AUTOMATED SEMANTIC FORGETTING FOR EXPRESSIVE DESCRIPTION LOGICS

A thesis submitted to the University of Manchester
for the degree of Doctor of Philosophy
in the Faculty of Science and Engineering

2018

Yizheng Zhao
School of Computer Science

Contents

Word Count: 54639

List of Tables

List of Figures

The University of Manchester

Yizheng Zhao
Doctor of Philosophy
Automated Semantic Forgetting for Expressive Description Logics
March 21, 2018

Ontologies, exploiting Description Logics (DLs) as the representational underpinning, provide a logic-based data model for knowledge processing thereby supporting intelligent reasoning of domain knowledge for various applications, most evidently for modern biomedical, life science and text mining applications. However, with their growing utilisation, not only has the number of available ontologies increased considerably, but they are also blowing up in size and becoming more complex to manage. Moreover, capturing domain knowledge in the form of ontologies is labour-intensive work which is expensive from an implementation perspective. There is a strong demand for techniques and automated tools for creating restricted views of ontologies while preserving complete information up to the restricted views.

Forgetting is a non-standard reasoning technique which provides such a service by eliminating concept and role symbols from ontologies in a way such that all logical consequences are preserved up to the remaining signature. It has turned out to be very useful in ontology-based knowledge processing, as it allows users to focus on specific parts of (usually very large) ontologies for easy reuse, or to zoom in on (usually very complex) ontologies for in-depth analysis. Other uses of forgetting are information hiding, explanation generation, abduction, ontology debugging and repair, and computing logical differences between ontology versions.

Despite its notable usefulness as described above, forgetting, on the other hand, is an inherently difficult problem — it is much harder than standard reasoning (satisfiability testing) — and very few logics are known to be complete for forgetting, there has been insufficient research on the topic and few forgetting tools are available.

This thesis investigates practical methods for semantic forgetting in expressive description logics not considered before. In particular, we present a practical method for forgetting concept symbols from ontologies expressible in the description logic $\mathcal{ALCOI}$, i.e., the basic $\mathcal{ALC}$ extended with nominals and inverse roles. Being based on a generalisation of a monotonicity property called Ackermann's Lemma, the method is the first and only approach to concept forgetting in description logics with nominals. We also present a practical method for forgetting role symbols from ontologies expressible in the description logic $\mathcal{ALCOIH}(\triangledown, \sqcap)$, i.e., $\mathcal{ALCOI}$ extended with role hierarchies, the universal role and role conjunction. The universal role and role conjunction enrich our target language, making it expressive enough to represent the forgetting solution which otherwise would have been lost. Being based on a non-trivial generalisation of Ackermann's Lemma, the method is the first and only approach so far that provides support for role forgetting in description logics with nominals.

Both methods are goal-oriented and incremental. They are terminating, and are sound in the sense that the forgetting solutions are equivalent to the original ontologies up to (the interpretations of) the symbols that have been forgotten, possibly with (the interpretations of) the symbols that have been introduced. These two methods can be used as a unifying method for forgetting both concept and role symbols from ontologies expressible in the description logic $\mathcal{ALCOIH}(\triangledown, \sqcap)$. The method has been

implemented in Java using the OWL API and the prototypical implementation, called FAME, has been evaluated on a corpus of real-world ontologies (in order to verify its practicality). Performance results have shown that FAME was successful (i.e., eliminated all specified concept and role symbols) in most of the test cases, and in most of these cases the elimination was done within a very short period of time.

Declaration

No portion of the work referred to in the 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 Statement

Acknowledgements

First of all, I would like to express my sincere gratitude and thanks to my supervisor Dr.-Ing. Renate A. Schmidt, you have been a tremendous mentor for me. I would like to thank you for encouraging my research and for allowing me to grow as a research scientist, for your patience, motivation and immense knowledge. Your guidance helped me in all the time of research and writing up of this thesis. I could not have imagined having a better supervisor and mentor for my Ph.D. study. I would also like to thank Dr. David E. Rydeheard and Prof. Andrzej Szalas for being willing to participate in my Ph.D. viva as respectively the internal examiner and external examiner.

I have to thank Dr. Zhanlin Ji, who encouraged me to undertake a Ph.D. and helped me address the financial concerns over the past three years. I would also like to thank all of my homeland friends who incented me to strive towards my goal.

I want to thank my office colleagues: Patrick Koopmann, Fabio Papacchini, Mohammad Khodadadi, Reyadh Alluhaibi, Ayo Adeniyi, Warren Del-Pinto, Julio Cesar Lopez Hernandez and Sen Zheng. These guys fertilised me with very insightful discussions on research and life matters.

Special thanks to my amazing family. Words cannot express how grateful I am to my mother, my father, my sister and my grandma for all of the sacrifices that you have made on my behalf. Your endless love was what sustained me thus far.

My time at The University of Manchester was made enjoyable in large part due to the many friends that became a part of my life. I am grateful for the leisure time spent with flatmates and the accompanying memories. I am also glad to have worked with a bunch of wonderful undergraduate and master students. I value their friendships.

Last but not least, I want to thank Manchester, the city where I have been living since August, 2011. The lifestyle the city influenced me has made me a real Mancunian.

Chapter 1

Introduction

Ontologies, exploiting description logics as the representational underpinning, are commonly used to model terminological domain knowledge using a set of appropriate concept and role symbols. They define the meaning of concepts and roles, and specify the relations between them. Being as one of the pillars of the Semantic Web, ontologies have practical deployments in a broad range of areas, in particular, the knowledge-intensive areas such as medicine, bioinformatics, software development, knowledge processing and many others. Because of the size and the knowledge-intensive nature, ontologies developed for applications in these areas can be monolithic and comprehensive, and often make use of a large number of concept and role symbols. For example, the SNOMED CT ontology, which plays a central role in health information system in different countries, contains more than 300000 concepts, the National Cancer Institute Thesaurus (NCI) contains more than 100000 concepts and the Gene Ontology (GO) contains more than 40000 concepts. This leads to, however, the ontologies being difficult to maintain and modify, and costly to reuse for later processing, where only a specific part of an ontology is of interest. For example, the Gene Ontology describes gene products across various kinds of species; a zoologist, meanwhile, is only interested in information concerned with endangered animals for research use. It is obvious in this case that working directly on the whole of the original ontology is not a sensible option and neither is building a new ontology from scratch only for use of this time, as both solutions are expensive from an implementation perspective. A wiser way, which has turned out to be more practical to this situation, is to create restricted views of the ontology, where the information appealing to the zoologist (i.e., the endangered

animals) has been well-preserved and those symbols not necessarily needed for the representation of this information have been gotten rid of.

Forgetting is a non-standard reasoning service that seeks to create restricted views of ontologies by eliminating concept and role symbols from description logic-based ontologies while preserving all logical consequences up to the remaining symbols in the ontologies. It allows users to focus on specific parts of (usually very large) ontologies for easy reuse, or to zoom in on (usually very complex) ontologies for in-depth analysis.

1.1 Applications of Forgetting

The investigation into forgetting is motivated by the high demand for advanced techniques for ontology-based knowledge processing. There are a multitude of real-world applications where forgetting can be useful. we enumerate some of them.

Ontology Summarisation: Creating a summary of an ontology is helpful when an ontology engineer wants to gain a quick understanding of the ontology, inexpensively examining its content (to decide whether the ontology is suitable for use). The summary may ignore more specific information and concentrate on more general terms that are expressed using only concepts and relations of high level.

Ontology Reuse: Knowledge modeled in large ontologies is often rich, heterogeneous, and multi-topic related, and applications are however interested in (or relevant to) specific parts. Compared to using existing ontologies or building new ontologies from scratch, extracting fragments w.r.t. specific interest (or relevance) from the existing ontologies and reusing them in specialised contexts is simpler, cheaper, and thus more useful to the ontology engineers.

Information Hiding: Ontologies for medical or military use may contain sensitive information that must be kept confidential to the public or the correspondences when the ontologies are published, shared, or disseminated. This is also relevant for uses in industry to ensure that proprietary information is sufficiently protected. This can be done by removing those concept and role symbols relative to sensitive information.

Logical Difference: In software engineering, the diff utility is a data comparison tool that computes and shows the differences between two files. In ontology engineering, such a tool is still useful. This is because ontologies constantly evolve and they

are regularly extended, updated, corrected and refined. An automated tool support for detecting the differences between different versions of ontologies becomes important. In particular, the syntactical difference is of little interest as same information may have different representation and some imformation is implicitly expressed. The semantic difference is however of greater interest. If different versions of ontologies give the same answers to a set of queries relevant to an application domain, then they may be of no difference, though they may syntactically look different. They are different, otherwise. Logic-based semantic approaches are required to detect whether two versions of ontologies are logically equivalent or different.

Forgetting is also useful for explanation generation, abduction, approximation, and ontology analysis, debugging, and repair.

1.2 Challenges and Contributions

Forgetting is research topic that has been actively investigated within the community over the last years. Foundational studies on its theoretical properties have shown that it is an inherently difficult problem and is not always solvable. [KLWW13] has shown that the solution of forgetting does not always exist for the description logic $\mathcal{ALC}$ and $\mathcal{EL}$, the existence of a solution of forgetting a concept or a role symbol is undecidable, even for the basic $\mathcal{ALC}$. For the solvable cases, the size of the forgetting solution can be triple exponential w.r.t. the size of the input ontology [NR14, LW11].

Other challenges include, e.g., in some cases, forgetting solutions cannot be represented by finitely many formulas, and often requires more expressivity than is available in the source logic. For example, the solution of semantically forgetting the role name r from the $\mathcal{ALC}$-ontology $\{A_1 \sqsubseteq \exists r.B, A_2 \sqsubseteq \forall r.\neg B\}$ is the set $\{A_1 \sqsubseteq \exists \nabla.B, A_1 \sqcap A_2 \sqsubseteq \bot\}$, whereas the uniform interpolant is $\{A_1 \sqcap A_2 \sqsubseteq \bot\}$, which is weaker. Observe in this case that the target language must include the universal role ∇ to represent the semantic forgetting solution.

Forgetting can be defined in two ways that are closely related; it can be defined on the syntactic level as the dual of *uniform interpolation* [Vis96] (i.e., related notions include weak forgetting [ZZ10], consequence-based inseparability [LW10] and deductive

conservative extensions [GLW06]) and it can be defined model-theoretically as *semantic forgetting* [WWT+14] (i.e., related notions include strong forgetting [LR94], model inseparability [KLWW13], model conservative extensions [LWW07] and second-order quantifier elimination [GSS08]). We explain the similarities and differences between these two notions in Chapter 3 (or see [BKL+16] a survey for their interrelations). The notion we take in this thesis is the semantic notion.

Practical methods such as NUI [LK13a, LK14] and LETHE [KS13d, KS14, Koo15, KS15a, KS15b, KS15c] have been developed for forgetting in expressive description logics, but both of them are focused on the syntactic notion (i.e., uniform interpolation). Methods for forgetting in description logics that follows the semantic notion include, e.g., [WWTP08, WWTP10, WWT+09]; however, these methods cannot be used practically and can only compute approximations of forgetting solutions. Moreover, the methods can only handle description logics as expressive as $\mathcal{ALC}$. For more expressive languages, they are impotent. At present, there are no practical methods for semantic forgetting in expressive description logics.

The main contributions of the thesis are summarised as follows:

- We present a practical method for forgetting concept symbols from ontologies expressible in the description logic $\mathcal{ALCOI}$. It is the first practical semantic approach that performs concept forgetting in expressive description logics. It is the first and only approach so far that provides support for forgetting concept symbols from ontologies specified in description logics with nominals.

- We present a practical method for forgetting role symbols from ontologies expressible in the description logic $\mathcal{ALCOIH}(\triangledown, \sqcap)$. It is the first practical semantic approach that performs role forgetting in expressive description logics. It is the first and only approach so far that provides support for forgetting role symbols from ontologies specified in description logics with nominals.

- The method for concept forgetting and the method for role forgetting can be used as a unifying method for forgetting both concept and role symbols for ontologies expressible in the description logic $\mathcal{ALCOIH}(\triangledown, \sqcap)$. It is the first practical semantic approach that performs both concept and role forgetting in expressive description logics. It is the first and only approach so far that provides support

for forgetting concept and role symbols from ontologies specified in description logics with nominals.

- The practicality of these methods is verified by an evaluation of a prototype of the methods on a large number of real-world biomedical ontologies.

- We also introduce a heuristic based on frequency analysis of the concept symbols specified to be forgotten. The heuristic allows specified concept symbols to be eliminated as quickly as possible.

1.3 Overview of the Thesis

Chapters 2 and 3 provide background material useful for the presentation of the remaining chapters. In Chapter 2, we introduce and explain the basics of description logics. In particular, we will first briefly review the history of the development and use of description logics, especially in the area of knowledge representation. Then, we will introduce the syntax and semantics of the basic description logic $\mathcal{ALC}$ and its extensions considered in this thesis. Next, we will look at the reasoning problems and services commonly considered in the research community of description logics. Finally, we will discuss the close relationship between description logics and other logics.

In Chapter 3, we introduce and explain the basics of forgetting. In particular, we will review the history of the investigation into forgetting. We will present significant (theoretical) results of the foundational studies of the properties of forgetting for description logics. We will describe existing methods and tools that have been developed for computing solutions of forgetting in a variety of description logics. We will also describe existing methods and tools developed for related problems, which have often been considered in other logics.

The main contributions of this thesis are due to Chapters 4, 5 and 6. In Chapter 4, we present a practical method for forgetting concept symbols from ontologies expressible in the description logic $\mathcal{ALCOI}$. We will describe the key ingredients of the method, with a particular focus on the calculus on which the method is based for eliminating a single concept symbol from a set of $\mathcal{ALCOI}$-clauses. We will show termination, soundness, and incompleteness of the method and we will demonstrate

the use of the method to solve concept forgetting problems.

In Chapter 5, we present a practical method for forgetting role symbols from ontologies expressible in the description logic $\mathcal{ALCOIH}(\triangledown, \sqcap)$. We describe the key ingredients of the method, with a particular focus on the calculus on which the method is based for eliminating a single role symbol from a set of $\mathcal{ALCOIH}(\triangledown, \sqcap)$-clauses. We will show termination, soundness, and partialcompleteness of the method and we will demonstrate the use of the method to solve role forgetting problems.

In order to make Chapter 4 and Chapter 5 a self-contained chapter and thus more readable and accessible to the reader, we provide a preliminary section at the beginning of each chapter (if necessary) where we introduce the source and target languages considered in that chapter, with a particular focus on their syntax and semantics, and in addition, we formalise the notion of forgetting w.r.t. the considered logics. Other important notions needed to the chapter are also established in this section.

In Chapter 6, we describe a prototype of our methods for concept and role forgetting. We evaluate the prototype on a corpus of real-world ontologies to verify the usefulness of our methods for various real-world applications.

The thesis is concluded with a summary of the results achieved in this thesis and an outlook on potential future research directions in Chapter 7.

1.4 Published Results

Most of the material presented in this thesis has been published at some conferences and workshops.

In Chapter 4, we present a practical method of semantic concept forgetting for ontologies expressible in the description logic $\mathcal{ALCOI}$. The work was first presented at the 28th International Workshop on Description Logics (DL-15) and then formally published at the 14th International Semantic Web Conference (ISWC-15) in 2015 [ZS15]. Chapter 4 is a significant extension of the work of [ZS15].

In Chapter 5, we present a practical method of semantic role forgetting for ontologies expressible in the description logic $\mathcal{ALCOIH}(\triangledown, \sqcap)$. The work was first presented at the 29th International Workshop on Description Logics (DL-16) and then formally

published at the 25th International Joint Conference on Artificial Intelligence (IJCAI-16) in 2016 [ZS16]. Chapter 5 is a significant extension of the work of [ZS16].

Part of the experiment results presented in Chapter 6 was also included in the work of [ZS15] and [ZS16], though the experiments were rerun with a later optimised implementation.

Until recently we have developed another practical method for semantic role forgetting in description logics with qualified number restrictions. The work was first presented at the 30th International Workshop on Description Logics (DL-17) and then formally published at the 26th International Joint Conference on Artificial Intelligence (IJCAI-17) in 2017 [ZS17].

Chapter 2

Basics of Description Logics

Description logics (DLs) are a family of knowledge representation languages that can be used to represent terminological knowledge of an application domain using a set of appropriate concept symbols, role symbols, and individual symbols [BN03]. Members of the DL family differ in expressivity as well as in computational complexity of reasoning. The name "description logics" is motivated by the fact that terms in the application domain are represented by (elementary and complex) *descriptions*. The most notable use of description logics is providing logical formalisms for ontologies and the Web Ontology Language (OWL). In this chapter, we introduce and explain the basic notions of description logics. In particular, in Section 2.1, we introduce the basic description logic $\mathcal{ALC}$, with a particular focus on the definitions of its syntax and semantics. In Section 2.2, we describe a number of important extensions of the basic $\mathcal{ALC}$. $\mathcal{ALC}$ with these extensions are the languages considered in this thesis. In Section 2.3, we explain the close relationships between description logics with other logics, namely, first-order logics and modal logics.

2.1 The Basic Description Logic $\mathcal{ALC}$

In this section, we introduce the basic description logic $\mathcal{ALC}$, the most important and the most widely considered description logic in the area. The name $\mathcal{ALC}$, standing for "Attributive concept Languages with Complements", was first introduced in [SS91]. A naming scheme for description logics was also proposed in this work: starting from the basic $\mathcal{ALC}$, the addition of a constructor is indicated by appending a corresponding

letter. For example, $\mathcal{ALCO}$ is obtained from $\mathcal{ALC}$ by allowing nominals in the language and $\mathcal{ALCOH}$ is obtained from $\mathcal{ALCO}$ by allowing role hierarchies in the language.

Elementary descriptions in $\mathcal{ALC}$ are *atomic concepts*, interpreted as sets of elements, and *atomic roles*, interpreted as binary relations between elements. Complex descriptions in $\mathcal{ALC}$ (i.e., concept expressions and role expressions) can be inductively built from atomic concepts and atomic roles using appropriate concept constructors and role constructors. The set a concept represents is referred to the *extension* of the concept. For instance, suppose Student and Teacher are atomic concepts, Mary is a student and John is a teacher. Then Mary is in the extension of Student and John is in the extension of Teacher. We use "is a" as a shorthand for "is in the extension of" to make the sentence rather shorter. A role r relates one element with another. The latter element is called an *r-filler* of the former one. For instance, suppose teaches is an atomic role and John teaches Mary. Then we say Mary is a teaches-filler of John.

In the remainder of this thesis, we use *concept symbols* to refer to atomic concepts and *role symbols* to refer to atomic roles. We use *concepts* to refer to concept expressions and *roles* to refer to role expressions. We use the symbols A and B to denote concept symbols, and the symbols C and D to denote concepts. We use the symbols r and s to denote role symbols, and the symbols R and S to denote roles. In each of these cases, subscripts and prime symbols may be used.

2.1.1 $\mathcal{ALC}$ Syntax

Definition 2.1.1. Let $\mathsf{N_C}$ and $\mathsf{N_R}$ be countably infinite and pairwise disjoint sets of *concept symbols* and *role symbols*. *Roles* in $\mathcal{ALC}$ ($\mathcal{ALC}$-roles) can be any role symbol $r \in \mathsf{N_R}$. *Concepts* in $\mathcal{ALC}$ ($\mathcal{ALC}$-concepts) have one of the following forms:

$$\begin{aligned}
&\mathrm{A} \mid \text{(atomic concept)}\\
&\top \mid \text{(the top concept)}\\
&\bot \mid \text{(the bottom concept)}\\
&\neg C \mid \text{(concept negation)}\\
&C \sqcap D \mid \text{(concept conjunction)}\\
&C \sqcup D \mid \text{(concept disjunction)}\\
&\exists r.C \mid \text{(existential restriction)}
\end{aligned}$$

$$\forall r.C \mid \text{(universal restriction)},$$

where $A \in \mathsf{N_C}$, $r \in \mathsf{N_R}$, C and D are arbitrary concepts. As usual, we use brackets to clarify the structure of concepts.

Suppose Lion and Female are atomic concepts. Then $\mathsf{Lion} \sqcap \mathsf{Female}$ and $\mathsf{Lion} \sqcap \neg\mathsf{Female}$ are concepts describing lions who are female and lions who are not female, respectively. Suppose further hasCub is an atomic role. Then $\mathsf{Lion} \sqcap \exists\mathsf{hasCub}.\mathsf{Female}$ and $\mathsf{Lion} \sqcap \forall\mathsf{hasCub}.\mathsf{Female}$ are concepts describing lions who have some cubs that are female and lions whose cubs are all female, respectively.

2.1.2 $\mathcal{ALC}$ Semantics

In order to define a formal semantics for concepts and roles in $\mathcal{ALC}$ (i.e., to fix the meaning of them), we consider using an *interpretation*, which is a structure that:

- consists of a non-empty set called its *interpretation domain*. The elements of the interpretation domain are simply called *elements*.
- fixes the extension for each concept symbol, that is, the interpretation tells us which elements of the interpretation domain are (or are not) in the extension of each concept symbol, and
- fixes the extension for each role symbol, that is, the interpretation tells us which pairs of elements of the interpretation domain are (or are not) related to each other by this role.

By making use of an interpretation, we formally define the semantics of concept and roles in $\mathcal{ALC}$ as follows.

Definition 2.1.2. Let $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ be an interpretation, where $\Delta^{\mathcal{I}}$ is a non-empty set (the *domain of the interpretation*), and $\cdot^{\mathcal{I}}$ is the *interpretation function*, which assigns to every concept symbol $A \in \mathsf{N_C}$ a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, and to every role symbol $r \in \mathsf{N_R}$ a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The interpretation function $\cdot^{\mathcal{I}}$ is inductively extended to concepts and roles as follows:

$$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$$

$$\begin{aligned}
\bot^{\mathcal{I}} &= \emptyset \\
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\exists r.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \exists y.(x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \\
(\forall r.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \forall y.(x,y) \in r^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\},
\end{aligned}$$

We say $C^{\mathcal{I}}$ is the extension of C in $\mathcal{I}$, and $b \in \Delta^{\mathcal{I}}$ an r-filler of a in $\mathcal{I}$ if $(a,b) \in \mathcal{I}'$.

Note that no restriction is imposed on the interpretation unless explicitly specified otherwise, that is, the interpretation domain must be non-empty, but can be of any cardinality, and in particular, the interpretation domain can be infinite; there can be any number of elements from "none" to "all" in the extension of a concept, and any number of pairs of elements from "none" to "all" can be related by a role.

2.1.3 $\mathcal{ALC}$ Knowledge Bases

Given an application domain, we use description logics to represent relevant notions and knowledge from this application domain. For example, given a sports application, we would build concepts describing football, tennis, Olympics etc., which would be stored in the knowledge base of the application. For different notions and knowledge, description logics build concepts in (at least) four different ways:

i. As in an encyclopedia, description logics define the meaning of a concept symbol in terms of a concept (expression). For example, we can define the meaning of Footballer and Olympian using the following DL expressions:

$$\begin{aligned}
\mathsf{Footballer} &\equiv \mathsf{Person} \sqcap \exists \mathsf{plays}.\mathsf{Football} \\
\mathsf{Olympian} &\equiv \mathsf{Person} \sqcap \exists \mathsf{participatesIn}.\mathsf{OlympicGames}
\end{aligned}$$

Intuitively, the first expression says that footballers are persons who play football and the second expression says that Olympians are persons who participate in Olympic Games.

ii. Description logics capture background knowledge. For example, we can state that a footballer is also a sportsperson, and that a person has sex either male or

female, using the following DL expressions:

$$\mathsf{Footballer} \sqsubseteq \mathsf{Sportsperson}$$

$$\mathsf{Person} \sqsubseteq \exists \mathsf{hasSex}.(\mathsf{Male} \sqcup \mathsf{Female})$$

iii. Description logics assert that individuals stand for instances of concepts. For example, we can assert that COMP60332 stands for an instance of Course, and that Phelps stands for an instance of person who participates in OlympicGames, using the following DL expressions:

$$\{\mathsf{COMP60332}\} \sqsubseteq \mathsf{Course}$$

$$\{\mathsf{Phelps}\} \sqsubseteq \mathsf{Person} \sqcap \exists \mathsf{participatesIn}.\mathsf{OlympicGames}$$

iv. Description logics relates individuals by roles. For example, we can say that John is a teacher who teaches COMP60332 using the following expressions:

$$\{\mathsf{John}\} \sqsubseteq \exists \mathsf{teaches}.\{\mathsf{COMP60332}\}.$$

Description logics separate domain knowledge into two components, i.e., a terminological component, namely, the *TBox*, and an assertional component, namely, the *ABox*. The TBox contains a set of statements of the form as described in Items (i) and (ii), and the ABox contains a set of statements of the form as described in Items (iii) and (iv). Together TBox statements and ABox statements make up a *knowledge base*. One can think of a knowledge base as a database. TBox statements thus correspond to the schema of the database and ABox statements thus correspond to the data of the database. The statements in the TBox are referred to as *TBox axioms* and the statements in the ABox are referred to as *ABox axioms*.

Next, we formally define $\mathcal{ALC}$ TBoxes, $\mathcal{ALC}$ ABoxes, and $\mathcal{ALC}$ knowledge bases.

Definition 2.1.3. An $\mathcal{ALC}$ *TBox* $\mathcal{T}$ is a finite set of *concept inclusions* of the form $C \sqsubseteq D$ and *concept equivalences* of the form $C \equiv D$, where C and D are concepts. Let $\mathsf{N_O}$ be a countably infinite set of individual symbols disjoint from $\mathsf{N_C}$ and $\mathsf{N_R}$. An $\mathcal{ALC}$ *ABox* $\mathcal{A}$ is a finite set of *concept assertions* of the form $C(a)$ and *role assertions* of the form $r(a, b)$, where $a, b \in \mathsf{N_O}$, C is a concept and r is a role symbol. An $\mathcal{ALC}$ knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is the union of an $\mathcal{ALC}$ TBox $\mathcal{T}$ and an $\mathcal{ALC}$ ABox $\mathcal{A}$.

A concept inclusion $C \sqsubseteq D$ is *true* in an interpretation $\mathcal{I}$, and we write $\mathcal{I} \models C \sqsubseteq D$, iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. A concept equivalence $C \equiv D$ is *true* in an interpretation $\mathcal{I}$, and we write $\mathcal{I} \models C \equiv D$, iff $C^{\mathcal{I}} \equiv D^{\mathcal{I}}$. $\mathcal{I}$ is a *model* of a TBox $\mathcal{T}$ iff every axiom in $\mathcal{T}$ is *true* in $\mathcal{I}$. In this case we write $\mathcal{I} \models \mathcal{T}$. A concept assertion $C(a)$ is *true*, and we write $\mathcal{I} \models C(a)$, iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$. A role assertion $r(a, b)$ is *true* in an interpretation $\mathcal{I}$, and we write $\mathcal{I} \models r(a, b)$, iff $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$. $\mathcal{I}$ is a *model* of an ABox $\mathcal{A}$ iff every axiom in $\mathcal{A}$ is *true* in $\mathcal{I}$. In this case we write $\mathcal{I} \models \mathcal{A}$. $\mathcal{I}$ is a *model* of a knowledge base $\mathcal{K}$ iff $\mathcal{I}$ is a model of both the TBox $\mathcal{T}$ and the ABox $\mathcal{A}$. In this case we write $\mathcal{I} \models \mathcal{K}$.

2.2 Extensions of the Basic $\mathcal{ALC}$

Next, we introduce three important extensions of the basic description logic $\mathcal{ALC}$, namely, nominals, inverse roles and role hierarchies. The languages we considered in this thesis are the description logic $\mathcal{ALC}$ with these extensions.

2.2.1 Nominals

In the previous section, we used individual symbols in ABox axioms, where the interpretations of the individual symbols were constrained by concepts and roles. However, there are situations where individual symbols need to occur inside concepts. For example, if we want to define PhDStudentsofJohn as those PhD students supervised by John, an intuitive way (of defining the notion) is the following:

$$\mathsf{PhDStudentsofJohn} \equiv \mathsf{PhDStudents} \sqcap \exists \mathsf{supervisedBy}.\mathsf{John}$$

However, there is a syntax error in this definition, i.e., John, as an individual symbol, occurs inside a concept (we have in contrast clearly stated that individual symbols are disjoint from concept symbols; hence John cannot occur both as an individual symbol and as a concept symbol). On the other hand, even if John were allowed to occur in place of a concept symbol in this definition, it is not clear how to interpret it. For an interpretation $\mathcal{I}$, $\mathsf{John}^{\mathcal{I}}$ is an element of the interpretation domain, whereas concepts are interpreted as sets of elements. To facilitate the use of individual symbols inside concepts, we introduce *nominals* [KSH12, KKS12]. Nominals are concepts that have exactly one instance, e.g., $\{\mathsf{John}\}$ is a concept that has only the instance John.

That a description logic incorporates nominals is indicated by the letter $\mathcal{O}$ in its name. Thus, $\mathcal{ALC}$ with nominals is denoted by $\mathcal{ALCO}$.

Definition 2.2.1. Let $\mathcal{I}$ be any interpretation. If a is an individual symbol in $\mathcal{I}$, then $\{a\}$ is called a *nominal*. The interpretation function $\cdot^{\mathcal{I}}$ is extended to nominals by assigning to every nominal $a \in \mathsf{N_O}$ a singleton $a^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, i.e., $(\{a\})^{\mathcal{I}} = \{a^{\mathcal{I}}\}$.

Thus, we can define PhDStudentsofJohn using the following DL expression:

$$\mathsf{PhDStudentsofJohn} \equiv \mathsf{PhDStudents} \sqcap \exists \mathsf{supervisedBy}.\{\mathsf{John}\}$$

An alternative way to define PhDStudentsofJohn, besides defining it in the traditional way as shown above, is to enumerate all its instances. For example, suppose there are five PhD students supervised by John, namely, Amy, Bruce, Carol, David and Eason. Then we can define PhDStudentsofJohn as follows:

$$\mathsf{PhDStudentsofJohn} \equiv \{\mathsf{Amy}\} \cup \{\mathsf{Bruce}\} \cup \{\mathsf{Carol}\} \cup \{\mathsf{David}\} \cup \{\mathsf{Eason}\}$$

More specifically, we define the notion by combining nominals with union.

2.2.2 Inverse Roles

Besides providing a variety of concept constructors (i.e., negation, concept conjunction and concept disjunction), description logics also provide a few role constructors for building complex roles. *Inverse roles* are (one of) the most important such constructor. Intuitively, the relationship between the roles teaches and isTaughtBy is that, for example, if John teaches the course COMP60332, then COMP60332 isTaughtBy John, and if the course COMP60332 isTaughtBy John, then John teaches COMP60332. Such a relationship can be captured by using the inverse operator $(^-)$:

$$\mathsf{teaches} \equiv \mathsf{isTaughtBy}^-$$

Inverse roles allow binary relations (i.e. roles) to be used symmetrically. There are situations where without inverse roles, a satisfactory knowledge base cannot be built and the expected knowledge cannot be derived.

Example 2.2.2. Consider the following knowledge base $\mathcal{K}$:

$$\mathsf{Teacher} \equiv \mathsf{Person} \sqcap \exists \mathsf{teaches}.\mathsf{Course}$$

$$\mathsf{Professor} \sqsubseteq \mathsf{Teacher}$$

$$\mathsf{Course} \sqsubseteq \forall \mathsf{isTaughtBy}.\neg\mathsf{Professor}$$

The second axiom states that professors are teachers, and the third axiom states that Courses are not taught by professors. Intuitively, professors are unsatisfiable w.r.t. $\mathcal{K}$, because due to the first axiom in $\mathcal{K}$, an element p in the extension of Professor should also be in the extension of Teacher. This implies that p has a teaches-filler c that is a Course. Normally (and intuitively), if p teachers c, then c isTaughtBy p. On the other hand, the third axiom implies that p is not a Professor, which contradicts with our initial assumption that p is a Professor. However, one may have neglected that there is a critical flaw that the roles teaches and isTaughtBy are treated as arbitrary binary relations in $\mathcal{K}$, and they are not related in a way that facilitates the soundness of the statement "if p teachers c, then c isTaughtBy p". Thus, Professor is satisfiable w.r.t. $\mathcal{K}$, which is undesired. This can be solved by extending the source language with inverse roles. In particular, the inverse roles allow teaches and isTaughtBy to be related (by the inverse operator) in such a way that "if p teachers c, then c isTaughtBy p".

That a description logic incorporates inverse roles is indicated by the letter $\mathcal{I}$ in its name. Thus, $\mathcal{ALCO}$ with inverse roles is denoted by $\mathcal{ALCOI}$. Roles in $\mathcal{ALCOI}$ can be a role symbol $r \in \mathsf{N_R}$ or the inverse r^- of a role symbol r. We assume w.l.o.g. that if R is already an inverse role, say S^-, then the inverse of R is written as $\mathcal{S}$, rather than R^-. We avoid double inverse operators in our language. The definition of the semantics of $\mathcal{ALCOI}$ can be obtained from that of $\mathcal{ALCO}$ by extending the interpretation function $\cdot^{\mathcal{I}}$ to inverse roles as follows:

$$(R^-)^{\mathcal{I}} = \{(y, x) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (x, y) \in R^{\mathcal{I}}\}$$

It has been known that concept satisfiability of $\mathcal{ALCOI}$ is ExpTime-hard [ABM99, ABM00]. Other logics allowing for inverse roles (or the converse operator) were investigated in the work of e.g. [Str82, Var85, DM96, De 96, Cal96, HS99, HST99].

2.2.3 Role Hierarchies

In many applications of knowledge representation, there are situations where two roles on the same concept are related in the way that every filler of one role is also a filler

of the other role. For example, in the university domain, the fillers of the role takesCSCourse, relating students to courses in computer science, are also fillers of the role takesCourse, relating students to courses in any discipline. This can be captured by the sub-role relationship between roles. Role hierarchies provide a mean to capture sub-role relationship between roles [HS99], i.e., takesCSCourse is a sub-role of takesCourse. Role hierarchies are also helpful when modeling sub-relations of the general part-whole relation [Sat96]. As with in the case of inverse roles, there are situations where without role hierarchies, a satisfactory knowledge base cannot be built and the expected knowledge cannot be correctly derived.

Example 2.2.3. Consider the following knowledge base $\mathcal{K}$:

$$\begin{aligned} \mathsf{OlympicMedalist} &\equiv \mathsf{person} \exists \mathsf{hasWonMedalIn}.\mathsf{OlympicGames} \\ \mathsf{Bolt} &: \mathsf{Person} \\ (\mathsf{Bolt}, \mathsf{men's100MetresSprint}) &: \mathsf{hasWonGoldMedalIn} \\ \mathsf{men's100MetresSprint} &: \mathsf{OlympicGames} \end{aligned}$$

The first axiom states that Olympic medalists are persons who have won medals in Olympic Games. The second axiom states that Bolt is an person. The third axiom states that Bolt has won gold medal in men's 100 metres sprint. The last axiom states that men's 100 metres sprint is an Olympic Game event. Then, we should derive that "Bolt is an Olympic medalist" from the knowledge base. However, this is not the case because the knowledge base did not capture the intended relatipnship between hasWonGoldMedalIn and hasWonMedalIn, i.e., hasWonGoldMedalIn implies hasWonMedalIn. In this case, we can use the following role inclusion to capture this relationship:

$$\mathsf{hasWonGoldMedalIn} \sqsubseteq \mathsf{hasWonMedalIn}$$

That a description logic incorporates role hierarchies is indicated by the letter $\mathcal{H}$ in its name. Thus, $\mathcal{ALCOI}$ with role hierarchies is denoted by $\mathcal{ALCOIH}$. Description logics with role hierarchies have additionally an RBox, defined as follows.

Definition 2.2.4. An *RBox* $\mathcal{R}$ is a finite set of *role inclusions* of the form $R \sqsubseteq S$ and *role equivalences* of the form $R \equiv S$, where R and S are roles.

We use $R \equiv S$ as an abbreviation of the pair $R \sqsubseteq S$ and $S \sqsubseteq R$. A role hierarchy is a set of role inclusions. Note that, sometimes, role inclusions and equivalences are considered as TBox axioms in some literature. We however distinguish TBox axioms and RBox axioms in this thesis. RBox axioms refer to properties of roles. In more expressive description logics (e.g., description logics with transitive property on roles), the RBox also contains other axioms such as transitive axioms of the form $\mathsf{trans}(R)$, where R is a role. The axiom $\mathsf{trans}(R)$ indicates that R is a transitive role [HG97].

A role inclusion $\mathcal{S} \sqsubseteq \mathcal{S}'$ is *true* in an interpretation $\mathcal{I}$, and we write $\mathcal{I} \models \mathcal{S} \sqsubseteq \mathcal{S}'$, iff $\mathcal{S}^{\mathcal{I}} \subseteq \mathcal{S}'^{\mathcal{I}}$. A role equivalence $\mathcal{S} \equiv \mathcal{S}'$ is *true* in an interpretation $\mathcal{I}$, and we write $\mathcal{I} \models \mathcal{S} \equiv \mathcal{S}'$, iff $\mathcal{S}^{\mathcal{I}} \subseteq \mathcal{S}'^{\mathcal{I}}$ and $\mathcal{S}'^{\mathcal{I}} \subseteq \mathcal{S}^{\mathcal{I}}$. $\mathcal{I}$ is a *model* of an RBox $\mathcal{R}$ iff every axiom in $\mathcal{R}$ is *true* in $\mathcal{I}$. In this case, we write $\mathcal{I} \models \mathcal{R}$.

Definition 2.2.5. An ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A}, \mathcal{R})$ is the union of an TBox $\mathcal{T}$, an ABox $\mathcal{A}$ and an RBox $\mathcal{R}$.

2.3 Relationships with Other Logics

In this section, we explain the close relationship between description logics and other interesting logics, in particular, with first-order logic and modal logic. The main motivation for establishing the correspondence between these logics is to exploit results of one logic to draw conclusions about another logic, and also to use existing methods and tools of a logic to solve problems for another logic. For example, the main result of this thesis is a practical method of semantic concept and role forgetting for expressive description logics. The result can be adapted to a modal method for computing correspondence properties for modal logics that correspond to the (description) logics considered in this thesis, because of the close relationship between these two logics, which we will discover in the sequel. Another motivation is to provide readers who are familiar with first-order logic and modal logic yet new to description logics a deeper understanding of the material and the field.

2.3.1 DLs as Decidable Fragments of First-Order Logics

Borgida [Bor96] presents a variety of results stating that most description logics are decidable fragments of first-order logic, though some of them allow for constructors

such as transitive closure of role or fixpoints which make them decidable fragments of second-order logic. By simply mapping concept symbols as unary predicates, role symbols as binary predicates and individual symbols as constants, TBox, ABox and RBox axioms can be translated into first-order logic formulas.

Example 2.3.1. Consider the following ontology $\mathcal{O}$:

$$\mathsf{Mother} \sqsubseteq \exists \mathsf{hasChild}.\top$$

$$\mathsf{Teacher} \equiv \mathsf{Person} \sqcap \exists \mathsf{teaches}.\mathsf{Course}$$

$$\mathsf{COMP60332} : \mathsf{Course}$$

$$\mathsf{hasMother} \sqsubseteq \mathsf{hasAncestor}$$

Using the mappings as mentioned above, $\mathcal{O}$ can be translated into the following first-order logic formulas:

$$\forall x.(\mathsf{Mother(x)} \rightarrow \exists y.\mathsf{hasChild(x,y)})$$

$$\forall x.(\mathsf{Teacher(x)} \leftrightarrow \mathsf{Person(x)} \wedge \exists y.(\mathsf{teaches(x,y)} \wedge \mathsf{Course(y)}))$$

$$\mathsf{Course(COMP60332)}$$

$$\forall x.\forall y.(\mathsf{hasMother(x,y)} \rightarrow \mathsf{hasAncestor(x,y)})$$

Note that TBox and RBox axioms correspond to universally quantified (bi-)implications without free variables, and ABox axioms correspond to ground facts.

The translation can be formalised using a *translation function*, π, which inductively maps concepts and roles to first-order logic formulas with one or two free variables:

$$\pi(\top) = \mathsf{true}$$

$$\pi(\bot) = \mathsf{false}$$

$$\pi(a) = x \approx a$$

$$\pi(A) = A(x)$$

$$\pi(\neg C) = \neg\pi(C)$$

$$\pi(C \sqcap D) = \pi(C) \wedge \pi(D)$$

$$\pi(C \sqcup D) = \pi(C) \vee \pi(D)$$

$$\pi(\exists R.C) = \exists y.R(x, y) \wedge \pi(C)$$

$$\pi(\forall R.C) = \forall y.R(x,y) \rightarrow \pi(C)$$
$$\pi(R) = R(x,y)$$
$$\pi(R^{-}) = R(y,x)$$

π maps TBox, ABox and RBox axioms to first-order logic formulas as follows:

$$\pi(C \sqsubseteq D) = \forall x.(\pi(C) \rightarrow \pi(D))$$
$$\pi(C \equiv D) = \forall x.(\pi(C) \leftrightarrow \pi(D))$$
$$\pi(R \sqsubseteq S) = \forall x.\forall y.(\pi(R) \rightarrow \pi(S))$$
$$\pi(R \equiv S) = \forall x.\forall y.(\pi(R) \leftrightarrow \pi(S))$$
$$\pi(a : C) = a : \pi(C)$$
$$\pi((a,b) : R) = (a,b) : \pi(R)$$

It is clear to see that this translation preserves the semantics. Thus, it provides an alternative way of defining the semantics of description logics (and vice versa).

The translation also indicates that all reasoning problems for $\mathcal{ALC}$ knowledge bases are decidable, because the translation of a knowledge base uses only two free variables, which makes the resulting formulas in the *two-variable fragment* of first-order logic. In addition, it is known that satisfiability is decidable for *two-variable fragment* of first-order logic in nondeterministic exponential time [GKV97, GOR99]. Even in the case of that the translation of a knowledge base uses quantification, it is used only in a restricted way. The resulting formula is in *guarded fragment* of first-order logic [ANvB98], for which satisfiability is known to be decidable in deterministic exponential time [Grä99].

2.3.2 DLs as Cousins of Modal Logics

Description logics are close relatives of modal logic, though they have been independently developed. This close relationship was first discovered by Schild in [Sch91], and transfer complexity and decidability result as well as reasoning techniques from one logic to the other [Sch94, DL94a, HP98, AdRdN01]. The most well-known result is that the correspondence between the description logic $\mathcal{ALC}$ and multi-modal logic **K**, i.e., $\mathcal{ALC}$ can be viewed as syntactic variants of formulas of **K**.

In particular, to connect $\mathcal{ALC}$ with multi-modal logic, all we need to do is the following. Semantically, from $\mathcal{ALC}$ to multi-modal logic, we correspond interpretations of description logics to Kripke structures and vice versa. We correspond interpretations of concept symbols to values of propositional variables, role symbols to binary relations used as interpretations for the modal operators. Syntactically, as with the correspondence from description logics to first-order logic, the correspondence from description logics to modal logic is realised via a function π as follows:

$$\pi(A) = p_A, \text{ for concept symbol } A$$
$$\pi(\neg C) = \neg\pi(C)$$
$$\pi(C \sqcap D) = \pi(C) \wedge \pi(D)$$
$$\pi(C \sqcup D) = \pi(C) \vee \pi(D)$$
$$\pi(\exists r.C) = \langle r \rangle \pi(C)$$
$$\pi(\forall r.C) = [r]\pi(C)$$

The translation of more expressive description logics is straightforward as well. For example, Schild [Sch94] and De Giacomo and Lenzerini [DL94b] identified a correspondence between description logics and the modal μ-calculus. Van der Hoek and De Rijke [vdHdR95] considered correspondence between description logics with qualified number restrictions and modal logic with counting expressions.

Chapter 3

Basics of Forgetting

The origins of interest in forgetting can be traced back to the work of Boole on propositional variable elimination and the seminal work of Ackermann [Ack35] who recognised that the problem amounts to the elimination of existential second-order quantifiers. In this chapter, we review the provenance and history of the forgetting problem and see how forgetting has been studied for various logics. For a deeper understanding of the origins and initial development of forgetting, we refer the reader to [Bro03].

3.1 Forgetting in Classical Logics

In propositional logic, forgetting has often been studied under the name of *variable elimination*. In particular, forgetting a propositional variable p from a set S of formulas is essentially the operation of checking satisfiability of S by assigning a truth value to p (i.e., the idea of the DPLL algorithm [DP60] and [DLL62]). The solution of forgetting a propositional variable p from a set S of formulas is simply the disjunction $S^p_\top \vee S^p_\bot$, where $S^p_\top$ denotes the set obtained from S by substituting $\top$ for every occurrence of p in S. Literature on forgetting/variable elimination for propositional logics include e.g. [KHM99, LM02, LLM03, BLM06]. It has been found that the size of the solution of forgetting in propositional logic can be exponential w.r.t. the size of the input in the worst cases [EW08], which is an undesirable property for forgetting. Zhou [Zho14] proposed a restricted notion of bounded forgetting, referred to as *polynomially bounded forgetting*, where the size of the forgetting solution is manageable for propositional logic and can be expressed polynomially.

In first-order logic, forgetting has often been studied as an instance of the *second-order quantifier elimination* (*SOQE*) problem. Since the problem is a research topic closely related to the material presented in this thesis, we introduce the problem in detail in the next section. The name "*forgetting*" was first proposed in the context of first-order logic in [LR94] to denote the operation of eliminating predicate symbols from first-order logic formulas. A formal definition of this forgetting is formulised as follows, which is also referred to as *strong forgetting* or *semantic forgetting* in other literature [ZFW05, EW08, WWT+09, ZZ10, WWTP10, PGJ11, WWT+14, FZ16, Lei17].

Definition 3.1.1 (**Strong/Semantic Forgetting**)**.** Let F be a first-order logic formula and let P be a predicate symbol. F' is a *solution of strongly forgetting* P from F iff for any interpretation M, $M \models F'$ iff there is some interpretation $M' \models F$ such that M' and M coincide but differ possibly on the truth value of P.

We say that the solution of strong/semantic forgetting is equivalent to the original formula up the truth value of P. Observe from the definition that forgetting does not require the given formula F to contain P and the forgetting solution F' to not contain P, though this is assumed in most of the work on forgetting (also in this thesis).

It is noted in [LR94] that the solution of strong/semantic forgetting a predicate symbol P from a first-order logic formula F is equivalent to the second-order formula $\exists X.F_X^P$, where F_X^P is our notation for substituting every occurrence of P in F by X. The task of forgetting in first-order logic, as a computational problem, is to find a first-order formula F' (without second-order quantification) that is equivalent to $\exists X.F_X^P$. Finding such a formula F' equivalent to $\exists X.F_X^P$ is an instance of the second-order quantifier elimination problem.

It is however observed that the solution of strong/semantic forgetting is not always expressible in first-order logic (i.e., the solution of strong/semantic forgetting does not always exist). Motivated by this observation, Zhang and Zhou [ZZ10] proposed an alternative notion of forgetting, referred to as *weak forgetting* in the work, where the forgetting solution is always expressible in first-order logic. The notion of weak forgetting can be formalised as follows.

Definition 3.1.2 (**Weak Forgetting**)**.** Let F be a first-order logic formula and let P be a predicate symbol. F' is a *solution of weakly forgetting* P iff for every formula

G that does not contain P, $F' \models G$ iff $F \models G$.

Intuitively, this means that if a formula G (not containing P) is a consequence of the solution F' of weakly forgetting P from the given formula F, then it is a consequence of F, and vice versa. We say that the strong/semantic forgetting preserves equivalence (up to certain signature), whereas the weak forgetting preserves only logical consequences (up to certain signature).

The solution of weak forgetting is always expressible in first-order logic, though there are cases the forgetting solution can only be represented by an infinite set of first-order logic formulas.

It is noted in [ZZ10] that the solution of strong/semantic forgetting is not always equivalent to that of weak forgetting, but if the solution of strong/semantic forgetting is first-order expressible, then the two notions coincide. We have also found that the solution of strong/semantic forgetting often requires more expressivity than is available in the source logic. Hence, the solution of strong/semantic forgetting (possibly expressed in an extended language) is often stronger than that of weak forgetting.

Because the solution of weak forgetting cannot always be represented by a finite set of first-order logic formulas, Zhang and Zhou [ZZ11] proposed a restricted notion of forgetting, referred to as *bounded forgetting*, where the solution only preserves logical consequences up to certain so-called *quantifier rank* (i.e., for the details of this notion we refer the reader to [ZZ11]). By making use of quantifier rank, the solution of bounded forgetting can always be represented by a finite set of formulas.

We apply the strong/semantic notion of forgetting in this thesis. In the remainder of the thesis, we will use the name "semantic forgetting" to refer to the notion.

3.2 Second-Order Quantifier Elimination

‘ Second-order quantifier elimination refers to a non-standard reasoning problem concerned with transforming second-order logic formulas into equivalent first-order logic formulas. The transformation is achieved by eliminating existentially quantified predicate symbols (universally quantified predicate symbols can be eliminated due to the equivalence $\forall X.F \equiv \neg\exists X.\neg F$). Therefore, the task of second-order quantifier elimination is to transform a (second-order logic) formula of the form $\exists X.F$ into an equivalent

(first-order logic) formula F' without existentially quantified second-order quantifiers.

It is observed that in this sense second-order quantifier elimination amounts to semantic forgetting. The result of eliminating existential second-order quantifiers is equivalent to the original formula up to the symbols that have been eliminated. This is consistent with the notion of semantic forgetting where the forgetting solution is equivalent to the original formula up to the symbols that have been forgotten. It is known that second-order quantifier elimination is undecidable [Ack35, GSS08] and very few logics are known to be complete for the problem. Nevertheless, this does not mean there is no need for the development of practical methods. Successful techniques from the area have shown very good results, even though these methods are not complete. In particular, these methods can directly be used for forgetting predicate symbols from first-order logic formulas.

Several second-order quantifier elimination methods have been proposed in the context of various logics and these methods can be generalised into two categories:

(i) *saturation approach* based on exhausitive deduction of consequences, and

(ii) *substitution-rewrite approach* based on monotonicity properties.

Methods following the saturation approach include the SCAN algorithm [GO92] and hierarchical resolution [BGW94a], and methods following the substitution-rewrite approach include the DLS algorithm [DLS97], the DLS* algorithm [NS98], the SQEMA algorithm [CGV06a], the MSQEL algorithm [Sch12], the Sahlqvist-van Benthem method for modal logic [Sza93, Hen75] and the method of simmons [Sim94]. In the sequel, we enumerate some of these methods, which are closely related to our method.

3.2.1 SOQE Based on Resolution

The SCAN algorithm, due to Gabbay and Ohlbach [GO92], is a saturation method for eliminating existential second-order quantifiers from a set of formulas. The method is based on the resolution calculus [Rob63] and works with formulas in clausal form. The idea of SCAN is to generate all possible logical consequences from the given formulas containing second-order variables and from the resulting formulas preserve only those non-redundant formulas in which no second-order variables occur [GSS08]. The method has been extended and implemented by Ohlbach [Ohl96] and Engel [Eng96].

The input of SCAN is a (second-order logic) formula of the following form:

$$\exists P_1 \exists P_2 \ldots \exists P_n.F,$$

where the P_i $(1 \leq i \leq n)$ are predicate variables and F is a set of quantifier-free first-order logic formula. We apply the SCAN algorithm to this set to eliminate all occurrences of P_i in this set, thereby yielding a set of (first-order logic) formulas equivalent to the original set up to the P_i.

SCAN was the first approach for forgetting in first-order logic. It is known that SCAN is complete for all Sahlqvist formulas in multi-modal logic [Hen75, GHSV04].

An alternative method based on resolution is the hierarchical resolution [BGW94b].

3.2.2 SOQE Based on Ackermann's Lemma

Second-order quantifier elimination methods following the substitution-rewrite approach are based on a monotonicity property found in [Ack35], referred to as *Ackermann's lemma*, whose main idea is captured by the following theorem.

Theorem 3.2.1 (Ackermann's Lemma)**.** *Let P be a predicate variable and $\alpha(\overline{x}, \overline{z})$, $\beta(P)$ be first-order formulas (without second-order quantification), where the number of distinct variables in $\overline{x}$ is equal to the arity of P and $\alpha(\overline{x}, \overline{z})$ does not contain P.*
If $\beta(P)$ is positive w.r.t. P, then

$$\exists P\{\forall \overline{x}[P(\overline{x}) \rightarrow \alpha(\overline{x}, \overline{z})] \wedge \beta(P)\} \equiv \beta(P)^{P(\overline{x})}_{\alpha(\overline{x},\overline{z})}.$$

If $\beta(P)$ is negative w.r.t. P, then

$$\exists P\{\forall \overline{x}[\alpha(\overline{x}, \overline{z}) \rightarrow P(\overline{x})] \wedge \beta(P)\} \equiv \beta(P)^{P(\overline{x})}_{\alpha(\overline{x},\overline{z})}.$$

By $\beta(P)^{P(\overline{x})}_{\alpha(\overline{x},\overline{z})}$, we denote the expression obtained from $\beta(P)$ by replacing all occurrences of $P(\overline{x})$ by $\alpha(\overline{x}, \overline{z})$.

Compared to those (second-order quantifier elimination) methods based on resolution, which work directly on the given clauses, Ackermann-based methods require all relevant clauses to be transformed into a certain form suitable for application of Ackermann's Lemma. In particular, Ackermann's Lemma requires the positive and negative occurrences of the eliminated second-order variable to be properly separated.

On the other hand, Ackermann-based methods do not generate as many clauses as the resolution-based methods do during the elimination process.

The DLS algorithm is a method based on Ackermann's Lemma for eliminating second-order quantifiers from first-order logic formulas. The method was developed by [DLS97], as an extension of the algorithm given by [Sza93].

The input of DLS is exactly the same with that of SCAN. Once a derivation (of DLS) is done, DLS either returns a set of formulas not containing any existential second-order quantifiers (if the derivation is successful), or reports failure (if the result still contains existential second-order quantifiers; in this case we say the derivation is unsuccessful or fails). The failure of a derivation does not mean that the second-order logic formulas are not reducible to first-order logic formulas, because as we mentioned earlier, this problem is undecidable.

Conradie has generalised a set of necessary and sufficient syntactic conditions on the formulas for the success of DLS [Con06]. It is shown that DLS is complete for all modal Sahlqvist formulas and inductive formulas. The class of inductive formulas is a large syntactic class defined by Goranko and Vakarelov [GV06], which extends the class of Sahlqvist formulas and subsumes the class of the polyadic Sahlqvist formulas found by Rijke and Venema [dRV95].

In [NS98], DLS has been extended to DLS* which is based on a generalisation of Ackermann's Lemma, referred to as *generalised Ackermann's Lemma*. This generalised Ackermann's Lemma has been formulated as the following theorem.

Theorem 3.2.2 (Generalised Ackermann's Lemma)**.** *Let P be a predicate variable and $\alpha(\overline{x},\overline{z})$, $\beta(P)$ be first-order formulas (without second-order quantification), where the number of distinct variables in $\overline{x}$ is equal to the arity of P and $\alpha(P,\overline{x},\overline{z})$ is positive w.r.t. P.*

If $\beta(P)$ is positive w.r.t. P, then

$$\exists P\{\forall\overline{x}[P(\overline{x})\rightarrow\alpha(P,\overline{x},\overline{z})]\wedge\beta(P)\}\equiv\beta(P)^{P(\overline{x})}_{\mathrm{GFP}P(\overline{x}).\alpha(P,\overline{x},\overline{z})}$$

If $\beta(P)$ is negative w.r.t. P, then

$$\exists P\{\forall\overline{x}[\alpha(P,\overline{x},\overline{z})\rightarrow P(\overline{x})]\wedge\beta(P)\}\equiv\beta(P)^{P(\overline{x})}_{\mathrm{LFP}P(\overline{x}).\alpha(P,\overline{x},\overline{z})}.$$

By $\beta(P)^{P(\overline{x})}_{\alpha(P,\overline{x},\overline{z})}$, we denote the expression obtained from $\beta(P)$ by replacing all occurrences of $P(\overline{x})$ by $\alpha(P,\overline{x},\overline{z})$.

The generalised Ackermann's Lemma allows $\alpha(P, \overline{x}, \overline{z})$ to contain the symbol to be eliminated, which is P in the theorem. As a consequence, the resulting formulas are not expressible in first-order logic but are expressible in fixpoint logics. The DLS* algorithm allows existential second-order variables to be eliminated in cases where positive and negative occurrences of the variable are not properly separated.

SCAN and DLS can be used in the area of correspondence theory for modal logics. In particular, they can be used to compute first-order correspondence properties for modal axioms. The area of correspondence theory was initiated by Sahlqvist [Hen75] and then by van Benthem [vB75, vB76, vB89, vBDMP97]. Sahlqvist has (syntactically) defined a class of modal formulas which are reducible to first-order logic. This class is called the Sahlqvist Class and the formulas of this class are called Sahlqvist formulas. The idea is to translate the given modal axioms to first-order formulas (with second-order quantification). The translation is based on the close relationship between modal logics and first-order logic, as discussed in the previous chapter. Then we apply SCAN or DLS to the second-order formulas to eliminate existential second-order quantifiers. The results reflect the first-order correspondence properties of the given modal axioms. The DLS* algorithm can be used for modal fixpoint correspondence theory.

Based on DLS, Conradie et al. [CGV06a] developed a method called SQEMA for computing first-order correspondence properties for modal axioms. Unlike SCAN, DLS and DLS*, which work with first-order translations of modal axioms, SQEMA works directly with modal formulas. This is due to a modal version of Ackermann's Lemma developed in this work. It is known that the SQEMA algorithm is complete for Sahlqvist formulas and monadic inductive formulas and any formulas for which SQEMA is complete are canonical [CGV06a, CGV06b, CG08, CGV09, Con09, CGV10].

Based on SQEMA, Schmidt [Sch12] developed an alternative method called MSQEL for computing first-order correspondence properties for modal axioms. Unlike SQEMA which uses negation normal form as the normal form, MSQEL uses a special normal form which makes the detection of potential simplifications (i.e., syntactic contradictions, redundancies, and tautologies etc.) much easier. Another difference is that MSQEL exploits orderings, which are compatible with the elimination order, to restrict the inference process. This provides more control and reduces non-determinism in derivations, thereby improving the efficiency and success rates of the method. It is

known that MSQEL is complete for two syntactically well-defined classes $\mathcal{C}$ and $\mathcal{C}^{\succ}$ of modal formulas, which subsume the Sahlqvist class and the monadic inductive class.

3.3 Forgetting in Modal Logics

Because of the close relationship between modal logics and description logics, research on forgetting in modal logics can be of interest to the material presented in this thesis. Direct correspondences between the two logics include, e.g. correspondence between the multi-modal logic K and the description logic $\mathcal{ALC}$, correspondence between hybrid logics and description logics with nominals, and correspondence between the modal μ-calculus and description logics with fixpoint operators.

In modal logics, forgetting has often been studied under the name of *uniform interpolation* [Vis96, DH00, HM08], a notion related to the *Craig interpolation* [Cra57], but stronger. The definition of Craig interpolation can be formalised as follows.

Definition 3.3.1 (**Craig Interpolation**)**.** Given two first-order logic formulas F_1 and F_2, there is always a first-order logic formula F that only uses predicate symbols occurring both in F_1 and F_2 such that $F_1 \models F$ and $F \models F_2$. F is then referred to as the *Craig interpolant* of F_1 and F_2.

Based on Craig interpolation, Henkin [Hen63] proposed a stronger property, which is the notion nowadays commonly known as *uniform interpolation*.

Definition 3.3.2 (**Uniform Interpolation**)**.** A logic $\mathcal{L}$ has has the *uniform interpolation* property iff for any formula F in $\mathcal{L}$ and any set S of symbols, there exists a formula F^S such that for every $\mathcal{L}$ formula G containing only symbols in S we have $F^S \models G$ iff $F \models G$. F^S is then referred to as the *uniform interpolant* of F for S.

It it observed that the notion of uniform interpolation coincides with that of weak forgetting in the sense that the uniform interpolant of F for S is equivalent to the solution of weakly forgetting the symbols not in S from F. Therefore, these two notions are often considered as dual notions. Following Definition 3.3.2 and the correspondence between uniform interpolation and weak forgetting, we can infer that if a logic has the uniform interpolation property, then the solution of weak forgetting for this logic is always finite. Since the solution of weak forgetting for first-order logic cannot always

be represented finitely, it is easy to conclude that first-order logic does not have the uniform interpolation property.

Early work on uniform interpolation in modal logics has been primarily focused on checking if a modal logic has the uniform interpolation property. Known results are listed below in chronological order:

- Pitts [Pit92] proved that IPC has the uniform interpolation property.
- Shavrukov [Sha03] proved that GL has the uniform interpolation property.
- Ghilardi and Zawadowski [GZ95] proved that K has uniform interpolation property and S4 does not.
- Visser [Vis96] proved indepedently that K has uniform interpolation property, and so does S4GRz.
- D'Agostino and Hollenberg [DH96] proved that the μ-calculus has uniform interpolation property.
- Bilkova [Bíl07] proved that KT has uniform interpolation and K4 does not.
- Based on the results above, there are seven intermediate logics which have the uniform interpolation property, including IPC, Sm, GSc, KC, LC, Bd2 and CPC.

Recent work on uniform interpolation in modal logics has been primarily focused on developing practical methods for computing uniform interpolants. Bílková [Bíl07] has developed a method based on a sequent-calculus for computing uniform interpolant in the modal logic K and T. Kracht [Kra07] has developed an alternative method based on a tableaux algorithm for computing uniform interpolant in the modal logic K and T. D'Agostino and Lenzi [DL06] has developed a method for computing uniform interpolants in the μ-calculus. The first two methods are based on implicit transformations of the input into the standard disjunctive normal form, whereas the last method is based on explicit transformations of the input into a special form of disjunctive normal form. Pointed out by Herzig and Mengin [HM08] that disjunctive normal form is not natural representation for most applications, methods based on transformations of the input into conjunctive normal form have been developed as well. An example of these methods is the one developed by Herzig and Mengin in the same work for computing

uniform interpolants in the modal logic K. The method is based on a clausal resolution calculus presented in [EdC89].

Second-order quantifier elimination techniques can also be used to compute uniform interpolants for various modal logics. For example, Szalas [Sza02] has presented techniques that allow one to eliminate second-order quantifiers from second-order formulas formulated in the modal logic K and its first-order extension Q1K. The techniques are purely syntactical and independent of any particular underlying semantics of modal logics, and thus can be easily implemented. Two Ackermann-based techniques [CGV06a] and [Sch12] performs second-order quantifier elimination for the modal hybrid logic.

3.4 Forgetting in Description Logics

In description logics, forgetting has been investigated under the names of forgetting, e.g. [WWTP10, ZS15, ZS16, ZS17], uniform interpolation, e.g., [LW11, KS13d, Koo15, LK14], inseparability, e.g. [LW10, KLWW13, BKL$^+$16, BLR$^+$16], and conservative extensisons, e.g. [GLW06, LW07, JLM$^+$17].

Early work on forgetting in description logics has been primarily focused on ontologies specified in some lightweight description logics such as $\mathcal{EL}$ and $\mathcal{ALC}$. The first method for forgetting in ontologies and knowledge bases was presented in [WWTP08, WWTP10]. The method is based on the resolution calculus and handles ontologies specified in the description logic $\mathcal{EL}$, which is a lightweight description logic that admits sound and complete reasoning in polynomial time. Compared to other description logics, $\mathcal{EL}$ does not allow concepts to occur under quantifiers. For this reason, forgetting in the description logic $\mathcal{EL}$ can be easily solved as in propositional logics. While most of the subsequent work was focused on forgetting only in TBoxes, forgetting in RBoxes was also considered in this work.

An alternative method for forgetting and uniform interpolation in the description logic $\mathcal{EL}$ was presented in [KWW09]. Unlike the method of [WWTP08, WWTP10], this method extends the source language with role inclusions and domain & range restrictions [BLB08], which can help formulate sufficient (acyclicity) conditions for computing the uniform interpolants. This is because, as we discussed in the previous

section, not all logics have the uniform interpolation property. It is known that $\mathcal{EL}$ does not have the uniform interpolation property [KWW08]. By imposing acyclicity conditions on the input, the uniform interpolants can always be computed and represented by a finite set of axioms. The uniform interpolants computed by this method, in the worst case, are of exponential size w.r.t. the size of the input. This has verified that forgetting is indeed an inherently difficult problem, even for the lightweight description logic $\mathcal{EL}$. The method was evaluated with a prototypical implementation on two large benchmark $\mathcal{EL}$ ontologies. The evaluation showed that the method could forget large sets of symbols from ontologies. This can be expected because $\mathcal{EL}$ is a Horn logic, and the method has imposed acyclicity conditions on the input to ensure that the input contains no cyclic definitions. Both these make the problem expectedly much easier than forgetting in more expressive and more general ontologies.

Another method that handles $\mathcal{EL}$ with extensions was presented in [Nik11]. The method computes uniform interpolants for general $\mathcal{EL}$ ontologies with greatest fixpoint operators. A method with the same ability but without using fixpoints was presented in [NR12, NR14]. In particular, the former method is based on computing most specific superconcepts and most general subconcepts of the concepts in the signature and the latter method is based on proof-theory and regular tree grammars. It is also known from the latter work that uniform interpolants can be of triple exponential size w.r.t. the size of the input for $\mathcal{EL}$ ontologies.

Lutz et al. [LSW12] studied uniform interpolation in $\mathcal{EL}$-TBoxes. In particular, they developed an algorithm based on tree automata representations of $\mathcal{EL}$-TBoxes for deciding the existence of uniform interpolants in $\mathcal{EL}$-TBoxes. In this way, they proved that deciding the existence of uniform interpolants of $\mathcal{EL}$-TBoxes is ExpTime-complete. This work also provided a simpler proof for the known result that deciding conservative extensions of $\mathcal{EL}$-TBoxes is in ExpTime [LW10].

Methods for forgetting in the description logic $\mathcal{ALC}$ has also been developed. For example, a second-order quantifier elimination method for forgetting in $\mathcal{ALC}$ has been developed in [Sza06]. The method uses a generalisation of Ackermann's Lemma to description logics and allows for the elimination of both concept and role names. Two applications, namely, circumscribing concepts in description logics and approximating

terminological axioms are also described in this work. Wang et al. [WWT+09] developed a method for computing uniform interpolants in $\mathcal{ALC}$ knowledge bases, which under certain circumstances can also handle ABoxes. The method is comparable to the method of [DL06] which computes uniform interpolants in the μ-calculus, except that the latter requires the input ontology to be transformed into a specialised disjunctive normal form and the former incrementally transforms the input ontology into disjunctive normal form during the process of computing the uniform interpolants. In the latter way, the input TBox is represented as a single concept in disjunctive normal form, which can make the elimination of concept and role symbols rather easier. However, such an unusual representation can be an infinite representation, and if this has been the case, the method incrementally approximates the representation and checking if the uniform interpolant has been computed using equivalence tests between increments. The method of [WWT+09] has been optimised in [WWTZ10, WWT+14] by exploiting a tableau-based reasoning [MH09].

These two methods have however a disadvantage that they represent the input ontology in disjunctive normal form, which, as we mentioned above, is an unusual way to represent ontologies. Ontologies are naturally represented in conjunctive normal form (i.e., as a set of small axioms), rather than a large single axiom in disjunctive normal form. This disadvantage makes the methods not practical enough for real-world use. For this reason, Wang et al. [WWT+14] developed another method for uniform interpolation in $\mathcal{ALC}$, which preserves the natural representation of ontologies.

Disproving the claim made in [WWT+09, WWTZ10, WWT+14] that a uniform interpolant is at most of double exponential size w.r.t. to size of the input ontology, Lutz and Wolter [LW11] claimed that this can be triple exponential for a family of ontologies. Based on this finding, in the same work they modified the method of [WWT+09] to accommodate the cases where the size of the uniform interpolants are triple exponential in the size of the input ontology. In addition, they proved that deciding the existence of uniform interpolant in $\mathcal{ALC}$ can be 2ExpTime-complete.

A major drawback of the methods of [WWT+09, WWTZ10, WWT+14] and the method of [LW11] is that they are not goal-oriented method, which means that they cannot eliminate symbols in a flexible way as the user wishes.

The first goal-oriented method for forgetting and uniform interpolation was presented in [LK13b, LK14]. The method is based on a clausal resolution algorithm and computes uniform interpolants in $\mathcal{ALC}$-TBoxes. Since $\mathcal{ALC}$ does not have the uniform interpolation property, the authors introduced a depth-bounded version of their core algorithm to guarantee the termination on all input ontologies. The method has been implemented as a tool which has been evaluated on a corpus of real-world ontologies. The experimental results suggest that despite a high cimputational complexity, unform interpolants can be computed in many practical cases.

More recently, Koopmann and Schmidt [KS13d, KS13c] developed a practical method for computing uniform interpolants in the description logic $\mathcal{ALC}$. The method is a saturation approach based on the resolution calculus and can eliminate concept symbols from $\mathcal{ALC}$-TBoxes. The method introduces fixpoint operators in the target language to ensure that uniform interpolants can be finitely represented. Based on this method, Koopmann and Schmidt [KS13b] presented an improved method that can eliminate both concept and role symbols from ontologies specified in the description logic $\mathcal{ALCH}$ with fixpoint operators. This method has then been significantly extended to $\mathcal{SIF}$, $\mathcal{SHI}$ and $\mathcal{SHQ}$ for uniform interpolation [KS15c, Koo15, KS14], though in $\mathcal{SHQ}$ the method can only eliminate concept symbols. While most of this work has been focused on TBox and RBox uniform interpolation, practical methods for computing uniform interpolants for description logics $\mathcal{ALC}$ and $\mathcal{SHI}$ with ABoxes are described in [KS15a, Koo15]. All these methods have been collectively implemented as a tool called Lethe [KS15b], which performs uniform interpolation and related tasks for OWL ontologies using the description logics mentioned above. A thorough evaluation of Lethe has been conducted with a large number of real-world ontologies from different sources, and performance results have shown that Lethe can compute the uniform interpolants in most of the test cases within a reasonable period of time [Koo15].

Chapter 4

Concept Forgetting for $\mathcal{ALCOI}$

In this chapter, we introduced a practical method for forgetting concept symbols from ontologies expressible in the description logic $\mathcal{ALCOI}$, i.e., the basic description logic $\mathcal{ALC}$ extended with nominals and inverse roles. The method is based on a calculus, namely, ACK$^{\mathsf{C}}$, which exploits a generalisation of Ackermann's Lemma for description logics as the fundamental rules to eliminate single concept symbols. Being based on ACK$^{\mathsf{C}}$, the method is the only approach so far that provides support for forgetting concept symbols in description logics with nominals. The method is goal-oriented and incremental. It always terminates and is sound in the sense that the forgetting solution is equivalent to the original ontology up to the interpretations of the symbols that have been forgotten, possibly with the interpretations of the nominals that have been introduced during the forgetting process.

This chapter is an extension of our published work of [ZS15].

4.1 The Description Logic $\mathcal{ALCOI}$

In this section, we introduce the description logic $\mathcal{ALCOI}$, which is the language considered in this chapter. Let $\mathsf{N_C}$, $\mathsf{N_R}$ and $\mathsf{N_O}$ be countably infinite and pairwise disjoint sets of *concept symbols*, *role symbols* and *individual symbols* (aka *nominals*), respectively. *Roles* in $\mathcal{ALCOI}$ can be a role symbol $r \in \mathsf{N_R}$ or the inverse r^- of a role symbol r (an inverted role). *Concepts* in $\mathcal{ALCOI}$ have one of the following forms:

$$a \mid \top \mid \bot \mid A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists R.C \mid \forall R.C,$$

where $a \in \mathsf{N_O}$, $A \in \mathsf{N_C}$, C and D are arbitrary concepts and R is an arbitrary role. We assume w.l.o.g. that concepts and roles are equivalent relative to associativity and commutativity of $\sqcap$ and $\sqcup$, $\neg$ and $^-$ are involutions, and $\top$ is a unit w.r.t. $\sqcap$.

An $\mathcal{ALCOI}$-ontology is mostly assumed to be comprised of a TBox and an ABox. A *TBox* $\mathcal{T}$ is a finite set of *concept axioms* of the form $C \sqsubseteq D$ (*concept inclusion*) and the form $C \equiv D$ (*concept equivalence*), where C and D are concepts. We use $C \equiv D$ as an abbreviation of the pair $C \sqsubseteq D$ and $D \sqsubseteq C$. An *ABox* $\mathcal{A}$ is a finite set of *concept assertions* of the form $C(a)$ and *role assertions* of the form $R(a, b)$, where $a, b \in \mathsf{N_O}$, C is a concept and R is a role. In a description logic with nominals, ABox assertions can be equivalently expressed as TBox axioms, namely, $C(a)$ as $a \sqsubseteq C$ and $R(a, b)$ as $a \sqsubseteq \exists R.b$. Hence, in this chapter, we assume w.l.o.g. that an $\mathcal{ALCOI}$-ontology contains only TBox axioms.

The semantics of $\mathcal{ALCOI}$ is defined in terms of an *interpretation* $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, where $\Delta^{\mathcal{I}}$ is a non-empty set (the *domain of the interpretation*), and $\cdot^{\mathcal{I}}$ is the *interpretation function*, which assigns to every nominal $a \in \mathsf{N_O}$ a singleton $a^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, to every concept symbol $A \in \mathsf{N_C}$ a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, and to every role symbol $r \in \mathsf{N_R}$ a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The interpretation function $\cdot^{\mathcal{I}}$ is inductively extended to concepts and roles as follows:

$$\top^{\mathcal{I}} = \Delta^{\mathcal{I}} \qquad \bot^{\mathcal{I}} = \emptyset \qquad (\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$$
$$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}} \qquad (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$$
$$(\exists R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \exists y.(x, y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$$
$$(\forall R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \forall y.(x, y) \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$$
$$(R^-)^{\mathcal{I}} = \{(y, x) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (x, y) \in R^{\mathcal{I}}\},$$

A concept inclusion $C \sqsubseteq D$ is *true* in an interpretation $\mathcal{I}$, and we write $\mathcal{I} \models C \sqsubseteq D$, iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. A concept equivalence $C \equiv D$ is *true* in an interpretation $\mathcal{I}$, and we write $\mathcal{I} \models C \equiv D$, iff $C^{\mathcal{I}} \equiv D^{\mathcal{I}}$. $\mathcal{I}$ is a *model* of an ontology $\mathcal{O}$ iff every axiom in $\mathcal{O}$ is *true* in $\mathcal{I}$. In this case we write $\mathcal{I} \models \mathcal{O}$.

Let $A \in \mathsf{N_C}$ be a designated concept symbol. An axiom (clause) that contains an occurrence of A is called an *A-axiom* (*A-clause*). An occurrence of A is assumed to be *positive* (*negative*) in an A-axiom (A-clause) if it is under an *even* (*odd*) number of explicit and implicit negations. For instance, A is assumed to be positive in $\exists r.A$

and $C \sqsubseteq A$, and negative in $\neg\forall r.A$ and $A \sqsubseteq C$. An axiom (clause) C is assumed to be *positive* (*negative*) w.r.t. A is every occurrence of A in C is positive (negative). An ontology $\mathcal{O}$ of axioms is assumed to be *positive* (*negative*) w.r.t. A if every A-axiom in $\mathcal{O}$ is positive (negative). A set $\mathcal{N}$ of clauses is assumed to be *positive* (*negative*) w.r.t. A if every A-clause in $\mathcal{N}$ is positive (negative). An axiom (clause) that contains a positive occurrence of A is referred to as an A^+-axiom (A^+-clause). An axiom (clause) that contains a negative occurrence of A is referred to as an A^--axiom (A^--clause).

Next, we formalise our notion of concept forgetting for $\mathcal{ALCOI}$. By $\text{sig}_{\mathsf{C}}(X)$ we denote the set of the concept symbols occurring in X (*excluding nominals*), where X ranges over concepts, axioms, clauses, sets of axioms, and sets of clauses. Let $A \in \mathsf{N_C}$ be any concept symbol, and let $\mathcal{I}$ and $\mathcal{I}'$ be any interpretations. We say $\mathcal{I}$ and $\mathcal{I}'$ are *equivalent up to* A, or A-*equivalent*, if $\mathcal{I}$ and $\mathcal{I}'$ coincide but differ possibly in the interpretations of A. More generally, $\mathcal{I}$ and $\mathcal{I}'$ are *equivalent up to a set* Σ *of concept symbols*, or Σ-*equivalent*, if $\mathcal{I}$ and $\mathcal{I}'$ coincide but differ possibly in the interpretations of the symbols in Σ. This can be understood as follows: (i) $\mathcal{I}$ and $\mathcal{I}'$ have the same domain, i.e., $\Delta^{\mathcal{I}} = \Delta^{\mathcal{I}'}$, and interpret every role symbol and every individual symbol identically, i.e., $r^{\mathcal{I}} = r^{\mathcal{I}'}$ for every $r \in \mathsf{N_R}$ and $a^{\mathcal{I}} = a^{\mathcal{I}'}$ for every $a \in \mathsf{N_O}$; (ii) for every concept symbol $A \in \mathsf{N_C}$ not in Σ, $A^{\mathcal{I}} = A^{\mathcal{I}'}$.

Definition 4.1.1 (**Concept Forgetting for $\mathcal{ALCOI}$**)**.** Let $\mathcal{O}$ be an $\mathcal{ALCOI}$-ontology and let Σ be a subset of $\text{sig}_{\mathsf{C}}(\mathcal{O})$. An ontology $\mathcal{O}'$ is a *solution of forgetting* Σ *from* $\mathcal{O}$, iff the following conditions hold:

(i) $\text{sig}_{\mathsf{C}}(\mathcal{O}') \subseteq \text{sig}_{\mathsf{C}}(\mathcal{O}) \backslash \Sigma$, and

(ii) for any interpretation $\mathcal{I}$: $\mathcal{I} \models \mathcal{O}'$ iff $\mathcal{I}' \models \mathcal{O}$, for some interpretation $\mathcal{I}'$ Σ-equivalent to $\mathcal{I}$.

It follows from Definition 4.1.1 that: (i) the original ontology $\mathcal{O}$ and the forgetting solution $\mathcal{O}'$ are equivalent up to (the interpretations of) the symbols in Σ. Also (ii) forgetting solutions are unique up to logical equivalence, that is, if both $\mathcal{O}'$ and $\mathcal{O}''$ are solutions of forgetting Σ from $\mathcal{O}$, then they are logically equivalent.

In this chapter, Σ is always assumed to be a set of concept symbols to be forgotten. The symbol in Σ under current consideration for forgetting is referred to as the *pivot*

in our method. An axiom (clause) that contains an occurrence of the symbols in Σ is referred to as a Σ-*axiom* (Σ-*clause*). An axiom (clause) that contains an occurrence of the pivot is referred to as a *pivot-axiom* (*pivot-clause*). An axiom (clause) that contains a positive occurrence of the pivot is referred to as a *pivot$^+$-axiom* (*pivot$^+$-clause*). An axiom (clause) that contains a negative occurrence of the pivot is referred to as a *pivot$^-$-axiom* (*pivot$^-$-clause*).

4.2 Generalised Ackermann's Lemma

Given an ontology $\mathcal{O}$ and a set $\Sigma \subseteq \text{sig}_{\mathsf{C}}(\mathcal{O})$ of concept symbols, computing a solution of forgetting Σ from $\mathcal{O}$ can be reduced to the problem of eliminating single symbols in Σ. For example, forgetting $\Sigma = \{A, B\}$ from the ontology $\{A \sqsubseteq \exists r.B, B \sqsubseteq \forall r.C\}$ can be reduced to the problem of first eliminating $\{A\}$ from $\{A \sqsubseteq \exists r.B, B \sqsubseteq \forall r.C\}$, and then eliminating $\{B\}$ from the intermediate solution obtained from the preceding elimination (or the other way around, i.e., first eliminating $\{B\}$ from $\{A \sqsubseteq \exists r.B, B \sqsubseteq \forall r.C\}$, and then eliminating $\{A\}$ from the intermediate solution). Thus, an approach to eliminating single concept symbols from ontologies is required for concept forgetting.

Such an approach can be based on the use of a monotonicity property found in [Ack35], referred to as *Ackermann's Lemma*. The original Ackermann's Lemma has been used in the context of the second-order quantifier elimination problem and the modal correspondence theory problem for eliminating single predicate symbols from first-order logic formulas. For generic description logic-based ontologies, Ackermann's Lemma can be generalised as the following theorem.

Theorem 4.2.1 (**Generalised Ackermann's Lemma**). *Let $\mathcal{O}$ be an ontology that contains the axioms $C_1 \sqsubseteq A, ..., C_n \sqsubseteq A$, where $A \in \text{sig}_{\mathsf{C}}(\mathcal{O})$ is a concept symbol, and the C_i ($1 \leq i \leq n$) are concepts that do not contain A. If $\mathcal{O}\backslash\{C_1 \sqsubseteq A, ..., C_n \sqsubseteq A\}$ is negative w.r.t. A, then $\mathcal{O}\backslash\{C_1 \sqsubseteq A, ..., C_n \sqsubseteq A\}^A_{C_1 \sqcup ... \sqcup C_n}$ is a* solution of forgetting $\{A\}$ from $\mathcal{O}$ *(i.e., Conditions (i) and (ii) of Definition 4.1.1 hold), where $\mathcal{O}\backslash\{C_1 \sqsubseteq A, ..., C_n \sqsubseteq A\}$ denotes the ontology $\mathcal{O}$ excluding the axioms $C_1 \sqsubseteq A, ..., C_n \sqsubseteq A$, and $\mathcal{O}\backslash\{C_1 \sqsubseteq A, ..., C_n \sqsubseteq A\}^A_{C_1 \sqcup ... \sqcup C_n}$ denotes the ontology obtained from $\mathcal{O}$ by substituting $C_1 \sqcup ... \sqcup C_n$ for every occurrence of A in $\mathcal{O}\backslash\{C_1 \sqsubseteq A, ..., C_n \sqsubseteq A\}$.*

This theorem can be formulated as the following rule in mathematical presentation,

referred to as the Ackermann rule:

$$\frac{\mathcal{O}\backslash\{C_1 \sqsubseteq A, ..., C_n \sqsubseteq A\}, C_1 \sqsubseteq A, ..., C_n \sqsubseteq A}{\mathcal{O}\backslash\{C_1 \sqsubseteq A, ..., C_n \sqsubseteq A\}^A_{C_1 \sqcup ... \sqcup C_n}} \tag{4.1}$$

Ackermann's Lemma has a dual form generalised as the following (dual) theorem.

Theorem 4.2.2 (**Generalised Ackermann's Lemma Dual**). *Let $\mathcal{O}$ be an ontology that contains the axioms $A \sqsubseteq C_1, ..., A \sqsubseteq C_n$, where $A \in \text{sig}_C(\mathcal{O})$ is a concept symbol, and the C_i ($1 \leq i \leq n$) are concepts that do not contain A. If $\mathcal{O}\backslash\{A \sqsubseteq C_1, ..., A \sqsubseteq C_n\}$ is positive w.r.t. A, then $\mathcal{O}\backslash\{A \sqsubseteq C_1, ..., A \sqsubseteq C_n\}^A_{C_1 \sqcap ... \sqcap C_n}$ is a* solution of forgetting $\{A\}$ from $\mathcal{O}$ *(i.e., Conditions (i) and (ii) of Definition 4.1.1 hold), where $\mathcal{O}\backslash\{A \sqsubseteq C_1, ..., A \sqsubseteq C_n\}$ denotes the ontology $\mathcal{O}$ excluding the axioms $A \sqsubseteq C_1, ..., A \sqsubseteq C_n$, and $\mathcal{O}\backslash\{A \sqsubseteq C_1, ..., A \sqsubseteq C_n\}^A_{C_1 \sqcap ... \sqcap C_n}$ denotes the ontology obtained from $\mathcal{O}$ by substituting $C_1 \sqcap ... \sqcap C_n$ for every occurrence of A in $\mathcal{O}\backslash\{A \sqsubseteq C_1, ..., A \sqsubseteq C_n\}$.*

The Ackermann rule corresponding to this dual theorem is the following:

$$\frac{\mathcal{O}\backslash\{A \sqsubseteq C_1, ..., A \sqsubseteq C_n\}, A \sqsubseteq C_1, ..., A \sqsubseteq C_n}{\mathcal{O}\backslash\{A \sqsubseteq C_1, ..., A \sqsubseteq C_n\}^A_{C_1 \sqcap ... \sqcap C_n}} \tag{4.2}$$

In order to distinguish the two Ackermann rules, we refer to the rule in (4.1) as the Ackermann$^+$ rule, and the rule in (4.2) as the Ackermann$^-$ rule.

In these Ackermann rules (and any other transformation rules, rewrite rules and simplification rules represented in line-dividing form), the expressions above the line are referred to as the *premises* of the rule and those under the line as the *conclusion*. In particular, in the Ackermann$^+$ rule, we refer to the axioms $C_1 \sqsubseteq A, ..., C_n \sqsubseteq A$ as the *positive premises* of the rule, and the axioms in $\mathcal{O}\backslash\{C_1 \sqsubseteq A, ..., C_n \sqsubseteq A\}$ as the *negative premises* of the rule. In the Ackermann$^-$ rule, we refer to the axioms $A \sqsubseteq C_1, ..., A \sqsubseteq C_n$ as the *negative premises* of the rule, and the axioms in $\mathcal{O}\backslash\{C_1 \sqsubseteq A, ..., C_n \sqsubseteq A\}$ as the *positive premises* of the rule.

In the sequel, we show that the Ackermann rules are *sound* in the sense that the premises of the rules are equivalent to their conclusion up to the interpretation of the symbol that has been eliminated (i.e., up to the interpretation of the pivot).

Theorem 4.2.3. *The Ackermann rules preserve equivalence up to the interpretation of the pivot.*

Proof. In order to show that the Ackermann rules preserve equivalence up to the interpretation of the pivot, we have to show that for any interpretation $\mathcal{I}$, the conclusion of the Ackermann rules is true in $\mathcal{I}$ iff for some interpretation $\mathcal{I}'$ pivot-equivalent to $\mathcal{I}$, the premises of the rules are true in $\mathcal{I}'$.

We prove the Ackermann$^+$ rule. First, we prove the "$\Rightarrow$" direction. Let $\mathcal{I}$ be an arbitrary interpretation. Assume

$$\mathcal{I} \models \mathcal{O}\backslash\{C_1 \sqsubseteq A, ..., C_n \sqsubseteq A\}^A_{C_1 \sqcup ... \sqcup C_n}.$$

Let $\mathcal{I}'$ be an interpretation extending $\mathcal{I}$ by additionally assigning A a subset of $\Delta^{\mathcal{I}}$ such that $A^{\mathcal{I}'} = (C_1 \sqcup \ldots \sqcup C_n)^{\mathcal{I}'}$. Directly, we have

$$\mathcal{I}' \models \mathcal{O}\backslash\{C_1 \sqsubseteq A, ..., C_n \sqsubseteq A\}.$$

It is obvious that $C_1^{\mathcal{I}'} \subseteq (C_1 \sqcup \ldots \sqcup C_n)^{\mathcal{I}'}$, $\ldots$, $C_1^{\mathcal{I}'} \subseteq (C_1 \sqcup \ldots \sqcup C_n)^{\mathcal{I}'}$, which implies $C_1^{\mathcal{I}'} \subseteq A^{\mathcal{I}'}$, $\ldots$, $C_n^{\mathcal{I}'} \subseteq A^{\mathcal{I}'}$. This implies $\mathcal{I}' \models C_1 \sqsubseteq A$, $\ldots$, $\mathcal{I}' \models C_n \sqsubseteq A$.

Next, we prove the "$\Leftarrow$" direction. Assume

$$\mathcal{I}' \models \mathcal{O}\backslash\{C_1 \sqsubseteq A, ..., C_n \sqsubseteq A\}, C_1 \sqsubseteq A, ..., C_n \sqsubseteq A.$$

Directly, we have $(C_1 \sqcup \ldots \sqcup C_n)^{\mathcal{I}'} \subseteq A^{\mathcal{I}'}$. Since $\mathcal{O}\backslash\{C_1 \sqsubseteq A, ..., C_n \sqsubseteq A\}^A_{C_1 \sqcup ... \sqcup C_n}$ is negative w.r.t. A, in order to show

$$\mathcal{I} \models \mathcal{O}\backslash\{C_1 \sqsubseteq A, ..., C_n \sqsubseteq A\}^A_{C_1 \sqcup ... \sqcup C_n},$$

we have to show

$$(\mathcal{O}\backslash\{C_1 \sqsubseteq A, ..., C_n \sqsubseteq A\})^{\mathcal{I}'} \subseteq (\mathcal{O}\backslash\{C_1 \sqsubseteq A, ..., C_n \sqsubseteq A\}^A_{C_1 \sqcup ... \sqcup C_n})^{\mathcal{I}'}.$$

Equivalently, we show that for every axiom α in $\mathcal{O}\backslash\{C_1 \sqsubseteq A, ..., C_n \sqsubseteq A\}$, we have

$$\alpha^{\mathcal{I}'} \subseteq (\alpha^A_{C_1 \sqcup ... \sqcup C_n})^{\mathcal{I}'}.$$

For cases where $A \notin \mathrm{sig}_{\mathsf{C}}(\alpha)$, this always holds. For cases where $A \in \mathrm{sig}_{\mathsf{C}}(\alpha)$, we do the proof by induction.

Base Case: If $\alpha = \neg A$, then $\alpha^A_{C_1 \sqcup ... \sqcup C_n} = \neg(C_1 \sqcup ... \sqcup C_n)$. Because

$$(C_1 \sqcup \ldots \sqcup C_n)^{\mathcal{I}'} \subseteq A^{\mathcal{I}'},$$

we have

$$\Delta^{\mathcal{I}'} \backslash A^{\mathcal{I}'} \subseteq \Delta^{\mathcal{I}'} \backslash (C_1 \sqcup \ldots \sqcup C_n)^{\mathcal{I}'}.$$

Thus, we have

$$(\neg A)^{\mathcal{I}'} \subseteq (\neg (C_1 \sqcup \ldots \sqcup C_n))^{\mathcal{I}'},$$

which implies that

$$\alpha^{\mathcal{I}'} \subseteq {\alpha^{A}_{C_1 \sqcup \ldots \sqcup C_n}}^{\mathcal{I}'}.$$

Induction Hypothesis: $(\alpha \sqcap \beta)$ Suppose the statement holds for α and β, i.e.,

$$\alpha^{\mathcal{I}'} \subseteq {\alpha^{A}_{C_1 \sqcup \ldots \sqcup C_n}}^{\mathcal{I}'} \text{ and } \beta^{\mathcal{I}'} \subseteq {\beta^{A}_{C_1 \sqcup \ldots \sqcup C_n}}^{\mathcal{I}'}.$$

We show

$$(\alpha \sqcap \beta)^{\mathcal{I}'} \subseteq {(\alpha \sqcap \beta)^{A}_{C_1 \sqcup \ldots \sqcup C_n}}^{\mathcal{I}'}.$$

Because

$$\alpha^{\mathcal{I}'} \subseteq {\alpha^{A}_{C_1 \sqcup \ldots \sqcup C_n}}^{\mathcal{I}'} \text{ and } \beta^{\mathcal{I}'} \subseteq {\beta^{A}_{C_1 \sqcup \ldots \sqcup C_n}}^{\mathcal{I}'},$$

we have

$$\alpha^{\mathcal{I}'} \cap \beta^{\mathcal{I}'} \subseteq {\alpha^{A}_{C_1 \sqcup \ldots \sqcup C_n}}^{\mathcal{I}'} \cap {\beta^{A}_{C_1 \sqcup \ldots \sqcup C_n}}^{\mathcal{I}'}.$$

Therefore, we have

$$(\alpha \sqcap \beta)^{\mathcal{I}'} \subseteq {(\alpha \sqcap \beta)^{A}_{C_1 \sqcup \ldots \sqcup C_n}}^{\mathcal{I}'}.$$

$(\alpha \sqcup \beta)$ We show $(\alpha \sqcup \beta)^{\mathcal{I}'} \subseteq {(\alpha \sqcup \beta)^{A}_{C_1 \sqcup \ldots \sqcup C_n}}^{\mathcal{I}'}$. Because

$$\alpha^{\mathcal{I}'} \subseteq {\alpha^{A}_{C_1 \sqcup \ldots \sqcup C_n}}^{\mathcal{I}'} \text{ and } \beta^{\mathcal{I}'} \subseteq {\beta^{A}_{C_1 \sqcup \ldots \sqcup C_n}}^{\mathcal{I}'},$$

we have

$$\alpha^{\mathcal{I}'} \cup \beta^{\mathcal{I}'} \subseteq {\alpha^{A}_{C_1 \sqcup \ldots \sqcup C_n}}^{\mathcal{I}'} \cup {\beta^{A}_{C_1 \sqcup \ldots \sqcup C_n}}^{\mathcal{I}'}.$$

Therefore, we have

$$(\alpha \sqcup \beta)^{\mathcal{I}'} \subseteq {(\alpha \sqcup \beta)^{A}_{C_1 \sqcup \ldots \sqcup C_n}}^{\mathcal{I}'}.$$

$(\exists R.\alpha)$ We show $(\exists R.\alpha)^{\mathcal{I}'} \subseteq {(\exists R.\alpha)^{A}_{C_1 \sqcup \ldots \sqcup C_n}}^{\mathcal{I}'}$. We do the proof by contradiction. Suppose there exists an element $d \in \Delta^{\mathcal{I}'}$ such that

$$d \notin {(\exists R.\alpha)^{A}_{C_1 \sqcup \ldots \sqcup C_n}}^{\mathcal{I}'}.$$

This means that for every $y \in \Delta^{\mathcal{I}'}$, we have

$$(d, y) \notin R^{\mathcal{I}'} \text{ or } y \notin {\alpha^{A}_{C_1 \sqcup \ldots \sqcup C_n}}^{\mathcal{I}'}.$$

Since $\exists R.\alpha$ is true in $\mathcal{I}'$, for every $x \in \Delta^{\mathcal{I}'}$, there exists an element $d' \in \Delta^{\mathcal{I}'}$ such that

$$(x, d') \in R^{\mathcal{I}'} \text{ and } d' \in \alpha^{\mathcal{I}'}.$$

$(x, d') \in R^{\mathcal{I}'}$ contradicts with $(d, y) \notin R^{\mathcal{I}'}$. Since $\alpha^{\mathcal{I}'} \subseteq {\alpha^A_{C_1 \sqcup \ldots \sqcup C_n}}^{\mathcal{I}'}$, we have

$$d' \in {\alpha^A_{C_1 \sqcup \ldots \sqcup C_n}}^{\mathcal{I}'},$$

which contradicts with

$$y \notin {\alpha^A_{C_1 \sqcup \ldots \sqcup C_n}}^{\mathcal{I}'}.$$

$(\forall R.\alpha)$ We show $(\forall R.\alpha)^{\mathcal{I}'} \subseteq {(\forall R.\alpha)^A_{C_1 \sqcup \ldots \sqcup C_n}}^{\mathcal{I}'}$. We do the proof by contradiction. Suppose there exists an element $d \in \Delta^{\mathcal{I}'}$ such that

$$d \notin {(\forall R.\alpha)^A_{C_1 \sqcup \ldots \sqcup C_n}}^{\mathcal{I}'}.$$

This means that there exists an element $d' \in \Delta^{\mathcal{I}'}$ such that

$$(d, d') \in R^{\mathcal{I}'} \text{ and } d' \notin {\alpha^A_{C_1 \sqcup \ldots \sqcup C_n}}^{\mathcal{I}'}.$$

Since $\forall R.\alpha$ is true in $\mathcal{I}'$, for every $x, y \in \Delta^{\mathcal{I}'}$, we have

$$(x, y) \notin R^{\mathcal{I}'} \text{ or } y \in \alpha^{\mathcal{I}'}.$$

$(x, y) \notin R^{\mathcal{I}'}$ contradicts with $(d, d') \in R^{\mathcal{I}'}$. Since $\alpha^{\mathcal{I}'} \subseteq {\alpha^A_{C_1 \sqcup \ldots \sqcup C_n}}^{\mathcal{I}'}$, we have

$$y \in {\alpha^A_{C_1 \sqcup \ldots \sqcup C_n}}^{\mathcal{I}'},$$

which contradicts with

$$d' \notin {\alpha^A_{C_1 \sqcup \ldots \sqcup C_n}}^{\mathcal{I}'}.$$

We prove the Ackermann$^-$ rule. First, we prove the "$\Leftarrow$" direction. Let $\mathcal{I}$ be an arbitrary interpretation. Assume

$$\mathcal{I} \models \mathcal{O} \backslash \{A \sqsubseteq C_1, ..., A \sqsubseteq C_n\}^A_{C_1 \sqcap \ldots \sqcap C_n}.$$

Let $\mathcal{I}'$ be an interpretation extending $\mathcal{I}$ by additionally assigning A a subset of $\Delta^{\mathcal{I}}$ such that $A^{\mathcal{I}'} = (C_1 \sqcap \ldots \sqcap C_n)^{\mathcal{I}'}$. Directly, we have

$$\mathcal{I}' \models \mathcal{O} \backslash \{A \sqsubseteq C_1, ..., A \sqsubseteq C_n\}.$$

It is obvious that

$$(C_1 \sqcap \ldots \sqcap C_n)^{\mathcal{I}'} \subseteq C_1^{\mathcal{I}'}, \ldots, (C_1 \sqcap \ldots \sqcap C_n)^{\mathcal{I}'} \subseteq C_n^{\mathcal{I}'}.$$

Thus, we have

$$A^{\mathcal{I}'} \subseteq {C_1}^{\mathcal{I}'}, \ldots, A^{\mathcal{I}'} \subseteq {C_n}^{\mathcal{I}'}.$$

Equivalently, we have

$$\mathcal{I}' \models A \sqsubseteq C_1, \ldots, \mathcal{I}' \models A \sqsubseteq C_n.$$

Then, we prove the "⇒" direction. Assume

$$\mathcal{I}' \models \mathcal{O}\backslash\{A \sqsubseteq C_1, ..., A \sqsubseteq C_n\}, A \sqsubseteq C_1, ..., A \sqsubseteq C_n.$$

Directly, we have

$$A^{\mathcal{I}'} \subseteq (C_1 \sqcap \ldots \sqcap C_n)^{\mathcal{I}'}.$$

Since $\mathcal{O}\backslash\{A \sqsubseteq C_1, ..., A \sqsubseteq C_n\}^A_{C_1 \sqcap ... \sqcap C_n}$ is positive w.r.t. A, in order to show

$$\mathcal{I} \models \mathcal{O}\backslash\{A \sqsubseteq C_1, ..., A \sqsubseteq C_n\}^A_{C_1 \sqcap ... \sqcap C_n},$$

we have to show

$$(\mathcal{O}\backslash\{A \sqsubseteq C_1, ..., A \sqsubseteq C_n\})^{\mathcal{I}'} \subseteq (\mathcal{O}\backslash\{A \sqsubseteq C_1, ..., A \sqsubseteq C_n\}^A_{C_1 \sqcap ... \sqcap C_n})^{\mathcal{I}'}.$$

Equivalently, we show that for every axiom α in $\mathcal{O}\backslash\{A \sqsubseteq C_1, ..., A \sqsubseteq C_n\}$, we have

$$\alpha^{\mathcal{I}'} \subseteq {\alpha^A_{C_1 \sqcap ... \sqcap C_n}}^{\mathcal{I}'}.$$

For cases where $A \notin \mathrm{sig}_{\mathsf{C}}(\alpha)$, this always holds. For cases where $A \in \mathrm{sig}_{\mathsf{C}}(\alpha)$, we do the proof by induction.

Base Case: If $\alpha = A$, then $\alpha^A_{C_1 \sqcap ... \sqcap C_n} = C_1 \sqcap ... \sqcap C_n$. Because

$$A^{\mathcal{I}'} \subseteq (C_1 \sqcap \ldots \sqcap C_n)^{\mathcal{I}'},$$

we have

$$\alpha^{\mathcal{I}'} \subseteq {\alpha^A_{C_1 \sqcap ... \sqcap C_n}}^{\mathcal{I}'}.$$

Induction Hypothesis: $(\alpha \sqcap \beta)$ Suppose the statement holds for the concepts α and β, i.e., $\alpha^{\mathcal{I}'} \subseteq {\alpha^A_{C_1 \sqcap ... \sqcap C_n}}^{\mathcal{I}'}$ and $\beta^{\mathcal{I}'} \subseteq {\beta^A_{C_1 \sqcap ... \sqcap C_n}}^{\mathcal{I}'}$. We show

$$(\alpha \sqcap \beta)^{\mathcal{I}'} \subseteq {(\alpha \sqcap \beta)^A_{C_1 \sqcap ... \sqcap C_n}}^{\mathcal{I}'}.$$

Because

$$\alpha^{\mathcal{I}'} \subseteq {\alpha^{A}_{C_1 \sqcap \ldots \sqcap C_n}}^{\mathcal{I}'} \text{ and } \beta^{\mathcal{I}'} \subseteq {\beta^{A}_{C_1 \sqcap \ldots \sqcap C_n}}^{\mathcal{I}'},$$

we have

$$\alpha^{\mathcal{I}'} \cap \beta^{\mathcal{I}'} \subseteq {\alpha^{A}_{C_1 \sqcap \ldots \sqcap C_n}}^{\mathcal{I}'} \cap {\beta^{A}_{C_1 \sqcap \ldots \sqcap C_n}}^{\mathcal{I}'}.$$

Therefore, we have

$$(\alpha \sqcap \beta)^{\mathcal{I}'} \subseteq {(\alpha \sqcap \beta)^{A}_{C_1 \sqcap \ldots \sqcap C_n}}^{\mathcal{I}'}.$$

$(\alpha \sqcup \beta)$ We show $(\alpha \sqcup \beta)^{\mathcal{I}'} \subseteq {(\alpha \sqcup \beta)^{A}_{C_1 \sqcap \ldots \sqcap C_n}}^{\mathcal{I}'}$. Because

$$\alpha^{\mathcal{I}'} \subseteq {\alpha^{A}_{C_1 \sqcap \ldots \sqcap C_n}}^{\mathcal{I}'} \text{ and } \beta^{\mathcal{I}'} \subseteq {\beta^{A}_{C_1 \sqcap \ldots \sqcap C_n}}^{\mathcal{I}'},$$

we have

$$\alpha^{\mathcal{I}'} \cup \beta^{\mathcal{I}'} \subseteq {\alpha^{A}_{C_1 \sqcap \ldots \sqcap C_n}}^{\mathcal{I}'} \cup {\beta^{A}_{C_1 \sqcap \ldots \sqcap C_n}}^{\mathcal{I}'}.$$

Therefore, we have

$$(\alpha \sqcup \beta)^{\mathcal{I}'} \subseteq {(\alpha \sqcup \beta)^{A}_{C_1 \sqcap \ldots \sqcap C_n}}^{\mathcal{I}'}.$$

$(\exists R.\alpha)$ We show $(\exists R.\alpha)^{\mathcal{I}'} \subseteq {(\exists R.\alpha)^{A}_{C_1 \sqcap \ldots \sqcap C_n}}^{\mathcal{I}'}$. We do the proof by contradiction. Suppose there exists an element $d \in \Delta^{\mathcal{I}'}$ such that $d \notin {(\exists R.\alpha)^{A}_{C_1 \sqcap \ldots \sqcap C_n}}^{\mathcal{I}'}$. This means that for every $y \in \Delta^{\mathcal{I}'}$, we have

$$(d, y) \notin R^{\mathcal{I}'} \text{ or } y \notin {\alpha^{A}_{C_1 \sqcap \ldots \sqcap C_n}}^{\mathcal{I}'}.$$

Since $\exists R.\alpha$ is true in $\mathcal{I}'$, for every $x \in \Delta^{\mathcal{I}'}$, there exists an element $d' \in \Delta^{\mathcal{I}'}$ such that

$$(x, d') \in R^{\mathcal{I}'} \text{ and } d' \in \alpha^{\mathcal{I}'}.$$

$(x, d') \in R^{\mathcal{I}'}$ contradicts with $(d, y) \notin R^{\mathcal{I}'}$. Since

$$\alpha^{\mathcal{I}'} \subseteq {\alpha^{A}_{C_1 \sqcap \ldots \sqcap C_n}}^{\mathcal{I}'},$$

we have

$$d' \in {\alpha^{A}_{C_1 \sqcap \ldots \sqcap C_n}}^{\mathcal{I}'},$$

which contradicts with

$$y \notin {\alpha^{A}_{C_1 \sqcap \ldots \sqcap C_n}}^{\mathcal{I}'}.$$

$(\forall R.\alpha)$ We show $(\forall R.\alpha)^{\mathcal{I}'} \subseteq {(\forall R.\alpha)^{A}_{C_1 \sqcap \ldots \sqcap C_n}}^{\mathcal{I}'}$. We do the proof by contradiction. Suppose there exists an element $d \in \Delta^{\mathcal{I}'}$ such that $d \notin {(\forall R.\alpha)^{A}_{C_1 \sqcap \ldots \sqcap C_n}}^{\mathcal{I}'}$. This means that there exists an element $d' \in \Delta^{\mathcal{I}'}$ such that

$$(d, d') \in R^{\mathcal{I}'} \text{ and } d' \notin {\alpha^{A}_{C_1 \sqcap \ldots \sqcap C_n}}^{\mathcal{I}'}.$$

Since $\forall R.\alpha$ is true in $\mathcal{I}'$, for every $x, y \in \Delta^{\mathcal{I}'}$, we have

$$(x, y) \notin R^{\mathcal{I}'} \text{ or } y \in \alpha^{\mathcal{I}'}.$$

$(x, y) \notin R^{\mathcal{I}'}$ contradicts with $(d, d') \in R^{\mathcal{I}'}$. Since

$$\alpha^{\mathcal{I}'} \subseteq {\alpha^{A}_{C_1 \sqcap \ldots \sqcap C_n}}^{\mathcal{I}'},$$

we have

$$y \in {\alpha^{A}_{C_1 \sqcap \ldots \sqcap C_n}}^{\mathcal{I}'},$$

which contradicts with

$$d' \notin {\alpha^{A}_{C_1 \sqcap \ldots \sqcap C_n}}^{\mathcal{I}'}.$$

□

4.3 The Normalisation

Most reasoning methods require the input ontology to be normalised in some way, so that further simplifications (i.e., syntactic equivalences, contradictions and tautologies) can be detected, and moreover, the inference rules for forgetting can be generalised. Our forgetting method requires the input ontology to be transformed into (clausal) normal form (i.e., a set of clauses). In this section, we introduce the notion of *clausal normal form*, and describe a number of transformation rules for transforming an ontology into clausal normal form. Since an $\mathcal{ALCOI}$-ontology is assumed to contain only TBox axioms, the notions defined in this chapter are based on TBox axioms.

Definition 4.3.1. A *literal* in $\mathcal{ALCOI}$ is a concept of one of the following forms:

$$a \mid \neg a \mid A \mid \neg A \mid \exists R.C \mid \forall R.C,$$

where $a \in \mathsf{N_O}$, $A \in \mathsf{N_C}$, C is a concept in negation normal form and R is an arbitrary role. A concept is in *negation normal form*, if the negation operator is only applied to concept symbols and nominals.

A concept can be transformed into an equivalent one in negation normal form by pushing the negation operators inwards using a combination of De Morgan's laws, the

De Morgan's laws:

$$\begin{array}{lcl} \neg(C \sqcap D) & \Longrightarrow & \neg C \sqcup \neg D \\ \neg(C \sqcup D) & \Longrightarrow & \neg C \sqcap \neg D \end{array}$$

Duality between $\exists$- and $\forall$-restrictions:

$$\begin{array}{lcl} \neg\exists r.C & \Longrightarrow & \forall r.\neg C \\ \neg\forall r.C & \Longrightarrow & \exists r.\neg C \end{array}$$

Duality between $\top$ and $\bot$:

$$\begin{array}{lcl} \neg\top & \Longrightarrow & \bot \\ \neg\bot & \Longrightarrow & \top \end{array}$$

Double negation elimination:

$$\neg\neg C \quad \Longrightarrow \quad C$$

Figure 4.1: Transformations of concepts into negation normal form

duality between existential and universal role restrictions, the duality between the top and the bottom concepts, and the double negation elimination (see Figure 4.1).

De Morgan's laws refer to a pair of transformation rules that allow the concepts of conjunctions and disjunctions to be expressed purely in terms of each other via negations. The duality between existential and universal role restrictions refers to a pair of transformation rules that allow the concepts of existential and universal role restrictions to be expressed purely in terms of each other via negations. The duality between the top and the bottom concepts refers to a pair of transformation rules that allow the top and the bottom concepts to be expressed purely in terms of each other via negations. The double negation elimination refers to a transformation rule that eliminates double negations. All these rules are standard negation normal form transformation rules which preserve logical equivalence. They are general enough to guarantee the success of the transformation of any $\mathcal{ALCOI}$-concepts into negation normal form. In addition, it is easy to check that the transformation of a concept into negation normal form can be performed in linear time in the size of the concept.

Theorem 4.3.2. *Using the transformation rules in Figures 4.1, any $\mathcal{ALCOI}$-concept can be transformed into a logically equivalent one in negation normal form.*

Proof. Since each of these rules preserves logical equivalence. We prove this by showing that there is no gap in the scope of the transformation rules for transforming $\mathcal{ALCOI}$-concepts into negation normal form. Due to the syntax restrictions of $\mathcal{ALCOI}$-concepts, the negation operator can be applied to a nominal, a concept symbol, the top concept, the bottom concept, a concept of a negation, a concept of a conjunction, a concept of a disjunction, a concept of an existential role restriction and a concept of a universal role restriction. For the first two cases, the concepts are already in negation normal form. For each of the other cases, we have a corresponding transformation rule in Figure 4.1 that allows the concept to be transformed into an equivalent one in negation normal form. □

Definition 4.3.3. A *clause* in $\mathcal{ALCOI}$ is a concept of the following form:

$$L_1 \sqcup \ldots \sqcup L_n,$$

where the L_i $(1 \leq i \leq n)$ are arbitrary $\mathcal{ALCOI}$-literals. In plain English, an $\mathcal{ALCOI}$-clause is a finite disjunction of $\mathcal{ALCOI}$-literals. The empty clause is denoted by $\bot$ and represents a contradiction. An axiom is in *clausal normal form* if it is an $\mathcal{ALCOI}$-clause. An ontology $\mathcal{O}$ is in *clausal normal form* if every axiom in $\mathcal{O}$ is an $\mathcal{ALCOI}$-clause.

Distributivity law:

$$C \sqcup (D_1 \sqcap D_2) \qquad \Longrightarrow \qquad C \sqcup D_1, C \sqcup D_2$$

Figure 4.2: Transformation of concepts into clausal normal form

The distributivity law in Figure 4.2 distributes disjunctions over conjunctions. A concept can be transformed into clausal normal form by applying the transformation rules in Figure 4.1, as well as the distributivity law.

Given an $\mathcal{ALCOI}$-ontology $\mathcal{O}$, not all concepts in $\mathcal{O}$ are in clausal normal form. Transforming concepts into clausal normal form is based on the use of the transformation rule in Figure 4.2, as well as the transformation rules in Figure 4.1. These rules are standard clausal normal form transformation rules, which preserve logical equivalence in the transformation, that is, the left-hand-side expression of the '$\Longrightarrow$' relation is logically equivalent to the right-hand-side expression of this relation.

Theorem 4.3.4. *Using the transformation rules in Figures 4.1 and 4.2, any $\mathcal{ALCOI}$-concept can be transformed into an logically equivalent one in clausal normal form.*

Example 4.3.5. Consider the following ontology $\mathcal{O}$:

1. $\neg(A \sqcap B) \sqcup C$
2. $\neg\exists r.B \sqcup A$
3. $\neg a \sqcup \neg\forall s.\neg B$
4. $\neg\forall r.C \sqcup \forall r.A$
5. $\neg\exists r.C \sqsubseteq \exists r.A$

Observe that Clauses 1 – 5 are not in clausal normal form. By applying the transformation rules in Figures 4.1 and 4.2, $\mathcal{O}$ is transformed into the following set of clauses in clausal normal form:

6. $\neg A \sqcup \neg B \sqcup C$
7. $\forall r.\neg B \sqcup A$
8. $\neg a \sqcup \exists s.B$
9. $\forall r.\neg C \sqcup \forall r.A$
10. $\exists r.\neg C \sqcup \exists r.A$

From this section onwards, we assume w.l.o.g. that an $\mathcal{ALCOI}$-ontology $\mathcal{O}$ is a set of axioms or a set of clauses. We sometimes use $\mathcal{N}$ to explicitly denote a set of clauses.

4.4 The Calculus – ACK$^{\mathsf{C}}$

In this section, we introduce a dedicated calculus, namely, ACK$^{\mathsf{C}}$, for eliminating a single concept symbol from a set of $\mathcal{ALCOI}$-clauses. The calculus is based on a generalisation of Ackermann's Lemma, and works directly on description logic expressions. In particular, the calculus has four key ingredients:

(i) transformation of the present clause set (in clausal normal form) into *pivot-reduced form*,

(ii) a pair of Ackermann$^{\mathsf{C}}$ rules,

(iii) a pair of Purify$^{\mathsf{C}}$ rules, and

(iv) a set of simplification rules.

The Ackermann$^{\mathsf{C}}$ rules reflect the generalisation of Ackermann's Lemma and allow a single concept symbol to be eliminated from a clause set in different pivot-reduced forms. The Purify$^{\mathsf{C}}$ rules are special cases of the Ackermann$^{\mathsf{C}}$ rules and allow a single concept symbol to be eliminated from a clause set where the pivot occurs only positively or only negatively. The simplification rules, as their name indicates, transform the current clauses into equivalent ones with simpler representation, so that the clause set is more accessible to the method. This is done for the efficiency of the method. We will describe these four ingredients in detail in the following subsections.

4.4.1 The Ackermann$^{\mathsf{C}}$ Rules

In the previous section, we have gained some insight into the generalised Ackermann's Lemma for generic description logic-based ontologies. We have proved that the generalised Ackermann's Lemma and the dual lemma can be used to compute the solution of eliminating a single concept symbol from a set of TBox axioms (i.e., the premises in 4.1 and 4.2). We incorporate these two lemmas in our calculus Ack$^{\mathsf{C}}$ as two inference rules, referred to as the Ackermann$^{\mathsf{C}}$ rules. In particular, the Ackermann$^{\mathsf{C}}$ rules are used to eliminate a single concept symbol from a set of clauses in pivot-reduced form. The rules are shown in Figure 4.3. They are represented in clausal form for consistency in presentation, since Ack$^{\mathsf{C}}$ works with axioms in clausal form.

The fundamental idea of the Ackermann$^{\mathsf{C}}$ rules is based on a notion of "*substitution*", which can informally yet intuitively be understood as follows: given a set $\mathcal{N}$ of clauses with $A \in \mathrm{sig}_{\mathsf{C}}(\mathcal{N})$ being the pivot, if there exists a concept C such that C does not contain A and C *defines* A w.r.t. $\mathcal{N}$, then we can substitute this *definition* for every occurrence of A in $\mathcal{N}$ (A is thus eliminated from $\mathcal{N}$). In the Ackermann$^{\mathsf{C},+}$ rule, the definition is the disjunctive concept $\neg C_1 \sqcup \ldots \sqcup \neg C_n$, and in the Ackermann$^{\mathsf{C},-}$ rule, the definition is the conjunctive concept $C_1 \sqcap \ldots \sqcap C_n$.

A crucial task in Ackermann-based methods, therefore, is to compute the *definition* of the pivot w.r.t. the present clause set, that is, to reformulate all pivot^{+}-clauses

Ackermann$^{\mathsf{C},+}$

$$\frac{\mathcal{N}\backslash\{C_1 \sqcup A, ..., C_n \sqcup A\}, C_1 \sqcup A, ..., C_n \sqcup A}{\mathcal{N}\backslash\{C_1 \sqcup A, ..., C_n \sqcup A\}^{A}_{\neg C_1 \sqcup ... \sqcup \neg C_n}}$$

provided: (i) $A \in \mathrm{sig}_{\mathsf{C}}(\mathcal{N})$ is the pivot,
(ii) A does not occur in the C_i $(1 \leq i \leq n)$, and
(iii) $\mathcal{N}\backslash\{C_1 \sqcup A, ..., C_n \sqcup A\}$ is negative w.r.t. A.

Ackermann$^{\mathsf{C},-}$

$$\frac{\mathcal{N}\backslash\{\neg A \sqcup C_1, ..., \neg A \sqcup C_n\}, \neg A \sqcup C_1, ..., \neg A \sqcup C_n}{\mathcal{N}\backslash\{\neg A \sqcup C_1, ..., \neg A \sqcup C_n\}^{A}_{C_1 \sqcap ... \sqcap C_n}}$$

provided: (i) $A \in \mathrm{sig}_{\mathsf{C}}(\mathcal{N})$ is the pivot,
(ii) A does not occur in the C_i $(1 \leq i \leq n)$, and
(iii) $\mathcal{N}\backslash\{\neg A \sqcup C_1, ..., \neg A \sqcup C_n\}$ is positive w.r.t. A.

Figure 4.3: The Ackermann$^{\mathsf{C}}$ rules for eliminating $A \in \mathsf{N_C}$ from a set of clauses

with every positive occurrence of the pivot being in the form $\neg C \sqcup A$ (or dually, to reformulate all pivot$^-$-clauses with every negative occurrence of the pivot being in the form $\neg A \sqcup C$), where $A \notin \mathrm{sig}(C)$. More specifically, this means reformulating all pivot$^+$-clauses with (i) every pivot$^+$-clause containing a single occurrence of the pivot, and (ii) the pivot occurring at the top level of the clause (or dually, reformulating all pivot$^-$-clauses with (i) every pivot$^-$-clause containing a single occurrence of the negated pivot, and (ii) the negated pivot occurring at the top level of the clause). Note that if one wants to apply the Ackermann$^{\mathsf{C},+}$ rule to eliminate the pivot, then only the pivot$^+$-clauses need to be reformulated into the required form, and the pivot$^-$-clauses remain unchanged. On the other hand, if one wants to apply the Ackermann$^{\mathsf{C},-}$ rule to eliminate the pivot, then only the pivot$^-$-clauses need to be reformulated into the required form, and the pivot$^+$-clauses remain unchanged.

In the remainder of this chapter, we assume w.l.o.g. that any present concepts are in negation normal form. We refer to a positive occurrence of the pivot as *the pivot*, and a negative occurrence of the pivot as *the negated pivot*.

Given a set $\mathcal{N}$ of clauses with $A \in \mathrm{sig}_{\mathsf{C}}(\mathcal{N})$ being the pivot, the only cases where

the pivot (or the negated pivot) does not occur at the top level of a clause are the cases where the pivot (or the negated pivot) occurs below an existential or universal role restriction. For instance, in the concept $\neg B \sqcup \exists r.A$, the pivot A occurs below the existential role restriction $\exists r$, and in the concept $\forall r.\neg A \sqcup B$, the negated pivot $\neg A$ occurs below the universal role restriction $\forall r$. In these cases, the Ackermann$^{\mathsf{C}}$ rules cannot be applied to the pivot-clauses (i.e., the premises) to eliminate the pivot.

An intuitive solution is to rewrite such pivot-clauses into equivalent ones with the pivot (or the negated pivot) occurring at the top level, and in addition, not below any role restrictions, that is, to move the pivot (or the negated pivot) outside of the preceded role restrictions. In the next subsection, we introduce two pairs of rewrite$^{\mathsf{C}}$ rules for the "surfacing" of the pivot (or the negated pivot) from role restrictions.

4.4.2 Transformation into Pivot-Reduced Form

An obstacle to concept forgetting is that the pivot (or the negated pivot) may occur below an existential or universal role restriction. This prevents the Ackermann$^{\mathsf{C}}$ rules from being applicable. We address this obstacle in this subsection. In particular, we present two pairs of rewrite$^{\mathsf{C}}$ rules, namely, a pair of Surfacing$^{\mathsf{C}}$ rules and a pair of Skolemisation$^{\mathsf{C}}$ rules. The Surfacing$^{\mathsf{C}}$ rules allow the pivot (or the negated pivot) to be moved outside the scope of a universal role restriction in any clauses. The Skolemisation$^{\mathsf{C}}$ rules allow the pivot (or the negated pivot) to be moved outside the scope of an existential role restriction only in existential clauses.

Before describing these rules, we first introduce the notion of *pivot-reduced form*.

Definition 4.4.1 (Pivot$^+$-Reduced Form). For $A \in \mathsf{N_C}$ the pivot, a pivot$^+$-clause is in *pivot$^+$-reduced form* if it has the form $C \sqcup A$, where C is a concept that does not contain A. A set $\mathcal{N}$ of clauses is in *pivot$^+$-reduced form* if A occurs both positively and negatively in $\mathcal{N}$ and every *pivot$^+$*-clause in $\mathcal{N}$ is in *pivot$^+$*-reduced form.

Observe that pivot-clauses in pivot$^+$-reduced form contain a single (positive) occurrence of the pivot and this pivot occurs at the top level of the clause.

Theorem 4.4.2. *Let $\mathcal{N}$ be a set of clauses and let $A \in \mathrm{sig}_{\mathsf{C}}(\mathcal{N})$ be the pivot. The Ackermann$^{\mathsf{C},+}$ rule is applicable to $\mathcal{N}$ (to eliminate A) iff $\mathcal{N}$ is in A^+-reduced form.*

Proof. The "⇒" direction is obvious. We only prove the "⇐" direction. Let $\mathcal{N}$ be a set of clauses in A^+-reduced form, i.e., A occurs both positively and negatively in $\mathcal{N}$ and every A^+-clause in $\mathcal{N}$ is in A^+-reduced form. This means that every A^+-clause has the form $C \sqcup A$, where C is a concept that does not contain A. Since A occurs only negatively in the A^--pivot clauses, according to the side conditions of the Ackermann$^{\mathsf{C},+}$ rule, the Ackermann$^{\mathsf{C},+}$ rule is applicable to $\mathcal{N}$ to eliminate A. □

Definition 4.4.3 (**Pivot$^-$-Reduced Form**). For $A \in \mathsf{N_C}$ the pivot, a pivot$^-$-clause is in *pivot$^-$-reduced form* if it has the form $\neg A \sqcup C$, where C is a concept that does not contain A. A set $\mathcal{N}$ of clauses is in *pivot$^-$-reduced form* if A occurs both positively and negatively in $\mathcal{N}$ and every A^--clause in $\mathcal{N}$ is in pivot$^-$-reduced form.

Observe that pivot-clauses in pivot$^-$-reduced form contain a single (negative) occurrence of the pivot and this negated pivot occurs at the top level of the clause.

Theorem 4.4.4. *Let $\mathcal{N}$ be a set of clauses and let $A \in \mathrm{sig}_{\mathsf{C}}(\mathcal{N})$ be the pivot. The Ackermann$^{\mathsf{C},-}$ rule is applicable to $\mathcal{N}$ (to eliminate A), iff $\mathcal{N}$ is in A^--reduced form.*

Proof. The "⇒" direction is obvious. We only prove the "⇐" direction. Let $\mathcal{N}$ be a set of clauses in A^--reduced form, i.e., A occurs both positively and negatively in $\mathcal{N}$ and every A^--clause in $\mathcal{N}$ is in A^--reduced form. This means that every A^--clause has the form $\neg A \sqcup C$, where C is a concept that does not contain A. Since A occurs only positively in the A^--pivot clauses, according to the side conditions of the Ackermann$^{\mathsf{C},-}$ rule, the Ackermann$^{\mathsf{C},-}$ rule is applicable to $\mathcal{N}$ to eliminate A. □

Let $\mathcal{N}$ be a set of clauses, and let $A \in \mathrm{sig}_{\mathsf{C}}(\mathcal{N})$ be the pivot occurring both positively and negatively in $\mathcal{N}$. In order for the Ackermann$^{\mathsf{C},+}$ rule to be applicable to $\mathcal{N}$ (to eliminate A), $\mathcal{N}$ has to be transformed into A^+-reduced form, i.e., every A^+-clause in $\mathcal{N}$ has to be transformed into A^+-reduced form. In order for the Ackermann$^{\mathsf{C},-}$ rule to be applicable to $\mathcal{N}$ (to eliminate A), $\mathcal{N}$ has to be transformed into A^--reduced form, i.e., every A^--clause in $\mathcal{N}$ has to be transformed into A^--reduced form. In the sequel, we introduce two pairs of rewrite$^{\mathsf{C}}$ rules for transforming pivot-clauses (not in reduced form) into pivot-reduced form.

The first pair of the rewrite$^{\mathsf{C}}$ rules are the Surfacing$^{\mathsf{C}}$ rules, shown in Figure 4.4. The intention of the Surfacing$^{\mathsf{C}}$ rules is to move the pivot (or the negated pivot)

Surfacing$^{\mathsf{C},+}$

$$\frac{\mathcal{N}, C \sqcup \forall R.D}{\mathcal{N}, (\forall R^-.C) \sqcup D}$$

provided: (i) $A \in \text{sig}_{\mathsf{C}}(\mathcal{N})$ is the pivot,
(ii) A does not occur in C, and
(iii) A occurs **positively** in D.

Surfacing$^{\mathsf{C},-}$

$$\frac{\mathcal{N}, C \sqcup \forall R.D}{\mathcal{N}, (\forall R^-.C) \sqcup D}$$

provided: (i) $A \in \text{sig}_{\mathsf{C}}(\mathcal{N})$ is the pivot,
(ii) A does not occur in C, and
(iii) A occurs **negatively** in D.

Figure 4.4: The Surfacing$^{\mathsf{C}}$ rules for transforming A-clauses into A-reduced form

occurring anywhere below a universal role restriction upwards in the clause tree and "closer to the surface". By applying the Surfacing$^{\mathsf{C},+}$ rule, a concept symbol occurring anywhere below a universal role restriction can be moved outside the scope of this role restriction. By applying the Surfacing$^{\mathsf{C},-}$ rule, a negated concept symbol occurring anywhere below a universal role restriction can be moved outside the scope of this role restriction. Note that the Surfacing$^{\mathsf{C}}$ rules can only be applied to axioms and cannot be used for replacing subformulas inside of axioms.

Theorem 4.4.5. *The Surfacing$^{\mathsf{C}}$ rules in Figure 4.4 preserve equivalence.*

Proof. First, we prove the "*top-down*" direction. We do the proof by contradiction. Suppose there exists an element $d \in \Delta^{\mathcal{I}}$ such that $d^{\mathcal{I}} \notin (\forall R^-.C \sqcup D)^{\mathcal{I}}$.

$$\begin{aligned} & d^{\mathcal{I}} \notin (\forall R^-.C \sqcup D)^{\mathcal{I}} \\ \Rightarrow\ & d^{\mathcal{I}} \notin (\forall R^-.C)^{\mathcal{I}} \text{ and } d^{\mathcal{I}} \notin D^{\mathcal{I}} \\ \Rightarrow\ & d^{\mathcal{I}} \notin (\forall R^-.C)^{\mathcal{I}} \end{aligned}$$

This means that there exists an element $d' \in \Delta^{\mathcal{I}}$ such that:

$$(d^{\mathcal{I}}, d'^{\mathcal{I}}) \in (R^-)^{\mathcal{I}} \text{ and } d'^{\mathcal{I}} \notin C^{\mathcal{I}}$$

$$\Rightarrow (d'^{\mathcal{I}}, d^{\mathcal{I}}) \in R^{\mathcal{I}} \text{ and } d'^{\mathcal{I}} \notin C^{\mathcal{I}}$$

Since the premise of the Surfacing$^{\mathsf{C}}$ rules is true in $\mathcal{I}$, for every $x \in \Delta^{\mathcal{I}}$, we have:

$$x^{\mathcal{I}} \in (C \sqcup \forall R.D)^{\mathcal{I}}$$

in particular, for $x^{\mathcal{I}} = d'^{\mathcal{I}}$, we have: $d'^{\mathcal{I}} \in C^{\mathcal{I}}$ or $d'^{\mathcal{I}} \in (\forall R.D)^{\mathcal{I}}$. However, $d'^{\mathcal{I}} \notin C^{\mathcal{I}}$

$$\Rightarrow d'^{\mathcal{I}} \in (\forall R.D)^{\mathcal{I}}$$
$$\Rightarrow \forall y[(d'^{\mathcal{I}}, y^{\mathcal{I}}) \in R^{\mathcal{I}} \rightarrow y^{\mathcal{I}} \in D^{\mathcal{I}}],$$

in particular, this holds for $y^{\mathcal{I}} = d^{\mathcal{I}}$

$$\Rightarrow (d'^{\mathcal{I}}, d^{\mathcal{I}}) \in R^{\mathcal{I}} \rightarrow d^{\mathcal{I}} \in D^{\mathcal{I}}$$
$$\Rightarrow d^{\mathcal{I}} \in D^{\mathcal{I}}$$

Contradiction. Then, we prove the "*bottom-up*" direction. Again we do the proof by contradiction. Suppose there exists an element $d \in \Delta^{\mathcal{I}}$ such that $d^{\mathcal{I}} \notin (C \sqcup \forall R.D)^{\mathcal{I}}$.

$$d^{\mathcal{I}} \notin (C \sqcup \forall R.D)^{\mathcal{I}}$$
$$\Rightarrow d^{\mathcal{I}} \notin C^{\mathcal{I}} \text{ and } d^{\mathcal{I}} \notin (\forall R.D)^{\mathcal{I}}$$
$$\Rightarrow d^{\mathcal{I}} \notin (\forall R.D)^{\mathcal{I}}$$

This means that there exists an element $d'^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ such that:

$$(d^{\mathcal{I}}, d'^{\mathcal{I}}) \in R^{\mathcal{I}} \text{ and } d'^{\mathcal{I}} \notin D^{\mathcal{I}}$$

Since the premise is true in $\mathcal{I}$, for every $x \in \Delta^{\mathcal{I}}$, we have:

$$x^{\mathcal{I}} \in (\forall R^{-}.C \sqcup D)^{\mathcal{I}}$$

in particular, for $x^{\mathcal{I}} = d'^{\mathcal{I}}$, we have: $d'^{\mathcal{I}} \in (\forall R^{-}.C)^{\mathcal{I}}$ or $d'^{\mathcal{I}} \in D^{\mathcal{I}}$. However, $d'^{\mathcal{I}} \notin D^{\mathcal{I}}$

$$\Rightarrow d'^{\mathcal{I}} \in (\forall R^{-}.C)^{\mathcal{I}}$$
$$\Rightarrow \forall y[(y^{\mathcal{I}}, d'^{\mathcal{I}}) \in R^{\mathcal{I}} \rightarrow y^{\mathcal{I}} \in C^{\mathcal{I}}],$$

in particular, this holds for $y^{\mathcal{I}} = d^{\mathcal{I}}$

$$\Rightarrow (d^{\mathcal{I}}, d'^{\mathcal{I}}) \in R^{\mathcal{I}} \rightarrow d^{\mathcal{I}} \in C^{\mathcal{I}}$$

$$\Rightarrow d^{\mathcal{I}} \in C^{\mathcal{I}}$$

Contradiction. Thus, the Surfacing$^{\mathsf{C}}$ rules preserve equivalence.

□

By repeatedly applying the Surfacing$^{\mathsf{C},+}$ rule, a concept symbol occurring anywhere below a sequence of universal role restrictions can be moved outside the scope of these role restrictions. By repeatedly applying the Surfacing$^{\mathsf{C},-}$ rule, a negated concept symbol occurring anywhere below a sequence of universal role restrictions can be moved outside the scope of these role restrictions. This can be illustrated with an example.

Example 4.4.6. Consider the following ontology $\mathcal{O}$:

1. $B \sqcup \forall r.A$
2. $\exists r.\neg A \sqcup C$
3. $\forall r.C \sqcup \forall r.\forall s.A$

Assume A is the pivot. We apply the Surfacing$^{\mathrm{C},+}$ rule to Clauses 1 and 3, thereby obtaining the following set:

4. $\forall r^{-}.B \sqcup A$
5. $\exists r.\neg A \sqcup C$
6. $\forall r^{-}.\forall r.C \sqcup \forall s.A$

Observe that A is still preceded by a universal role restriction in Clause 6. We apply the Surfacing$^{\mathsf{C},+}$ rule again to Clause 6, thereby obtaining the following set:

7. $\forall r^{-}.B \sqcup A$
8. $\exists r.\neg A \sqcup C$
9. $\forall s^{-}.\forall r^{-}.\forall r.C \sqcup A$

$\mathcal{O}$ is in A^{+}-reduced form. We apply the Ackermann$^{\mathsf{C},+}$ rule to $\mathcal{O}$ to eliminate A.

As shown in the example above, inverse roles are playing a special role in the Surfacing$^{\mathsf{C},+}$ rules, because they allow the relationships between two concepts to be

represented in both directions. This would suggest that an Ackermann-based method does not seem feasible in general for concept forgetting in description logics without inverse roles, unless one is willing to extend the target language with inverse roles.

Although the Surfacing$^{\mathsf{C}}$ rules are very helpful in our method, they cannot guarantee the success of surfacing the pivot (or the negated pivot) from a universal role restriction. This can be illustrated with the following example.

Example 4.4.7. Consider the following ontology $\mathcal{O}$:

$$1.\ \neg B \sqcup \exists r.\neg A$$

$$2.\ \neg C \sqcup \exists s.\forall r.A$$

Assume A is the pivot. Observe that A occurs below a universal role restriction in Clause 2, but the Surfacing$^{\mathsf{C},+}$ rule is not applicable to this clause, because A is also preceded by an existential role restriction. Thus, the Surfacing$^{\mathsf{C}}$ rules are applicable to a clause only if the pivot is only preceded by universal restrictions in this clause.

Note that the Surfacing$^{\mathsf{C}}$ rules are not applicable to "$C \sqcup \forall R.D$" if "C" contains the pivot (or the negated pivot), though the rules are still sound.

Example 4.4.8. Consider the following ontology $\mathcal{O}$:

$$1.\ \neg B \sqcup \exists r.\neg A$$

$$2.\ \neg A \sqcup \forall r.A$$

Assume A is the pivot. Observe that the second occurrence of A in Clause 2 occurs below a universal role restriction. We apply the Surfacing$^{\mathsf{C},+}$ rule to Clause 2 to transform it into A^+-reduced form, thereby yielding the following set:

$$3.\ \neg B \sqcup \exists r.\neg A$$

$$4.\ \forall r^{-}.\neg A \sqcup A$$

Although the second occurrence of A has been passed up to the top level of Clause 4, the first occurrence of A is preceded by a universal role restriction. It is easy to see that there is an infinite loop in the derivation. We avoid this infinite loop by imposing a side condition that the pivot is not allowed in the concept "C".

Skolemisation$^{\mathsf{C},+}$

$$\frac{\mathcal{N}, \neg a \sqcup \exists R.C}{\mathcal{N}, \neg a \sqcup \exists R.b, \neg b \sqcup C}$$

provided: (i) $A \in \text{sig}_{\mathsf{C}}(\mathcal{N})$ is the pivot,
(ii) A occurs **positively** in C, and
(iii) b is a fresh nominal.

Skolemisation$^{\mathsf{C},-}$

$$\frac{\mathcal{N}, \neg a \sqcup \exists R.C}{\mathcal{N}, \neg a \sqcup \exists R.b, \neg b \sqcup C}$$

provided: (i) $A \in \text{sig}_{\mathsf{C}}(\mathcal{N})$ is the pivot,
(ii) A occurs **negatively** in C, and
(iii) b is a fresh nominal.

Figure 4.5: The Skolemisation$^{\mathsf{C}}$ rules for transforming A-clauses into A-reduced form

The other pair of the rewrite rules are the Skolemisation$^{\mathsf{C}}$ rules, shown in Figure 4.5. The intention of the Skolemisation$^{\mathsf{C}}$ rules is to move the pivot (or the negated pivot) occurring anywhere below an existential role restriction upwards in the clause tree and "closer to the surface". By applying the Skolemisation$^{\mathsf{C},+}$ rule, a concept symbol occurring anywhere below an existential role restriction can be moved outside the scope of the restriction in an existential clause. By applying the Surfacing$^{\mathsf{C},-}$ rule, a negated concept symbol occurring anywhere below a universal role restriction can be moved outside the scope of the restriction in an existential clause. In particular, the Skolemisation$^{\mathsf{C}}$ rules replace an existential clause of the form $\neg a \sqcup \exists R.C$ (i.e., a concept assertion) by two new clauses. One is an existential clause (i.e., a role assertion) that does not contain the pivot (or the negated pivot), i.e., $\neg a \sqcup \exists r.b$, and the other is an existential clause (i.e., a concept assertion) that contains the pivot (or the negated pivot), e.g., $\neg b \sqcup C$, where C a concept that contains the pivot (or the negated pivot). If C is a concept of an existential restriction, then we apply the Skolemisation$^{\mathsf{C}}$ rule again to $\neg b \sqcup C$ to surface the pivot (or the negated pivot). The Skolemisation$^{\mathsf{C}}$ rules are repeatedly applied until "C" is no longer an existential restriction, or the pivot (or

the negated pivot) has been passed up to the top level of the clause.

In a sense, the Skolemisation$^{\mathsf{C}}$ rules can be regarded as the Surfacing$^{\mathsf{C}}$ rules for existential role restrictions, but they are not as general as the Surfacing$^{\mathsf{C}}$ rules for universal role restrictions. Specifically, the Skolemisation$^{\mathsf{C}}$ rules allow for the surfacing of the pivot (or the negated pivot) only in clauses of the form $\neg a \sqcup \exists r.C$ where $a \in \mathsf{N_O}$ is a nominal, that is, the Skolemisation is only applied to existentially quantified clauses (i.e., existential clauses). This is different from the Surfacing$^{\mathsf{C}}$ rules, which, in contrast, allow for the surfacing of the pivot (or the negated pivot) in both existential and universal clauses. This restriction makes the Skolemisation$^{\mathsf{C}}$ rules less powerful than the Surfacing$^{\mathsf{C}}$ rules. Therefore, there is a gap in the scope of the Skolemisation$^{\mathsf{C}}$ rules for the surfacing of the pivot (or the negated pivot) occurring below an existential role restriction, a gap in the scope of the rewrite$^{\mathsf{C}}$ rules for transforming the clause set into pivot-reduced form, a gap in the scope of ACK$^{\mathsf{C}}$ for eliminating a single concept symbol from a set of $\mathcal{ALCOI}$-clauses, and a gap in the scope of our forgetting method for computing the solution of forgetting a set Σ of concept symbols from an $\mathcal{ALCOI}$-ontology. Another notable difference (between the Skolemisation$^{\mathsf{C}}$ rules and the Surfacing$^{\mathsf{C}}$ rules) is that the Skolemisation$^{\mathsf{C}}$ rules introduce new nominals, which means that if we apply the Skolemisation$^{\mathsf{C}}$ rules to existential pivot-clauses, then it is very likely that the forgetting solution would be expressed in an extended language with these introduced nominals. The Skolemisation$^{\mathsf{C}}$ rules, therefore, do not preserve logical equivalence. Instead, as with the Ackermann$^{\mathsf{C}}$ rules and the Purify$^{\mathsf{C}}$ rules, they preserve equivalence up to the interpretation of a specific symbol.

Theorem 4.4.9. *The Skolemisation$^{\mathsf{C}}$ rules preserve equivalence up to the interpretation of the introduced nominal.*

Proof. Observe that the Skolemisation$^{\mathsf{C}}$ rules are essentially the Ackermann$^{\mathsf{C}}$ rules in the reserved direction. The nominal introduced in the Skolemisation$^{\mathsf{C}}$ rules corresponds to the pivot in the Ackermann$^{\mathsf{C}}$ rules. Since the Ackermann$^{\mathsf{C}}$ rules preserve equivalence up to the interpretation of the pivot, the Skolemisation$^{\mathsf{C}}$ rules preserve equivalence up to the interpretation of the introduced nominal. □

The Skolemisation$^{\mathsf{C}}$ rules introduce new nominals into the present clause set. This seems to be away from our original purpose of "forgetting" symbols. Moreover, these

introduced nominals cannot be eliminated from the clause set, which means that the introduced nominals would occur in the forgetting solution (if the forgetting is successful). This is a weakness of the Skolemisation$^{\mathsf{C}}$ rules. In this sense, frequent use of the Skolemisation$^{\mathsf{C}}$ rules should be avoided; we use the Skolemisation$^{\mathsf{C}}$ rules in a conservative manner (i.e., as less frequently as possible).

Example 4.4.10. Consider the following ontology $\mathcal{O}$:

$$1.\ B_1 \sqcup \exists r.A$$

$$2.\ B_2 \sqcup \forall r.\neg A$$

Assume A is the pivot. Observe that A is preceded by an existential role restriction in Clause 1, and $\neg A$ is preceded by a universal role restriction. We can either apply the Skolemisation$^{\mathsf{C},+}$ rule to Clause 1, or apply the Surfacing$^{\mathsf{C},-}$ rule to Clause 2, to transform $\mathcal{O}$ into A^+-reduced form or A^--reduced form. In this case, we should apply the Surfacing$^{\mathsf{C},-}$ rule to avoid introducing new nominals. The following set is obtained thereby:

$$3.\ B \sqcup \exists r.A$$

$$4.\ \forall r^-.B_2 \sqcup \neg A$$

We apply the Ackermann$^{\mathsf{C},-}$ rule to $\mathcal{O}$ to eliminate A, thereby obtaining the following clause, which is the solution of forgetting $\{A\}$ from $\mathcal{O}$:

$$5.\ B \sqcup \exists r.\forall r^-.B_2$$

Despite this trivial weakness, the Skolemisation$^{\mathsf{C}}$ rules are very important to our calculus, because they allow concept symbols to be eliminated from ontologies with ABox assertions (and nominals). Skolemisation requires only the introduction of nominals, and not Skolem terms with dependencies on universally quantified variables. No other form of Skolemisation is performed in Ack$^{\mathsf{C}}$.

4.4.3 The Purify$^{\mathsf{C}}$ Rules

The Ackermann$^{\mathsf{C}}$ rules are used when the pivot occurs both positively and negatively in the present clause set. For the cases where the pivot occurs only positively or only

negatively (in the present clause set), we apply the Purify$^{\mathsf{C}}$ rules (to the clause set) to eliminate the pivot (i.e., in this case, we say that the pivot is *purifiable* w.r.t. the present clause set). The Purify$^{\mathsf{C}}$ rules are shown in Figure 4.6.

Purify$^{\mathsf{C},+}$

$$\frac{\mathcal{N}}{\mathcal{N}^{A}_{\bot}}$$

provided: (i) $A \in \mathrm{sig}_{\mathsf{C}}(\mathcal{N})$ is the pivot, and
(ii) $\mathcal{N}$ is **negative** w.r.t. A.

Purify$^{\mathsf{C},-}$

$$\frac{\mathcal{N}}{\mathcal{N}^{A}_{\top}}$$

provided: (i) $A \in \mathrm{sig}_{\mathsf{C}}(\mathcal{N})$ is the pivot, and
(ii) $\mathcal{N}$ is **positive** w.r.t. A.

Figure 4.6: The Purify$^{\mathsf{C}}$ rules for eliminating $A \in \mathsf{N}_{\mathsf{C}}$ from a set of clauses

Specifically, the Purify$^{\mathsf{C},+}$ rule says that if the pivot occurs only negatively in $\mathcal{N}$, then we substitute the bottom concept for every occurrence of the pivot in $\mathcal{N}$. The Purify$^{\mathsf{C},-}$ rule says that if the pivot occurs only positively in $\mathcal{N}$, then we substitute the top concept for every occurrence of the pivot in $\mathcal{N}$. Note that the Purify$^{\mathsf{C}}$ rules do not require the present clause set to be transformed into pivot-reduced form (i.e., the Purify$^{\mathsf{C}}$ rules are form-independent). They can be applied at any time as long as the pivot has been found purifiable in the present clause set (i.e., the pivot occurs only positively or only negatively in the present clause set).

Observe that the Purify$^{\mathsf{C},+}$ rule is a special case of the Ackermann$^{\mathsf{C},+}$ rule in the sense that the set of positive premises is empty. The Purify$^{\mathsf{C},-}$ rule is a special case of the Ackermann$^{\mathsf{C},-}$ rule in the sense that the set of negative premises is empty. This can be seen by adapting the Ackermann$^{\mathsf{C}}$ rules in the way as shown in Figure 4.7.

Theorem 4.4.11. *The Purify$^{\mathsf{C}}$ rules preserve equivalence up to the interpretation of the pivot.*

Purify$^{\mathsf{C},+}$

$$\frac{\mathcal{N}\backslash\{\top \sqsubseteq A, ..., \top \sqsubseteq A\}, \top \sqsubseteq A, ..., \top \sqsubseteq A}{\mathcal{N}\backslash\{\top \sqsubseteq A, ..., \top \sqsubseteq A\}^{A}_{\neg\top\sqcup...\sqcup\neg\top}}$$

provided: (i) $A \in \mathrm{sig}_{\mathsf{C}}(\mathcal{N})$ is the pivot, and
(ii) $\mathcal{N}\backslash\{\top \sqsubseteq A, ..., \top \sqsubseteq A\}$ is **negative** w.r.t. A.

Purify$^{\mathsf{C},-}$

$$\frac{\mathcal{N}\backslash\{\neg A \sqsubseteq \top, ..., \neg A \sqsubseteq \top\}, \neg A \sqsubseteq \top, ..., \neg A \sqsubseteq \top}{\mathcal{N}\backslash\{\neg A \sqsubseteq \top, ..., \neg A \sqsubseteq \top\}^{A}_{\top\sqcap...\sqcap\top}}$$

provided: (i) $A \in \mathrm{sig}_{\mathsf{C}}(\mathcal{N})$ is the pivot, and
(iii) $\mathcal{N}\backslash\{\neg A \sqsubseteq \top, ..., \neg A \sqsubseteq \top\}$ is **positive** w.r.t. A.

Figure 4.7: The Purify$^{\mathsf{C}}$ rules in the sense of the Ackermann$^{\mathsf{C}}$ rules

Proof. The Purify$^{\mathsf{C}}$ rules are special cases of the Ackermann$^{\mathsf{C}}$ rules (see Figure 4.7). In particular, the Purify$^{\mathsf{C},+}$ rule preserves equivalence up to the interpretation of the pivot in the sense of the Ackermann$^{\mathsf{C},+}$ rule. The Purify$^{\mathsf{C},-}$ rule preserves equivalence up to the interpretation of the pivot in the sense of the Ackermann$^{\mathsf{C},-}$ rule. □

Because of the nature of the Purify$^{\mathsf{C}}$ rules, one may reasonably expect that the result of the purification is a set of clauses that has been significantly reduced (compared to the original set). Specifically, using the top concept or the bottom concept to replace every occurrence of the pivot in the present clause set can lead to significant redundancies, contradictions and tautologies in the resulting clause set, which can be further simplified. Therefore, given a set of clauses and a set Σ of concept symbols to be forgotten, it is justifiable to eliminate the purifiable Σ-symbols first, and then the non-purifiable Σ-symbols. This can improve the efficiency and success rates of the method, because eliminating non-purifiable Σ-symbols from a relatively small clause set is easier than eliminating them from a relatively large clause set.

Example 4.4.12. Consider the following ontology $\mathcal{O}$:

1. $\neg A \sqcup \forall r.B$

$$2.\ \neg A \sqcup \exists r.\neg B$$

Assume $\Sigma = \{A, B\}$. Observe that A occurs only negatively in Clauses 1 and 2, and B occurs positively in Clause 1 and negatively in Clause 2. If we eliminate A first, then we apply the Purify$^{\mathsf{C},+}$ rule to $\mathcal{O}$, thereby obtaining the clause set $\{\top\}$. Observe that B does not occur in $\{\top\}$. $\{\top\}$ is thus the solution of forgetting $\{A, B\}$ from $\mathcal{O}$. If we eliminate B first, we apply the Surfacing$^{\mathsf{C},+}$ rule to Clause 1 to transform it into B^+-reduced form, and then apply the Ackermann$^{\mathsf{C},+}$ rule to $\mathcal{O}$ to eliminate A. We apply the Purify$^{\mathsf{C},-}$ rule to the intermediate result to eliminate A. It is obvious that eliminating the purifiable symbols first makes the problem rather easier.

4.4.4 The Simplification Rules

To ensure that the current clauses are always simpler representation, ACK$^{\mathsf{C}}$ involves a set of equivalence-preserving simplification rules, shown in Figure 4.8. The purpose of the simplification rules is twofold. On the one hand, they are used to simplify clauses, making the clause set more accessible to the forgetting method and thus improving the efficiency of the method. On the other hand, they are used to transform the clause set into a set of general clauses, which facilitates the transformation of the clause set into pivot-reduced form. This improves the success rates of the method.

The purpose of seeking efficiency is self-explained by the rules. We show how the simplification rules facilitate the transformation of the clause set into pivot-reduced form, and how the simplification rules improve the success rates of the method.

Example 4.4.13. Consider the following ontology $\mathcal{O}$:

$$1.\ \exists r.C \sqcup \exists r.A$$

$$2.\ \forall r.C \sqcup \exists r.\neg A$$

$$3.\ \neg E \sqcup \exists r.(\forall r.B \sqcup \exists r.\neg D \sqcup \neg B)$$

$$4.\ \neg B \sqcup D$$

Assume $\Sigma = \{A, B, C, D\}$. First, we attempt to eliminate A. Observe that A occurs positively below an existential role restriction in Clause 1 and occurs negatively below an existential restriction in Clause 2. The Skolemisation$^{\mathsf{C}}$ rules are not applicable to Clauses 1 and 2 because these two clauses are not existential clauses.

Obvious Simplifications

$$C \sqcup C \equiv C \qquad C \sqcup \neg C \equiv \top$$
$$C \sqcup \top \equiv \top \qquad C \sqcup \bot \equiv C$$
$$C \sqcap C \equiv C \qquad C \sqcap \neg C \equiv \bot$$
$$C \sqcap \top \equiv C \qquad C \sqcap \bot \equiv \bot$$
$$\forall R.\top \equiv \top \qquad \exists R.\bot \equiv \bot$$

Simplifications based on Absorption

$$C \sqcup (C \sqcap D) \equiv C \qquad C \sqcap (C \sqcup D) \equiv C$$
$$\exists R_1. \ldots \exists R_n.C \sqcup \exists R_1. \ldots \exists R_n.(C \sqcap D) \equiv \exists R_1. \ldots \exists R_n.C$$
$$\forall R_1. \ldots \forall R_n.C \sqcup \forall R_1. \ldots \forall R_n.(C \sqcap D) \equiv \forall R_1. \ldots \forall R_n.C$$

Simplifications based on Distributivity

$$\exists R.C \sqcup \exists R.D \equiv \exists R.(C \sqcup D)$$
$$\forall R.C \sqcap \forall R.D \equiv \forall R.(C \sqcap D)$$

Simplifications based on Surfacing

$$C \sqcup \forall R_1. \ldots \forall R_n.(\forall {R_n}^{-}. \ldots \forall {R_1}^{-}.C \sqcup D) \equiv C \sqcup \forall R_1. \ldots \forall R_n.D$$

Simplifications based on interaction between $\exists$ and $\forall$

$$\exists R.C \sqcap \forall R.\neg C \equiv \bot \qquad \exists R.\neg C \sqcap \forall R.C \equiv \bot$$
$$\exists R.C \sqcup \forall R.\neg C \equiv \top \qquad \exists R.\neg C \sqcup \forall R.C \equiv \top$$

Simplifications based on double negation elimination

$$\neg\neg C \equiv C$$

Figure 4.8: The simplification rules in ACK$^{\mathsf{C}}$

We attempt to eliminate B. Observe that B occurs both positively and negatively in Clause 3, but there are no rules applicable to Clause 3 to transform it into B^+-reduced form or B^--reduced form.

We attempt to eliminate C. Observe that C occurs only positively in $\mathcal{O}$. We apply the Purify$^{\mathsf{C},-}$ rule to $\mathcal{O}$ to eliminate C, thereby obtaining the following set:

5. $\exists r.\top \sqcup \exists r.A$
6. $\forall r.\top \sqcup \exists r.\neg A$
7. $\neg E \sqcup \exists r.(\forall r.B \sqcup \exists r.\neg D \sqcup \neg B)$
8. $\neg B \sqcup D$

Due to the simplification rule $\forall R.\top \equiv \top$, Clause 6 is a tautology and thus can be deleted. Then A occurs only positively in $\mathcal{O}$. We apply the Purify$^{\mathsf{C},-}$ rule to $\mathcal{O}$ to eliminate A, thereby obtaining the following set:

9. $\exists r.\top \sqcup \exists r.\top$
10. $\neg E \sqcup \exists r.(\forall r.B \sqcup \exists r.\neg D \sqcup \neg B)$
11. $\neg B \sqcup D$

We attempt to eliminate D. Observe that $\mathcal{O}$ is already in D^+-reduced form. We apply the Ackermann$^{\mathsf{C},+}$ rule to $\mathcal{O}$ to eliminate D, thereby obtaining the following set:

12. $\exists r.\top \sqcup \exists r.\top$
13. $\neg E \sqcup \exists r.(\forall r.B \sqcup \exists r.\neg B \sqcup \neg B)$

Due to the simplification rule $\forall R.C \sqcup \exists R.\neg C \equiv \top$ and the simplification rule $C \sqcup \top \equiv \top$, Clause 13 can be simplified as $\neg E \sqcup \exists r.\top$. Due to the simplification rule $C \sqcup C \equiv C$, Clause 12 can be simplified as $\exists r.\top$, which subsumes Clause 13. Therefore, the solution of forgetting $\{A, B, C, D\}$ from $\mathcal{O}$ is $\{\exists r.\top\}$.

This example has also demonstrated the significance of the order of eliminating Σ-symbols. we will discuss this in more detail in the next section.

The obvious simplification rules in Figure 4.8, which appear to be the simplest and most intuitive rules, are most beneficial, because they can simplify clauses, they can facilitate the transformation of the clause set into pivot-reduced form, and most

importantly, they are cheap to implement and they are used very often. The other simplification rules in Figure 4.8 are also very useful (in simplifying clauses and transforming the clause set into pivot-reduced form), but they are rarely used and they are expensive to implement. Frequent use of them could harm the efficiency of our forgetting method (in this sense, these rules are not worth their names). Therefore, we are faced with a dilemma about whether we should use these functional yet expensive simplification rules. The principle is that: (i) if we are pursuing good success rates of the method, then we should switch on these rules, because, as the preceding example shows, they can help transform the clause set into pivot-reduced form, which cannot be computed using only the rewrite$^{\mathsf{C}}$ rules. On the other hand, (ii) if we are pursuing the efficiency of the method, then we should switch off these rules, though this may lower the likelihood of the method successfully computing a solution of concept forgetting.

Theorem 4.4.14. *The simplification rules in Figure 4.8 preserve logical equivalence.*

Proof. The simplification rules in Figure 4.8 are standard transformations. □

So far, we have introduced all the key ingredients of the calculus Ack$^{\mathsf{C}}$. These ingredients include two pairs of rewrite$^{\mathsf{C}}$ rules for transforming a set of clauses into pivot-reduced form, a pair of Ackermann$^{\mathsf{C}}$ rules for eliminating a single concept symbol from a set of clauses in pivot-reduced form, a pair of Purify$^{\mathsf{C}}$ rules for eliminating a single concept symbol from a set of clauses positive or negative w.r.t. this symbol, and a set of simplification rules for improving the efficiency and success rates of the method. In the next subsection, we investigate several important properties of Ack$^{\mathsf{C}}$.

4.4.5 Properties of Ack$^{\mathsf{C}}$

Ack$^{\mathsf{C}}$ is a calculus for eliminating a single concept symbol from a set of clauses expressible in $\mathcal{ALCOI}$. Let $\mathcal{N}$ be a set of clauses expressible in $\mathcal{ALCOI}$ and let A be a concept symbol in $\mathcal{N}$. We say a derivation in Ack$^{\mathsf{C}}$ is *successful* w.r.t. A, if A does not occur in the result $\mathcal{N}'$ of the derivation. We say a derivation in Ack$^{\mathsf{C}}$ is *unsuccessful* (or *fails*) w.r.t. A, otherwise.

In this subsection, we show termination, soundness, and incompleteness of Ack$^{\mathsf{C}}$. In particular, we show that: (i) Ack$^{\mathsf{C}}$ is *terminating*, i.e. any Ack$^{\mathsf{C}}$-derivation terminates, (ii) Ack$^{\mathsf{C}}$ is *sound*, i.e., the resulting set $\mathcal{N}'$ of any successful Ack$^{\mathsf{C}}$-derivation

is equivalent to the original set $\mathcal{N}$ up to the interpretation of the pivot in the current derivation, possibly with the interpretations of the introduced nominals (iii) ACK$^{\mathsf{C}}$ is (concept forgetting) *incomplete* for $\mathcal{ALCOI}$-ontologies.

Theorem 4.4.15. ACK$^{\mathsf{C}}$ *is terminating and sound.*

Proof. In order to show termination of ACK$^{\mathsf{C}}$, we have to show that there is no infinite loop in each step of the ACK$^{\mathsf{C}}$-derivation. If the pivot does not occur in the present clause set, then the current ACK$^{\mathsf{C}}$-derivation terminates directly and returns the present clause set. If the pivot occurs only positively or only negatively in the present clause set (in normal form), then one of the Purify$^{\mathsf{C}}$ rules of ACK$^{\mathsf{C}}$ can be applied to eliminate the pivot. The current ACK$^{\mathsf{C}}$-derivation terminates and returns the resulting clause set. If the pivot occurs both positively and negatively in the present clause set (in normal form), the clause set is first transformed into pivot-reduced form using the Surfacing$^{\mathsf{C}}$ rules in Figure 4.4 and the Skolemisation$^{\mathsf{C}}$ rules in Figure 4.5. Observe that no rules could be applicable to the conclusion of the Surfacing$^{\mathsf{C}}$ rules or the Skolemisation$^{\mathsf{C}}$ rules except for the Ackermann$^{\mathsf{C}}$ rules. This means that there is no infinite loop in the transformation of the present clause set into pivot-reduced form. If the present clause set cannot be transformed into pivot-reduced form, then the current ACK$^{\mathsf{C}}$-derivation terminates and returns the present clause set, which still contains the pivot. If the present clause set has been transformed into pivot-reduced form, then the Ackermann$^{\mathsf{C}}$ rule can be applied to the clause set to eliminate the pivot. The current ACK$^{\mathsf{C}}$-derivation terminates and returns the resulting ontology. The simplification rules in Figure 4.8 are standard transformations.

We show the soundness of ACK$^{\mathsf{C}}$. ACK$^{\mathsf{C}}$ is defined in terms of a pair of Surfacing$^{\mathsf{C}}$ rules, a pair of Skolemisation$^{\mathsf{C}}$ rules, a pair of Ackermann$^{\mathsf{C}}$ rules, a pair of Purify$^{\mathsf{C}}$ rules and a set of simplification rules. We have shown that the Surfacing$^{\mathsf{C}}$ rules preserve logical equivalence, the Skolemisation$^{\mathsf{C}}$ rules preserve equivalence up to the interpretations of the introduced nominals, and the Ackermann$^{\mathsf{C}}$ and Purify$^{\mathsf{C}}$ rules preserve the equivalence up to the interpretations of the pivot. The simplification rules are standard transformation rules which preserve logical equivalence. Thus, the result of any successful ACK$^{\mathsf{C}}$-derivation is equivalent to the original ontology up to the interpretation of the pivot, possibly with the interpretations of the introduced nominals. □

Theorem 4.4.16. ACK$^{\mathsf{C}}$ *is (concept forgetting) incomplete for* $\mathcal{ALCOI}$*-ontologies.*

Proof. ACK$^{\mathsf{C}}$ is incomplete for $\mathcal{ALCOI}$-ontologies, because there is a gap in the scope of the rewrite$^{\mathsf{C}}$ rules for transforming pivot-clauses into pivot-reduced form. □

4.5 The Forgetting Method

In the previous section, we introduced a dedicated calculus ACK$^{\mathsf{C}}$ for eliminating a single concept symbol from a set of clauses expressible in $\mathcal{ALCOI}$. In this section, we present a practical method based on ACK$^{\mathsf{C}}$ for forgetting a set Σ of concept symbols from $\mathcal{ALCOI}$-ontologies. Following ACK$^{\mathsf{C}}$, the method is (concept forgetting) incomplete for $\mathcal{ALCOI}$-ontologies.

The method is not simply the process of one by one eliminating the symbols in Σ (using ACK$^{\mathsf{C}}$ as we described in the previous section). It also involves improvement of the efficiency and success rates of the method. This is because the most critical requirement for the method is its practical applicability, that is, we expect the method to be able to compute a solution of forgetting in as many cases as possible, and if the method is successful, we expect the forgetting is done in as short a time duration as possible. To this end, we have to adopt a global perspective to develop the method.

4.5.1 The Forgetting Process

Given an $\mathcal{ALCOI}$-ontology $\mathcal{O}$ of axioms and a set $\Sigma \subseteq \mathsf{sig_C}(\mathcal{O})$ of concept symbols to be forgotten, the forgetting process in our method consists of three main phases (see Figure 5.6): the conversion of $\mathcal{O}$ into a set $\mathcal{N}$ of clauses (**the clausification phase**), the conversion of Σ-clauses into normal form (**the normalisation phase**), the Σ-symbol elimination phase (**the central phase**), and the conversion of the resulting set $\mathcal{N}'$ into an ontology $\mathcal{O}'$ of axioms (**the declausification phase**). It is assumed that as soon as a forgetting solution has been computed, then the remaining phases are skipped. Our method is equipped with a frequency counter, dedicated to counting the frequencies of positive and negative occurrences of each Σ-symbol in $\mathcal{N}$.

Input: Given as input to the method are an $\mathcal{ALCOI}$-ontology $\mathcal{O}$ of TBox and ABox axioms and a set $\Sigma \subseteq \mathsf{sig_C}(\mathcal{O})$ of concept symbols to be forgotten. An important feature of the method is that Σ-symbols can be flexibly specified.

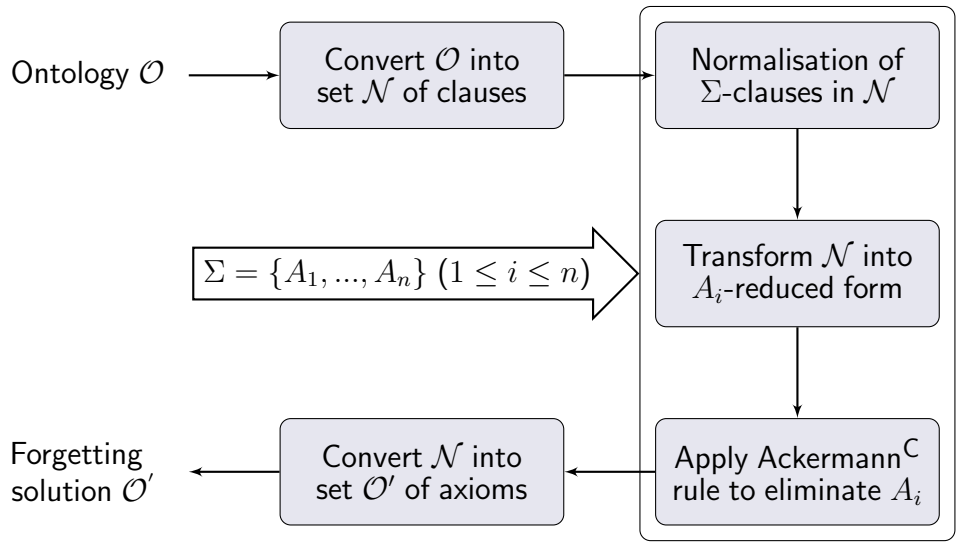

Figure 4.9: The main phases in concept forgetting process

TBox axioms into TBox clauses:

$$\begin{array}{lll} C \sqsubseteq D & \Longrightarrow & \neg C \sqcup D \\ C \equiv D & \Longrightarrow & \neg C \sqcup D, \neg D \sqcup C \end{array}$$

ABox assertions into TBox clauses:

$$\begin{array}{lll} C(a) & \Longrightarrow & \neg a \sqcup C \\ r(a,b) & \Longrightarrow & \neg a \sqcup \exists r.b \end{array}$$

Figure 4.10: Transformation of TBox axioms and ABox assertions into TBox clauses

The clausification phase: The initial phase of the forgetting process in the method is the clausification phase, where the method transforms $\mathcal{O}$ into a set $\mathcal{N}$ of clauses. The transformation is based on the use of the rules in Figure 4.10.

The first two rules in Figure 4.10 transform TBox axioms into TBox clauses and the last two rules in Figure 4.10 transform ABox assertions into TBox clauses. The last two transformations reflect the internalisation of ABox assertions into TBox axioms.

Theorem 4.5.1. *The transformation rules in Figure 4.2 preserve logical equivalence.*

Proof. The transformation rules in Figure 4.2 are standard transformation rules. □

The normalisation phase: The normalisation phase of the forgetting process transforms all Σ-clauses in $\mathcal{N}$ into normal form using the rules in Figures 4.1 and 4.2. The normalisation is not applied to the other clauses because they are not involved

in the elimination of the Σ-symbols. As a result, for the efficiency of our method, non-Σ-clauses will not be processed during the forgetting process.

The central phase: Central to the forgetting process is the Σ-symbol elimination phase, which is an iteration of several rounds (i.e., ACK$^{\mathsf{C}}$-derivations) in which the elimination of Σ-symbols is attempted. More specifically, the method attempts to eliminate the symbols in Σ one by one using the calculus ACK$^{\mathsf{C}}$ as described in the previous section. In each elimination round, the method (normally) performs two steps. The first step attempts to transform every (TBox) pivot-clause (not in pivot-reduced form) into pivot-reduced form using the Surfacing$^{\mathsf{C}}$ rules in Figure 4.4 and the Skolemisation$^{\mathsf{C}}$ rules in Figure 4.5, so that one of the Ackermann$^{\mathsf{C}}$ rules can be applied. If the transformation is successful, then the second step applies the Ackermann$^{\mathsf{C}}$ rule to the pivot-clauses (i.e., the premises) to eliminate the pivot. If the transformation is not successful, then the method skips the current round and attempts to eliminate another symbol in Σ (using ACK$^{\mathsf{C}}$). Upon the intermediate result being returned at the end of each round, the method repeats the same steps in the next round for the elimination of the remaining symbols in Σ (if necessary). If the pivot is found purifiable w.r.t. the present clause set (i.e., the pivot occurs only positively or only negatively in the present clause set), then the method performs only one single step in the elimination round. In particular, the method applies one of the Purify$^{\mathsf{C}}$ rules to the present clause set to eliminate the pivot.

The declausification phase: The final phase of the forgetting process is the declausification phase, where the method transforms $\mathcal{N}'$ into an ontology $\mathcal{O}'$ of axioms. The transformation is based on the use of the rules in Figure 4.11.

The first two rules in Figure 4.11 transform TBox clauses into TBox axioms and the last two rules in Figure 4.11 transform TBox clauses into ABox assertions. The last two transformations reflect the internalisation of TBox axioms into ABox assertions.

Theorem 4.5.2. *The transformation rules in Figure 4.2 preserve logical equivalence.*

Proof. The transformation rules in Figure 4.2 are standard transformation rules. □

In order to ensure that the present clauses are always simpler representations and thus more accessible to the method, a number of equivalence-preserving simplification rules are applied throughout the forgetting process.

TBox clauses into TBox axioms:

$$\begin{aligned} \neg C_1 \sqcup \ldots \sqcup \neg C_m \sqcup D_1 \sqcup \ldots \sqcup D_n \quad &\Longrightarrow \quad (C_1 \sqcap \ldots \sqcap C_m) \sqsubseteq (D_1 \sqcup \ldots \sqcup D_n) \\ \neg C_1 \sqcup \ldots \sqcup \neg C_m \quad &\Longrightarrow \quad (C_1 \sqcap \ldots \sqcap C_m) \sqsubseteq \bot \\ D_1 \sqcup \ldots \sqcup D_n \quad &\Longrightarrow \quad \top \sqsubseteq (D_1 \sqcup \ldots \sqcup D_n) \end{aligned}$$

ABox clauses into ABox assertions:

$$\begin{aligned} \neg a \sqcup C \quad &\Longrightarrow \quad C(a) \\ \neg a \sqcup \exists R.C \quad &\Longrightarrow \quad R(a, C) \end{aligned}$$

provided: (i) C, the C_i $(1 \leq i \leq m)$ and the D_j $(1 \leq j \leq n)$ are arbitrary positive literals excluding nominals,
(ii) R is an arbitrary role, and
(iii) $a \in \mathsf{N_O}$ is an arbitrary nominal.

Figure 4.11: Transformation of TBox clauses into TBox axioms and ABox assertions

Output: What the method returns as output at the end of the forgetting process is an ontology $\mathcal{O}'$ of axioms. If $\mathcal{O}'$ does not contain any symbols in Σ, then the method is *successful* (in computing a solution of forgetting Σ from $\mathcal{O}$). If $\mathcal{O}'$ still contains some symbols in Σ, then the method is *unsuccessful* (or the method *fails*).

The frequency counter: The applicability of the rules in ACK$^{\mathsf{C}}$ depends greatly on the polarity of the pivot w.r.t. the present clause set. For example, the Purify$^{\mathsf{C}}$ rules are applicable when the pivot occurs only positively or only negatively in the present clause set, and the Surfacing$^{\mathsf{C},+}$ rules are applicable to the clause $C \sqcup \forall R.D$ when the pivot occurs positively in the concept D. The polarity of the pivot can be found by counting the frequencies of positive occurrence and negative occurrence of the pivot in the present clause set. Specifically, if both the frequencies of positive occurrence and negative occurrence of the pivot in the present clause (or the present clause set) counted by the method are zero, then the pivot does not occur in the present clause (or the present clause set). If the frequency of positive occurrence of the pivot counted by the method is zero and the frequency of negative occurrence of the pivot counted by the method is non-zero in the present clause (or the present clause set), then the pivot occurs only negatively in the present clause (or the present clause set). If the frequency of positive occurrence of the pivot counted by the method is non-zero and

the frequency of negative occurrence of the pivot counted by the method is zero in the present clause (or the present clause set), then the pivot occurs only positively in the present clause (or the present clause set). If both the frequencies of positive occurrence and negative occurrence of the pivot in the present clause (or the present clause set) counted by the method are non-zero, then the pivot occurs both positively and negatively in the present clause (or the present clause set).

4.5.2 The Elimination Order

The elimination order refers to the order in which the Σ-symbols are eliminated. Previous work has shown that, in concept forgetting, changing the order of eliminating Σ-symbols could affect the efficiency and success of the method [CGV06a, Sch12, ZS15, ZS16]. In particular, Example 4.4.12 has shown that changing the elimination order affects the efficiency of the method. Now we show that changing the elimination order affects the success of the method.

Example 4.5.3. Consider the following ontology $\mathcal{O}$:

$$1.\ \neg a \sqcup \forall r.(\exists r.\neg A \sqcup B)$$

$$2.\ \neg a \sqcup \forall r.(\forall r.A \sqcup \neg B)$$

$$3.\ \neg a \sqcup \forall r.\exists r.A$$

Assume $\Sigma = \{A, B\}$ and A is the pivot. Observe that A occurs negatively in Clause 1 and positively in Clauses 2 and 3. This means that the Purify$^{\mathsf{C}}$ rules are not applicable. We attempt to transform $\mathcal{O}$ into A^--reduced form and then apply the Ackermann$^{\mathsf{C},-}$ rule to $\mathcal{O}$ to eliminate A. We apply the Surfacing$^{\mathsf{C},-}$ rule to Clause 1, thereby obtaining the clause $\forall r^-.\neg a \sqcup \exists r.\neg A \sqcup B$. Observe that $\neg A$ is still preceded by an existential role restriction, but the Skolemisation$^+$ rule cannot be applied to this clause because it is not an existential clause. No other rules are applicable to the present $\mathcal{O}$. Alternatively, we attempt to transform $\mathcal{O}$ into A^+-reduced form and then apply the Ackermann$^{\mathsf{C},+}$ rule to $\mathcal{O}$ to eliminate A. We apply the Surfacing$^{\mathsf{C},+}$ rule to Clauses 2 and 3, thereby obtaining the following clauses:

$$4.\ \forall r^-.\neg a \sqcup \exists r.\neg A \sqcup B$$

$$5.\ \forall r^-.\neg a \sqcup \forall r.A \sqcup \neg B$$

$$6.\ \forall r^{-}.\neg a \sqcup \exists r.A$$

The Surfacing$^{\mathsf{C},+}$ rule is applicable to Clause 5, but no rules are applicable to Clause 6. This means that $\mathcal{O}$ cannot be transformed into A^{+}-reduced form either.

Assume B is the pivot. Observe that the present $\mathcal{O}$ is in both B^{+}-reduced form and B^{-}-reduced form. We apply the Ackermann$^{\mathsf{C},+}$ rule (or the Ackermann$^{\mathsf{C},-}$ rule) to $\mathcal{O}$ to eliminate B, thereby obtaining the following clauses:

$$7.\ \forall r^{-}.\neg a \sqcup \exists r.\neg A \sqcup \forall r^{-}.\neg a \sqcup \forall r.A$$

$$8.\ \neg a \sqcup \forall r.\exists r.A$$

Observe that Clause 7 is a tautology because $\exists r.\neg A \sqcup \forall r.A \equiv \top$, and thus can be deleted. Then A occurs only positively in Clause 8, which is the only clause remaining in $\mathcal{O}$. We apply the Purify$^{\mathsf{C},+}$ rule to $\mathcal{O}$ to eliminate A, thereby yielding the clause $\neg a \sqcup \forall r.\exists r.\top$, which is the solution of forgetting $\{A, B\}$ from the ontology $\mathcal{O}$.

Thus, finding a good elimination order for the method to follow to eliminate Σ-symbols is crucial to the efficiency and success rates of the method.

Definition 4.5.4. An *ordering* on a set Σ (of concept symbols) is a transitive and irreflexive binary relation on Σ, denoted by $\succ$. A symbol A of Σ is *(strictly) maximal* w.r.t. $\succ$, if for all $B \in \Sigma$ different from A, $A \succ B$.

In the sequel, we introduce a heuristic for computing orderings $\succ$ on Σ, which is compatible with the order of eliminating Σ-symbols. The ordering $\succ$ provides local guidance to the elimination of Σ-symbols. In particular, a good ordering on Σ allows all Σ-symbols to be eliminated from the present clause set and allows the Σ-symbols to be eliminated as quickly as possible.

Lemma 4.5.5. For any non-empty set Σ of concept symbols, in the worst case there are $n!$ possible orderings, where $n = \#\Sigma$ and $n!$ denotes the factorial of n.

Proof. The following is an inductive method of generating all possible orderings on a set of n concept symbols.
Base Case: There is clearly one way to place one symbol in order.
Induction Hypothesis: Assume that there are $n!$ ways to place n symbols in order.

Inductive Step: Without loss of generality, we assume that $\Sigma = \{A_1, A_2, \ldots, A_n\}$. A permutation of Σ is the following: $A_1 \succ A_2 \succ \ldots \succ A_n$. Next, we take the number $n+1$. We can construct $n+1$ permutations from this one by placing A_{n+1} in all possible positions: $A_{n+1} \succ A_1 \succ \ldots \succ A_n$, $A_1 \succ A_{n+1} \succ A_2 \succ \ldots \succ A_n$, ..., $A_1 \succ \ldots \succ A_n \succ A_{n+1}$.

It is clear that all permutations of $n+1$ symbols can be obtained in this manner and no permutations are obtained more than once. Assume that there are P_n permutations on n symbols. Then there are $(n+1) \times P_n$ permutations on $n+1$ symbols. Hence by induction, and the recursive definition of the factorial: $P_n = n!$ □

The maximal symbol of $\succ$ is assumed to be eliminated first. The heuristic is based on a frequency analysis of positive occurrence and negative occurrence of all Σ-symbols. Specifically, let $\mathcal{N}$ be a set of clauses and let $\Sigma = \{A_1, \ldots, A_n\}$. The heuristic first counts the frequency $\mathsf{fp}(A_i)$ of positive occurrence and the frequency $\mathsf{fn}(A_i)$ of negative occurrence of each A_i $(1 \leq i \leq n)$ in $\mathcal{N}$. Then the heuristic compares $\mathsf{fp}(A_i)$ and $\mathsf{fn}(A_i)$ and assigns the smaller one to A_i as its *actual count*. In this way, every A_i is assigned an actual count. Finally, the heuristic sorts the actual counts of the Σ-symbols in an ascending order. A heuristic ordering $\succ$ is thus generated. Following $\succ$ and starting with the maximal symbol, the method eliminates the Σ-symbols one by one.

A justification of this heuristic is twofold: (i) the heuristic ensures that the purifiable symbols in Σ are eliminated first (i.e., there is a good chance that the intermediate solution after purification is a clause set much smaller than the original set), and (ii) one can reasonably expect that the Σ-symbols occurring less frequently in the present clause set should be eliminated more easily (i.e., because if a Σ-symbol occurs less frequently in the clause set, then there will be fewer pivot-clauses needing to be transformed into pivot-reduced form. This reduces the risk of failure in the transformation).

An optimisation that can be made to the heuristic is checking the existence of Σ-symbols occurring in cases where ACK$^{\mathsf{C}}$ is impotent. More specifically, it is clear that ACK$^{\mathsf{C}}$ cannot handle the cases where the pivot and the negated pivot both occur below existential role restrictions; see the following example.

Example 4.5.6. Consider the following ontology $\mathcal{O}$:

$$1.\ \neg B \sqcup \exists r.A$$

2. $\neg B \sqcup \forall r.\neg A$

Assume A is the pivot. Observe that A occurs positively below an existential role restriction in Clause 1. The Skolemisation$^{\mathsf{C},+}$ rule is not applicable in this case because Clause 1 is not an existential clause. Nevertheless, we can apply the Surfacing$^{C,-}$ rule to Clause 2 to transform it into A^--reduced form, and then apply the Ackermann$^{\mathsf{C},-}$ rule to $\mathcal{O}$ to eliminate A. In contrast, consider the following ontology $\mathcal{O}$:

1. $\neg B \sqcup \exists r.A$
2. $\neg B \sqcup \exists r.\neg A$

Assume A is the pivot. Observe that A occurs positively below an existential role restriction in Clause 1 and occurs negatively below an existential role restriction in Clause 2. The Skolemisation$^{\mathsf{C},+}$ rule is not applicable in this case because Clause 1 is not an existential clause and the Skolemisation$^{\mathsf{C},-}$ rule is not applicable in this case because Clause 2 is not an existential clause. Hence, $\mathcal{O}$ cannot be transformed into pivot-reduced form, and this problem cannot be solved by ACK$^{\mathsf{C}}$.

As a by-product of the frequency count of each Σ-symbol, the heuristic also checks if the symbol occurs both positively and negatively below existential role restrictions in non-existential clauses. If it does, then the symbol is flagged as a presently-ineliminable symbol and is moved to the tail end of the ordering (and the sequence of the other symbols in the ordering remains unchanged). Once those eliminable Σ-symbols have been eliminated, the flagged symbols may become eliminable (i.e., we have shown that in concept forgetting, the elimination of a concept symbol may affect the success of the elimination of another concept symbol). For this reason, (following the current ordering) our method will make another attempt to eliminate these symbols.

Example 4.5.7. Consider the following ontology $\mathcal{O}$:

1. $\neg A \sqcup \forall r.B$
2. $\forall r.\neg B \sqcup \forall r.\exists r.C$
3. $\neg A \sqcup \exists r.B$
4. $\neg B \sqcup \exists r.\neg C$
5. $\neg D \sqcup \forall r.C$

$$6.\ \neg B \sqcup \exists r.D$$

Assume $\Sigma = \{A, B, C, D\}$. The heuristic first counts the frequencies of positive occurrence and negative occurrence of A, B, C and D. The results are shown in Table 4.1.

$\mathsf{fp}(A)$	**0**	$\mathsf{fp}(B)$	**2**	$\mathsf{fp}(C)$	2	$\mathsf{fp}(D)$	**1**
$\mathsf{fn}(A)$	2	$\mathsf{fn}(B)$	3	$\mathsf{fn}(C)$	**1**	$\mathsf{fn}(D)$	**1**

Table 4.1: Frequency counts of Σ-symbols

Based on the frequency counts of the Σ-symbols, the heuristic assigns actual count values to A, B, C and D, which are 0, 2, 1 and 1, respectively. The heuristic compares these values and sorts them in an ascending order: $0 \succ 1 \succ 1 \succ 2$. In line with this order, the ordering $\succ$ is $A \succ C \succ D \succ B$.

While counting the frequencies of positive occurrence and negative occurrence of the Σ-symbols, the heuristic also checks the eliminability of them. It is easily observed that C occurs both positively and negatively below existential role restrictions in two non-existential clauses. Hence, C should be flagged as a presently-ineliminable symbol and moved to the tail end of the ordering $\succ$, which has become $A \succ D \succ B \succ C$. Following $\succ$ (and starting with A), the method eliminates the Σ-symbols one by one. If a symbol is successfully eliminated from $\mathcal{O}$, then the method attempts to eliminate the next symbol in $\succ$. If not, then the method skips the current elimination round and attempts to eliminate the next symbol in $\succ$. The symbol in the current elimination round is then moved to the tail end of $\succ$.

Observe that A occurs only negatively in $\mathcal{O}$. We apply the Purify$^{\mathsf{C},+}$ rule to $\mathcal{O}$ to eliminate A, thereby obtaining the following clauses:

$$7.\ \forall r.\neg B \sqcup \forall r.\exists r.C$$

$$8.\ \neg B \sqcup \exists r.\neg C$$

$$9.\ \neg D \sqcup \forall r.C$$

$$10.\ \neg B \sqcup \exists r.D$$

Then the method attempts to eliminate D. Observe that the present clause set is already in D^--reduced form, we apply the Ackermann$^{\mathsf{C},-}$ rule to $\mathcal{O}$ to eliminate D, thereby obtaining the following clauses:

$$11.\ \forall r.\neg B \sqcup \forall r.\exists r.C$$

$$12.\ \neg B \sqcup \exists r.\neg C$$

$$13.\ \neg B \sqcup \exists r.\forall r.C$$

Next, the method attempts to eliminate B. Observe that B occurs only negatively in the present clause set, we apply the Purify$^{\mathsf{C},-}$ rule to $\mathcal{O}$ to eliminate B, thereby obtaining the following clause set:

$$14.\ \forall r.\top \sqcup \forall r.\exists r.C$$

Clause 14 is a tautology because $\forall R.\top \equiv \top$. Thus, the forgetting solution computed by the method is $\{\top\}$.

This example has shown the significance of the order of eliminating concept symbols. In particular, if the method started with C, then the method would skip the current elimination round, because C was ineliminable. If the method started with B, although B was eliminable, the intermediate result would contain clauses of complex forms, which might increase the difficulty of the subsequent elimination. Note that the elimination order generated by the heuristic is not guaranteed the optimal solution. There might be other orders leading to better performance of the method.

Using the heuristic (based on frequency counts of Σ-symbols) as described above, Σ-symbols occurring only positively or only negatively in the given clause set will always be eliminated first (by applying the Purify$^{\mathsf{C}}$ rules). One may reasonably expect that the purification result is a clause set that is much simpler than the original one, from which eliminating other symbols in Σ becomes fairly easier.

4.5.3 Termination, Soundness and Incompleteness

In the previous subsections, we have presented a practical method for forgetting concept symbols from ontologies expressible in the description logic $\mathcal{ALCOIH}(\triangledown, \sqcap)$. In this subsection, we show a number of important properties of the method. In particular, we show that: (i) the method is *terminating*, i.e., for any $\mathcal{ALCOI}$-ontologies $\mathcal{O}$ of axioms and any set $\Sigma \subseteq \mathrm{sig}_{\mathsf{C}}(\mathcal{O})$ of concept symbols to be forgotten, the method always terminates and returns an ontology $\mathcal{O}'$ of axioms, (ii) the method is *sound*, i.e., if the method is successful, then the forgetting solution $\mathcal{O}'$ computed by the method is equivalent to the original ontology $\mathcal{O}$ up to the interpretations of the symbols in

Σ, possibly with the interpretations of the newly-introduced nominals, and (iii) the method is not concept forgetting *complete* for $\mathcal{ALCOI}$-ontologies.

Theorem 4.5.8. *For any $\mathcal{ALCOI}$-ontology $\mathcal{O}$ and any set $\Sigma \subseteq \text{sig}_{\mathsf{C}}(\mathcal{O})$ of concept symbols to be forgotten, the method always terminates and returns an ontology $\mathcal{O}'$. (i) If $\mathcal{O}'$ does not contain any symbols in Σ or any introduced nominals, then $\mathcal{O}'$ is a solution of forgetting Σ from $\mathcal{O}$ (i.e., $\mathcal{O}'$ is equivalent to the original ontology $\mathcal{O}$ up to the interpretations of the symbols in Σ). (ii) If $\mathcal{O}'$ does not contain any symbols in Σ but it contains introduced nominals, then $\mathcal{O}'$ is a solution of forgetting Σ from $\mathcal{O}$ in an extended language (and $\mathcal{O}$ and $\mathcal{O}'$ are equivalent up to the interpretations of the symbols in Σ, as well as the interpretations of the introduced nominals present in $\mathcal{O}'$).*

Proof. In order to show our method is terminating, we have to show that there is no infinite loop in each phase of the forgetting process in our method. The forgetting process in our method comprises four main phases. The initial phase of the method is the clausification phase, which is a standard transformation, i.e., any $\mathcal{ALCOI}$-ontology of axioms can be transformed into a set of clauses using the transformation rules in Figure 4.10. The second phase of the method is the normalisation phase. We have shown that any $\mathcal{ALCOI}$-ontologies can be transformed into a set of clauses in normal form using the rules in Figures 4.1 and 4.2. The third phase of the method is the Σ-symbol elimination phase, which is iteration of ACK$^{\mathsf{C}}$-derivations in each of which a single Σ-symbol is eliminated. Thus the termination of this phase follows the termination of ACK$^{\mathsf{C}}$. The final phase of the method is the declausification phase, which is a standard transformation, i.e., any set of clauses in $\mathcal{ALCOI}$ can be transformed into an $\mathcal{ALCOI}$-ontology by using the transformation rules in Figure 4.11.

Next, we show that our method is sound. The clausification, normalisation and declausification are standard equivalence-preserving transformations. The soundness of the Σ-symbol elimination phase follows the soundness of ACK$^{\mathsf{C}}$. Therefore, our method for concept forgetting is sound in the sense that the forgetting solution is equivalent to the original ontology up to the interpretations of the symbols in Σ, possibly with the interpretations of the introduced nominals. □

The incompleteness of the method for $\mathcal{ALCOI}$-ontologies follows the incompleteness of ACK$^{\mathsf{C}}$ for $\mathcal{ALCOI}$-ontologies. Given an $\mathcal{ALCOI}$-ontology $\mathcal{O}$ of axioms and

a set $\Sigma \subseteq \mathrm{sig}_{\mathsf{C}}(\mathcal{O})$ of concept symbols to be forgotten (as input to the method), the method may return an ontology $\mathcal{O}'$ that still contains some symbols in Σ. In this case, the method was *unsuccessful* (or the method *fails*). This is because there is a gap in the scope of the rewrite rules for handling existential role restrictions, and for handling cases where a Σ-symbol occurs multiple times in a clause.

4.6 Examples

We conclude this chapter with an example illustrating the usage of our method to forget concept symbols from ontologies expressible in the description logic $\mathcal{ALCOI}$.

Example 4.6.1. Consider the following ontology $\mathcal{O}$:

1. $A \sqsubseteq (\neg C \sqcap D)$
2. $B \sqsubseteq \forall r.\forall r^{-}.\neg B$
3. $A \sqsubseteq \exists r.B$
4. $a \sqsubseteq \exists s.\exists r.D$
5. $E \sqsubseteq \forall t.C$
6. $\forall s.D \sqsubseteq \exists s.B$

Assume $\Sigma = \{A, B, C, D\}$. The method transforms $\mathcal{O}$ into the following set of clauses:

7. $\neg A \sqcup (\neg C \sqcap D)$
8. $\neg B \sqcup \forall r.\forall r^{-}.\neg B$
9. $\neg A \sqcup \exists t.B$
10. $\neg a \sqcup \exists s.\exists r.D$
11. $\neg E \sqcup \exists t.C$
12. $\neg\forall s.D \sqcup \exists r.B$

Observe that Clauses 7 and 12 are not in normal form. The methods transforms them into normal form, thereby yielding the following set:

13. $\neg A \sqcup \neg C$
14. $\neg A \sqcup D$

15. $\neg B \sqcup \forall r.\forall r^{-}.\neg B$
16. $\neg A \sqcup \exists t.B$
17. $\neg a \sqcup \exists s.\exists r.D$
18. $\neg E \sqcup \exists t.C$
19. $\exists s.\neg D \sqcup \exists r.B$

Then the frequency counter of our method counts the frequencies of positive occurrence and negative occurrence of each Σ-symbol, thereby yielding the following statistics:

$\mathsf{fp}(A)$	**0**	$\mathsf{fp}(B)$	**2**	$\mathsf{fp}(C)$	**1**	$\mathsf{fp}(D)$	**2**
$\mathsf{fn}(A)$	3	$\mathsf{fn}(B)$	**2**	$\mathsf{fn}(C)$	**1**	$\mathsf{fn}(D)$	**1**

Table 4.2: Frequency counts of Σ-symbols in Example 4.6.1

Based on the results in Table 4.2 (i.e., in particular, based on the actual counts of the Σ-symbols), the heuristic computes the ordering $\succ$: $A \succ C \succ D \succ B$. Following $\succ$ and starting with the maximal symbol of $\succ$, the methods attemps to eliminate the Σ-symbols one by one. Hence, A becomes the pivot. Observa that A occurs only negatively in $\mathcal{O}$. The method applies the Purify$^{\mathsf{C},-}$ rule to $\mathcal{O}$ to eliminate A, thereby yielding the following set (after simplification):

20. $\neg B \sqcup \forall r.\forall r^{-}.\neg B$
21. $\neg a \sqcup \exists s.\exists r.D$
22. $\neg E \sqcup \exists t.C$
23. $\exists s.\neg D \sqcup \exists r.B$

Then C becomes the pivot. Observe that C occurs only positively in $\mathcal{O}$. The method applies the Purify$^{\mathsf{C},+}$ rule to $\mathcal{O}$ to eliminate C, thereby yielding the following set:

24. $\neg B \sqcup \forall r.\forall r^{-}.\neg B$
25. $\neg a \sqcup \exists s.\exists r.D$
26. $\neg E \sqcup \exists t.\top$
27. $\exists s.\neg D \sqcup \exists r.B$

Then D becomes the pivot. Observe that D occurs positively below two consecutive existential role restrictions in an existential clause, and occurs negatively below an

existential role restriction in a non-existential caluse. Clause 23 cannot be transformed into D^--reduced form. By repeated application of the Skolemisation$^{\mathsf{C},+}$ rule, Clause 24 can be transformed into D^+-reduced form, where $b, c \in \mathsf{N_O}$ are fresh nominals.

28. $\neg B \sqcup \forall r.\forall r^-.\neg B$
39. $\neg a \sqcup \exists s.b$
30. $\neg b \sqcup \exists r.c$
31. $\neg c \sqcup D$
32. $\neg E \sqcup \exists t.\top$
33. $\exists s.\neg D \sqcup \exists r.B$

The method applies the Ackermann$^{\mathsf{C},+}$ rule to $\mathcal{O}$ to eliminate D, thereby yielding the following set:

34. $\neg B \sqcup \forall r.\forall r^-.\neg B$
35. $\neg a \sqcup \exists s.b$
36. $\neg b \sqcup \exists r.c$
37. $\neg E \sqcup \exists t.\top$
38. $\exists s.\neg c \sqcup \exists r.B$

Now B is the only symbol that needs to be forgotten. Observe that B occurs negatively in Clause 34, and occurs positively below an existential role restriction in a non-existential clause. Clause 38 cannot be transformed into B^+-reduced form because the Skolemisation$^{\mathsf{C},+}$ rule is not applicable to a non-existential clause. Clause 34 cannot be transformed into B^--reduced form using the Surfacing$^{\mathsf{C},-}$ rule, because ire contains two occurrences of the pivot. Nevertheless, the method can apply the simplification rule based on Surfacing to Clause 34 to transform it into D^--reduced form.

39. $\neg B \sqcup \forall r.\bot$
40. $\neg a \sqcup \exists s.b$
41. $\neg b \sqcup \exists r.c$
42. $\neg E \sqcup \exists t.\top$
43. $\exists s.\neg c \sqcup \exists r.B$

Then the method applies the Ackermann$^{\mathsf{C},-}$ rule to $\mathcal{O}$ to eliminate B, thereby yielding the following set:

44. $\neg a \sqcup \exists s.b$
45. $\neg b \sqcup \exists r.c$
46. $\neg E \sqcup \exists t.\top$
47. $\exists s.\neg c \sqcup \exists r.\forall t.\bot$

The clause set above does not contain any symbols in Σ, and is thus the solution of forgetting Σ from the given ontology.

Chapter 5

Role Forgetting for $\mathcal{ALCOIH}(\triangledown, \sqcap)$

In the previous chapter, we introduced a practical method of semantic concept forgetting for ontologies expressible in the description logic $\mathcal{ALCOI}$. The method is goal-oriented, incremental and is based on a calculus, namely, ACK$^{\mathsf{C}}$, which exploits a generalisation of Ackermann's Lemma for description logics. Despite its practical applicability, a limitation of the method is that it does not support the forgetting of role symbols, which is also an important part of the forgetting problem. We fill this gap in this chapter. In particular, we present a practical method for (semantically) forgetting role symbols from ontologies expressible in the description logic $\mathcal{ALCOIH}(\triangledown, \sqcap)$, the description logic $\mathcal{ALCOI}$ extended with role hierarchies $\mathcal{H}$, role conjunctions $\sqcap$ and the universal role $\triangledown$. Whereas $\mathcal{ALCOI}$-ontologies have only a TBox and an ABox, $\mathcal{ALCOIH}(\triangledown, \sqcap)$-ontologies have additionally an RBox that contains axioms of role inclusions ($r \sqsubseteq s$) and role equivalences ($r \equiv s$).

We incorporate role hierarchies in the language because role symbols occur largely in them, and it would be interesting to see how role hierarchies interact with other expressivity in role forgetting problems. The other two added expressivity, the universal role and role conjunctions, play an indispensable role in our method, because they enrich our target language, making it expressive enough to represent the forgetting solution, which otherwise would have been lost. This will be explained in detail in a subsequent section.

The method is based on a calculus, namely, ACK$^{\mathsf{R}}$, which exploits a non-trivial generalisation of Ackermann's Lemma, namely, the Ackermann$^{\mathsf{R}}$ rule, as the fundamental rule to eliminate single role symbols. Being based on ACK$^{\mathsf{R}}$, the method is one

of the only few approaches that can forget role symbols, while also handling inverse roles and ABox axioms (via nominals), and the only approach so far providing support for role forgetting in description logics with nominals. The method is goal-oriented and incremental. It always terminates and is sound in the sense that the forgetting solution is equivalent to the original ontology up to the symbols that have been forgotten, possibly with introduced definer symbols and nominals. Definer symbols are auxiliary concept symbols which play a special role in the transformation of ontologies into normal form, i.e., that an ontology is in normal form is a necessary condition for application of the Ackermann$^{\mathsf{R}}$ rule.

Our method of concept forgetting for $\mathcal{ALCOI}$ can directly be used as a (incomplete) method for forgetting concept symbols from $\mathcal{ALCOIH}$-ontologies. This is because concept symbols do not occur in role inclusions and no additional rules are needed for RBox axioms. Also, it is easy to verify that the rules of ACK$^{\mathsf{C}}$ are sound for clauses incorporating role conjunctions and the universal role. Thus, the method is also a (incomplete) method for forgetting concept symbols in $\mathcal{ALCOIH}(\triangledown, \sqcap)$. We use this method (for concept forgetting), as well as the method introduced in this chapter (for role forgetting), as a unifying method for forgetting concept and role symbols from ontologies expressible in the description logic $\mathcal{ALCOIH}(\triangledown, \sqcap)$.

This chapter is an extension of our published work of [ZS16].

5.1 The Description Logic $\mathcal{ALCOIH}(\triangledown, \sqcap)$

In this section, we introduce the description logic $\mathcal{ALCOIH}(\triangledown, \sqcap)$, the language considered in this chapter, and formalise the notion of role forgetting for $\mathcal{ALCOIH}(\triangledown, \sqcap)$. Let $\mathsf{N_C}$, $\mathsf{N_R}$ and $\mathsf{N_O}$ be countably infinite and pairwise disjoint sets of *concept symbols*, *role symbols* and *individual symbols* (*nominals*), respectively. *Roles* in $\mathcal{ALCOIH}(\triangledown, \sqcap)$ can be any role symbol $r \in \mathsf{N_R}$, the inverse r^- of any role symbol r (i.e., an inverted role symbol), the universal role $\triangledown$ or formed with conjunction. The universal role relates any two individuals in the domain and also every individual with itself. *Concepts* in $\mathcal{ALCOIH}(\triangledown, \sqcap)$ have one of the following forms:

$$a \mid \top \mid \bot \mid A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists R.C \mid \forall \mathcal{S}.C,$$

where $a \in \mathsf{N_O}$, $A \in \mathsf{N_C}$, C and D are arbitrary concepts, R is an arbitrary role, and $\mathcal{S}$ is an arbitrary role symbol or the inverse of an arbitrary role symbol. Additional concepts and roles are defined as abbreviations: $\vartriangle = \neg\triangledown$. We assume w.l.o.g. that concepts and roles are equivalent relative to associativity and commutativity of $\sqcap$ and $\sqcup$, $\top$ and $\triangledown$ are units w.r.t. $\sqcap$, and $\neg$ is an involution.

One may have noticed that there is a difference in the roles between an existential role restriction ($\exists R.C$) and a universal role restriction ($\forall \mathcal{S}.C$), i.e., the role in an existential role restriction can be an arbitrary role in $\mathcal{ALCOIH}(\triangledown, \sqcap)$, whereas the role in a universal role restriction is restricted to be a role symbol or an inverted role symbol. This is because: (i) the given ontology is always an $\mathcal{ALCOIH}$-ontology, which means the roles in our source logic can only be a role symbol or an inverted role symbol; (ii) the universal role and role conjunction only occur in the forgetting solution or in some intermediate results, and more importantly, (when they occur in the forgetting solution or in the intermediate results) they only occur in existential role restrictions. This will be explained in detail in a subsequent section.

An $\mathcal{ALCOIH}(\triangledown, \sqcap)$-ontology is mostly assumed to be comprised of a TBox, an RBox and an ABox. A *TBox* $\mathcal{T}$ is a finite set of *concept axioms* of the form $C \sqsubseteq D$ (*concept inclusion*) and $C \equiv D$ (*concept equivalence*), where C and D are arbitrary concepts. An *RBox* $\mathcal{R}$ is a finite set of *role axioms* of the form $\mathcal{S} \sqsubseteq \mathcal{S}'$ (*role inclusion*) and $\mathcal{S} \equiv \mathcal{S}'$ (*role equivalence*), where $\mathcal{S}$ and $\mathcal{S}'$ are arbitrary role symbols or the inverses of arbitrary role symbols. We define $C \equiv D$ and $\mathcal{S} \equiv \mathcal{S}'$ as abbreviations for the pair of $C \sqsubseteq D$ and $D \sqsubseteq C$ and the pair of $\mathcal{S} \sqsubseteq \mathcal{S}'$ and $\mathcal{S} \sqsubseteq \mathcal{S}'$, respectively. An *ABox* $\mathcal{A}$ is a finite set of *concept assertions* of the form $C(a)$ and *role assertions* of the form $R(a, b)$, where $a, b \in \mathsf{N_O}$, C is an arbitrary concept and R is an arbitrary role. In a description logic with nominals, ABox assertions can be equivalently expressed as TBox axioms, namely, $C(a)$ as $a \sqsubseteq C$ and $R(a, b)$ as $a \sqsubseteq \exists R.b$. Hence, in this chapter, we assume w.l.o.g. that an ontology contains only TBox and RBox axioms.

The semantics of $\mathcal{ALCOQH}(\triangledown, \sqcap)$ is defined in terms of an *interpretation* $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, where $\Delta^{\mathcal{I}}$ is a non-empty set (the *domain of the interpretation*), and $\cdot^{\mathcal{I}}$ is the *interpretation function*, which assigns to every nominal $a \in \mathsf{N_O}$ a singleton $a^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, to every concept symbol $A \in \mathsf{N_C}$ a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, and to every role symbol $r \in \mathsf{N_R}$ a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The interpretation function $\cdot^{\mathcal{I}}$ is inductively extended

to concepts and roles as follows:

$$\top^{\mathcal{I}} = \Delta^{\mathcal{I}} \qquad \bot^{\mathcal{I}} = \emptyset \qquad \triangledown^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$$
$$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}} \qquad (R \sqcap S)^{\mathcal{I}} = R^{\mathcal{I}} \cap S^{\mathcal{I}}$$
$$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}} \qquad (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$$
$$(\exists R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \exists y.(x, y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$$
$$(\forall \mathcal{S}.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \forall y.(x, y) \in \mathcal{S}^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$$
$$(R^{-})^{\mathcal{I}} = \{(y, x) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (x, y) \in R^{\mathcal{I}}\},$$

where C and D denote concepts, R and S denote roles, $\mathcal{S}$ denotes a role symbol or the inverse of a role symbol, and x and y denote variables.

A concept inclusion $C \sqsubseteq D$ is *true* in an interpretation $\mathcal{I}$, and we write $\mathcal{I} \models C \sqsubseteq D$, iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. A concept equivalence $C \equiv D$ is *true* in an interpretation $\mathcal{I}$, and we write $\mathcal{I} \models C \equiv D$, iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ and $D^{\mathcal{I}} \subseteq C^{\mathcal{I}}$. A role inclusion $\mathcal{S} \sqsubseteq \mathcal{S}'$ is *true* in an interpretation $\mathcal{I}$, and we write $\mathcal{I} \models \mathcal{S} \sqsubseteq \mathcal{S}'$, iff $\mathcal{S}^{\mathcal{I}} \subseteq \mathcal{S}'^{\mathcal{I}}$. A role equivalence $\mathcal{S} \equiv \mathcal{S}'$ is *true* in an interpretation $\mathcal{I}$, and we write $\mathcal{I} \models \mathcal{S} \equiv \mathcal{S}'$, iff $\mathcal{S}^{\mathcal{I}} \subseteq \mathcal{S}'^{\mathcal{I}}$ and $\mathcal{S}'^{\mathcal{I}} \subseteq \mathcal{S}^{\mathcal{I}}$. $\mathcal{I}$ is a *model* of an ontology $\mathcal{O}$, and we write $\mathcal{I} \models \mathcal{O}$, iff every axiom in $\mathcal{O}$ is *true* in $\mathcal{I}$.

Our method works with TBox and RBox axioms in clausal normal form. We assume w.l.o.g. that a *TBox literal* in $\mathcal{ALCOIH}(\triangledown, \sqcap)$ is a concept of the form a, $\neg a$, A, $\neg A$, $\exists R.C$ or $\forall \mathcal{S}.C$, where $a \in \mathsf{N_O}$, $A \in \mathsf{N_C}$, C is an arbitrary concept, R is an arbitrary role and $\mathcal{S}$ is an arbitrary role symbol or the inverse of an arbitrary role symbol. A *TBox clause* in $\mathcal{ALCOIH}(\triangledown, \sqcap)$ is a disjunction of a finite number of TBox literals. An *RBox clause* in $\mathcal{ALCOIH}(\triangledown, \sqcap)$ is a disjunction of a role symbol and a negated role symbol. TBox and RBox clauses are obtained by clausification of TBox and RBox axioms, where in the case of RBox axioms role negation is introduced. This is done for consistency in presentation, replacing role inclusion by disjunction as the main operator. Nominals are used as regular concept symbols in our method, because the method is only concerned with role forgetting in this chapter.

Let $r \in \mathsf{N_R}$ be a designated role symbol. An axiom (clause) that contains an occurrence of r is called an *r-axiom* (*r-clause*). An occurrence of r is assumed to be *positive* (*negative*) in an r-axiom (r-clause) if it is under an *even* (*odd*) number of explicit and implicit negations. For instance, r is assumed to be positive in $\exists r.A$ and $s \sqsubseteq r$, and negative in $\forall r.A$ and $r \sqsubseteq s$. An ontology $\mathcal{O}$ of axioms is assumed to be

positive (*negative*) w.r.t. r if every occurrence of r in $\mathcal{O}$ is positive (negative). A set $\mathcal{N}$ of clauses is assumed to be *positive* (*negative*) w.r.t. r if every occurrence of r in $\mathcal{N}$ is positive (negative). An axiom (clause) that contains a positive occurrence of r is called an r^+-axiom (r^+-clause). An axiom (clause) that contains a negative occurrence of r is called an r^--axiom (r^--clause).

We now formalise our notion of role forgetting for $\mathcal{ALCOIH}(\triangledown, \sqcap)$. By $\mathrm{sig}_{\mathsf{C}}(X)$ we denote the set of the concept symbols occurring in X and by $\mathrm{sig}_{\mathsf{R}}(X)$ we denote the set of the role symbols occurring in X, where X ranges over concepts, roles, axioms, clauses, sets of axioms and sets of clauses. Let $r \in \mathsf{N}_{\mathsf{R}}$ be any role symbol, and let $\mathcal{I}$ and $\mathcal{I}'$ be any interpretations. We say $\mathcal{I}$ and $\mathcal{I}'$ are *equivalent up to* r, or r*-equivalent*, if $\mathcal{I}$ and $\mathcal{I}'$ coincide but differ possibly in the interpretations of r. More generally, $\mathcal{I}$ and $\mathcal{I}'$ are *equivalent up to a set* Σ *of role symbols*, or Σ*-equivalent*, if $\mathcal{I}$ and $\mathcal{I}'$ coincide but differ possibly in the interpretations of the symbols in Σ. This can be understood as follows: (i) $\mathcal{I}$ and $\mathcal{I}'$ have the same domain, i.e., $\Delta^{\mathcal{I}} = \Delta^{\mathcal{I}'}$, and interpret every concept symbol and every individual symbol identically, i.e., $A^{\mathcal{I}} = A^{\mathcal{I}'}$ for every $A \in \mathsf{N}_{\mathsf{C}}$ and $a^{\mathcal{I}} = a^{\mathcal{I}'}$ for every $a \in \mathsf{N}_{\mathsf{O}}$; (ii) for every role symbol $r \in \mathsf{N}_{\mathsf{R}}$ not in Σ, $r^{\mathcal{I}} = r^{\mathcal{I}'}$.

Definition 5.1.1 (Role Forgetting for $\mathcal{ALCOIH}(\triangledown, \sqcap)$)**.** Let $\mathcal{O}$ be an $\mathcal{ALCOIH}(\triangledown, \sqcap)$-ontology and let Σ be a subset of $\mathrm{sig}_{\mathsf{R}}(\mathcal{O})$. An ontology $\mathcal{O}'$ is a *solution of forgetting* Σ *from* $\mathcal{O}$, iff the following conditions hold:

(i) $\mathrm{sig}_{\mathsf{R}}(\mathcal{O}') \subseteq \mathrm{sig}_{\mathsf{R}}(\mathcal{O}) \backslash \Sigma$, and

(ii) for any interpretation $\mathcal{I}$: $\mathcal{I} \models \mathcal{O}'$ iff $\mathcal{I}' \models \mathcal{O}$, for some interpretation $\mathcal{I}'$ Σ-equivalent to $\mathcal{I}$.

It follows from Definition 5.1.1 that: (i) the original ontology $\mathcal{O}$ and the forgetting solution $\mathcal{O}'$ are equivalent up to (the interpretations of) the symbols in Σ. Also (ii) forgetting solutions are unique up to logical equivalence, that is, if both $\mathcal{O}'$ and $\mathcal{O}''$ are solutions of forgetting Σ from $\mathcal{O}$, then they are logically equivalent.

In this chapter, Σ is always assumed to be a set of symbols to be forgotten. The symbol in Σ under current consideration for forgetting is referred to as the *pivot* in our method. An axiom (clause) that contains an occurrence of the symbols in Σ is called a Σ*-axiom* (Σ*-clause*). An axiom (clause) that contains an occurrence of the pivot is called a *pivot-axiom* (*pivot-clause*).

5.2 Obstacles to Role Forgetting

Unlike concept symbols which represent unary elements, role symbols are binary relations that represent relationships between unary elements. This makes the problem of forgetting role symbols even more challenging. In this section, we point out the inherent obstacles to role forgetting and the obstacles we encountered when using an Ackermann-based method to solve role forgetting problems.

Previous work has been primarily focused on forgetting concept symbols, as role forgetting was realised to be significantly harder than concept forgetting [WWTP10]. A known reason is that the solution of forgetting role symbols is not expressible in the source logic in general and often requires more expressivity than is available in the source logic [ZS16]. For example, the solution of forgetting the role symbols $\{r, s\}$ from the $\mathcal{ALCH}$-ontology $\{A_1 \sqsubseteq \exists r.B_1, \exists r.B_1 \sqsubseteq B_1, r \sqsubseteq t_1, r \sqsubseteq t_2, A_2 \sqsubseteq \exists s.B_2, \exists s.B_2 \sqsubseteq B_2\}$ in quantifier-free first-order logic is the set (i.e., the solution is computed by the DLS algorithm [DLS97]):

$$\begin{aligned}\{&\forall x(A_1(x) \sqsubseteq B_1(f(x))), \forall x(A_1(x) \sqsubseteq t_1(x, f(x)))\\ &\forall x(A_1(x) \sqsubseteq t_2(x, f(x))), \forall x(A_1(x) \sqsubseteq B_1(x))\\ &\forall x(A_2(x) \sqsubseteq B_2(g(x))), \forall x(A_2(x) \sqsubseteq B_2(x))\}\end{aligned}$$

where $f(x)$ and $g(x)$ are Skolem terms. Because of the Skolem terms, this first-order logic solution is not expressible in purely the description logic $\mathcal{ALCH}$.

Nevertheless, it has been noticed that often such a solution can be expressed in a language extending the source logic with the universal role $\triangledown$ and role conjunction $\sqcap$. For example, the first-order logic solutions of the example above can be expressed as the following description logic expressions:

$$\begin{aligned}\{&A_1 \sqsubseteq \exists(t_1 \sqcap t_2).B_1, A_1 \sqsubseteq B_1,\\ &A_2 \sqsubseteq \exists\triangledown.B_2, A_2 \sqsubseteq B_2\}\end{aligned}$$

In particular, we have found that representing the solution of forgetting role symbols from $\mathcal{ALCOI}$-ontologies often requires the universal role to be in the target logic, and when role hierarchies are involved in the source logic (i.e., forgetting role symbols from $\mathcal{ALCOIH}$-ontologies), role conjunctions are additionally needed. Thus, we incorporate the universal role and role conjunctions in our target language, making it

expressive enough to represent the forgetting solution.

Given an ontology $\mathcal{O}$ and a set Σ of role symbols to be forgotten, computing a solution of forgetting Σ from $\mathcal{O}$ can be reduced to the problem of eliminating single symbols in Σ. As with in concept forgetting, this should be based on the use of the generalised Ackermann's Lemma for description logics (i.e., the Ackermann$^{\mathsf{C}}$ rules).

Recall that a sufficient condition for applying the Ackermann$^{\mathsf{C}}$ rules is that all pivot-clauses must be transformed into pivot-reduced form, that is, every positive occurrence of the pivot (i.e., every pivot symbol) or every negative occurrence of the pivot (i.e., every negated pivot symbol) must occur at the top level of the clause. In the case of concept forgetting, a concept symbol (or a negated concept symbol) deep inside a universal role restriction could be moved outwards using the Surfacing$^{\mathsf{C}}$ rules, and a concept symbol (or a negated concept symbol) deep inside an existential role restriction could be moved outwards using the Skolemisation$^{\mathsf{C}}$ rules when the existential role restriction is disjunctively connected with a negated nominal. These rules have been explained with details in the previous chapter.

Whereas in the case of role forgetting, since every role symbol that occurs in a TBox clause is always preceded by a universal or existential role restriction operator, it is not obvious how to rewrite TBox pivot-clauses (with every occurrence of the pivot being at the top level of the clause). Thus a direct method based on the generalised Ackermann's Lemma does not seem feasible for role forgetting in ontologies with TBoxes. The only cases where this direct method is feasible are the cases where the pivot occurs positively (or negatively) only in RBox axioms, in particular, role inclusions, where (negated) role symbols always occur at the top level.

An alternative approach to role forgetting is to exploit the translation of ontologies in first-order logic, where concept symbols and role symbols are treated equally (i.e., as with concept symbols, role symbols always occur at the top level of a flat clause, and are not preceded by role restrictions in first order logic formulas), and then apply Ackermann's Lemma for first-order logic (e.g., as in the DLS algorithm of [DLS97]) to eliminate single role symbols (i.e., binary predicate symbols). Such an indirect approach requires suitable back-translation however, which is absent at present for expressive description logics. Moreover, translating first-order logic formulas back into equivalent description logic expressions is not straightforward, especially when

Skolem terms are present in the first-order logic formulas. For example, the solution of forgetting the role symbol $\{r\}$ from the $\mathcal{ALC}$-ontology $\{A_1 \sqsubseteq \exists r.B_1, A_2 \sqsubseteq \forall r.B_2\}$ in quantifier-free first-order logic is the set:

$$\{\forall x(\neg A_1(x) \vee B_1(f(x))), \forall x(\neg A_1(x) \vee \neg A_2(x) \vee \neg B_2(f(x)))\},$$

where $f(x)$ is a Skolem term. Because of the presence of the Skolem term (i.e., $f(x)$ occurs in both clauses, which can make the back-translation even harder), it is not obvious whether this first-order logic solution can be expressed in a description logic.

Another major obstacle that blocks our way to solving role forgetting problems (using a conventional Ackermann-based method) is that the pivot-clauses in the given ontology may contain multiple occurrences of the pivot. For instance, the pivot r occurs twice in the TBox clauses $\exists r.A \sqcup \forall r.B$ and $A \sqcup \exists r.\forall r.B$, and occurs twice in the RBox clauses $\neg r \sqcup r^-$ and $\neg r^- \sqcup r$. The presence of such clauses makes role forgetting a seemingly unsolvable problem (when using a conventional Ackermann-based method) because a conventional Ackermann-based method requires the present ontology $\mathcal{O}$ to be transformed into one of the following forms:

$$\mathcal{O}\backslash\{\alpha_1 \sqcup \mathcal{S}, ..., \alpha_n \sqcup \mathcal{S}\}, \alpha_1 \sqcup \mathcal{S}, ..., \alpha_n \sqcup \mathcal{S} \tag{5.1}$$

where $\mathcal{S} \in \mathsf{N_C}$ ($\mathcal{S} \in \mathsf{N_R}$) is the pivot, every α_i ($1 \leq i \leq n$) is a concept that does not contain $\mathcal{S}$, and $\mathcal{O}\backslash\{\alpha_1 \sqcup \mathcal{S}, ..., \alpha_n \sqcup \mathcal{S}\}$ is negative w.r.t. $\mathcal{S}$, or

$$\mathcal{O}\backslash\{\neg\mathcal{S} \sqcup \alpha_1, ..., \neg\mathcal{S} \sqcup \alpha_n\}, \neg\mathcal{S} \sqcup \alpha_1, ..., \neg\mathcal{S} \sqcup \alpha_n \tag{5.2}$$

where $\mathcal{S} \in \mathsf{N_C}$ ($\mathcal{S} \in \mathsf{N_R}$) is the pivot, every α_i ($1 \leq i \leq n$) is a concept that does not contain $\mathcal{S}$, and $\mathcal{O}\backslash\{\neg\mathcal{S} \sqcup \alpha_1, ..., \neg\mathcal{S} \sqcup \alpha_n\}$ is positive w.r.t. $\mathcal{S}$.

It can be observed from (5.1) and (5.2) that: (i) every $\alpha_i \sqcup \mathcal{S}$ contains a single occurrence of $\mathcal{S}$ and every $\neg\mathcal{S} \sqcup \alpha_i$ contains a single occurrence of $\mathcal{S}$ ($1 \leq i \leq n$), and (ii) positive occurrences and negative occurrences of $\mathcal{S}$ are properly separated, i.e., in (5.1), $\mathcal{S}$ occurs only positively in $\alpha_1 \sqcup \mathcal{S}, ..., \alpha_n \sqcup \mathcal{S}$, and only negatively in the remainder of $\mathcal{O}$; in (5.2), $\mathcal{S}$ occurs only negatively in $\neg\mathcal{S} \sqcup \alpha_1, ..., \neg\mathcal{S} \sqcup \alpha_n$ and only positively in the remainder of $\mathcal{O}$. Therefore, an approach that transforms pivot-clauses into simpler forms with only a single occurrence of the pivot is desired.

In this section, we have described an inherent obstacle to role forgetting, as well as two obstacles we encountered in developing our Ackermann-based method for role

forgetting. We have addressed the first one by incorporating the universal role and role conjunction in our target language. They have enriched our target language, making it expressive enough to represent the forgetting solution, and thus avoiding information loss in the forgetting solution. We address the other two obstacles in the next sections. In the remainder of this chapter, we assume w.l.o.g that an ontology is a set of (TBox and RBox) clauses.

5.3 The Normalisation

One of the major obstacles to role forgetting (when using an Ackermann-based method to solve role forgetting problems), as noted in the previous section, is that the pivot-clauses in the given ontology may contain multiple occurrences of the pivot. In TBox pivot-clauses, the pivot may occur multiple times in a single role restriction because role conjunction is incorporated in our language (e.g., $\neg A \sqcup \exists(r \sqcap r^{-}).B$, for $r \in \mathsf{N_R}$ the pivot), or in a number of different role restrictions (e.g., $\forall r.A \sqcup \exists r.B$, for $r \in \mathsf{N_R}$ the pivot). In RBox pivot-clauses, the pivot may occur multiple times only in cases such as $\neg r \sqcup r^{-}$ and $\neg r^{-} \sqcup r$, for $r \in \mathsf{N_R}$ the pivot, i.e., the pivot and the inverse pivot occur simultaneously in an RBox pivot-clause. In contrast, an Ackermann-based method requires every "$\alpha_i \sqcup \mathcal{S}$" (i.e., every positive premise) or every "$\neg\mathcal{S} \sqcup \alpha_i$" (i.e., every negative premise) to contain a single occurrence of the pivot. Given an ontology $\mathcal{O}$ of clauses and a set Σ of role symbols to be forgotten, in order to transform the pivot-clauses in $\mathcal{O}$ into a form with a single occurrence of the pivot, first, we need to transform the given ontology into normal form (i.e., ontology normalisation).

Definition 5.3.1 (**Normal Form**)**.** A TBox clause is in *normal form* if it does not contain role restrictions or it has the form $C \sqcup \exists R.D$ or $C \sqcup \forall\mathcal{S}.D$, where R is an arbitrary role, $\mathcal{S}$ is an arbitrary role symbol or inverted role symbol, and C and D are concepts that do not contain role restrictions. An RBox clause is in *normal form* if it has the form $\neg\mathcal{S} \sqcup \mathcal{S}'$ or $\neg\mathcal{S}' \sqcup \mathcal{S}$, where $\mathcal{S}$ and $\mathcal{S}'$ are arbitrary role symbols or inverted role symbols. An ontology $\mathcal{O}$ is in *normal form* if every clause in $\mathcal{O}$ is in normal form. We refer to such an ontology as an *normalised ontology*. Let $\Sigma \subseteq \mathrm{sig}_{\mathsf{R}}(\mathcal{O})$ be a set of role symbols. An ontology $\mathcal{O}$ is in *normal form w.r.t.* Σ if every Σ-clause in $\mathcal{O}$ is in normal form. We refer to such an ontology as a *normalised ontology* w.r.t. Σ.

By definition, a TBox clause in normal form contains at most one role restriction. However, since role conjunction is incorporated in our language, a role restriction may contain multiple role symbols. Thus, a TBox pivot-clause in normal form may contain multiple occurrences of the pivot. For example, for $r \in \mathsf{N_R}$ the pivot, the clause $\neg A \sqcup \exists(r \sqcap r^-).B$ is in normal form, but it contains two occurrences of the pivot. An RBox pivot-clause in normal form may contain multiple occurrences (more precisely, two occurrences) of the pivot. For example, for $r \in \mathsf{N_R}$ the pivot, the clauses $\neg r \sqcup r^-$ and $\neg r^- \sqcup r$ contain two occurrences of the pivot. Because of the incorporation of inverse roles in our language, a pivot-clause in normal form may contain multiple occurrences of the pivot.

Observe that an RBox clause is always in normal form, but this is not true for a TBox clause. TBox clauses not in normal form have one of the following forms:

$$C \sqcup \exists R.D \text{ or } C \sqcup \forall \mathcal{S}.D$$

where R is an arbitrary role, $\mathcal{S}$ is an arbitrary role symbol or inverted role symbol, and C and D are concepts with at least one of them containing a (or a number of) role restriction(s). In some cases, TBox clauses not in normal form could be transformed into normal form by exploiting the Surfacing$^{\mathsf{C}}$ and Skolemisation$^{\mathsf{C}}$ rules.

Example 5.3.2. Consider the following ontology $\mathcal{O}$:

1. $\forall r^-.(\forall r.A \sqcup \neg B) \sqcup A$
2. $\neg a \sqcup \exists s.\exists r.C$
3. $\neg r \sqcup s$

Observe that Clauses 1 and 2 are not in normal form. We apply the Surfacing$^{\mathsf{C}}$ rule to Clause 1, thereby yielding the clause $\neg B \sqcup \forall r.A$. We apply the Skolemisation$^{\mathsf{C}}$ rule to Clause 2, thereby yielding the clauses $\neg a \sqcup \exists s.b$ and $\neg b \sqcup \exists r.C$, where b is a fresh nominal. In this way, Clauses 1 and 2 are transformed into normal form, and the ontology $\mathcal{O}$ is transformed into a normalised ontology.

However, the Surfacing$^{\mathsf{C}}$ and Skolemisation$^{\mathsf{C}}$ rules are helpful only in a limited number of cases, i.e., the Surfacing$^{\mathsf{C}}$ rules are helpful in the cases where the result of Surfacing can be further simplified, and the Skolemisation$^{\mathsf{C}}$ rules are helpful only in

the existential clauses where there exists a sequence of existential role restrictions. In cases such as $\neg A \sqcup \exists s.\forall r.B$, where there exists a sequence of universal and existential role restrictions, or cases such as $\forall r.A \sqcup \forall r.B$, where there exist a number of separated role restrictions, they are impotent.

Substitution of C with a definer symbol

$$\frac{\mathcal{N}, C \sqcup \exists R.D}{\mathcal{N}, \mathsf{D_d} \sqcup \exists R.D, \neg \mathsf{D_d} \sqcup C} \qquad \frac{\mathcal{N}, C \sqcup \forall \mathcal{S}.D}{\mathcal{N}, \mathsf{D_d} \sqcup \forall \mathcal{S}.D, \neg \mathsf{D_d} \sqcup C}$$

provided: (i) R is a role,
(ii) $\mathcal{S}$ is a role symbol or an inverted role symbol,
(iii) C is a concept that contains role restrictions, and
(iv) $\mathsf{D_d} \in \mathsf{N_D}$ is a fresh definer symbol.

Substitution of D with a definer symbol

$$\frac{\mathcal{N}, C \sqcup \exists R.D}{\mathcal{N}, C \sqcup \exists R.\mathsf{D_d}, \neg \mathsf{D_d} \sqcup D} \qquad \frac{\mathcal{N} C \sqcup \forall \mathcal{S}.D}{\mathcal{N} C \sqcup \forall \mathcal{S}.\mathsf{D_d}, \neg \mathsf{D_d} \sqcup D}$$

provided: (i) R is a role,
(ii) $\mathcal{S}$ is a role symbol or an inverted role symbol,
(iii) D is a concept that contains role restrictions, and
(iv) $\mathsf{D_d} \in \mathsf{N_D}$ is a fresh definer symbol.

Figure 5.1: Transformation of TBox clauses into normal form

In the sequel, we introduce an approach that can transform any TBox clause into normal form. The approach is based on the use of a countably infinite set of auxiliary symbols, namely, *definer symbols* (or *definers* for short). In particular, definer symbols are specialised concept symbols that do not occur in the present ontology [KS13a]. Playing a special role in our method, definer symbols are introduced as follows (to facilitate the transformation of TBox clauses into normal form): let $\mathcal{O}$ be a non-normalised $\mathcal{ALCOIH}(\triangledown, \sqcap)$-ontology (i.e., $\mathcal{O}$ contains clauses of the form $C \sqcup \exists R.D$ or $C \sqcup \forall \mathcal{S}.D$, where R is an arbitrary role, $\mathcal{S}$ is an arbitrary role symbol or inverted role symbol, and C and D are concepts with at least one of them containing role restrictions), and let $\mathsf{N_D} \subset \mathsf{N_C}$ be a set of definer symbols disjoint with $\mathrm{sig}_{\mathsf{C}}(\mathcal{O})$. Definer

symbols are used as substitutes, incrementally replacing "C" and "D" for every TBox clause not in normal form until neither "C" nor "D" contain role restrictions. A new clause $\neg \mathsf{D}_1 \sqcup C$ is added to $\mathcal{O}$ for each replaced subconcept C and a new clause $\neg \mathsf{D}_2 \sqcup D$ is added to $\mathcal{O}$ for each replaced subconcept D, where $\mathsf{D}_1, \mathsf{D}_2 \in \mathsf{N}_\mathsf{D}$ are fresh definer symbols. $\mathcal{O}$ is thus transformed into a set of clauses in normal form. The transformation can be simulated by the rules in Figure 5.1.

Theorem 5.3.3. *The transformation rules in Figure 5.1 preserve equivalence up to the interpretations of the introduced definer symbols.*

Proof. Observe that these transformation rules are essentially the Ackermann$^\mathsf{C}$ rules in the reserved direction. The definer symbol introduced in the transformation rules corresponds to the pivot in the Ackermann$^\mathsf{C}$ rules. Since the Ackermann$^\mathsf{C}$ rules preserve equivalence up to the interpretation of the pivot, these transformation rules preserve equivalence up to the interpretations of the introduced definer symbols. □

Example 5.3.4. Consider the following ontology $\mathcal{O}$:

1. $\neg A \sqcup \forall s.\exists r.B$
2. $\forall r.A \sqcup \exists r^-.B$
3. $\neg r \sqcup s$

Observe that Clauses 1 and 2 are not in normal form. In this case, two fresh definer symbols $\mathsf{D}_1 \in \mathsf{N}_\mathsf{C}$ and $\mathsf{D}_2 \in \mathsf{N}_\mathsf{C}$ are introduced into the ontology to replace the concept $\exists r.B$ in Clause 1 and the concept $\forall r.A$ in Clause 2, respectively. Two additional clauses $\neg \mathsf{D}_1 \sqcup \exists r.B$ and $\neg \mathsf{D}_2 \sqcup \forall r.A$ are added to $\mathcal{O}$:

1. $\neg A \sqcup \forall s.\mathsf{D}_1$
2. $\neg \mathsf{D}_1 \sqcup \exists r.B$
3. $\mathsf{D}_2 \sqcup \exists r^-.B$
4. $\neg \mathsf{D}_2 \sqcup \forall r.A$
5. $\neg r \sqcup s$

In this way $\mathcal{O}$ is transformed into a set of clauses in normal form.

Normal form transformation is only applied to TBox clauses. If a TBox clause does not contain role restrictions or contains only a single role restriction, then it is already in normal form. If a TBox clause contains two role restrictions, then one definer symbol needs to be introduced to replace one of the two role restrictions. If a TBox clause contains three role restrictions, then two definer symbols need to be introduced to replace two of the three role restrictions. Likewise, if a TBox clause contains n role restrictions, then $n-1$ definer symbols need to be introduced to replace $n-1$ of the n role restrictions. Let $\mathcal{O}$ be an $\mathcal{ALCOIH}(\triangledown, \sqcap)$-ontology that contains TBox clauses $\alpha_1, \alpha_2, \ldots, \alpha_n$. By $|\alpha_i|^{\exists\forall}$ we denote the number of role restrictions occurring in α_i ($1 \leq i \leq n$). Then $(|\alpha_i|^{\exists\forall}-1)+\ldots+(|\alpha_n|^{\exists\forall}-1)$ (i.e., $|\alpha_i|^{\exists\forall}+\ldots+|\alpha_n|^{\exists\forall}-n$) definer symbols need to be introduced for the transformation of $\mathcal{O}$ into normal form. It is easy to see that introducing a fresh definer symbol leads to an additional clause in the present ontology. Thus, transforming $\mathcal{O}$ into normal form will lead to $|\alpha_i|^{\exists\forall} + \ldots + |\alpha_n|^{\exists\forall} - n$ additional clauses in $\mathcal{O}$.

Theorem 5.3.5. *Using definer introduction as described above, any $\mathcal{ALCOIH}(\triangledown, \sqcap)$-ontology can be transformed into a set of clauses in normal form in finite steps.*

Proof. For any $\mathcal{ALCOIH}(\triangledown, \sqcap)$-ontology that contains a finite number of clauses (and thus contains a finite number of role restrictions), the result of the transformation (of the ontology into normal form using definer introduction) is a finite set of clauses. Thus, the normal form transformation can be done in finite steps. □

Given an ontology $\mathcal{O}$ and a set $\Sigma \in \mathsf{sig}_{\mathsf{R}}(\mathcal{O})$ of role symbols to be forgotten, there is no need for every clause in $\mathcal{O}$ to be transformed into normal form, because in the process of eliminating the pivot, only the pivot-clauses will be (and need to be) processed. Therefore, only the Σ-clauses in $\mathcal{O}$ need to be transformed into normal form for the elimination of the symbols in Σ. If all Σ-clauses in $\mathcal{O}$ have been transformed into normal form, then we say $\mathcal{O}$ is a normalised ontology w.r.t. Σ. In the next section, we present a dedicated calculus for eliminating a single role symbol from $\mathcal{ALCOIH}(\triangledown, \sqcap)$-ontologies. The calculus works with TBox and RBox clauses in normal form.

5.4 The Calculus – ACK$^{\mathsf{R}}$

In this section, we introduce a dedicated calculus, namely, ACK$^{\mathsf{R}}$, for eliminating a single role symbol from a set of $\mathcal{ALCOIH}(\triangledown, \sqcap)$-clauses. The calculus is based on a non-trivial generalisation of Ackermann's Lemma and works directly with description logic expressions. In particular, the calculus has three key ingredients: (i) transformation of the present clause set (in normal form) into *pivot-reduced form*, (ii) an Ackermann-based rule, namely, the Ackermann$^{\mathsf{R}}$ rule, and (iii) a pair of Purify$^{\mathsf{R}}$ rules. The Ackermann$^{\mathsf{R}}$ rule reflects the generalisation of Ackermann's Lemma and allows a single role symbol to be eliminated from a clause set in pivot-reduced form. The Purify$^{\mathsf{R}}$ rules are special cases of the Ackermann$^{\mathsf{R}}$ rule and allow a single role symbol to be eliminated from a clause set where the pivot occurs only positively or only negatively.

5.4.1 Transformation into Pivot-Reduced Form

Given an ontology $\mathcal{O}$ and a set $\Sigma \in \mathsf{sig}_{\mathsf{R}}(\mathcal{O})$ of role symbols to be forgotten, once $\mathcal{O}$ has been transformed into normal form (using definer introduction), we obtain a new ontology with every pivot-clause in the ontology being in normal form. However, since a pivot-clause in normal form may still contain multiple occurrences of the pivot or contain the inverse pivot (i.e., because of the presence of role conjunctions and inverse roles), that every pivot-clause in $\mathcal{O}$ is in normal form is not a sufficient condition for the application of the Ackermann$^{\mathsf{R}}$ rule (to $\mathcal{O}$ to eliminate the pivot). It is only a necessary condition because the Ackermann$^{\mathsf{R}}$ rule requires further that every pivot-clause contains only a single occurrence of the pivot, and does not contain the inverse pivot. There are cases where the pivot-clauses are in normal form, but the Ackermann$^{\mathsf{R}}$ rule is not applicable. For example, all r-clauses in $\{A \sqcup \exists r.B, B \sqcup \forall r^{-}.B, \neg r^{-} \sqcup s\}$ are in normal form, but the Ackermann$^{\mathsf{R}}$ rule is not applicable (to this ontology to eliminate the pivot r) because r is preceded by an inverse operator in the last two clauses. In order for the Ackermann$^{\mathsf{R}}$ rule to be applied, we need to transform the present ontology into pivot-reduced form, i.e., a restriction of the normal form by ruling out multiple occurrences of the pivot and occurrences of the inverse pivot.

Definition 5.4.1 (**Pivot-Reduced Form**)**.** For $r \in \mathsf{N}_{\mathsf{R}}$ the pivot, a TBox pivot-clause is in *pivot-reduced form* if it has the form

$$C \sqcup \exists(r \sqcap Q).D \text{ or } C \sqcup \forall r.D,$$

where C and D are concepts that do not contain any role restrictions and Q is a role that does not contain r. An RBox clause is in *pivot-reduced form* if it has the form

$$\neg\mathcal{S} \sqcup r \text{ or } \neg r \sqcup \mathcal{S},$$

where $\mathcal{S} \neq r$ is a role symbol or $\mathcal{S} \neq r^-$ is an inverted role symbol. An ontology $\mathcal{O}$ is in *pivot-reduced form* if every pivot-clause in $\mathcal{O}$ is in pivot-reduced form.

Pivot-clauses in pivot-reduced form contain a single occurrence of the pivot and does not contain any occurrences of the inverse pivot. The Ackermann$^{\mathsf{R}}$ rule is applicable to the pivot-clauses to eliminate the pivot iff all pivot-clauses are in pivot-reduced form. In the sequel, we present three rewrite rules to transform pivot-clauses (in normal form) with an inverse pivot into equivalent clauses with the inverse pivot being inverted back to a role symbol. Note that the rewrite rules are applied (to appropriate pivot-clauses) only when the pivot-clauses have been transformed into normal form. The rules are shown in Figure 5.2.

The first rule in Figure 5.2 is an instance of the Surfacing$^{\mathsf{C}}$ rules of ACK$^{\mathsf{C}}$ for concept forgetting. It turns out to be useful for role forgetting as well. We incorporate this rule in ACK$^{\mathsf{R}}$ and refer to it as the Surfacing$^{\mathsf{R}}$ rule. In particular, it is used to transform a TBox clause with an inverted role symbol below a universal role restriction into an equivalent clause with the inverted role symbol being inverted back into a role symbol. In the context of ACK$^{\mathsf{R}}$, this allows a TBox pivot-clause of the form $C \sqcup \forall r^-.D$ to be transformed into a form suitable for application of the Ackermann$^{\mathsf{R}}$ rule. Since the intention of these rewrite rules is to invert an inverse role back into a role symbol, we refer to the other two rules as the Inverting$^{\mathsf{R}}$ rules in our calculus. In particular, the Inverting$^{\mathsf{R}}$ rules are used to transform an RBox clause with an inverse role into an equivalent clause with the inverse role being inverted back into a role symbol. In the context of ACK$^{\mathsf{R}}$, they allow an RBox pivot-clause of the form $\neg\mathcal{S} \sqcup r^-$ or $\neg r^- \sqcup \mathcal{S}$ to be transformed into a form suitable for application of the Ackermann$^{\mathsf{R}}$ rule.

Theorem 5.4.2. *The rewrite rules in Figure 5.2 preserve equivalence.*

Surfacing$^{\mathsf{R}}$

$$\frac{\mathcal{N}, C \sqcup \forall r^{-}.D}{\mathcal{N}, (\forall r.C) \sqcup D}$$

provided: (i) $r \in \mathsf{N_R}$ is the pivot, and
(ii) $C \sqcup \forall r^{-}.D$ is in normal form.

Inverting$^{\mathsf{R}}$

$$\frac{\mathcal{N}, \neg\mathcal{S} \sqcup r^{-}}{\mathcal{N}, \neg\mathcal{S}^{-} \sqcup r} \qquad \frac{\mathcal{N}, \neg r^{-} \sqcup \mathcal{S}}{\mathcal{N}, \neg r \sqcup \mathcal{S}^{-}}$$

provided: (i) $r \in \mathsf{N_R}$ is the pivot,
(ii) $\neg\mathcal{S} \sqcup r^{-}$ and $\neg r^{-} \sqcup \mathcal{S}$ are in normal form, and
(iii) $\mathcal{S} \neq r$.

Figure 5.2: The rewrite$^{\mathsf{R}}$ rules for transforming r-clauses into r-reduced form

Proof. We have proved in the previous chapter that the Surfacing$^{\mathsf{R}}$ rule preserves equivalence. For the Inverting$^{\mathsf{R}}$ rules, we do the proof by contradiction. We first prove the "top-down" direction for the first Inverting$^{\mathsf{R}}$ rule. Suppose there exists an element $d \in \Delta^{\mathcal{I}}$ and an element $d' \in \Delta^{\mathcal{I}}$ such that $(d, d') \notin (\neg\mathcal{S}^{-} \sqcup r)^{\mathcal{I}}$.

$$\begin{aligned}
&(d, d') \notin (\neg\mathcal{S}^{-} \sqcup r)^{\mathcal{I}} \\
\Rightarrow\ &(d, d') \notin (\neg\mathcal{S}^{-})^{\mathcal{I}} \text{ and } (d, d') \notin r^{\mathcal{I}} \\
\Rightarrow\ &(d, d') \in (\mathcal{S}^{-})^{\mathcal{I}} \text{ and } (d, d') \notin r^{\mathcal{I}} \\
\Rightarrow\ &(d', d) \in \mathcal{S}^{\mathcal{I}} \text{ and } (d, d') \notin r^{\mathcal{I}}
\end{aligned}$$

Since the premise of the rule is true in $\mathcal{I}$, then for every x and y:

$$\begin{aligned}
&(x, y) \in (\neg\mathcal{S} \sqcup r^{-})^{\mathcal{I}} \\
\Rightarrow\ &(x, y) \in (\neg\mathcal{S})^{\mathcal{I}} \text{ or } (x, y) \in (r^{-})^{\mathcal{I}} \\
\Rightarrow\ &(x, y) \notin \mathcal{S}^{\mathcal{I}} \text{ or } (y, x) \in r^{\mathcal{I}}
\end{aligned}$$

This contradicts with $(d', d) \in \mathcal{S}^{\mathcal{I}}$ and $(d, d') \notin r^{\mathcal{I}}$. Next, we prove the "bottom-up" direction. Suppose there exists an element $d \in \Delta^{\mathcal{I}}$ and an element $d' \in \Delta^{\mathcal{I}}$ such that $(d, d') \notin (\neg\mathcal{S} \sqcup r^{-})^{\mathcal{I}}$.

$$\begin{aligned} & (d, d') \notin (\neg\mathcal{S} \sqcup r^{-})^{\mathcal{I}} \\ \Rightarrow\ & (d, d') \notin (\neg\mathcal{S})^{\mathcal{I}} \text{ and } (d, d') \notin (r^{-})^{\mathcal{I}} \\ \Rightarrow\ & (d, d') \in \mathcal{S}^{\mathcal{I}} \text{ and } (d', d) \notin r^{\mathcal{I}} \end{aligned}$$

Since the conclusion of the rule is true in $\mathcal{I}$, then for every x and y:

$$\begin{aligned} & (x, y) \in (\neg\mathcal{S}^{-} \sqcup r)^{\mathcal{I}} \\ \Rightarrow\ & (x, y) \in (\neg\mathcal{S}^{-})^{\mathcal{I}} \text{ or } (x, y) \in r^{\mathcal{I}} \\ \Rightarrow\ & (x, y) \notin \mathcal{S}^{-\mathcal{I}} \text{ or } (x, y) \in r^{\mathcal{I}} \\ \Rightarrow\ & (y, x) \notin \mathcal{S}^{\mathcal{I}} \text{ or } (x, y) \in r^{\mathcal{I}} \end{aligned}$$

This contradicts with $(d, d') \in \mathcal{S}^{\mathcal{I}}$ and $(d', d) \notin r^{\mathcal{I}}$. The other Inverting$^{\mathsf{R}}$ rule in Figure 5.2 can be proved similarly. □

One may have noticed that there is a gap in the scope of the rewrite rules, i.e., there are no rules available for inverting an inverse role occurring in TBox clauses of the form $C \sqcup \exists(r^{-} \sqcap Q).D$ and there are no rules available for handling cases such as $\neg r \sqcup r^{-}$ and $\neg r^{-} \sqcup r$. The Inverting$^{\mathsf{R}}$ rules cannot be applied to the latter two cases because otherwise, the other occurrence of "r" in the clause will be inverted. This leads to an infinite loop in the derivation. We avoid this by imposing a side condition $\mathcal{S} \neq r$ on the Inverting$^{\mathsf{R}}$ rules. The gap leads to the incompleteness of Ack$^{\mathsf{R}}$ (in eliminating a single role symbol from an $\mathcal{ALCOIH}(\triangledown, \sqcap)$-ontology) and the incompleteness of our method (in forgetting a set of role symbols from an $\mathcal{ALCOIH}(\triangledown, \sqcap)$-ontology).

Theorem 5.4.3. *Any normalised $\mathcal{ALCOH}(\triangledown, \sqcap)$-ontology is in pivot-reduced form.*

Proof. A TBox pivot-clause in $\mathcal{ALCOH}(\triangledown, \sqcap)$ in normal form has the form $C \sqcup \exists R.D$ or $C \sqcup \forall\mathcal{S}.D$, where R is an arbitrary role in $\mathcal{ALCOIH}(\triangledown, \sqcap)$ excluding inverse roles, $\mathcal{S}$ is an arbitrary role symbol, and C and D are concepts that do not contain any role restrictions. If the pivot occurs multiple times in R, then we can remove multiple occurrences of the pivot in R by repeatedly applying the simplification rule $r \sqcap r \equiv r$,

for $r \in \mathsf{N_R}$ an arbitrary role symbol. Thus, a TBox pivot-clause in $\mathcal{ALCOH}(\triangledown, \sqcap)$ in normal form is also in r-reduced form. An RBox pivot-clause in $\mathcal{ALCOH}(\triangledown, \sqcap)$ in normal form has the form $\neg\mathcal{S} \sqcup r$ and $\neg r \sqcup \mathcal{S}$, where $\mathcal{S}$ is an arbitrary role symbol. If $\mathcal{S} = r$, then this clause can be simplified to $\top$ by applying the simplification rule $\neg r \sqcup r \equiv r$, for $r \in \mathsf{N_R}$ an arbitrary role symbol. Thus, an RBox pivot-clause in $\mathcal{ALCOH}(\triangledown, \sqcap)$ in normal form is also in pivot-reduced form. □

We have shown that any $\mathcal{ALCOH}(\triangledown, \sqcap)$-ontology can be transformed into normal form using definer introduction. In this sense, any $\mathcal{ALCOH}(\triangledown, \sqcap)$-ontology can be transformed into pivot-reduced form using definer introduction.

In this subsection, we introduced the notion of pivot-reduced form. Pivot-clauses in pivot-reduced form contain a single occurrence of the pivot and do not contain any occurrences of the inverse pivot. An ontology $\mathcal{O}$ is in pivot-reduced form if every pivot-clause in $\mathcal{O}$ is in pivot-reduced form. The Ackermann$^{\mathsf{R}}$ rule is applicable to $\mathcal{O}$ to eliminate the pivot iff $\mathcal{O}$ is in pivot-reduced form. We presented three rewrite rules to transform pivot-clauses into pivot-reduced form. The rules are applied only when the pivot-clauses have been transformed into normal form. Note that not every pivot-clause in $\mathcal{ALCOIH}(\triangledown, \sqcap)$ can be transformed into pivot-reduced form. This is because there is a gap in the scope of the rewrite rules. Nevertheless, we have found that using definer introduction, any $\mathcal{ALCOH}(\triangledown, \sqcap)$-ontologies can be transformed into pivot-reduced form. In the next subsection, we describe the Ackermann$^{\mathsf{R}}$ rule.

5.4.2 The Ackermann$^{\mathsf{R}}$ Rule

The Ackermann$^{\mathsf{R}}$ rule is the most important rule in Ack$^{\mathsf{R}}$. It leads to the elimination of a role symbol from a set of clauses. Unlike the Ackermann$^{\mathsf{C}}$ rules for concept forgetting, the Ackermann$^{\mathsf{R}}$ rule is not an obvious generalisation of Ackermann's Lemma for description logics. It has a different flavour which makes the rule not easy to understand. Before we describe the Ackermann$^{\mathsf{R}}$ rule, we first introduce the notion of *premises*, useful for the presentation of the Ackermann$^{\mathsf{R}}$ rule.

Let $\mathcal{N}$ be a set of (TBox and RBox) clauses exhibiting all different forms of r-reduced form, where $r \in \mathrm{sig}_{\mathsf{R}}(\mathcal{N})$ is the pivot. We refer to the clauses of the form $C \sqcup \exists(r \sqcap Q).D$ and clauses of the form $C \sqcup \forall r.D$ as *positive TBox premises* and

negative TBox premises of the Ackermann$^{\mathsf{R}}$ rule, respectively. We refer to the clauses of the form $\neg\mathcal{S} \sqcup r$ and clauses of the form $\mathcal{S} \sqcup \neg r$ as *positive RBox premises* and *negative RBox premises* of the Ackermann$^{\mathsf{R}}$ rule, respectively. By $\mathcal{P}^{+}_{\mathcal{T}}(r)$ and $\mathcal{P}^{-}_{\mathcal{T}}(r)$ we denote the set of positive TBox premises and the set of negative TBox premises, respectively. By $\mathcal{P}^{+}_{\mathcal{R}}(r)$ and $\mathcal{P}^{-}_{\mathcal{R}}(r)$ we denote the set of positive RBox premises and the set of negative RBox premises, respectively. By $\mathcal{P}^{+}(r)$ and $\mathcal{P}^{-}(r)$ we denote the union of the set of positive TBox premises and the set of positive RBox premises, and the union of the set of negative TBox premises and the set of negative RBox premises, respectively, i.e., $\mathcal{P}^{+}(r) = \mathcal{P}^{+}_{\mathcal{T}}(r) \cup \mathcal{P}^{+}_{\mathcal{R}}(r)$ and $\mathcal{P}^{-}(r) = \mathcal{P}^{-}_{\mathcal{T}}(r) \cup \mathcal{P}^{-}_{\mathcal{R}}(r)$.

Ackermann$^{\mathsf{R}}$

$$\frac{\mathcal{N}, \overbrace{C_1 \sqcup \exists(r \sqcap Q_1).D_1, \ldots, C_m \sqcup \exists(r \sqcap Q_m).D_m}^{\mathcal{P}^{+}_{\mathcal{T}}(r)}, \overbrace{\neg s_1 \sqcup r, \ldots, \neg s_v \sqcup r}^{\mathcal{P}^{+}_{\mathcal{R}}(r)}, \overbrace{E_1 \sqcup \forall r.F_1, \ldots, E_n \sqcup \forall r.F_n}^{\mathcal{P}^{-}_{\mathcal{T}}(r)}, \overbrace{\neg r \sqcup t_1, \ldots, \neg r \sqcup t_w}^{\mathcal{P}^{-}_{\mathcal{R}}(r)}}{\mathcal{N}, \mathbf{Block}(\mathcal{P}^{-}(r), C_1 \sqcup \exists(r \sqcap Q_1).D_1), ..., \mathbf{Block}(\mathcal{P}^{-}(r), C_m \sqcup \exists(r \sqcap Q_m).D_m), \mathbf{Block}(\mathcal{P}^{-}(r), \neg s_1 \sqcup r), ..., \mathbf{Block}(\mathcal{P}^{-}(r), \neg s_v \sqcup r)}$$

provided: (i) $r \in \mathsf{N_R}$ is the pivot,
(ii) r does not occur in $\mathcal{N}$, and
(iii) the premises are in r-reduced form.

Notation in the Ackermann$^{\mathsf{R}}$ rule ($1 \leq i \leq m$, $1 \leq j \leq n$, $1 \leq k \leq v$, $1 \leq l \leq w$):

1. $\mathbf{Block}(\mathcal{P}^{-}(r), C_i \sqcup \exists(r \sqcap Q_i).D_i)$ denotes the union of the following sets:

Ground-tier: $\{C_i \sqcup \exists(t_1 \sqcap \ldots \sqcap t_w \sqcap Q_i).D_i\}$

1st-tier: $\bigcup_{1 \leq j \leq n} \{C_i \sqcup E_j \sqcup \exists(t_1 \sqcap \ldots \sqcap t_w \sqcap Q_i).(D_i \sqcap F_j)\}$

2nd-tier: $\bigcup_{1 \leq j_1 \leq j_2 \leq n} \{C_i \sqcup E_{j_1} \sqcup E_{j_2} \sqcup \exists(t_1 \sqcap \ldots \sqcap t_w \sqcap Q_i).(D_i \sqcap F_{j_1} \sqcap F_{j_2})\}$

...

nth-tier: $\{C_i \sqcup E_1 \sqcup \ldots \sqcup E_n \sqcup \exists(t_1 \sqcap \ldots \sqcap t_w \sqcap Q_i).(D_i \sqcap F_1 \sqcap \ldots \sqcap F_n)\}$.

2. $\mathbf{Block}(\mathcal{P}^{-}(r), \neg s_k \sqcup r)$ denotes the following set:

$\{E_1 \sqcup \forall s_k.F_1, \ldots, E_n \sqcup \forall s_k.F_n, \neg s_k \sqcup t_1, \ldots, \neg s_k \sqcup t_w\}$.

$\star$ $E_j = \begin{cases} \bot & \text{if } \mathcal{P}^{-}_{\mathcal{T}} = \emptyset \\ E_j & \text{otherwise,} \end{cases}$ $F_j = \begin{cases} \top & \text{if } \mathcal{P}^{-}_{\mathcal{T}} = \emptyset \\ F_j & \text{otherwise, and} \end{cases}$ $t_l = \begin{cases} \nabla & \text{if } \mathcal{P}^{-}_{\mathcal{R}} = \emptyset \\ t_l & \text{otherwise.} \end{cases}$

Figure 5.3: The Ackermann$^{\mathsf{R}}$ rule for eliminating $r \in \mathsf{N_R}$ from a set of clauses

The Ackermann$^{\mathsf{R}}$ rule, shown in Figure 5.3, is based on an idea of "*combination*".

Specifically, the idea is to combine all negative premises $\mathcal{P}^-(r)$ with every positive premise $\alpha(r)$ in $\mathcal{P}^+(r)$. The combination result is a finite set of $\mathcal{ALCOIH}(\triangledown, \sqcap)$-clauses in normal form, denoted by $\mathbf{Block}(\mathcal{P}^-(r), \alpha(r))$. Recall that the Ackermann$^{\mathsf{C}}$ rules (for concept forgetting) are based on an idea of "substitution", where we derive a definition of the pivot from all positive premises and substitute the definition for every negative occurrence of the pivot (or dually, we derive a definition of the pivot from all negative premises and substitute the definition for every positive occurrence of the pivot). In this way, the pivot is eliminated from the clause set. However, in the context of role forgetting, since a role symbol is always preceded by a role restriction operator, it is not obvious how to reformulate the pivot-clauses so that the pivot (role symbol) occurs at the top level of the clause. Thus, a definition of the pivot cannot be derived. To this end, we propose the notion "combination", where the substitution is performed in an implicit way. More specifically, combining all negative premises with every positive premise amounts to deriving a definition of the pivot from all negative premises and then substituting the definition for every positive occurrence of the pivot. Since the negative TBox premises cannot be transformed into the form suitable for deriving the definition of the pivot, we derive the definition via the interpretations of the negative TBox and RBox premises, shown in Table 5.1 and Table 5.2, respectively.

Negative TBox premises	Interpretations of negative TBox premises
$E_1 \sqcup \forall r.F_1$	$\forall x, y : (x, y) \in r^{\mathcal{I}} \rightarrow x \in E_1^{\mathcal{I}} \vee y \in F_1^{\mathcal{I}}$
$E_2 \sqcup \forall r.F_2$	$\forall x, y : (x, y) \in r^{\mathcal{I}} \rightarrow x \in E_2^{\mathcal{I}} \vee y \in F_2^{\mathcal{I}}$
$\ldots$	$\ldots$
$E_n \sqcup \forall r.F_n$	$\forall x, y : (x, y) \in r^{\mathcal{I}} \rightarrow x \in E_n^{\mathcal{I}} \vee y \in F_n^{\mathcal{I}}$

Table 5.1: Interpretations of negative TBox premises

Negative RBox premises	Interpretations of negative RBox premises
$\neg r \sqcup t_1$	$\forall x, y : (x, y) \in r^{\mathcal{I}} \rightarrow (x, y) \in {t_1}^{\mathcal{I}}$
$\neg r \sqcup t_2$	$\forall x, y : (x, y) \in r^{\mathcal{I}} \rightarrow (x, y) \in {t_2}^{\mathcal{I}}$
$\ldots$	$\ldots$
$\neg r \sqcup t_w$	$\forall x, y : (x, y) \in r^{\mathcal{I}} \rightarrow (x, y) \in {t_w}^{\mathcal{I}}$

Table 5.2: Interpretations of negative RBox premises

In particular, the definition of the pivot can be derived as follows:

$$\forall x, y : (x, y) \in r^{\mathcal{I}} \rightarrow (x \in E_1^{\mathcal{I}} \vee y \in F_1^{\mathcal{I}}) \wedge$$

$$
\begin{aligned}
&(x \in E_2^{\mathcal{I}} \vee y \in F_2^{\mathcal{I}}) \wedge \\
&\ldots \\
&(x \in E_n^{\mathcal{I}} \vee y \in F_n^{\mathcal{I}}) \wedge \\
&(x, y) \in {t_1}^{\mathcal{I}} \wedge \\
&(x, y) \in {t_2}^{\mathcal{I}} \wedge \\
&\ldots \\
&(x, y) \in {t_w}^{\mathcal{I}}
\end{aligned}
$$

We substitute this definition, derived from the interpretations of the negative premises, for the interpretation of every positive occurrence of the pivot. We first substitute this definition for the interpretation of the pivot in every $\neg s_k \sqcup r$ $(1 \leq k \leq v)$.

$$
\begin{aligned}
\forall x, y : (x, y) \in {s_k}^{\mathcal{I}} \rightarrow &(x \in E_1^{\mathcal{I}} \vee y \in F_1^{\mathcal{I}}) \wedge \\
&(x \in E_2^{\mathcal{I}} \vee y \in F_2^{\mathcal{I}}) \wedge \\
&\ldots \\
&(x \in E_n^{\mathcal{I}} \vee y \in F_n^{\mathcal{I}}) \wedge \\
&(x, y) \in {t_1}^{\mathcal{I}} \wedge \\
&(x, y) \in {t_2}^{\mathcal{I}} \wedge \\
&\ldots \\
&(x, y) \in {t_w}^{\mathcal{I}}
\end{aligned}
$$

Decoding the interpretation above into description logic expressions, we obtain:

$$\{E_1 \sqcup \forall s_k.F_1, \ldots, E_n \sqcup \forall s_k.F_n, \neg s_k \sqcup t_1, \ldots, \neg s_k \sqcup t_w\}.$$

The substitution of the interpretation of the pivot in every $C_i \sqcup \exists(r \sqcap Q_i).D_i$ $(1 \leq i \leq m)$ with the definition derived above is not as intuitive as in every $\neg s_k \sqcup r$ $(1 \leq k \leq v)$. We show the Ackermann$^{\mathsf{R}}$ rule is sound in the sense that the conclusion of the rule is equivalent to the premises of the rule up to the interpretation of the pivot.

Theorem 5.4.4. *The Ackermann$^{\mathsf{R}}$ rule preserves equivalence up to the interpretation of the pivot.*

Proof. In order to prove the Ackermann$^{\mathsf{R}}$ rule preserves equivalence up to the pivot, we need to prove that for any $\mathcal{ALCOIH}(\triangledown, \sqcap)$-interpretation $\mathcal{I}$, the conclusion of the

Ackermann$^{\mathsf{R}}$ rule is true in $\mathcal{I}$ iff for some interpretation $\mathcal{I}'$ pivot-equivalent to $\mathcal{I}$, the premises are true in $\mathcal{I}'$. We first prove the "if" direction, namely, the conclusion of the Ackermann$^{\mathsf{R}}$ rule is true in $\mathcal{I}$ if for some interpretation $\mathcal{I}'$ pivot-equivalent to $\mathcal{I}$, the premises are true in $\mathcal{I}'$. Let $\mathcal{I}$ be an $\mathcal{ALCOIH}(\triangledown, \sqcap)$-interpretation. Assume the conclusion of the rule is true in $\mathcal{I}$. Let $\mathcal{I}'$ be an interpretation extending $\mathcal{I}$ by additionally assigning r a binary relation of $\Delta^{\mathcal{I}'} \times \Delta^{\mathcal{I}'}$ such that

$$\forall x, y : (x, y) \in r^{\mathcal{I}'} = (x \in {E_1}^{\mathcal{I}'} \vee y \in {F_1}^{\mathcal{I}'}) \wedge \ldots \wedge (x \in {E_n}^{\mathcal{I}'} \vee y \in {F_n}^{\mathcal{I}'}) \wedge (x, y) \in {t_1}^{\mathcal{I}'} \wedge \ldots \wedge (x, y) \in {t_w}^{\mathcal{I}'}.$$

Directly, we have $\mathcal{I}' \models E_1 \sqcup \forall r.F_1, \ldots, E_n \sqcup \forall r.F_n, \neg r \sqcup t_1, \ldots, \neg r \sqcup t_w$. Because

$$\mathcal{I} \models E_1 \sqcup \forall s_k.F_1, \ldots, E_n \sqcup \forall s_k.F_n, \neg s_k \sqcup t_1, \ldots, \neg s_k \sqcup t_w,$$

we have

$$\mathcal{I}' \models E_1 \sqcup \forall s_k.F_1, \ldots, E_n \sqcup \forall s_k.F_n, \neg s_k \sqcup t_1, \ldots, \neg s_k \sqcup t_w.$$

Then we have:

$$\begin{aligned} \forall x, y : (x, y) \in {s_k}^{\mathcal{I}'} \rightarrow\ & (x \in E_1^{\mathcal{I}'} \vee y \in F_1^{\mathcal{I}'}) \wedge \\ & (x \in E_2^{\mathcal{I}'} \vee y \in F_2^{\mathcal{I}'}) \wedge \\ & \ldots \\ & (x \in E_n^{\mathcal{I}'} \vee y \in F_n^{\mathcal{I}'}) \wedge \\ & (x, y) \in {t_1}^{\mathcal{I}'} \wedge \\ & (x, y) \in {t_2}^{\mathcal{I}'} \wedge \\ & \ldots \\ & (x, y) \in {t_w}^{\mathcal{I}'}, \end{aligned}$$

which implies $\forall x, y : (x, y) \in {s_k}^{\mathcal{I}'} \rightarrow (x, y) \in r^{\mathcal{I}'}$. Similarly, we have

$$\forall x, y : (x, y) \in {s_1}^{\mathcal{I}'} \rightarrow (x, y) \in r^{\mathcal{I}'}, \ldots, (x, y) \in {s_v}^{\mathcal{I}'} \rightarrow (x, y) \in r^{\mathcal{I}'}.$$

Therefore, we have $\mathcal{I}' \models \neg s_1 \sqcup r, \ldots, \neg s_v \sqcup r$. Next, we need to prove

$$\mathcal{I}' \models C_1 \sqcup \exists r.D_1, \ldots, C_m \sqcup \exists r.D_m.$$

We prove this using SPASS,[1] an automated theorem prover for first-order logic with equality. In particular, we prove this as follows.

[1] https://www.mpi-inf.mpg.de/departments/automation-of-logic/software/spass-workbench/

1. Translating the premises and the conclusion of the Ackermann$^{\mathsf{R}}$ rule into first-order logic, denoted by FOL(premises) and FOL(Conclusion), respectively.

2. Encoding the assignment of the pivot into first-order logic:
   $\forall x \forall y(r(x,y) \equiv (E_1(x) \vee F_1(y)) \wedge \ldots \wedge (E_n(x) \vee F_n(y)) \wedge t_1(x,y) \wedge \ldots \wedge t_w(x,y))$,
   denoted by FOL(assignment).

3. Using SPASS to show: (FOL(assignment) $\wedge$ FOL(conclusion)) $\rightarrow$ FOL(premises).

Since $\mathcal{I} \models \mathcal{N}$ and the pivot r does not occur in $\mathcal{N}$, we have $\mathcal{I}' \models \mathcal{N}$. Therefore, the premises of the Ackermann$^{\mathsf{R}}$ rule are true in $\mathcal{I}'$.

Then we prove the "only-if" direction. Assume the premises are true in $\mathcal{I}'$. Directly, we have

$$\mathcal{I}' \models E_1 \sqcup \forall r.F_1, \ldots, E_n \sqcup \forall r.F_n \text{ and } \mathcal{I}' \models \neg r \sqcup t_1, \ldots, \neg r \sqcup t_w.$$

Since $\mathcal{I}' \models \neg s_k \sqcup r$ $(1 \leq k \leq v)$, we have

$$\mathcal{I}' \models E_1 \sqcup \forall s_k.F_1, \ldots, E_n \sqcup \forall s_k.F_n \text{ and } \mathcal{I}' \models \neg s_k \sqcup t_1, \ldots, \neg s_k \sqcup t_w.$$

Since r does not occur in the conclusion of the Ackermann$^{\mathsf{R}}$ rule, we have

$$\mathcal{I} \models E_1 \sqcup \forall s_k.F_1, \ldots, E_n \sqcup \forall s_k.F_n \text{ and } \mathcal{I} \models \neg s_k \sqcup t_1, \ldots, \neg s_k \sqcup t_w.$$

Therefore, we have

$$\mathcal{I} \models \mathbf{Block}(\mathcal{P}^-(r), \neg s_k \sqcup r)(1 \leq k \leq v).$$

Next, we show $\mathcal{I} \models \mathbf{Block}(\mathcal{P}^-(r), C_i \sqcup \exists(r \sqcap Q_i).D_i)$ $(1 \leq i \leq m)$. We first show the ground tier of $\mathbf{Block}(\mathcal{P}^-(r), C_i \sqcup \exists(r \sqcap Q_i).D_i)$ is true in $\mathcal{I}$. Since

$$\mathcal{I}' \models C_i \sqcup \exists(r \sqcap Q_i).D_i \text{ and } \mathcal{I}' \models \neg r \sqcup t_1, \ldots, \neg r \sqcup t_w,$$

we have

$$\mathcal{I}' \models C_i \sqcup \exists(t_1 \sqcap \ldots \sqcap t_w \sqcap Q_i).D_i.$$

Since r does not occur in $C_i \sqcup \exists(t_1 \sqcap \ldots \sqcap t_w \sqcap Q_i).D_i$, we have

$$\mathcal{I} \models C_i \sqcup \exists(t_1 \sqcap \ldots \sqcap t_w \sqcap Q_i).D_i.$$

Next, we show the first tier is true in $\mathcal{I}$. We do the proof by contradiction. Suppose there exists an element $d \in \Delta^{\mathcal{I}}$ such that

$$d \notin (C_i \sqcup E_j \sqcup \exists(t_1 \sqcap \ldots \sqcap t_w \sqcap Q_i).(D_i \sqcap F_j))^{\mathcal{I}} (1 \leq j \leq n).$$

Since the premises are true in $\mathcal{I}'$, we have

$$\mathcal{I}' \models C_i \sqcup \exists(r \sqcap Q_i).D_i \text{ and } \mathcal{I}' \models E_j \sqcup \forall r.F_j.$$

This means for every element $x \in \Delta^{\mathcal{I}'}$, we have

$$x \in (C_i \sqcup \exists(r \sqcap Q_i).D_i)^{\mathcal{I}'} \text{ and } x \in (E_j \sqcup \forall r.F_j)^{\mathcal{I}'},$$

which implies

$$x \in (C_i \sqcup E_j \sqcup \exists(r \sqcap Q_i).(D_i \sqcap F_j))^{\mathcal{I}'}.$$

Since $\mathcal{I}' \models r \sqsubseteq (t_1 \sqcap \ldots \sqcap t_w)$, we have

$$x \in (C_i \sqcup E_j \sqcup \exists(t_1 \sqcap \ldots \sqcap t_w \sqcap Q_i).(D_i \sqcap F_j))^{\mathcal{I}'}.$$

Since r does not occur in $C_i \sqcup E_j \sqcup \exists(t_1 \sqcap \ldots \sqcap t_w \sqcap Q_i).(D_i \sqcap F_j)^{\mathcal{I}'}$, we have

$$x \in (C_i \sqcup E_j \sqcup \exists(t_1 \sqcap \ldots \sqcap t_w \sqcap Q_i).(D_i \sqcap F_j)^{\mathcal{I}}.$$

Contradiction. The other tiers can be proved similarly. □

The conclusion of the Ackermann$^{\mathsf{R}}$ rule is the solution of forgetting the pivot from the premises of the rule. Let $\mathcal{O}$ be an $\mathcal{ALCOIH}(\triangledown, \sqcap)$-ontology in pivot-reduced form without the universal role and role conjunctions (i.e. an $\mathcal{ALCOIH}$-ontology), applying the Ackermann$^{\mathsf{R}}$ rule to $\mathcal{O}$ (to eliminate the pivot) could lead to the universal role and role conjunctions occurring in the conclusion. Specifically, applying the Ackermann$^{\mathsf{R}}$ rule to $\mathcal{O}$ lead to the universal role occurring in the conclusion if $\mathcal{P}_{\mathcal{T}}^{+} \neq \emptyset$, $\mathcal{P}_{\mathcal{T}}^{-} \neq \emptyset$ and $\mathcal{P}_{\mathcal{R}}^{-} = \emptyset$. Applying the Ackermann$^{\mathsf{R}}$ rule to $\mathcal{O}$ lead to role conjunctions occurring in the conclusion if $\mathcal{P}_{\mathcal{T}}^{+} \neq \emptyset$ and $|\mathcal{P}_{\mathcal{R}}^{-}| \geq 2$. Because of the nature of the Ackermann$^{\mathsf{R}}$ rule, role conjunctions only occur below the existential role restriction operator (i.e., they could never occur below the universal role restriction operator). The conclusion of the Ackermann$^{\mathsf{R}}$ rule is always expressible in $\mathcal{ALCOIH}(\triangledown, \sqcap)$.

Given a set of clauses in pivot-reduced form with n negative TBox premises ($|\mathcal{P}_{\mathcal{T}}^{-}| = n$), w negative RBox premises ($|\mathcal{P}_{\mathcal{R}}^{-}| = w$), m positive TBox premises ($|\mathcal{P}_{\mathcal{T}}^{+}| = m$) and

v positive RBox premises ($|\mathcal{P}^{+}_{\mathcal{R}}| = v$), combining $\mathcal{P}^{-}$ with every positive TBox premise yields $m \times 2^n$ clauses (**exponential growth**); combining $\mathcal{P}^{-}$ with every positive RBox premise yields $vn + vw$ clauses (**polynomial growth**). The size of the forgetting solution, therefore, depends largely on the number n of negative TBox premises.

5.4.3 The Purify$^{\mathsf{R}}$ Rules

The Ackermann$^{\mathsf{R}}$ rule is used when the pivot occurs both positively and negatively in the present clause set. For the cases where the pivot occurs only positively or only negatively (in the present clause set), we apply the Purify$^{\mathsf{R}}$ rules (to the present clause set) to eliminate the pivot. The Purify$^{\mathsf{R}}$ rules are shown in Figure 5.4.

Purify$^{\mathsf{R},+}$

$$\frac{\mathcal{N}}{\mathcal{N}^{r}_{\triangledown}}$$

provided: (i) $r \in \mathrm{sig}_{\mathsf{R}}(\mathcal{N})$ is the pivot, and
(ii) $\mathcal{N}$ is **positive** w.r.t. r.

Purify$^{\mathsf{R},-}$ (dual)

$$\frac{\mathcal{N}}{\mathcal{N}^{r}_{\neg\triangledown}}$$

provided: (i) $r \in \mathrm{sig}_{\mathsf{R}}(\mathcal{N})$ is the pivot, and
(ii) $\mathcal{N}$ is **negative** w.r.t. r.

Figure 5.4: The Purify$^{\mathsf{R}}$ rules for eliminating $r \in \mathsf{N}_{\mathsf{R}}$ from a set of clauses

The Purify$^{\mathsf{R}}$ rules do not require the present ontology to be transformed into pivot-reduced form (i.e., the Purify$^{\mathsf{R}}$ rules are form-independent). They can be applied at any time as long as the pivot has been found purifiable w.r.t. the present ontology (i.e., the present ontology is positive or negative w.r.t. the pivot). In particular, if the pivot occurs only positively in $\mathcal{N}$, we substitute the universal role for every occurrence of the pivot in $\mathcal{N}$. If the pivot occurs only negatively in $\mathcal{N}$, we substitute the negated universal role for every occurrence of the pivot in $\mathcal{N}$.

The Purify$^{\mathsf{R}}$ rules are special cases of the Ackermann$^{\mathsf{R}}$ rule in the sense that the sets of negative premises are empty (or the sets of positive premises are empty). This can be seen by adapting the Ackermann$^{\mathsf{R}}$ rule in the way as shown in Figure 5.5.

Purify$^{\mathsf{R,+}}$

$$\frac{\mathcal{N}, \overbrace{C_1 \sqcup \exists(r \sqcap Q_1).D_1, \ldots, C_m \sqcup \exists(r \sqcap Q_m).D_m}^{\mathcal{P}^+_{\mathcal{T}}(r)}, \overbrace{\neg s_1 \sqcup r, \ldots, \neg s_v \sqcup r}^{\mathcal{P}^+_{\mathcal{R}}(r)}, \quad \overbrace{\bot \sqcup \forall r.\top, \ldots, \bot \sqcup \forall r.\top}^{\mathcal{P}^-_{\mathcal{T}}(r)}, \overbrace{\neg r \sqcup \triangledown, \ldots, \neg r \sqcup \triangledown}^{\mathcal{P}^-_{\mathcal{R}}(r)}}{\begin{array}{r}\mathcal{N}, \mathbf{Block}(\mathcal{P}^-(r), C_1 \sqcup \exists(r \sqcap Q_1).D_1), ..., \mathbf{Block}(\mathcal{P}^-(r), C_m \sqcup \exists(r \sqcap Q_m).D_m), \\ \mathbf{Block}(\mathcal{P}^-(r), \neg s_1 \sqcup r), ..., \mathbf{Block}(\mathcal{P}^-(r), \neg s_v \sqcup r)\end{array}}$$

Purify$^{\mathsf{R,-}}$ (dual)

$$\frac{\mathcal{N}, \overbrace{\top \sqcup \exists(r \sqcap Q_1).D_1, \ldots, \top \sqcup \exists(r \sqcap Q_m).D_m}^{\mathcal{P}^+_{\mathcal{T}}(r)}, \overbrace{\triangledown \sqcup r, \ldots, \triangledown \sqcup r}^{\mathcal{P}^+_{\mathcal{R}}(r)}, \quad \overbrace{E_1 \sqcup \forall r.F_1, \ldots, E_n \sqcup \forall r.F_n}^{\mathcal{P}^-_{\mathcal{T}}(r)}, \overbrace{\neg r \sqcup t_1, \ldots, \neg r \sqcup t_w}^{\mathcal{P}^-_{\mathcal{R}}(r)}}{\begin{array}{r}\mathcal{N}, \mathbf{Block}(\mathcal{P}^-(r), \top \sqcup \exists(r \sqcap Q_1).D_1), ..., \mathbf{Block}(\mathcal{P}^-(r), \top \sqcup \exists(r \sqcap Q_m).D_m), \\ \mathbf{Block}(\mathcal{P}^-(r), \triangledown \sqcup r), ..., \mathbf{Block}(\mathcal{P}^-(r), \triangledown \sqcup r)\end{array}}$$

Figure 5.5: The Purify$^{\mathsf{R}}$ rules in the sense of the Ackermann$^{\mathsf{R}}$ rule

Note that the pivot occurs negatively only in negative TBox and RBox premises. In particular, if the pivot occurs in negative TBox premises, then every occurrence of the pivot in them is replaced by the negated universal role. This yields the clause set $\{E_1 \sqcup \forall\neg\triangledown.F_1, \ldots, E_n \sqcup \forall\neg\triangledown.F_n\}$, where every $E_j \sqcup \forall\neg\triangledown.F_j$ $(1 \leq j \leq n)$ is a tautology. If the pivot occurs in negative RBox premises, then every occurrence of the pivot in them is replaced by the negated universal role. This yields the clause set $\{\triangledown \sqcup t_1, \ldots, \triangledown \sqcup t_w\}$, where every $\triangledown \sqcup t_k$ $(1 \leq k \leq w)$ is a tautology. This means that the negated universal role will never occur in the conclusion of the Purify$^{\mathsf{R}}$ rules. The conclusion of the Purify$^{\mathsf{R}}$ rules is always expressible in $\mathcal{ALCOIH}(\triangledown, \sqcap)$.

Theorem 5.4.5. *The Purify$^{\mathsf{R}}$ rules preserve equivalence up to the interpretation of the pivot.*

Proof. The Purify$^{\mathsf{R}}$ rules are special cases of the Ackermann$^{\mathsf{R}}$ rule (see Figure 5.5). They preserve equivalence up to the interpretation of the pivot in the sense of the

Ackermann$^{\mathsf{R}}$ rule. They preserve equivalence up to the interpretation of the pivot also in the sense of the Ackermann$^{\mathsf{C}}$ rules. □

So far, we have described all the key ingredients of the calculus ACK$^{\mathsf{R}}$. These ingredients include three rewrite rules for transforming normalised ontologies into pivot-reduced form, an Ackermann$^{\mathsf{R}}$ rule for eliminating a single role symbol from ontologies in pivot-reduced form, and a pair of Purify$^{\mathsf{R}}$ rules for eliminating a single role symbol from ontologies that are positive or negative w.r.t. this symbol. The conclusions of these rules are always in normal form. This means that ontology normalisation can be performed only once. In particular, given an $\mathcal{ALCOIH}(\triangledown, \sqcap)$-ontology $\mathcal{O}$ (i.e., usually the given ontology is an $\mathcal{ALCOIH}$-ontology, because most real-world ontologies do not include the universal role and role conjunctions), once $\mathcal{O}$ has been clausified into a set of clauses, the immediate next step is to normalise $\mathcal{O}$ w.r.t. Σ using definer introduction. The result is a finite set $\mathcal{N}$ of clauses with Σ-clauses being in normal form. Since definer symbols might have been introduced during the normalisation process, the input to ACK$^{\mathsf{R}}$ is a set $\mathcal{N}$ of clauses possibly with definer symbols. From $\mathcal{N}$, we eliminate the symbols in Σ using the rules in ACK$^{\mathsf{R}}$. Since the conclusions of the rules in ACK$^{\mathsf{R}}$ are always normalised ontologies expressible in $\mathcal{ALCOIH}(\triangledown, \sqcap)$, Σ-symbols can be eliminated by repeated use of ACK$^{\mathsf{R}}$. In the next subsection, we investigate several important properties of ACK$^{\mathsf{R}}$.

5.4.4 Properties of Ack$^{\mathsf{R}}$

ACK$^{\mathsf{R}}$ is a calculus for eliminating a single role symbol from a normalised set of clauses expressible in $\mathcal{ALCOIH}(\triangledown, \sqcap)$. Let $\mathcal{N}$ be a normalised set of clauses expressible in $\mathcal{ALCOIH}(\triangledown, \sqcap)$ and let r be a role symbol in $\mathcal{N}$. We say a derivation in ACK$^{\mathsf{R}}$ is *successful* w.r.t. r, if r does not occur in the result $\mathcal{N}'$ of the derivation. We say a derivation in ACK$^{\mathsf{R}}$ is *unsuccessful* (or *fails*) w.r.t. r, otherwise.

In this subsection, we show termination, soundness and partialcompleteness of ACK$^{\mathsf{R}}$. In particular, we show that: (i) ACK$^{\mathsf{R}}$ is *terminating*, i.e. any ACK$^{\mathsf{R}}$-derivation terminates, (ii) ACK$^{\mathsf{R}}$ is *sound*, i.e., the resulting set $\mathcal{N}'$ of any successful ACK$^{\mathsf{R}}$-derivation is equivalent to the original set $\mathcal{N}$ up to the interpretation of the pivot in the derivation, and (iii) ACK$^{\mathsf{R}}$ is *partially complete* for $\mathcal{ALCOIH}(\triangledown, \sqcap)$-ontologies, namely,

ACK$^{\mathsf{R}}$ is complete for $\mathcal{ALCO}(\triangledown)$-ontologies and *incomplete* for $\mathcal{ALCOIH}(\triangledown, \sqcap)$-ontologies.

Theorem 5.4.6. ACK^{R} *is terminating and sound.*

Proof. In order to show ACK$^{\mathsf{R}}$ is terminating, we need to show there is no infinite loop in each step of the ACK$^{\mathsf{R}}$-derivation. If the pivot does not occur in the present ontology, then the current ACK$^{\mathsf{R}}$-derivation terminates immediately and returns the present ontology. If the pivot occurs only positively or only negatively in the present ontology (in normal form), then one of the Purify$^{\mathsf{R}}$ rules of ACK$^{\mathsf{R}}$ can be applied to eliminate the pivot. The current ACK$^{\mathsf{R}}$-derivation terminates and returns the resulting ontology. If the pivot occurs both positively and negatively in the present ontology (in normal form), the ontology is first transformed into pivot-reduced form using the rewrite rules in Figure 5.2. Observe that no rules are applicable to the conclusions of these rewrite rules except for the Ackermann$^{\mathsf{R}}$ rule. This means that there is no infinite loop in the transformation of the present ontology into pivot-reduced form. If the present ontology cannot be transformed into pivot-reduced form, then the current ACK$^{\mathsf{R}}$-derivation terminates and returns the present ontology, which still contains the pivot. If the present ontology has been transformed into pivot-reduced form, then the Ackermann$^{\mathsf{R}}$ rule can be applied to the ontology to eliminate the pivot. The current ACK$^{\mathsf{R}}$-derivation terminates and returns the resulting ontology.

We show ACK$^{\mathsf{R}}$ is sound. ACK$^{\mathsf{R}}$ is defined in terms of three rewrite rules, the Ackermann$^{\mathsf{R}}$ rule and a pair of Purify$^{\mathsf{R}}$ rules. We have shown that the rewrite rules preserve logical equivalence, the Ackermann$^{\mathsf{R}}$ rule preserves equivalence up to the interpretation of the pivot, and the Purify$^{\mathsf{R}}$ rules preserve equivalence up to the interpretation of the pivot. Thus, the result of any successful ACK$^{\mathsf{R}}$-derivation is equivalent to the original ontology up to the interpretation of the pivot. □

Theorem 5.4.7. ACK^{R} *is (role forgetting) complete for* $\mathcal{ALCO}(\triangledown)$*-ontologies.*

Proof. Let $\mathcal{O}$ be any $\mathcal{ALCO}(\triangledown)$-ontology and let r be any role symbol in $\mathcal{O}$. If r occurs only positively or only negatively in $\mathcal{O}$, then the Purify$^{\mathsf{R}}$ rules can be applied to $\mathcal{O}$ to eliminate r. If r occurs both positively and negatively in $\mathcal{O}$, then $\mathcal{O}$ needs to be transformed into r-reduced form first. We have shown in Section 5.4 that any $\mathcal{ALCO}(\triangledown)$-ontologies can be transformed into pivot-reduced form (using definer introduction). Thus, the transformation always succeeds. The Ackermann$^{\mathsf{R}}$ rule can be

applied to $\mathcal{O}$ (in r-reduced form) to eliminate r. □

ACK$^{\mathsf{R}}$ is (role forgetting) incomplete for $\mathcal{ALCOIH}(\triangledown, \sqcap)$-ontologies due to the presence of inverse roles and role inclusions. In particular, cases where ACK$^{\mathsf{R}}$ will definitely fail include:

(i) if the present ontology contains TBox clauses of the form $C \sqcup \exists(r^- \sqcap Q).D$, where r is the pivot and occurs both positively and negatively in the ontology, and $r^- \sqcap Q$ is a role that cannot be further simplified (i.e., the inverse pivot occurs below an existential role restriction and cannot be eliminated via the use of simplification rules);

(ii) if the present ontology contains RBox clauses of the form $\neg r \sqcup r^-$ or $\neg r^- \sqcup r$ (i.e., r is a symmetric role), where the pivot and the inverse pivot occur simultaneously in an RBox clause.

A justification of this is the following: the pivot r in these cases occurs both positively and positively in the present ontology, which means only the Ackermann$^{\mathsf{R}}$ rule can be applied to eliminate the pivot (i.e., the Purify$^{\mathsf{R}}$ rules are not applicable). Applying the Ackermann$^{\mathsf{R}}$ rule requires the present ontology to be transformed into pivot-reduced form, but there are no rules in ACK$^{\mathsf{R}}$ allowing clauses of the form $C \sqcup \exists(r^- \sqcap Q).D$, $r \sqcup r^-$ or $\neg r^- \sqcup r$ to be transformed into pivot-reduced form.

5.5 The Forgetting Method

In the previous section, we introduced a dedicated calculus ACK$^{\mathsf{R}}$ for eliminating a single role symbol from a normalised set of clauses expressible in $\mathcal{ALCOIH}(\triangledown, \sqcap)$. In this section, we present a practical method based on ACK$^{\mathsf{R}}$ for forgetting a set of role symbols from an ontology expressible in $\mathcal{ALCOIH}(\triangledown, \sqcap)$. Following ACK$^{\mathsf{R}}$, the method is (role forgetting) complete for $\mathcal{ALCOH}(\triangledown, \sqcap)$-ontologies and is (role forgetting) incomplete for $\mathcal{ALCOIH}(\triangledown, \sqcap)$-ontologies.

5.5.1 The Forgetting Process

Given an $\mathcal{ALCOIH}(\triangledown, \sqcap)$-ontology $\mathcal{O}$ of axioms and a set $\Sigma \subseteq \mathrm{sig}_{\mathsf{R}}(\mathcal{O})$ of role symbols to be forgotten, the forgetting process in our method consists of five main phases (see

Figure 5.6): the conversion of $\mathcal{O}$ into a set $\mathcal{N}$ of clauses (**the clausification phase**), the conversion of Σ-clauses into normal form (**the normalisation phase**), the Σ-symbol elimination phase (**the central phase**), **the definer elimination phase**, and the conversion of the resulting clause set $\mathcal{N}'$ into an ontology $\mathcal{O}'$ of axioms (**the declausification phase**). It is assumed that as soon as a forgetting solution has been computed, then the remaining phases are skipped.

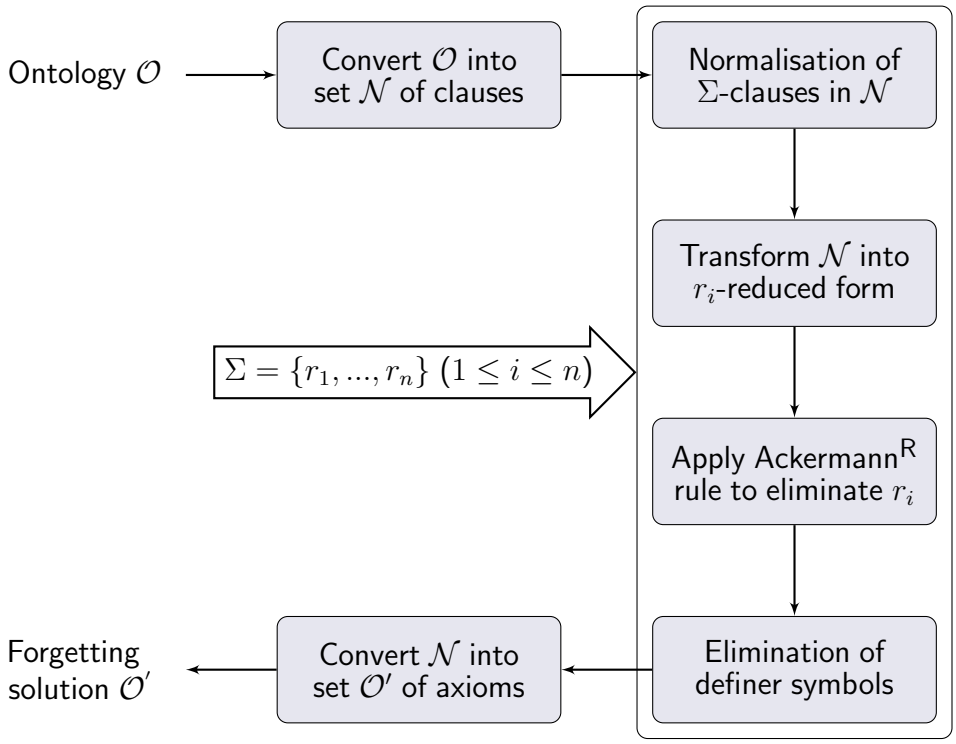

Figure 5.6: The main phases in role forgetting process

Input: Given as input to the method are an $\mathcal{ALCOIH}(\triangledown, \sqcap)$-ontology $\mathcal{O}$ of TBox, ABox and RBox axioms and a set $\Sigma \subseteq \text{sig}_{\mathsf{R}}(\mathcal{O})$ of role symbols to be forgotten. An important feature of the method is that Σ-symbols can be flexibly specified.

The clausification phase: The initial phase of the forgetting process is the clausification phase, where the method internalises all ABox assertions in $\mathcal{O}$ (if they are present in $\mathcal{O}$) into TBox axioms, and then transforms $\mathcal{O}$ into a set $\mathcal{N}$ of clauses using the standard (clausal form) transformation rules in Figures 4.1 and 4.2, as well as the rules in Figure 5.7, which are used to transform RBox axioms into RBox clauses (i.e., because $\mathcal{ALCOIH}(\triangledown, \sqcap)$-ontologies additionally have an RBox).

The normalisation phase: The normalisation phase of the forgetting process

RBox axioms into clauses:

$$\mathcal{S} \sqsubseteq \mathcal{S}' \quad \Longrightarrow \quad \neg\mathcal{S}' \sqcup \mathcal{S}$$
$$\mathcal{S} \equiv \mathcal{S}' \quad \Longrightarrow \quad \neg\mathcal{S} \sqcup \mathcal{S}', \neg\mathcal{S}' \sqcup \mathcal{S}$$

Figure 5.7: Transformation of RBox axioms into RBox clauses

transforms all Σ-clauses in $\mathcal{N}$ into normal form using definer introduction. The normalisation is not applied to the other clauses because they are not involved in the elimination of Σ-symbols. As a result, for efficiency of our method, non-Σ-clauses will not be processed. We have shown in the previous section that the result of an ACK$^{\mathsf{R}}$-derivation is always a set of clauses in normal form. This means that the normalisation does not need to be repeatedly performed during the Σ-symbol elimination process. Our method performs the normalisation immediately after the clausification. This ensures that ACK$^{\mathsf{R}}$ works with a normalised clause set when eliminating symbols in Σ. By normalisation, TBox Σ-clauses are transformed into a specialised form that contains a single role restriction (i.e., the pivot occurs in this role restriction). RBox Σ-clauses remain unchanged (i.e., because they are always in normal form).

The central phase: Central to the forgetting process is the Σ-symbol elimination phase, which is an iteration of several rounds (i.e., ACK$^{\mathsf{R}}$-derivations) in which the elimination of Σ-symbols is attempted. Specifically, the method attempts to eliminate the symbols in Σ one by one using the calculus ACK$^{\mathsf{R}}$ as described in the previous section. In each elimination round, the method (normally) performs two steps. The first step attempts to transform every (TBox and RBox) pivot-clause (not in pivot-reduced form) into pivot-reduced form using the rewrite$^{\mathsf{R}}$ rules in Figure 5.2, so that the Ackermann$^{\mathsf{R}}$ rule can be applied. If the transformation is successful, then the second step applies the Ackermann$^{\mathsf{R}}$ rule to the pivot-clauses (i.e., the premises) to eliminate the pivot. If the transformation is not successful, then the method skips the current round and attempts to eliminate another symbol in Σ (using ACK$^{\mathsf{R}}$). On the intermediate result being returned at the end of each round, the method repeats the same steps in the next round for the elimination of the remaining symbols in Σ (if necessary). If the pivot is found purifiable w.r.t. the present clause set (i.e., the pivot occurs only positively or only negatively in the present clause set), then the method

performs only one single step in the elimination round. In particular, the method applies one of the Purify$^{\mathsf{R}}$ rules to the present clause set to eliminate the pivot.

As with our method for concept forgetting, the method for role forgetting is additionally equipped with a frequency counter, which is used to count the frequency of positive and negative occurrences of the pivot. In particular, that both the frequencies of positive and negative occurrences of the pivot are zero means that the pivot does not occur in the present clause set. That the frequency of negative occurrence of the pivot is zero and the frequency of positive occurrence of the pivot is non-zero means that the pivot occurs only positively in the present clause set. That the frequency of positive occurrence of the pivot is zero and the frequency of negative occurrence of the pivot is non-zero means that the pivot occurs only negatively in the present clause set. That both the frequencies of positive and negative occurrences of the pivot are non-zero means that the pivot occurs both positively and negatively in the present ontology. Based on the frequency counts of the pivot, the method determines which elimination rule is to be applied (to eliminate the pivot).

The definer elimination phase: To facilitate the transformation of TBox pivot-clauses (not in normal form) into normal form, definer symbols might have been introduced during the normalisation process. However, the forgetting solution is supposed not to contain these definer symbols, because they are not part of the desired signature. This phase attempts to eliminate these definer symbols using our method for concept forgetting as described in the previous chapter, which can directly be used for forgetting concept definer symbols from $\mathcal{ALCOIH}(\triangledown, \sqcap)$-ontologies. For improved success rates of the method, a new pair of rewrite rules, namely, the Skolemisation$^{\triangledown}$ rules, are added to Ack$^{\mathsf{C}}$ to handle the universal role. They are shown in Figure 5.8.

Theorem 5.5.1. *The Skolemisation$^{\triangledown}$ rules equivalence up to the interpretation of the introduced nominal.*

Proof. We do the proof by contradiction. We first prove the "top-down" direction. Let $\mathcal{I}$ be any $\mathcal{ALCOIH}(\triangledown, \sqcap)$-interpretation such that $\mathcal{I} \models C \sqcup \exists\triangledown.D$. Suppose for every element $x \in \Delta^{\mathcal{I}}$ we have $x \notin (D \sqcup \exists\triangledown.C)^{\mathcal{I}}$

$$x \notin (D \sqcup \exists\triangledown.C)^{\mathcal{I}}$$
$$\Rightarrow x \notin D^{\mathcal{I}} \text{ and } x \notin (\exists\triangledown.C)^{\mathcal{I}}$$

Skolemisation$^{\nabla,+}$

$$\frac{\mathcal{N}, C \sqcup \exists \nabla.D}{\mathcal{N}, \neg a \sqcup D \sqcup \forall \nabla.C}$$

provided: (i) $A \in \text{sig}_{\mathsf{C}}(\mathcal{N})$ is the pivot,
(ii) A does not occur in C,
(iii) A occurs **positively** in D, and
(iv) $a \in \mathsf{N_O}$ is a fresh nominal.

Skolemisation$^{\nabla,-}$

$$\frac{\mathcal{N}, C \sqcup \exists \nabla.D}{\mathcal{N} \neg a \sqcup D \sqcup \forall \nabla.C}$$

provided: (i) $A \in \text{sig}_{\mathsf{C}}(\mathcal{N})$ is the pivot,
(ii) A does not occur in C,
(iii) A occurs **negatively** in D, and
(iv) $a \in \mathsf{N_O}$ is a fresh nominal.

Figure 5.8: The Skolemisation$^{\nabla}$ rules for transforming A-clauses into A-reduced form

This means there exists an element $d \in \Delta^{\mathcal{I}}$ such that $d \notin C^{\mathcal{I}}$. Since the premise of the rule is true in $\mathcal{I}$, for every element $x' \in \Delta^{\mathcal{I}}$, we have $x' \in (C \sqcup \exists \nabla.D)^{\mathcal{I}}$.

$$x' \in (C \sqcup \exists \nabla.D)^{\mathcal{I}}$$
$$\Rightarrow x' \in C^{\mathcal{I}} \text{ or } x' \in (\exists \nabla.D)^{\mathcal{I}}$$

This means that there exists an element $d' \in \Delta'$ such that $d' \in D^{\mathcal{I}}$. Observe that $d' \in D^{\mathcal{I}}$ contradicts with $x \in D'$, and $x' \in C^{\mathcal{I}}$ contradicts with $d \notin C^{\mathcal{I}}$. **Contradiction**. Next, we prove the "bottom-up" direction (by contradiction). Let $\mathcal{I}$ be any $\mathcal{ALCOIH}(\nabla, \sqcap)$-interpretation such that $\mathcal{I} \models \neg a \sqcup D \sqcup \forall \nabla.C$. Suppose there exists an element $d \in \Delta^{\mathcal{I}}$ such that $d \notin (C \sqcup \exists \nabla.D)$.

$$d \notin (C \sqcup \exists \nabla.D)$$
$$\Rightarrow d \notin C^{\mathcal{I}} \text{ and } d \notin (\exists \nabla.D)^{\mathcal{I}}$$

This means that for every $y \in \Delta^{\mathcal{I}}$, we have $y \notin D^{\mathcal{I}}$. Since the conclusion of the rule is true in $\mathcal{I}$, for every element $x' \in \Delta^{\mathcal{I}}$, we have $x' \in (\neg a \sqcup D \sqcup \forall \nabla.C)^{\mathcal{I}}$.

$$x \in (\neg a \sqcup D \sqcup \forall \nabla.C)^{\mathcal{I}}$$

$$\Rightarrow a \in (D \sqcup \forall \triangledown.C)^{\mathcal{I}}$$
$$\Rightarrow a \in D^{\mathcal{I}} \text{ or } a \in (\forall \triangledown.C)^{\mathcal{I}}$$

This means that for every $y' \in \Delta^{\mathcal{I}}$, we have $y' \in C^{\mathcal{I}}$. Observe that $y' \in C^{\mathcal{I}}$ contradicts with $d \notin C^{\mathcal{I}}$, and $y \in D^{\mathcal{I}}$ contradicts with $a \notin C^{\mathcal{I}}$. **Contradiction**. □

The method, based on ACK$^{\mathsf{C}}$ extended with the Skolemisation$^{\triangledown}$ rules, allows definer symbols to be eliminated in many cases, but there is no guarantee that definer symbols can be eliminated in all cases. This is because our method (for concept forgetting) is an incomplete method. Nevertheless, in practice, most real-world ontologies are normalised ontologies where the Σ-clauses are already in normal form. This means for most real-world ontologies definer introduction and elimination are obsolete.

The declausification phase: The declausification phase of the forgetting process transforms the clause set obtained from the previous phase into $\mathcal{ALCOIH}(\triangledown, \sqcap)$-ontology axioms. The transformation is based on the use of the rules in Figures 4.11, as well as the rules in Figure 5.9.

RBox clauses into axioms:

$$\begin{array}{rcl} \neg\mathcal{S}' \sqcup \mathcal{S} & \Longrightarrow & \mathcal{S} \sqsubseteq \mathcal{S}' \\ \neg\mathcal{S} \sqcup \mathcal{S}', \neg\mathcal{S}' \sqcup \mathcal{S} & \Longrightarrow & \mathcal{S} \equiv \mathcal{S}' \end{array}$$

Figure 5.9: Transformation of RBox clauses into RBox axioms

Output: What the method returns as output at the end of the forgetting process is an ontology $\mathcal{O}'$ of axioms. If $\mathcal{O}'$ does not contain any symbols in Σ, then the method has been *successful* (in computing a solution of forgetting Σ from $\mathcal{O}$). If $\mathcal{O}'$ still contains some symbols in Σ, then the method has been *unsuccessful* (or *has failed*).

5.5.2 The Elimination Order

The elimination order refers to the order in which the Σ-symbols are eliminated. We have shown that the elimination order is crucial to our method for concept forgetting because changing the order of eliminating Σ-symbols may affect the efficiency and success rates of the method. While in role forgetting, changing the order of eliminating

Σ-symbols does not affect the efficiency and success rates of our method too much. An important reason is that when using our Ackermann-based method to solve a role forgetting problem, in most cases, the elimination of a Σ-symbol is independent with the elimination of another Σ-symbol in the forgetting process. Thus, the elimination of a Σ-symbol will not affect (the efficiency and success of) the elimination of another Σ-symbol. This is different for concept forgetting, where the elimination of a concept Σ-symbol may significantly affect (the efficiency and success of) the elimination of the another concept Σ-symbol; see Examples 4.5.3 and 4.5.7. In this sense, no heuristic analysis is needed for role forgetting.

Why is the elimination of two role symbols mutually independent in role forgetting? In order to justify this, we consider two cases, i.e., the case of eliminating role symbols from $\mathcal{ALCOI}$-ontologies and the case of eliminating role symbols from $\mathcal{ALCOIH}$-ontologies (i.e., most given ontologies are real-world ontologies which do not include the universal role and role conjunctions).

In the case of $\mathcal{ALCOI}$-ontologies, when the given ontology has been transformed into a set of clauses in normal form, each Σ-clause in the clause set contains a single role restriction with a single role symbol. In this case, the clause set can be split into several clause subsets with one of them not containing any symbols in Σ, and with each of the other subsets containing only a specific symbol in Σ. For example, let $\mathcal{N}$ be a set of $\mathcal{ALCOI}$-clauses in normal form, and let $\Sigma = \{r_1, \ldots r_n\}$ be any subset of $\mathsf{sig}_{\mathsf{R}}(\mathcal{N})$. Then $\mathcal{N}$ can be split into several clause subsets as follows:

$$\mathcal{N} = \mathcal{N}^{-\Sigma} \cup \mathcal{N}(r_1) \cup \ldots \cup \mathcal{N}(r_n),$$

where $\mathcal{N}^{-\Sigma}$ denotes the clause set that does not contain any symbols in Σ, and each $\mathcal{N}(r_i)$ $(1 \leq i \leq n)$ denotes the clause set that contains only the role symbol r_i. Thus the elimination of a Σ-symbol is performed on each $\mathcal{N}(r_i)$, rather than the entire clause set $\mathcal{N}$ (i.e., the elimination of r_i is *locally* performed). The forgetting solution is the union of the results obtained from each elimination of r_i. In this way, it is easy to see that eliminating a role symbol is independent with eliminating another role symbol from an $\mathcal{ALCOI}$-ontology. This can also be illustrated with the following example.

Example 5.5.2. Consider the following set of $\mathcal{ALCOI}$-clauses $\mathcal{N}$:

1. $A \sqcup \forall s.\exists r.B$

2. $\forall r.B \sqcup \exists r.\neg C$

3. $\forall s.C \sqcup \exists s.A$

Assume $\Sigma = \{r, s\}$. Observe that Clauses 1, 2 and 3 are not in normal form. Using definer introduction, $\mathcal{N}$ is transformed into the following clause set:

1. $A \sqcup \forall s.\mathsf{D}_1$

2. $\neg\mathsf{D}_1 \sqcup \exists r.B$

3. $\mathsf{D}_2 \sqcup \exists r.\neg C$

4. $\neg\mathsf{D}_2 \sqcup \forall r.B$

5. $\mathsf{D}_3 \sqcup \exists s.A$

6. $\neg\mathsf{D}_3 \sqcup \forall s.C$

Then $\mathcal{N}$ can be splitted into two clause subsets $\mathcal{N}(r)$ and $\mathcal{N}(s)$ as follows:

2. $\neg\mathsf{D}_1 \sqcup \exists r.B$	1. $A \sqcup \forall s.\mathsf{D}_1$
3. $\mathsf{D}_2 \sqcup \exists r.\neg C$	5. $\mathsf{D}_3 \sqcup \exists s.A$
4. $\neg\mathsf{D}_2 \sqcup \forall r.B$	6. $\neg\mathsf{D}_3 \sqcup \forall s.C$

We apply the Ackermann$^{\mathsf{R}}$ rule to $\mathcal{N}(r)$ to eliminate r and to $\mathcal{N}(s)$ to eliminate s. The elimination of these two symbols is mutually independent. The forgetting solution is the union of the results obtained from each elimination.

The only cases where the elimination of a Σ-symbol interacts with the elimination of another Σ-symbol are the cases where the two Σ-symbols occur simultaneously in RBox clauses (i.e., eliminating role symbols from $\mathcal{ALCOIH}$-ontologies), because RBox clauses in normal form contain two role symbols (i.e., whereas TBox clauses in normal form contain only one role symbol and thus different Σ-symbols that occur in TBox clauses can be isolated into different clause subsets).

Example 5.5.3. Consider the following set of $\mathcal{ALCOIH}$-clauses $\mathcal{N}$:

1. $\neg A_1 \sqcup \exists r.B_1$

2. $\neg A_2 \sqcup \forall r.B_2$

3. $\neg C_1 \sqcup \exists s.D_1$

4. $\neg C_2 \sqcup \forall s.D_2$

5. $\neg r \sqcup s$

Assume $\Sigma = \{r, s\}$. Observe that the two Σ-symbols r and s occur simultaneously in Clause 5. If $r \succ s$, we apply the Ackermann$^{\mathsf{R}}$ rule to $\mathcal{N}$ to eliminate r, thereby yielding the following clause set:

1. $\neg A_1 \sqcup \exists s.B_1$

2. $\neg A_1 \sqcup \neg A_2 \sqcup \exists s.(B_1 \sqcap B_2)$

3. $\neg C_1 \sqcup \exists s.D_1$

4. $\neg C_2 \sqcup \forall s.D_2$

Then we apply the Ackermann$^{\mathsf{R}}$ rule to the clause set above to eliminate s, thereby yielding the following forgetting solution $\mathcal{N}'$:

1. $\neg A_1 \sqcup \exists \nabla.B_1$

2. $\neg A_1 \sqcup \neg C_2 \sqcup \exists \nabla.(B_1 \sqcap D_2)$

3. $\neg A_1 \sqcup \neg A_2 \sqcup \exists \nabla.(B_1 \sqcap B_2)$

4. $\neg A_1 \sqcup \neg A_2 \sqcup \neg C_2 \sqcup \exists \nabla.(B_1 \sqcap B_2 \sqcap D_2)$

5. $\neg C_1 \sqcup \exists \nabla.D_1$

6. $\neg C_1 \sqcup \neg C_2 \sqcup \exists \nabla.(D_1 \sqcap D_2)$

If $s \succ r$, we apply the Ackermann$^{\mathsf{R}}$ rule to the original $\mathcal{N}$ to eliminate s, thereby yielding the following clause set:

1. $\neg A_1 \sqcup \exists r.B_1$

2. $\neg A_2 \sqcup \forall r.B_2$

3. $\neg C_1 \sqcup \exists \nabla.D_1$

4. $\neg C_1 \sqcup \neg C_2 \sqcup \exists \nabla.(D_1 \sqcap D_2)$

5. $\neg C_2 \sqcup \forall r.D_2$

Then we apply the Ackermann$^{\mathsf{R}}$ rule to the clause set above to eliminate r, thereby yielding the following forgetting solution $\mathcal{N}''$:

1. $\neg A_1 \sqcup \exists \nabla.B_1$

2. $\neg A_1 \sqcup \neg C_2 \sqcup \exists \triangledown.(B_1 \sqcap D_2)$
3. $\neg A_1 \sqcup \neg A_2 \sqcup \exists \triangledown.(B_1 \sqcap B_2)$
4. $\neg A_1 \sqcup \neg A_2 \sqcup \neg C_2 \sqcup \exists \triangledown.(B_1 \sqcap B_2 \sqcap D_2)$
5. $\neg C_1 \sqcup \exists \triangledown.D_1$
6. $\neg C_1 \sqcup \neg C_2 \sqcup \exists \triangledown.(D_1 \sqcap D_2)$

Observe that $\mathcal{N}'$ and $\mathcal{N}''$ are identical. Thus, changing the elimination order has not affected the efficiency and success of the method too much.

Although the elimination order in role forgetting is not as important as it is in concept forgetting, there are still search heuristics worth implementing. For example, Σ-symbols that are purifiable w.r.t. the present clause set should be eliminated first, because there is a great chance that the clause set after purification is significantly reduced. A justification of this is the following: recall that clauses in normal form are of the form $C \sqcup \exists R.D$, $C \sqcup \forall \mathcal{S}.D$, $\neg \mathcal{S} \sqcup \mathcal{S}'$ or $\neg \mathcal{S}' \sqcup \mathcal{S}$. If the pivot occurs only positively in the normalised clause set, and in particular, the pivot occurs in clauses of the form $C \sqcup \exists R.D$, then substituting the universal role for every occurrence of the pivot in $C \sqcup \exists R.D$ yields $C \sqcup \exists \triangledown.D$. In this way, the other role symbols occurring in the role R are eliminated as well. If the pivot occurs in $\mathcal{S}$ of the clauses $\neg \mathcal{S}' \sqcup \mathcal{S}$, then substituting the universal role for every occurrence of the pivot in $\neg \mathcal{S}' \sqcup \mathcal{S}$ yields $\top$. On the other hand, if the pivot occurs only negatively in the normalised clause set, and in particular, the pivot occurs in clauses of the form $C \sqcup \forall \mathcal{S}.D$, then substituting the negated universal role for every occurrence of the pivot in $C \sqcup \forall \mathcal{S}.D$ yields $\top$. If the pivot occurs in $\mathcal{S}$ in the clauses of the form $\neg \mathcal{S} \sqcup \mathcal{S}'$, then substituting the negated universal role for every occurrence of the pivot in $\neg \mathcal{S} \sqcup \mathcal{S}'$ yields $\top$. Therefore, eliminating the purifiable Σ-symbols first could improve the efficiency of the method.

5.5.3 Termination, Soundness and Partialcompleteness

In the previous subsections, we have presented a practical method for forgetting role symbols from ontologies expressible in the description logic $\mathcal{ALCOIH}(\triangledown, \sqcap)$. In this subsection, we show a number of important properties of the method. In particular, we show that: (i) the method is *terminating*, i.e., for any $\mathcal{ALCOIH}(\triangledown, \sqcap)$-ontology $\mathcal{O}$

and any set $\Sigma \subseteq \text{sig}_{\mathsf{R}}(\mathcal{O})$ of role symbols to be forgotten, the method always terminates and returns an ontology $\mathcal{O}'$ of axioms, (ii) the method is *sound*, i.e., if the method is successful, then the forgetting solution $\mathcal{O}'$ computed by the method is equivalent to the original ontology $\mathcal{O}$ up to the interpretations of the symbols in Σ, possibly with the interpretations of the newly-introduced definer symbols and nominals, and (iii) the method is (role forgetting) *complete* for $\mathcal{ALCO}(\triangledown)$-ontologies and is (role forgetting) *incomplete* for $\mathcal{ALCOIH}(\triangledown, \sqcap)$-ontologies.

Theorem 5.5.4. *For any $\mathcal{ALCOIH}(\triangledown, \sqcap)$-ontology $\mathcal{O}$ and any set $\Sigma \subseteq \text{sig}_{R}(\mathcal{O})$ of role symbols to be forgotten, the method always terminates and returns an ontology $\mathcal{O}'$.*

(i) If $\mathcal{O}'$ does not contain any symbols in Σ or any newly-introduced definer symbols or nominals, then $\mathcal{O}'$ is a solution of forgetting Σ from $\mathcal{O}$ (i.e., $\mathcal{O}'$ is equivalent to the original ontology $\mathcal{O}$ up to the interpretations of the symbols in Σ).

(ii) If $\mathcal{O}'$ does not contain any symbols in Σ but it contains newly-introduced definer symbols or nominals, then $\mathcal{O}'$ is a solution of forgetting Σ from $\mathcal{O}$ in an extended language (and $\mathcal{O}$ and $\mathcal{O}'$ are equivalent up to the interpretations of the symbols in Σ, as well as the interpretations of the newly-introduced definer symbols and nominals present in $\mathcal{O}'$).

Proof. In order to show our method is terminating, we have to show that there is no infinite loop in each phase of the forgetting process in our method. The forgetting process in our method comprises five main phases. The initial phase of the method is the clausification phase, which is a standard transformation, i.e., any $\mathcal{ALCOIH}(\triangledown, \sqcap)$-ontology of axioms can be transformed into a set of clauses using the transformation rules in Figures 4.1, 4.2 and 5.7. The second phase of the method is the normalisation phase. We have shown that any $\mathcal{ALCOIH}(\triangledown, \sqcap)$-ontologies can be transformed into a set of clauses in normal form in finite steps (using definer introduction). The third phase of the method is the Σ-symbol elimination phase, which is an iteration of ACK^{R}-derivations in each of which one single Σ-symbol is eliminated. Thus the termination of this phase follows the termination of ACK^{R}. The fourth phase of the method is the definer elimination phase, where the method attempts to eliminate the definer symbols introduced during the normalisation process (if they were introduced) using our method for concept forgetting. The termination of this phase thus follows the

termination of our method for concept forgetting. The last phase of the method is the declausification phase, which is a standard transformation, i.e., any set of clauses in $\mathcal{ALCOIH}(\triangledown, \sqcap)$ can be transformed into an $\mathcal{ALCOIH}(\triangledown, \sqcap)$-ontology by using the transformation rules in Figures 4.11 and 5.9.

Next, we show that our method is sound. The clausification and declausification are standard equivalence-preserving trasformations. The normalisation preserves equivalence up to the interpretations of the introduced definer symbols. The soundness of the Σ-symbol elimination phase follows the soundness of ACK^{R}. The definer symbols elimination preserves equivalence up to the interpretations of the definer symbols that have been eliminated, possibly with new-introduced nominals. Therefore, our method for role forgetting is sound in the sense that the forgetting solution is equivalent to the original ontology up to the interpretations of the symbols in Σ, possibly with the interpretations of the newly-introduced definer symbols and nominals. □

Theorem 5.5.5. *The method is (role forgetting) complete for $\mathcal{ALCO}(\triangledown)$-ontologies.*

Proof. We have shown that any $\mathcal{ALCO}(\triangledown)$-ontologies can be transformed into a set of clauses in normal form, and any set of clauses in $\mathcal{ALCO}(\triangledown)$ can be transformed back into an $\mathcal{ALCO}(\triangledown)$-ontology of axioms. We have also shown that ACK^{R} is (role forgetting) complete for $\mathcal{ALCO}(\triangledown)$-ontologies. □

Note that the completeness of the method for $\mathcal{ALCO}(\triangledown)$-ontologies is based on the assumption that definer symbols are allowed to occur in the forgetting solutions. If definer symbols are disallowed in the forgetting solutions, then our method is (role forgetting) complete only for normalised $\mathcal{ALCO}(\triangledown)$-ontologies. Since most real-world ontologies are normalised ontologies, this gives us best benefits of using our Ackermann-based method to solve role forgetting problems on real-world ontologies.

The incompleteness of the method for $\mathcal{ALCOIH}(\triangledown, \sqcap)$-ontologies follows the incompleteness of ACK^{R} for $\mathcal{ALCOIH}(\triangledown, \sqcap)$-ontologies. Given an $\mathcal{ALCOIH}(\triangledown, \sqcap)$-ontology $\mathcal{O}$ and a set $\Sigma \subseteq \text{sig}_{\mathsf{R}}(\mathcal{O})$ of role symbols to be forgotten (as input to the method), the method may return an ontology $\mathcal{O}'$ that still contains some symbols in Σ. In this case, the method was *unsuccessful* (or *failed*). This is because there is a gap in the scope of the rewrite rules for handling inverse pivots.

5.6 Examples

We conclude the chapter with an example illustrating the usage of our method to forget role symbols from ontologies expressible in the description logic $\mathcal{ALCOIH}(\triangledown, \sqcap)$.

Example 5.6.1. Consider the following $\mathcal{ALCOIH}$-ontology $\mathcal{O}$:

1. $A \sqsubseteq \exists s.\exists r.B$
2. $B \sqsubseteq \forall r^{-}.\neg C$
3. $a \sqsubseteq \forall s.\forall r.B$
4. $s \sqsubseteq t_1$
5. $s \sqsubseteq t_2$

Assume $\Sigma = \{r, s\}$. The method first transforms $\mathcal{O}$ into the following set $\mathcal{N}$ of clauses:

6. $\neg A \sqcup \exists s.\exists r.B$
7. $\neg B \sqcup \forall r^{-}.\neg C$
8. $\neg a \sqcup \forall s.\forall r.B$
9. $\neg s \sqcup t_1$
10. $\neg s \sqcup t_2$

Observe that Clauses 6 and 8 are not in normal form. Thus two fresh definer symbols $\mathsf{D}_1 \in \mathsf{N}_\mathsf{D}$ and $\mathsf{D}_2 \in \mathsf{N}_\mathsf{D}$ are introduced by the method to replace respectively $\exists r.B$ and $\forall r.B$, and two clauses $\neg\mathsf{D}_1 \sqcup \exists s.B$ and $\neg\mathsf{D}_2 \sqcup \forall r.B$ are accordingly added to $\mathcal{N}$.

11. $\neg A \sqcup \exists s.\mathsf{D}_1$
12. $\neg\mathsf{D}_1 \sqcup \exists r.B$
13. $\neg B \sqcup \forall r^{-}.\neg C$
14. $\neg a \sqcup \forall s.\mathsf{D}_2$
15. $\neg\mathsf{D}_2 \sqcup \forall r.B$
16. $\neg s \sqcup t_1$
17. $\neg s \sqcup t_2$

Assume r is the pivot. The method attempts to transform the set into pivot-reduced form. Observe that Clause 13 is not in r-reduced form because of the inverse r. The

method applies the Surfacing$^{\mathsf{R}}$ rule to Clause 13 to transform it into r-reduced form:

18. $\neg A \sqcup \exists s.\mathsf{D}_1$
19. $\neg \mathsf{D}_1 \sqcup \exists r.B$
20. $\neg C \sqcup \forall r.\neg B$
21. $\neg a \sqcup \forall s.\mathsf{D}_2$
22. $\neg \mathsf{D}_2 \sqcup \forall r.B$
23. $\neg s \sqcup t_1$
24. $\neg s \sqcup t_2$

The method applies the Ackermann$^{\mathsf{R}}$ rule to Clauses 19, 20 and 22 to eliminate r, thereby yielding the following set:

25. $\neg A \sqcup \exists s.\mathsf{D}_1$
26. $\neg a \sqcup \forall s.\mathsf{D}_2$
27. $\neg \mathsf{D}_1 \sqcup \exists \triangledown.B$
28. $\neg \mathsf{D}_1 \sqcup \neg C \sqcup \exists \triangledown.(B \sqcap \neg B)$
29. $\neg \mathsf{D}_1 \sqcup \neg \mathsf{D}_2 \sqcup \exists \triangledown.(B \sqcap B)$
30. $\neg \mathsf{D}_1 \sqcup \neg C \sqcup \neg \mathsf{D}_2 \sqcup \exists \triangledown.(B \sqcap \neg B \sqcap B)$
31. $\neg s \sqcup t_1$
32. $\neg s \sqcup t_2$

The pivot r has been eliminated from $\mathcal{N}$. Observe that Clauses 28, 29 and 30 can be further simplified. In particular, Clause 27 subsumes Clause 29, and Clause 28 subsumes Clause 30. By simplification, the following clauses are obtained:

33. $\neg A \sqcup \exists s.\mathsf{D}_1$
34. $\neg a \sqcup \forall s.\mathsf{D}_2$
35. $\neg \mathsf{D}_1 \sqcup \exists \triangledown.B$
36. $\neg \mathsf{D}_1 \sqcup \neg C$
37. $\neg s \sqcup t_1$
38. $\neg s \sqcup t_2$

Now s becomes the pivot. Since the set is already in s-reduced form, the Ackermann$^{\mathsf{R}}$ rule can be applied to eliminate s, thereby yielding the following set:

$$39.\ \neg \mathsf{D}_1 \sqcup \exists \nabla.B$$

$$40.\ \neg \mathsf{D}_1 \sqcup \neg C$$

$$41.\ \neg A \sqcup \exists (t_1 \sqcap t_2).\mathsf{D}_1$$

$$42.\ \neg A \sqcup \neg a \sqcup \exists (t_1 \sqcap t_2).(\mathsf{D}_1 \sqcap \mathsf{D}_2)$$

The definer symbols D_1 and D_2 were introduced during the normalisation process. The method attempts to eliminate them using the method for concept forgetting. Observe that D_2 occurs only positively in $\mathcal{N}$. Then the method applies the Purify$^{\mathsf{R},+}$ rule to eliminate D_2, thereby yielding the following clauses:

$$43.\ \neg \mathsf{D}_1 \sqcup \exists \nabla.B$$

$$44.\ \neg \mathsf{D}_1 \sqcup \neg C$$

$$45.\ \neg A \sqcup \exists (t_1 \sqcap t_2).\mathsf{D}_1$$

$$46.\ \neg A \sqcup \neg a \sqcup \exists (t_1 \sqcap t_2).(\mathsf{D}_1 \sqcap \top)$$

Clause 46 is subsumed by Clause 45, and thus can be deleted. The method attempts to eliminate the other introduced definer symbol D_1. Since the set is already in D_1^--reduced form, the method applies the Ackermann$^{\mathsf{C},-}$ rule to eliminate D_1. In particular, the method substitutes $\exists \nabla.B \sqcap \neg C$ for the D_1 in Clause 45:

$$47.\ \neg A \sqcup \exists (t_1 \sqcap t_2).(\exists \nabla.B \sqcap \neg C)$$

The method declausifies the clause above into the following ontology $\mathcal{O}'$, which is the solution of forgetting $\{s\}$ from the given ontology.

$$\mathcal{O}' = \{A \sqsubseteq \exists (t_1 \sqcap t_2).(\exists \nabla.B \sqcap \neg C)\}$$

The forgetting solution $\mathcal{O}'$ is not expressible in the source logic $\mathcal{ALCOIH}$, but it can be expressed in $\mathcal{ALCOIH}$ with the universal role and role conjunction as shown above.

Chapter 6

Implementation and Evaluation

The main topic of this thesis is the development of practical methods for forgetting concept and role symbols from ontologies specified in expressive description logics.

In Chapter 4, we presented a method for forgetting concept symbols from ontologies expressible in the description logic $\mathcal{ALCOI}$. The method is based on the calculus ACK$^{\mathsf{C}}$ for eliminating a single concept symbol from a set of $\mathcal{ALCOI}$-clauses. The elimination is based on the use of two direct generalisations of Ackermann's Lemma for description logics, referred to as the Ackermann$^{\mathsf{C}}$ rules. The method is terminating, and is sound in the sense that the forgetting solution is equivalent to the original ontology up to the interpretations of the symbols that have been forgotten, possibly with the interpretations of the nominals that have been introduced during the forgetting process.

In chapter 5, we presented a method for forgetting role symbols from ontologies expressible in the description logic $\mathcal{ALCOIH}(\triangledown, \sqcap)$. The method is based on the calculus ACK$^{\mathsf{R}}$ for eliminating a single role symbol from a set of $\mathcal{ALCOIH}(\triangledown, \sqcap)$-clauses. The elimination is based on the use of a non-trivial generalisation of Ackermann's Lemma for description logics, referred to as the Ackermann$^{\mathsf{R}}$ rule. The method is terminating, and is sound in the sense that the forgetting solution is equivalent to the original ontology up to the interpretations of the symbols that have been forgotten, possibly with the interpretations of the concept definer symbols and nominals that have been introduced during the forgetting process. The method is role forgetting complete for $\mathcal{ALCO}(\triangledown)$-ontologies.

Collectively, these two methods can be used as a (incomplete) unifying method for

forgetting concept and role symbols from ontologies expressible in the description logic $\mathcal{ALCOIH}(\triangledown, \sqcap)$. Being based on ACK$^{\mathsf{C}}$ and ACK$^{\mathsf{R}}$, the method is goal-oriented and incremental. It is terminating, and is sound in the sense that the forgetting solution is equivalent to the original ontology up to the interpretations of the symbols that have been forgotten, possibly with the interpretations of the concept definer symbols and nominals that have been introduced in the forgetting process (i.e., specifically, definer symbols are introduced in role forgetting, during the process of transforming the ontology into normal form, and nominals are introduced in concept forgetting, during the process of transforming the ontology into pivot-reduced form).

To gain insight into the practicality of the method, we implemented a prototype FAME and evaluated FAME on a corpus of real-world ontologies. FAME, which stands for **F**orgetting using an **A**ckermann-based **ME**thod, fully realises ACK$^{\mathsf{C}}$ and ACK$^{\mathsf{R}}$. The experiments were run on a desktop computer with an Intel® Core™ i7-4790 processor, four cores running at up to 3.60 GHz and 8 GB of DDR3-1600 MHz RAM.

6.1 The Implementation – FAME

FAME was implemented in Java – a platform-independent and object-oriented programming language. It can be used as a standalone tool useful for tasks such as the performance evaluation of the method presented in this chapter, and it can also be used as a library for platform-independent Java applications. In this section, we describe the general design of FAME and discuss several notable features of FAME.

The framework of FAME is shown in Figure 6.1, where it can be seen that FAME has two major components, i.e., Role forgetting and Concept forgetting, and four minor components, i.e., Load ontology, Parse into own data structure, Unparse into OWL/XML document and Save ontology. We employed the OWL API Version 3.5.2[1] for the tasks of loading, parsing, unparsing and saving OWL ontologies. The ontology to be loaded must be specified as an OWL file or as a URL pointing to the ontology, though own data structure was used in the main forgetting life cycle. This was done for the convenience of programming and also for efficiency of FAME. When the forgetting process is finished, the result will be transformed back into an OWL/XML document,

[1] http://owlapi.sourceforge.net/

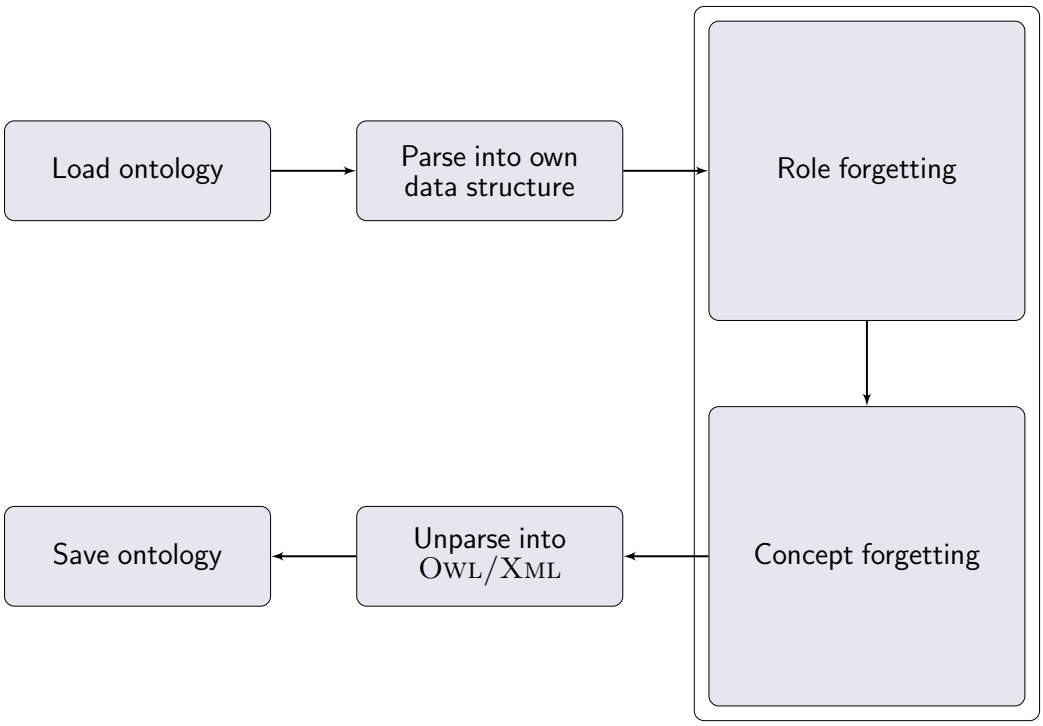

Figure 6.1: The framework of FAME

which can be used as a standard OWL ontology for further processing.

FAME can perform purely concept forgetting, purely role forgetting, and both concept and role forgetting for ontologies expressible in $\mathcal{ALCOIH}(\triangledown, \sqcap)$. In particular, if one wants to forget only concept symbols from the ontology, then the functionalities for role forgetting are switched off and only the functionalities for concept forgetting are switched on. If one wants to forget only role symbols from the ontology, then the functionalities for concept forgetting are switched off and only the functionalities for role forgetting are switched on. If one wants to forget both concept and role symbols from the ontology, then both the functionalities for concept forgetting and the functionalities for role forgetting are switched on.

In the case of forgetting only concept symbols, a heuristic is used to compute good elimination orders, which facilitate successful and quick elimination of Σ-symbols. The heuristic was implemented in FAME to guide the elimination process.

In the case of forgetting only role symbols, no heuristic is used to compute good elimination orders and Σ-symbols are eliminated in the order as returned by an OWL API method that gets role symbols in the signature of an ontology. This is because a good elimination order is not crucial in role forgetting, i.e., the elimination of a role symbol is independent of the elimination of another role symbol.

In the case of forgetting both concept and role symbols, an important problem is how these concept and role symbols are eliminated from the ontology, or more precisely, what elimination order should be used when eliminating both concept and role symbols. Our method allows the concept and role symbols in Σ to be eliminated in any order, but as shown in Chapter 4, a good elimination order can improve the efficiency and success rates of the method.

Given an ontology $\mathcal{O}$ of axioms and a set $\Sigma \in \text{sig}(\mathcal{O})$ of concept and role symbols to be forgotten, FAME implements the following strategy to eliminate Σ-symbols:

1. First, FAME counts the frequencies of positive occurrence and negative occurrence of each symbol in Σ, and flags up those symbols that occur only positively or only negatively in $\mathcal{O}$ (i.e., the purifiable symbols).

2. Then, FAME eliminates all purifiable (concept and role) symbols in Σ from $\mathcal{O}$. After purification, it is very likely that the result would contain massive syntactic redundancies, contradictions and tautologies, which can be further simplified. Therefore, one can reasonably expect that after simplification the purification result is a significantly reduced ontology with fewer symbols and clauses. This makes the subsequent elimination rather easier.

3. Next, FAME eliminates all role symbols in the current Σ. During the role forgetting process, concept definer symbols might be introduced in the present clause set to facilitate the transformation of the set into normal form, which are however not in the desired signature. This means that they should be eliminated at some point during the entire forgetting process. FAME places them in Σ and treats them as regular concept Σ-symbols, which will be eliminated in the subsequent concept forgetting. No heuristic is used in the role forgetting process.

4. Last, FAME eliminates all concept symbols (including the concept definer symbols) in the current Σ. A heuristic based on frequency analysis of concept Σ-symbols is used in the concept forgetting process, leading to successful and faster elimination of concept Σ-symbols.

This is the most effective strategy we have found to eliminate Σ-symbols when both concept and role symbols are in Σ. Other strategies include: (i) FAME first eliminates

all concept symbols in Σ, and then eliminates all role symbols in Σ, and (ii) FAME randomly selects (concept and role) symbols in Σ to eliminate. In the former case, FAME may have to perform concept forgetting again when role forgetting is finished, because concept definer symbols might have been introduced during the role forgetting process. While in the latter case, the heuristic used in concept forgetting cannot be used well because new concept definer symbols may be introduced at any time, whereas the heuristic is based on the frequency analysis of the original Σ-symbols.

In the case of role forgetting, the rewrite$^{\mathsf{R}}$ rules can be flexibly applied to the pivot-clauses (to transform them into pivot-reduced form), because it makes no difference to the forgetting solution; see the following example.

Example 6.1.1. Consider the following ontology $\mathcal{O}$:

$$1.\ \neg A_1 \sqcup \exists r.B_1$$
$$2.\ \neg B_2 \sqcup \forall r^{-}.A_2$$
$$3.\ \neg r^{-} \sqcup s$$

Assume $\Sigma = \{r\}$. Observe r is preceded by an inverse operator in Clauses 2 and 3. We can first apply the Surfacing$^{\mathsf{R}}$ rule to Clause 2, and then apply the Inverting$^{\mathsf{R},-}$ rule to Clause 3 to transform them into r-reduced form. Alternatively, we can first apply the Inverting$^{\mathsf{R},-}$ rule to Clause 3, and then apply the Surfacing$^{\mathsf{R}}$ rule to Clause 2 to transform them into r-reduced form. We obtain the same clauses:

$$1.\ \neg A_1 \sqcup \exists r.B_1$$
$$2.\ \neg A_2 \sqcup \forall r.B_2$$
$$3.\ \neg r \sqcup s^{-}$$

We apply the Ackermann$^{\mathsf{R}}$ rule to $\mathcal{O}$ to eliminate r, thereby obtaining the following clause set, which is solution of forgetting $\{r\}$ from $\mathcal{O}$.

$$4.\ \neg A_1 \sqcup \exists s^{-}.B_1$$
$$5.\ \neg A_1 \sqcup \neg A_2 \sqcup \exists s^{-}.(B_1 \sqcap B_2)$$

In the case of concept forgetting, there are three pairs of rewrite$^{\mathsf{C}}$ rules for transforming pivot-clauses into pivot-reduced form, i.e., a pair of Surfacing$^{\mathsf{C}}$ rules, a pair of

Skolemisation$^{\mathsf{C}}$ rules and a pair of Skolemisation$^{\triangledown}$ rules. Applying the Skolemisation$^{\mathsf{C}}$ rules or the Skolemisation$^{\triangledown}$ rules (i.e., the Skolemisation rules) introduces new nominals in the result and these nominals are not in the desired signature however. Therefore, we should avoid introducing nominals during the forgetting process, which means that we should avoid applying the Skolemisation rules (if possible) to facilitate the transformation of pivot-clauses into pivot-reduced form. There are cases where both the Surfacing$^{\mathsf{C}}$ rules and the Skolemisation rules are capable of transforming pivot-clauses into pivot-reduced form. This can be illustrated with the following example.

Example 6.1.2. Consider the following ontology $\mathcal{O}$:

$$1.\ A \sqcup \forall r.\neg B$$

$$2.\ \neg a \sqcup \exists r.B$$

Assume $\Sigma = \{B\}$. Observe that $\neg B$ occurs below purely a universal role restriction in Clause 1, which means that the Surfacing$^{\mathsf{C},-}$ rule can be applied to Clause 1 to transform $\mathcal{O}$ into B^{-}-reduced form. B occurs below purely an existential role restriction Clause 2, which means that the Skolemisation$^{\mathsf{C},+}$ rule can be applied to Clause 2 to transform $\mathcal{O}$ into B^{+}-reduced form. In particular, applying the Surfacing$^{\mathsf{C},-}$ rule to Clause 1 and then applying the Ackermann$^{\mathsf{C},-}$ rule to $\mathcal{O}$ yields the following solution:

$$3.\ \neg a \sqcup \exists r.\forall r^{-}.A$$

In contrast, applying the Skolemisation$^{\mathsf{C},+}$ rule to Clause 2 and then applying the Ackermann$^{\mathsf{C},+}$ rule to $\mathcal{O}$ yields the following solution:

$$4.\ A \sqcup \forall r.\neg b$$

$$5.\ \neg a \sqcup \exists r.b$$

where b is a fresh nominal. The two solutions are logically equivalent, but the second one contains a fresh nominal, which is not in the desired signature.

It is obvious in this case that using the Surfacing$^{\mathsf{C},-}$ rule to transform the present ontology into pivot-reduced form is more appropriate, because it avoided the introduction of new nominals. FAME implements a strategy that applying the Surfacing$^{\mathsf{C}}$ rules takes precedence over applying the Skolemisation rules when transforming the present ontology into pivot-reduced form in concept forgetting.

More specifically, when transforming the present ontology into pivot-reduced form, there are in general three different cases: (i) the transformation involves the use of only the Surfacing$^{\mathsf{C}}$ rules, (ii) the transformation involves the use of only the Skolemisation rules, and (iii) the transformation involves the use of both the Surfacing$^{\mathsf{C}}$ rules and the Skolemisation rules. Let $\mathcal{O}$ be an $\mathcal{ALCOIH}(\triangledown, \sqcap)$-ontology, and suppose $\mathcal{O}$ can be transformed into either the pivot^{+}-reduced form or the pivot^{-}-reduced form. If transforming $\mathcal{O}$ into the pivot^{+}-reduced form involves the use of only the Surfacing$^{\mathsf{C}}$ rules (i.e., Case (i)), whereas transforming $\mathcal{O}$ into the pivot^{-}-reduced form involves the use of the Skolemisation rules (i.e., Case (ii) or Case (iii)), then Fame will apply the Surfacing$^{\mathsf{C}}$ rules to transform $\mathcal{O}$ into the pivot^{+}-reduced form. If transforming $\mathcal{O}$ into the pivot^{+}-reduced form involves the use of both the Surfacing$^{\mathsf{C}}$ rules and the Skolemisation$^{\mathsf{C}}$ rules (i.e., Case (ii)), whereas transforming $\mathcal{O}$ into the pivot^{-}-reduced form involves the use of only the Skolemisation rules (i.e., Case (iii)), then Fame will apply the Surfacing$^{\mathsf{C}}$ rules and the Skolemisation rules to transform $\mathcal{O}$ into the pivot^{+}-reduced form. In this way, the Skolemisation rules can be applied as infrequently as possible, and new nominals can be introduced as few as possible.

6.2 The Corpus

We evaluated the performance of Fame on a corpus of real-world ontologies from the NCBO BioPortal repository [WNS+11, MNS+12]. The NCBO BioPortal repository is a resource that includes more than 500 biomedical ontologies originally developed for use in clinical and translational research. In particular, they cover a wide range of topics in biomedicine such as gene, cancer, organology, and anatomy. Differing in size, structure, and expressivity, the BioPortal ontologies offer a rich, diverse and realistic test data set for the evaluation of Fame. In addition, the NCBO BioPortal repository is the most widely used resource for evaluation of automated tools in ontology engineering.

Nevertheless, not all ontologies in the NCBO BioPortal repository are suitable for use in the evaluation. For example, some of the ontologies in the repository were not compatible with the OWL API (i.e., they could not be parsed using the corresponding methods in the OWL API), and some of the ontologies in the repository provided corrupted files of ontologies. The corpus used for the evaluation of Fame was based

on a snapshot of the repository taken in March 2017, containing 422 ontologies. In particular, we selected ontologies from the snapshot based on the following criteria:

1. They were compatible with the OWL API,
2. They contained at most 100000 axioms, and
3. They contained at least 100 axioms.

The first criterion is necessary because otherwise, we could have selected ontologies that are unparsable with the OWL API. Running FAME on unparsable ontologies will lead to the program being interrupted with an UnparsableOntologyException.

The second criterion is justified by an observation that running FAME on ontologies with more than 100000 axioms has always led to a timeout or space explosion. From time to time, this could happen even in the process of loading, parsing and converting the ontologies into our internal representation. In order to make the evaluation manageable, we should get rid of ontologies that are too large.

The third criterion is justified by an observation that ontologies with less than 100 axioms often contain only a few symbols. We have found that FAME always computes a solution of forgetting (concept and role) symbol for such ontologies. Thus, running FAME on very small ontologies is neither interesting nor challenging. In order to obtain statistically significant results, we should get rid of ontologies that are too small.

Item	Mean	Median	90th percentile
logical axioms per ontology	4651	1096	12570
concept symbols per ontology	2110	502	5598
role symbols per ontology	54	12	144
individual symbols per ontology	216	0	206
average axiom size	3.72	3.54	4.82

Table 6.1: Statistics of ontologies selected from BioPortal

Following these three criteria, we selected ontologies (i.e., OWL/XML files) from the snapshot, thereby obtaining a corpus of 396 OWL/XML files of various sizes, ranging from 10 KB up to 60000 KB. Table 6.1 shows the statistical information about these ontologies, where "logical axioms per ontology" represents the number of logical axioms occurring in each test ontology, "concept symbols per ontology" represents the number of concept symbols occurring in each test ontology, "role symbols per ontology" represents the number of role symbols occurring in each test ontology, "individual

symbols per ontology" represents the number of individual symbols occurring in each test ontology, "average axiom size" represents the number of occurrences of concept symbols, role symbols, individual symbols and operators in each axiom, "mean" is the most commonly used measure of average, which is calculated by adding all values together and dividing the sum by the number of values, "median" (i.e., the middle value) is a measure of statistical distribution, which is calculated by ordering all values and picking out the one in the middle (i.e., if there are two middle values, then the median is the mean of those two values), and the 90th percentile is an alternative measure of statistical distribution. The median is the value for which 50% of the values are larger and 50% are smaller. The 90th percentile is the value for which 90% of the values are smaller, and 10% are larger.

The ontologies selected from the snapshot have the DL expressivity ranging from $\mathcal{EL}$ and $\mathcal{ALC}$ to $\mathcal{SHOIN}$ and $\mathcal{SROIQ}$. On the other hand, FAME handles ontologies as expressive as $\mathcal{ALCOIH}(\triangledown, \sqcap)$. Hence, we adjusted these selected ontologies to the language of $\mathcal{ALCOIH}(\triangledown, \sqcap)$. The adjustment was done using simple simulations and simple replacement. Specifically, if the concepts not in $\mathcal{ALCOIH}(\triangledown, \sqcap)$ can be simulated in $\mathcal{ALCOIH}(\triangledown, \sqcap)$, then they are replaced by their simulations in $\mathcal{ALCOIH}(\triangledown, \sqcap)$. If the concepts not in $\mathcal{ALCOIH}(\triangledown, \sqcap)$ cannot be simulated in $\mathcal{ALCOIH}(\triangledown, \sqcap)$, then they are replaced by the top concept. For example, the disjointness between the concepts C and D can be simulated by $C \sqsubseteq \neg D$, and the symmetry of the role R can be simulated by $R \sqsubseteq R^-$.

6.3 Forgetting Concept Symbols

In this subsection, we evaluate the performance of FAME for forgetting only concept symbols from ontologies expressible in the description logic $\mathcal{ALCOIH}(\triangledown, \sqcap)$.

To fit in with real-world applications, we evaluate the performance of FAME on forgetting different numbers of concept symbols from each test ontology. In particular, we forgot 10% (a relatively small number), 40% (a relatively moderate number) and 70% (a relatively large number) of concept symbols in the signature of each ontology whereby we consider how different numbers of Σ-concept symbols affect the results of concept forgetting, in particular, the time duration and the success rates.

We also evaluate the performance of FAME on forgetting concept symbols of different importance from each test ontology. The *importance* of a concept (or role) symbol is defined in the sense that if a concept (or role) symbol $\mathcal{S}$ occurs more frequently than another concept (or role) symbol $\mathcal{S}'$ in an ontology $\mathcal{O}$, then we say that $\mathcal{S}$ is *more important* than $\mathcal{S}'$ w.r.t. $\mathcal{O}$. We refer to the symbols of relatively high importance as *central symbols* to the ontologies. We hypothesise that it is generally harder to eliminate a central symbol from an ontology than eliminate a non-central symbol. In order to verify this, we set up a series of experiments where the Σ-symbols are randomly selected (from the signature of each ontology), and we set up another series of experiments where the Σ-symbols based on their importance. For example, in the case of forgetting 10% of concept symbols, we randomly selected 10% of the concept symbols from each test ontology as the Σ-symbols to be forgotten, and we also selected the central 10% of concept symbols from each test ontology as the Σ-symbols to be forgotten. If our hypothesis turns out to be true, then the result would be insightful to applications such as ontology reuse, information hiding and ontology analysis where it may always be necessary to eliminate non-central symbols.

In addition, we evaluate the performance of FAME on forgetting a fixed number of concept symbols from each test ontology. Since the ontologies used for the evaluation contain different numbers of axioms, the evaluation is intended to see how the number of axioms affects the results of concept forgetting, in particular, the duration of time and the frequency of timeout.

Since a heuristic based on frequency analysis for computing good elimination orders is implemented and used in FAME, finally, we evaluate the performance of FAME on forgetting concept symbols from each ontology with and without the heuristic being used. We consider how this heuristic improves the efficiency and success rates of FAME in concept forgetting.

We ran the experiments 100 times on each test ontology and averaged the results to verify the accuracy of our findings. A timeout of 1000 seconds was imposed on each run of the experiment.

First, we evaluate the performance of FAME on forgetting 10% of concept symbols in the signature of each test ontology. The results shed light on the usefulness of FAME for real-world applications such as ontology debugging and computing logical

differences of different ontology versions where the expected task is to forget a small number of symbols (i.e., from one version to another usually only a small number of symbols will be added or removed).

Settings			Results				
Σ (10%)	≻	!	Timeouts	Duration	Success Rate	New Nominals	Clause ↑
211 (Avg.)	✗	✗	1.5%	3.509 sec.	93.2%	1.26	-9.3%
	✓	✗	1.3%	2.614 sec.	93.4%	1.25	-9.4%
	✗	✓	2.8%	4.576 sec.	90.4%	1.49	-8.7%
	✓	✓	2.3%	3.679 sec.	91.0%	1.49	-8.9%

Table 6.2: Results of forgetting 10% of concept symbols in the signature

The results are shown in Table 6.2, which are rather revealing in several ways. The most encouraging result is that FAME was successful (i.e., forgot all symbols in Σ) in more than 90% test cases within a short period of time (see the Duration Column). It is evident from the table that using the heuristic based on frequency analysis of Σ-symbols (see the ≻ Column) led to a modest decrease in the average duration of the runs of every experiment, which means that it took less time to complete the same task than when the heuristic was not used. Also, it can be seen from the Success Rate Column and the Timeouts Column that basing the elimination order to the frequency analysis has brought a positive effect on the overall success rates (i.e., increased by 0.2%) and the frequency of timeouts (i.e., decreased by 0.2%). Because of the presence of the clauses of the form $\neg a \sqcup \exists R.C$ and the application of the Skolemisation rules, new nominals were introduced in the ontologies. These nominals are however not expected to occur in the signatures of the resulting ontologies (i.e., the forgetting solutions). The New Nominals Column has shown that only a small number of new nominals were introduced into each ontology during the forgetting process. In addition, we compared the number of clauses in the forgetting solutions with that in the given ontologies (see the Clause ↑ column). Compared to the given ontologies, there was a modest decrease in the number of clauses in the forgetting solutions. This is quite different from the resolution-based methods, where there is always an obvious increase in the number of clauses in the forgetting solutions. The ✗ symbol in the ! Column indicates that the Σ-symbols were randomly selected from the signature of each test ontology, and the ✓ symbol in the ! Column indicates that only the central 10% of concept symbols were selected as the Σ-symbols to be forgotten. The comparison results have verified

that, in general, eliminating central symbols takes longer time than eliminating the same number of randomly selected symbols.

Then, we evaluate the performance of FAME on forgetting 40% of concept symbols in the signature of each ontology. The results shed light on the usefulness of FAME for real-world applications such as ontology analysis and explanation generation (abduction), where the expected task is to forget a moderate number of symbols.

Settings			Results				
Σ (40%)	$\succ$	!	Timeouts	Duration	Success Rate	New Nominals	Clause $\uparrow$
844 (Avg.)	✗	✗	6.2%	6.295 sec.	84.0%	5.29	-39.4%
	✓	✗	5.6%	4.931 sec.	85.1%	5.28	-39.7%
	✗	✓	8.7%	9.028 sec.	80.8%	5.55	-32.9%
	✓	✓	7.9%	7.187 sec.	82.3%	5.53	-34.8%

Table 6.3: Results of forgetting 40% of concept symbols in the signature

The results are shown in Table 6.3, from which it can be seen that eliminating a larger number of concept symbols is (expectedly) harder. The success rates were not as good as in the case of forgetting 10% of concept symbols, i.e., there was, on average, a 10% reduction in the success rates, where half of the additional failures were due to timeout and the other half failures were due to the non-existence of the forgetting solutions and the incompleteness of our forgetting method. A 10% reduction in success rates was not as substantial as expected, i.e., we expected that there would be an approximately 30% reduction in the success rates because we eliminated four times as many concept symbols as in the preceding experiments. The reasons might be multifold, but an important reason that has been found is that many presently ineliminable symbols might become eliminable as the elimination of other symbols; see the following example.

Example 6.3.1. Consider the following ontology $\mathcal{O}$:

1. $\neg A \sqcup \exists r.B$
2. $\neg C \sqcup \exists r.\neg B$

Assume $\Sigma = \{B\}$. Observe that $\mathcal{O}$ cannot be transformed into pivot-reduced form by using the (rewrite$^{\mathsf{C}}$) rules in ACK$^{\mathsf{C}}$ because the pivot and the negated pivot both occur below existential role restrictions in Clauses 1 and 2, which are not existential clauses

however (i.e., hence the Skolemisation$^{\mathsf{C}}$ rules are not applicable to Clauses 1 and 2 to transform them into pivot-reduced form). Thus B is ineliminable w.r.t. $\mathcal{O}$. Assume $\Sigma = \{A, B\}$. We apply the Purify$^{\mathsf{C},+}$ rule to Clause 1 to eliminate A whereby Clause 1 becomes a tautology. Then B occurs only negatively in $\mathcal{O}$ (i.e., B is purifiable w.r.t. $\mathcal{O}$), and thus B is eliminable w.r.t. $\mathcal{O}$.

One might also expect that eliminating 40% of concept symbols in the signature should take four times as long as eliminating 10% of concept symbols in the signature. Our experimental results have shown that this is not exactly true (see the Duration Column) i.e., in practice, eliminating 40% of concept symbols in the signature took approximately two times as long as eliminating 10% of concept symbols. This is because as the elimination of Σ-symbols, the ontology became decreasingly smaller (i.e., containing decreasingly fewer clauses), and the subsequent elimination might thus become easier and take less time. What was as expected was that more nominals were introduced in the ontologies when more symbols were selected as the Σ-symbols to be forgotten. Another important observation is that there was a notable reduction (i.e., an approximately 40% reduction) in the number of clauses in the forgetting solutions (see the Clause ↑ Column). This is because usually when the forgetting was finished, most Σ-clauses became tautologies and were removed from the ontologies.

Next, we evaluate the performance of FAME on forgetting 70% of concept symbols from the signature of each ontology. The results shed light on the usefulness of FAME for real-world applications such as ontology summary and reuse, where the expected task is to forget a large number of symbols.

Settings			Results				
Σ (70%)	≻	!	Timeouts	Duration	Success Rate	New Nominals	Clause ↑
1477 (Avg.)	✗	✗	16.7%	12.893 sec.	68.3%	7.06	-70.1%
	✓	✗	15.2%	9.827 sec.	70.7%	7.04	-70.3%
	✗	✓	20.2%	13.684 sec.	63.1%	7.49	-67.9%
	✓	✓	19.2%	10.075 sec.	64.9%	7.47	-68.3%

Table 6.4: Results of forgetting 70% of concept symbols in the signature

The results are shown in Figure 6.4, from which it can be observed that as more (concept) symbols were selected as the Σ-symbols to be forgotten, the (averaged) success rates dropped to approximately 70% for the cases where Σ-symbols were randomly

selected and dropped to approximately 64% for the cases where only the central 70% of concept symbols were selected, the (averaged) frequency of timeout rose to approximately 16% for the cases where Σ-symbols were randomly selected and rose to approximately 20% for the cases where only the central 70% of concept symbols were selected, and the (averaged) time duration rose to approximately three times as long as in the case of eliminating 10% of concept symbols. This was expected because eliminating 70% of concept symbols in the signature of each ontology was obviously an extremely challenging task. For example, in the case of forgetting 70% of concept symbols in the signature of the HUGO ontology, this means eliminating 23041 concept symbols from 32917 (TBox and RBox) axioms. The number of nominals introduced in each ontology rose to between 7 and 7.5, nearly six times as many as in the case of forgetting 10% of concept symbols. As in Table 6.3, there was a notable reduction (i.e., an approximately 70% reduction) in the number of clauses in the forgetting solutions (see the Clause ↑ Column). From Tables 6.2, 6.3 and 6.4, we have noticed that the reduction rate of the clauses in the forgetting solutions was almost identical to the percentage of concept symbols selected as the Σ-symbols to be forgotten.

Finally, we evaluate the performance of FAME on forgetting a fixed number (i.e., 100) of concept symbols from the signature of each ontology. The results shed light on how different numbers of axioms affect the performance (i.e., in particular, the time duration) of FAME on forgetting the same number of concept symbols. Considering that not all test ontologies contained at least 100 concept symbols, we filtered out those containing less than 100 concept symbols. As a result, 326 ontologies stood out from the corpus. Then we split these ontologies into three (sub)corpora with each of them containing ontologies with the number of axioms ranging from 100 to 1000, 1001 to 5000, and more than 5000, respectively. In this way, we obtained a corpus (**Corpus I**) of 190 ontologies with the number of axioms ranging from 100 to 1000, a corpus (**Corpus II**) of 120 ontologies with the number of axioms ranging from 1001 to 5000, and a corpus (**Corpus III**) of 86 ontologies with more than 5000 axioms.

The results are shown in Figure 6.5, from which it can be seen that forgetting 100 concept symbols from the ontologies with more axioms took much longer than forgetting the same number of concept symbols from the ontologies with fewer axioms. This

Settings			Results				
Corpora	$\succ$	!	Timeouts	Duration	Success Rate	New Nominals	Clause ↑
I	✗	✗	0.0%	0.839 sec.	97.9%	0.73	-5.1%
	✓	✗	0.0%	0.827 sec.	97.9%	0.72	-5.3%
	✗	✓	0.5%	1.195 sec.	96.8%	1.03	-4.9%
	✓	✓	0.5%	1.183 sec.	96.8%	1.02	-5.0%
II	✗	✗	0.8%	1.356 sec.	96.7%	1.13	-5.2%
	✓	✗	0.8%	1.327 sec.	96.7%	1.12	-5.1%
	✗	✓	1.7%	2.183 sec.	95.8%	1.44	-5.1%
	✓	✓	1.7%	2.095 sec.	95.8%	1.43	-5.0%
III	✗	✗	1.2%	2.204 sec.	93.0%	1.53	-4.9%
	✓	✗	1.2%	2.010 sec.	93.0%	1.52	-4.8%
	✗	✓	2.3%	3.356 sec.	91.9%	1.70	-4.7%
	✓	✓	2.3%	3.094 sec.	91.9%	1.69	-4.6%

Table 6.5: Results of forgetting 100 concept symbols in the signature

is because (i) processing ontologies with more axioms usually takes longer than processing ontologies with fewer axioms, and (ii) in general, there were more occurrences of Σ-symbols in the ontologies with more axioms, which means that the rewrite$^{\mathsf{C}}$ rules needed to be applied more often than in the ontologies with fewer axioms.

6.4 Forgetting Role Symbols

In this subsection, we evaluate the performance of FAME for forgetting only role symbols from ontologies expressible in the description logic $\mathcal{ALCOIH}(\triangledown, \sqcap)$.

We implement the following strategy for the evaluation of FAME for role forgetting: (i) we evaluate the performance of FAME for forgetting different numbers of role symbols from each ontology. The results shed light on the usefulness of FAME for different real-world applications as discussed in Section 1.1. In particular, we forgot 10% (a relatively small number), 40% (a relatively moderate number) and 70% (a relatively large number) of role symbols in the signature of each ontology whereby we consider how different numbers of role symbols selected as Σ-symbols to be forgotten affect the results of role forgetting, (ii) we evaluate the performance of FAME for forgetting central role symbols from each ontology whereby we consider how different frequencies of Σ-symbols occurring in the ontologies affect the results of role forgetting, (iii) we evaluate the performance of FAME for forgetting a fixed number of role symbols

from each ontology whereby we consider how different numbers of axioms contained in the ontologies affect the results of role forgetting, and (iv) we compare the performance results of role forgetting to those of concept forgetting. This is to verify our consideration that forgetting role symbols is a harder than forgetting concept symbols.

Out of the 396 ontologies in the corpus, only 333 ontologies contain role symbols. We filtered out those ontologies not containing role symbols. We ran the experiments 100 times on each test ontology and averaged the results to verify the accuracy of our findings. A timeout of 1000 seconds was imposed on each run of the experiment.

Settings			Results					
Σ (10%)	$\succ$	!	Timeouts	Duration	Success Rate	D. In	D. Left	Clause $\uparrow$
5 (Avg.)	✗	✗	0.0%	2.129 sec.	100.0%	1.25	0.0	4.2%
	✓	✗	0.0%	2.127 sec.	100.0%	1.25	0.0	4.2%
	✗	✓	0.0%	10.491 sec.	85.0%	5.02	0.0	13.7%
	✓	✓	0.0%	10.488 sec.	85.0%	5.02	0.0	13.7%

Table 6.6: Results of forgetting 10% of role symbols in the signature

First, we evaluate the performance of FAME for forgetting 10% of role symbols in the signature of each test ontology. The results are shown in Table 6.6, which is rather revealing in several ways. The most encouraging result is that FAME was successful (i.e., forgot all symbols in Σ) in all test cases when the Σ-symbols were randomly selected. In the cases where only the central 10% of role symbols were selected as the Σ-symbols to be forgotten, the success rates were 85%. The failures were due to space explosion caused by the high frequency of some role symbols in Σ. Without these symbols in Σ, the success rates were 100%. It can be observed from Table 6.6 that only a small number of concept definer symbols were introduced in the ontologies to facilitate the transformation of pivot-clauses into pivot-reduced form (see the D. In Column). This indicates that most clauses in the test ontologies were flat (i.e., most test ontologies were normalised ontologies). The introduced definer symbols were eliminated from the ontologies by using our method for concept forgetting (see the D. Left Column). Because of the nature of the Ackermann$^{\mathsf{R}}$ rule, role forgetting may lead to growth of clauses in the forgetting solutions, which was modest compared to the theoretical worst case (i.e., 4.2% and 13.7%, respectively). It is apparent from Table 6.6 that eliminating central role symbols was significantly harder than eliminating the same number of randomly selected role symbols. For example, eliminating the

randomly selected 10% of role symbols in the signature of each test ontology took approximately 2 seconds (averaged), whereas eliminating the central 10% of role symbols in the signature of each test ontology took approximately 10 seconds (averaged), five times as long as in the "randomly selected" cases. This means that the more frequently the role symbols occur in an ontology, the more difficult (i.e., the more time used and the lower success rates achieved) the symbols are eliminated from the ontology. This is because of the nature of the Ackermann$^{\mathsf{R}}$ rule (recall the complexity analysis of the Ackermann$^{\mathsf{R}}$ rule discussed in Subsection 5.4.2).

Settings			Results					
Σ (40%)	$\succ$	!	Timeouts	Duration	Success Rate	D. In	D. Left	Clause $\uparrow$
22 (Avg.)	✗	✗	0.0%	9.913 sec.	90.1%	5.43	0.0	18.5%
	✓	✗	0.0%	9.866 sec.	90.1%	5.43	0.0	18.5%
	✗	✓	0.0%	38.683 sec.	73.0%	14.52	0.0	40.7%
	✓	✓	0.0%	38.556 sec.	73.0%	14.52	0.0	40.7%

Table 6.7: Results of forgetting 40% of role symbols in the signature

Then, we evaluate the performance of Fame for forgetting 40% of role symbols in the signature of each test ontology. The results are shown in Table 6.7, from which we can conclude that eliminating a larger number of role symbols can significantly affect the efficiency and success rates of Fame. In particular, eliminating 40% of role symbols in the signature of each ontology took approximately four times as long as eliminating 10% of role symbols in the signature of each ontology. The success rates decreased by approximately 10% for the cases where the Σ-symbols were randomly selected and by approximately 12% for the cases where only the central 40% of role symbols were selected as the Σ-symbols to be forgotten. As in the cases of forgetting 10% of role symbols in the signature of each ontology, the failures were due to space explosion caused by the high frequency of some role symbols in Σ. Without these symbols in Σ, the success rates were 100%. With more role symbols being selected as the Σ-symbols to be forgotten, more concept definer symbols were introduced in the ontologies to facilitate the transformation of pivot-clauses into pivot-reduced form. As in the cases of forgetting 10% of role symbols in the signature of each ontology, these definer symbols were eliminated from the resulting ontologies using our method for concept forgetting. It can be observed that there was a notable growth in the number of clauses in the forgetting solutions, in particular in the cases where only the central

70% of role symbols were selected as the Σ-symbols to be forgotten.

Settings			Results					
Σ (10%)	≻	!	Timeouts	Duration	Success Rate	D. In	D. Left	Clause ↑
38 (Avg.)	✗	✗	0.0%	16.385 sec.	81.1%	8.54	0.0	30.2%
	✓	✗	0.0%	16.375 sec.	81.1%	8.54	0.0	30.2%
	✗	✓	0.0%	52.491 sec.	63.1%	24.68	0.0	76.7%
	✓	✓	0.0%	52.428 sec.	63.1%	24.68	0.0	76.7%

Table 6.8: Results of forgetting 70% of role symbols in the signature

Next, we evaluate the performance of FAME for forgetting 70% of role symbols in the signature of each test ontology. The results are shown in Table 6.8, from which it can be seen that eliminating randomly selected 70% of role symbols (in the signature of each ontology) took more than seven times as long as eliminating randomly selected 10% of role symbols (in the signature of each ontology), and eliminating the central 70% of role symbols (in the signature of each ontology) took five times as long as eliminating the central 10% of role symbols (in the signature of each ontology). With further more role symbols being selected as the Σ-symbols to be forgotten, the success rates dropped to 81.1% for the cases where the Σ-symbols were randomly selected, and 63.1% for the cases where the central 70% of role symbols were selected as the Σ-symbols to be forgotten, and the number of the definer symbols introduced in each ontology rose to 8.54 for the randomly selected cases and 24.68 for the central cases. Although more definer symbols were introduced in the ontologies, they were all eliminated using our method for concept forgetting (once the Σ-symbols had been eliminated). A significant growth of clauses in the forgetting solutions is observed (see the Clauses ↑ column).

Results from Table 6.8 can be compared with the results in Tables 6.6 and 6.7 which shows that:

1. forgetting role symbols is, in general, a harder task than forgetting concept symbols (i.e., longer time duration was used and lower success rates were obtained when forgetting same numbers of symbols),

2. the performance of FAME depends greatly on the number of the symbols in Σ (more precisely, the frequency of Σ-symbols occurring in the ontologies),

3. the more the role symbols are selected as the Σ-symbols to be forgotten, the lower the likelihood of FAME successfully computing the forgetting solutions,

the more the concept definer symbols are introduced in the ontologies and the more the clauses are generated in the forgetting solutions, and

4. the order of eliminating Σ-symbols is not important for role forgetting.

Settings			Results					
Corpora	$\succ$	!	Timeouts	Duration	Success Rate	D. In	D. Left	Clause ↑
I	✗	✗	0.0%	5.746 sec.	94.9%	1.37	0	18.36%
	✓	✗	0.0%	5.745 sec.	94.9%	1.37	0	18.36%
	✗	✓	0.0%	19.474 sec.	79.7%	3.47	0	40.02%
	✓	✓	0.0%	19.473 sec.	79.7%	3.47	0	40.02%
II	✗	✗	0.0%	9.336 sec.	90.0%	5.58	0	18.64%
	✓	✗	0.0%	9.332 sec.	90.0%	5.58	0	18.64%
	✗	✓	0.0%	36.375 sec.	73.3%	13.98	0	39.28%
	✓	✓	0.0%	36.374 sec.	73.3%	13.98	0	39.28%
III	✗	✗	0.0%	14.725 sec.	85.1%	11.31	0	18.48%
	✓	✗	0.0%	14.723 sec.	85.1%	11.31	0	18.48%
	✗	✓	0.0%	57.583 sec.	66.0%	22.57	0	40.26%
	✓	✓	0.0%	57.582 sec.	66.0%	22.57	0	40.26%

Table 6.9: Results of forgetting 20 role symbols in the signature

Finally, we evaluate the performance of FAME for forgetting a fixed number (i.e., 20) of role symbols from the signature of each ontology. Considering that not all test ontologies contained at least 20 role symbols, we filtered out those ontologies containing less than 20 role symbols. As a result, 166 ontologies stood out from the current corpus. Then we split these ontologies into three (sub)corpora with each of them containing ontologies with the number of axioms ranging from 100 to 1000, 1001 to 5000, and more than 5000, respectively. In this way, we obtained a corpus (**Corpus I**) of 59 ontologies with the number of axioms ranging from 100 to 1000, a corpus (**Corpus II**) of 60 ontologies with the number of axioms ranging from 1001 to 5000, and a corpus (**Corpus III**) of 47 ontologies with more than 5000 axioms.

The results are shown in Figure 6.9, from which it can be seen that forgetting 20 role symbols from the ontologies with more axioms took much longer than forgetting the same number of role symbols from the ontologies with fewer axioms. This is for the same reasons as we found in concept forgetting.

6.5 Forgetting Concept and Role Symbols

In this subsection, we evaluate the performance of FAME for forgetting both concept and role symbols from ontologies expressible in the description logic $\mathcal{ALCOIH}(\triangledown, \sqcap)$. This is motivated by the fact that in most real-world applications, usually concept and role symbols are required to be eliminated together. Given an ontology $\mathcal{O}$ and a set $\Sigma \in \text{sig}(\mathcal{O})$ of concept and role symbols to be forgotten, FAME defaults to first eliminating all role symbols in Σ, and then all concept symbols in Σ. In this way, the possibly introduced definer symbols can be eliminated as part of the subsequent concept forgetting. The aim of the experiments is to investigate how concept forgetting and role forgetting affect each other.

We implement the same strategy for this evaluation. In particular, we evaluate the performance of FAME for forgetting 10%, 40% and 70% of concept symbols and 10%, 40% and 70% of role symbols in the signature of each test ontology. Unlike the experiments conducted in the previous two sections where concept symbols and role symbols were eliminated separately, in this section, we conducted a series of experiments where concept symbols and role symbols are eliminated together. All other settings remain unchanged.

The ontologies used for this evaluation are the ontologies used for the evaluation of FAME for forgetting only concept symbols. We ran the experiments 100 times on each test ontology and averaged the results to verify the accuracy of our findings. A timeout of 1000 seconds was imposed on each run of the experiment.

Settings			Results				
Σ (10%)	$\succ$	!	Timeouts	Duration	Success Rate	New Nominals	Clause ↑
211 (Avg.)	✗	✗	1.5%	3.489 sec.	95.0%	1.24	-9.4%
	✓	✗	1.3%	2.601 sec.	95.2%	1.24	-9.4%
	✗	✓	2.8%	4.487 sec.	92.2%	1.47	-8.9%
	✓	✓	2.3%	3.598 sec.	92.7%	1.47	-9.0%
Σ (10%)	$\succ$	!	Timeouts	Duration	Success Rate	Definers Left	Clause ↑
5 (Avg.)	✗	✗	0.0%	2.089 sec.	100.0%	0.0	4.3%
	✓	✗	0.0%	2.088 sec.	100.0%	0.0	4.3%
	✗	✓	0.0%	10.278 sec.	85.0%	0.0	13.8%
	✓	✓	0.0%	10.273 sec.	85.0%	0.0	13.8%

Table 6.10: Results of forgetting 10% of concept symbols and 10% of role symbols

Settings			Results				
Σ (40%)	≻	!	Timeouts	Duration	Success Rate	New Nominals	Clause ↑
844 (Avg.)	✗	✗	5.8%	6.145 sec.	86.4%	5.34	-39.7%
	✓	✗	5.6%	4.903 sec.	87.1%	5.33	-40.1%
	✗	✓	7.3%	8.892 sec.	80.8%	5.59	-34.1%
	✓	✓	6.8%	7.012 sec.	82.3%	5.58	-34.4%
Σ (40%)	≻	!	Timeouts	Duration	Success Rate	Definers Left	Clause ↑
22 (Avg.)	✗	✗	0.0%	9.487 sec.	90.1%	0.0	18.8%
	✓	✗	0.0%	9.485 sec.	90.1%	0.0	18.8%
	✗	✓	0.0%	37.983 sec.	73.0%	0.0	41.2%
	✓	✓	0.0%	37.975 sec.	73.0%	0.0	41.2%

Table 6.11: Results of forgetting 40% of concept symbols and 10% of role symbols

Settings			Results				
Σ (70%)	≻	!	Timeouts	Duration	Success Rate	New Nominals	Clause ↑
1477 (Avg.)	✗	✗	15.2%	12.786 sec.	70.7%	7.13	-70.9%
	✓	✗	13.9%	9.743 sec.	72.2%	7.12	-71.1%
	✗	✓	18.9%	13.102 sec.	64.9%	7.53	-68.6%
	✓	✓	17.4%	10.002 sec.	67.2%	7.52	-68.9%
Σ (10%)	≻	!	Timeouts	Duration	Success Rate	Definers Left	Clause ↑
38 (Avg.)	✗	✗	0.0%	15.873 sec.	81.1%	0.0	31.0%
	✓	✗	0.0%	15.487 sec.	81.1%	0.0	31.0%
	✗	✓	0.0%	51.304 sec.	63.1%	0.0	77.9%
	✓	✓	0.0%	51.302 sec.	63.1%	0.0	77.9%

Table 6.12: Results of forgetting 70% of concept symbols and 70% of role symbols

The results obtained from forgetting 10% of concept symbols and forgetting 10% of role symbols in the signature of each test ontology are shown in Table 6.10. The results obtained from forgetting 40% of concept symbols and forgetting 40% of role symbols in the signature of each test ontology are shown in Table 6.11. The results obtained from forgetting 70% of concept symbols and forgetting 70% of role symbols in the signature of each test ontology are shown in Table 6.12. We compare the results from these tables with those in Tables 6.2, 6.3 and 6.4 (i.e., the results for concept forgetting) and the results in Tables 6.7, 6.8 and 6.9 (i.e., the results for role forgetting), respectively. Careful inspection of all relevant tables shows that there was a slight increase in the success rates of FAME for forgetting concept symbols. The most likely cause of this is that the preceding role forgetting could have eliminated some concept symbols in Σ which could not be eliminated by our method for concept forgetting. This justifies FAME of eliminating role symbols first and then concept symbols.

Example 6.5.1. Consider the following set of clauses $\mathcal{N}$:

$$1.\ \forall s.A \sqcup \exists r.B$$

$$2.\ \neg C \sqcup \exists r.\neg B$$

Assume $\Sigma = \{B\}$. Observe that $\mathcal{N}$ cannot be transformed into pivot-reduced form by using the (rewrite$^{\mathsf{C}}$) rules in ACK$^{\mathsf{C}}$ because the pivot and the negated pivot both occur below existential role restrictions in Clauses 1 and 2, which are not existential clauses however (i.e., hence the Skolemisation$^{\mathsf{C}}$ rules are not applicable to Clauses 1 and 2 to transform them into pivot-reduced form). Assume $\Sigma = \{s, B\}$ and s is the pivot. Observe that s occurs only negatively in $\mathcal{N}$. We apply the Purify$^{\mathsf{R},-}$ rule to $\mathcal{N}$ to eliminate s, thereby leading to Clause 1 a tautology. Then B occurs only negatively in Clause 2. We apply the Purify$^{\mathsf{C},-}$ rule to Clause 2 to eliminate B, thereby yielding the set $\{\neg C \sqcup \exists r.\top\}$, which is the solution of forgetting $\{s, B\}$ from $\mathcal{N}$.

6.6 Comparison of Fame with Lethe

To verify the correctness of FAME, we have conducted a preliminary comparison of FAME with the verified tool LETHE on the $\mathcal{ALC}$-fragments of ontologies taken from the Oxford Ontology Repository [2] and the results have shown that:

1. If FAME successfully forgot all the symbols in the forgetting signature and computes a solution that is expressible in $\mathcal{ALC}$, rather than a more expressive description logic such as $\mathcal{ALCI}$, then the solutions computed by FAME and LETHE entail each other (i.e., they are logically equivalent).

2. If FAME successfully forgot all the symbols in the forgetting signature, but computes a solution in a more expressive description logic, then the solutions computed by FAME entail those computed by LETHE.

These results have verified the correctness of FAME. We have also compared the two tools in terms of their speed. The result has shown that FAME was considerably faster than LETHE, i.e., in average FAME was six times faster than LETHE for the forgetting tasks with the test ontologies. This has shown the speed advantage of the Ackermann-based forgetting approach over the resolution-based ones.

[2] https://www.cs.ox.ac.uk/isg/ontologies/

Chapter 7

Conclusions and Future Directions

Forgetting is a non-standard reasoning service that has a broad range of potential applications such as ontology summarisation, ontology analysis and reuse, ontology debugging and repair, information hiding, explanation generation (abduction) and computing logical differences between different versions of ontologies. Basically, it allows users to focus on specific parts of ontologies in order to create decompositions and restricted views of ontologies and it allows implicit information to be inferred from ontologies. Thus, forgetting can be used as a very useful tool in the area of (description logic-based) ontology engineering. However, on the other hand, forgetting is an inherently difficult problem — it is much harder than standard reasoning (satisfiability checking) — and very few logics are known to be complete for forgetting (or have the uniform interpolation property). There has been insufficient research on the topic and very few forgetting methods/tools are available at present.

Existing methods such as Nui and Lethe compute uniform interpolants for ontologies specified in a number of OWL language dialects ranging from the basic $\mathcal{ALC}$ to more expressive ones such as $\mathcal{ALCH}$, $\mathcal{SHI}$, $\mathcal{SIF}$ and $\mathcal{SHQ}$ etc. These methods, which are saturation approach based on resolution, do not provide support for forgetting in description logics with nominals. We have filled this gap in this thesis.

7.1 Conclusions

In this thesis, we investigated practical methods of (semantic) concept and role forgetting for ontologies expressible in the description logic $\mathcal{ALCOIH}(\triangledown, \sqcap)$, i.e., the basic

description logic $\mathcal{ALC}$ extended with nominals, inverse roles, role hierarchies, the universal role and role conjunctions. In particular, we developed a practical method for forgetting concept symbols from ontologies specified in description logics with inverse roles, and we developed a practical method for forgetting role symbols from ontologies specified in description logics with the universal role and role conjunctions. These two methods can be used as an integrated method for forgetting both concept and role symbols from ontologies specified in description logics with inverse roles, the universal role and role conjunctions.

The method for concept forgetting is based on a dedicated calculus, namely, ACK$^{\mathsf{C}}$, which is based on generalisations of Ackermann's Lemma for description logics and allows a single concept symbol to be eliminated from a set of $\mathcal{ALCOIH}(\triangledown, \sqcap)$-clauses. The calculus is comprised of a pair of the Ackermann$^{\mathsf{C}}$ rules, a pair of the Purify$^{\mathsf{C}}$ rules, a pair of the Surfacing$^{\mathsf{C}}$ rules, a pair of Skolemisation$^{\mathsf{C}}$ rules, a pair of the Skolemisation$^{\triangledown}$ rules, and a set of simplification rules. The Ackermann$^{\mathsf{C}}$ rules reflect the generalisations of Ackermann's Lemma and allow a single concept symbol to be eliminated from a set of clauses in pivot-reduced form. The Purify$^{\mathsf{C}}$ rules are special cases of the Ackermann$^{\mathsf{C}}$ rules and allow a single concept symbol to be eliminated from a set of clauses that is positive or negative w.r.t. this symbol (i.e., the pivot occurs only positively or only negatively in the clause set). The Surfacing$^{\mathsf{C}}$ rules, the Skolemisation$^{\mathsf{C}}$ rule and the Skolemisation$^{\triangledown}$ rules facilitate the transformation of the clause set into pivot-reduced form. In particular, the Surfacing$^{\mathsf{C}}$ rules allow the pivot (or the negated pivot) to be moved outside the scope of a universal role restriction in any clauses. The Skolemisation$^{\mathsf{C}}$ rules allow the pivot (or the negated pivot) to be moved outside the scope of an existential role restriction in an existential clause. The Skolemisation$^{\triangledown}$ rules allow the pivot (or the negated pivot) to be moved outside the scope of the universal role. The Skolemisation rules introduce new nominals in the ontologies, which are not in the desired signatures. Practical methods for forgetting nominals are not available at present, which means that we should avoid applying the Skolemisation rules (if possible). Crucial to the method are the simplification rules, which have a twofold purpose. On the one hand, the simplification rules are used to simplify clauses, making the clause set more accessible to the forgetting method and thus improving the efficiency of the method. On the other hand, the simplification

rules are used to transform the clause set into a set of general clauses, which facilitates the transformation of the clause set into pivot-reduced form. This improves the success rates of the method. We have shown that ACK$^{\mathsf{C}}$ is terminating and is sound in the sense that the elimination result is equivalent to the original ontology up to the interpretations of the symbol that has been eliminated, possibly with the interpretation of the nominals that have been introduced during the elimination process.

In order to gain an insight into the practicality of the method for concept forgetting, we implemented a prototype of the method in Java using the OWL API and evaluated the prototype on a corpus of 396 real-world ontologies. These ontologies are taken from the NCBO BioPortal repository, a resource so far including more than 600 biomedical ontologies developed for use in clinical and translational research. The evaluation results have shown that our method (for concept forgetting) can compute a solution of forgetting concept symbols from $\mathcal{ALCOIH}(\triangledown, \sqcap)$-ontologies in most test cases within a short period of time. Only a small number of nominals are introduced during the elimination process. This verifies the practicality of our method for concept forgetting.

The method for role forgetting is based on a dedicated calculus, namely, ACK$^{\mathsf{R}}$, which is based on a non-trivial generalisation of Ackermann's Lemma and allows a single role symbol to be eliminated from a set of $\mathcal{ALCOIH}(\triangledown, \sqcap)$-clauses. The calculus is comprised of the Ackermann$^{\mathsf{R}}$ rule, a pair of the Purify$^{\mathsf{R}}$ rules, a pair of the Inverting$^{\mathsf{R}}$ rules and a set of simplification rules. The Ackermann$^{\mathsf{R}}$ rule reflects the generalisation of Ackermann's Lemma and allows a single role symbol to be eliminated from a set of clauses in pivot-reduced form. The Purify$^{\mathsf{R}}$ rules are special cases of the Ackermann$^{\mathsf{R}}$ rule and allow a single role symbol to be eliminated from a set of clauses that is positive or negative w.r.t. this symbol (i.e., the pivot occurs only positively or only negatively in the clause set). The Inverting$^{\mathsf{R}}$ rules facilitate the transformation of the clause set into pivot-reduced form. In particular, the Inverting$^{\mathsf{R}}$ rules allow the pivot (or the negated pivot) to be moved outside the scope of the inverse operator in an RBox clause. The simplification rules in ACK$^{\mathsf{R}}$ are exactly the same as those in ACK$^{\mathsf{C}}$ and thus have the same use and effect as in concept forgetting, i.e., they are used to simplify the clauses and transform the pivot-clauses into pivot-reduced form, thereby improving the efficiency and success rates of the method. We have shown that the method is (role forgetting) complete for $\mathcal{ALCOH}(\triangledown, \sqcap)$-ontologies.

In order to gain an insight into the practicality of the method for role forgetting, we implemented a prototype of the method in Java using the OWL API and evaluated the prototype on a corpus of 333 real-world ontologies taken from the NCBO BioPortal repository. The evaluation results have shown that our method (for role forgetting) can compute a solution of forgetting role symbols from $\mathcal{ALCOIH}(\triangledown, \sqcap)$-ontologies in most test cases within a reasonable period of time. Concept definer symbols may be introduced in the ontologies during the forgetting process (to transform the Σ-clauses into normal form), which are not in the desired signatures. We attempt to eliminate them using our method for concept forgetting. Although, using this method, there is no guarantee that all introduced definer symbols can be eliminated from the ontologies, the evaluation results have shown that the introduced definer symbols have been successfully eliminated from the ontologies in all test cases.

The method for concept forgetting and the method for role forgetting can be integrated and used as a practical method for forgetting both concept and role symbols from $\mathcal{ALCOIH}(\triangledown, \sqcap)$-ontologies. This is because, in the method, concept and role symbols are eliminated in a focused way, that is, the rules for concept forgetting and those for role forgetting are mutually independent. The elimination of a (concept or role) symbol in Σ can be followed by the elimination of any other (concept or role) symbol in Σ. During the process of eliminating the role symbols in Σ, concept definer symbols may be introduced in the ontologies when the Σ-clauses are transformed into normal form. Our implementation FAME defaults to first forgetting all role symbols (in Σ) and then all concept symbols (in Σ) so that the introduced definer symbols can be eliminated as part of the subsequent concept forgetting. The evaluation results have shown that in this way, the efficiency of the method can be significantly improved.

7.2 Future Directions

Forgetting for more expressive description logics. The description logics considered in this thesis are those expressible in $\mathcal{ALCOIH}(\triangledown, \sqcap)$. A natural next step for future work is to extend the method to accommodate more expressivity such as qualified number restrictions and transitive and functional properties on roles, i.e., a practical method of concept and role forgetting for $\mathcal{SHOIQ}(\triangledown, \sqcap)$-ontologies. The

obstacles to the extension of the method to $\mathcal{SHOIQ}(\triangledown, \sqcap)$ are due to (i) the interaction between nominals, inverse roles, and qualified number restrictions, which lead to the almost complete loss of the tree model property [Tob00] (see [HS07] for this interaction), and (ii) nominals, inverse roles, and qualified number restrictions together have a dramatic influence on complexity: satisfiability of $\mathcal{ALCOIQ}$-concepts is NExpTime-hard [Tob00]. It has also been realised that when forgetting role symbols, the interaction between transitivity and role inclusions can lead to results where it is not clear how to represent them finitely [Koo15]. Recently, we have developed a practical method for forgetting role symbols from ontologies expressible in the description logic $\mathcal{ALCOQH}(\triangledown, \sqcap)$ [ZS17]. The method is terminating and is sound in the sense that the forgetting solution is equivalent to the original ontology up to the interpretations of the symbols that have been forgotten, possibly with the interpretations of the definer symbols that have been introduced during the forgetting process. We have shown that the method is (role forgetting) complete for $\mathcal{ALCOQ}(\triangledown)$-ontologies. Only problematic are the cases where forgetting a role symbol would require the combinations of certain qualified number restrictions and role inclusions.

Selection of Σ-symbols. An important feature of our method is that Σ-symbols can be flexibly specified. This means that the Σ-symbols are determined entirely by the user and their application demands. Nevertheless, there are situations where it is not clear which symbols should be selected as the Σ-symbols to be forgotten. For example, the BBC Sports Ontology provides a set of hierarchical controlled vocabulary split into many sports categories. However, one may be only interested in the information relating to football. In this scenario, we can use forgetting to create a restricted view of the ontology, where the information relating to football is fully preserved and those symbols not necessarily needed for the representation of this information (i.e. symbols not relating to football) are gotten rid of. However, it is not obvious in this case which information is not relevant to football and which symbols should be selected as the Σ-symbols to be forgotten. Even if we already know this, it is difficult to select them purely by hand, especially in the cases where a large number of symbols need to be forgotten. One may expect an easier and automatic way to do the selection.

Comparison of FAME with other systems. FAME is a prototypical implementation that fully realises the functions of our forgetting method. It would be of interest

to see how FAME is like compared to other forgetting tools such as LETHE [KS15b], SCAN [Ohl96] and DLS [Gus96]. For the comparison with LETHE, used as test data are ontologies restricted to the expressivity that both FAME and LETHE can handle. We have obtained preliminary results which show that FAME is faster than the current version of LETHE in some cases where a corpus of $\mathcal{ALC}$-ontologies were considered. For the comparison with SCAN and DLS, the strategy is first translating the test ontologies into first-order logic formulas, and then applying SCAN and DLS to the formulas to eliminate the (predicate) symbols in Σ. The results are first-order logic formulas that do not contain any (predicate) symbols in Σ. Since LETHE can handle some DL expressivity that FAME cannot handle, and FAME can handle some DL expressivity that LETHE cannot handle, it would be beneficial from both the LETHE and FAME perspective to integrate these two systems into one framework and use the integrated system to solve forgetting problems in more expressive description logics.

Use of Fame in real-world applications. Evaluation results have shown that FAME achieve good results in a lot of real-world use cases. The evaluation was however conducted with the test data set being a corpus of adjusted ontologies from the NCBO BioPortal repository where the ontologies are mainly used for system evaluation. It would be of interest to see how FAME performs in real-world applications. This could unfold the full potential of forgetting and FAME, especially in applications dealing with large and monolithic ontologies. An open question is whether Ackermann-based forgetting approaches can be used to approximate ontologies into new ontologies that are less expressive, since saturation-approaches have been found feasible in approximating ontologies [PT07, RPZ10, PRZ16, BCR10, LSW12, CFG$^+$14].

Bibliography

[ABM99] Carlos Areces, Patrick Blackburn, and Maarten Marx. A Road-Map on Complexity for Hybrid Logics. In *Computer Science Logic, 13th International Workshop, 8th Annual Conference of the EACSL*, volume 1683 of *Lecture Notes in Computer Science*, pages 307–321. Springer, 1999.

[ABM00] Carlos Areces, Patrick Blackburn, and Maarten Marx. The Computational Complexity of Hybrid Temporal Logics. *Logic Journal of the IGPL*, 8(5):653–679, 2000.

[Ack35] Wilhelm Ackermann. Untersuchungen *ü*ber das Eliminationsproblem der mathematischen Logik. *Mathematische Annalen*, 110(1):390–413, 1935.

[AdRdN01] Carlos Areces, Maarten de Rijke, and Hans de Nivelle. Resolution in Modal, Description and Hybrid Logic. *Journal of Logic and Computation*, 11(5):717–736, 2001.

[ANvB98] Hajnal Andréka, István Németi, and Johan van Benthem. Modal Languages and Bounded Fragments of Predicate Logic. *Journal of Philosophical Logic*, 27(3):217–274, 1998.

[BCR10] Elena Botoeva, Diego Calvanese, and Mariano Rodriguez-Muro. Expressive Approximations in *DL-Lite* Ontologies. In *Artificial Intelligence: Methodology, Systems, and Applications, 14th International Conference, AIMSA 2010*, volume 6304 of *Lecture Notes in Computer Science*, pages 21–31. Springer, 2010.

[BGW94a] Leo Bachmair, Harald Ganzinger, and Uwe Waldmann. Refutational theorem proving for hierarchic first-order theories. *Applicable Algebra in Engineering, Communication and Computing*, 5(3-4):193–212, 1994.

[BGW94b] Leo Bachmair, Harald Ganzinger, and Uwe Waldmann. Refutational Theorem Proving for Hierarchic First-Order Theories. *Applicable Algebra in Engineering, Communication and Computing*, 5:193–212, 1994.

[Bíl07] Marta Bílková. Uniform Interpolation and Propositional Quantifiers in Modal Logics. *Studia Logica*, 85(1):1–31, 2007.

[BKL+16] 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: Logical Foundation of Knowledge Graph Construction and Query Answering - 12th International Summer School 2016*, volume 9885 of *Lecture Notes in Computer Science*, pages 27–89. Springer, 2016.

[BLB08] Franz Baader, Carsten Lutz, and Sebastian Brandt. Pushing the EL Envelope Further. In *Proceedings of the Fourth OWLED Workshop on OWL: Experiences and Directions*, volume 496 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.

[BLM06] Philippe Besnard, Jérôme Lang, and Pierre Marquis. Variable Forgetting in Preference Relations over Propositional Domains. In *ECAI 2006, 17th European Conference on Artificial Intelligence*, volume 141 of *Frontiers in Artificial Intelligence and Applications*, pages 763–764. IOS Press, 2006.

[BLR+16] Elena Botoeva, Carsten Lutz, Vladislav Ryzhikov, Frank Wolter, and Michael Zakharyaschev. Query-Based Entailment and Inseparability for $\mathcal{ALC}$ Ontologies. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016* [DBL16], pages 1001–1007.

[BN03] Franz Baader and Werner Nutt. Basic Description Logics. In *The Description Logic Handbook: Theory, Implementation, and Applications*, pages 43–95. Cambridge University Press, 2003.

[Bor96] Alexander Borgida. On the Relative Expressiveness of Description Logics and Predicate Logics. *Artificial Intelligence*, 82(1-2):353–367, 1996.

[Bro03] Frank Markham Brown. *Boolean Reasoning: The Logic of Boolean Equations*. Dover Publications, 2003.

[Cal96] Diego Calvanese. Finite Model Reasoning in Description Logics. In *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning (KR'96), 1996.*, pages 292–303. Morgan Kaufmann, 1996.

[CFG+14] David Carral, Cristina Feier, Bernardo Cuenca Grau, Pascal Hitzler, and Ian Horrocks. *EL*-ifying ontologies. In *Automated Reasoning - 7th International Joint Conference, IJCAR 2014* [DBL14], pages 464–479.

[CG08] Willem Conradie and Valentin Goranko. IV. Semantic extensions of SQEMA. *Journal of Applied Non-Classical Logics*, 18(2-3):175–211, 2008.

[CGV06a] Willem Conradie, Valentin Goranko, and Dimiter Vakarelov. Algorithmic correspondence and completeness in modal logic. I. The core algorithm SQEMA. *Logical Methods in Computer Science*, 2(1), 2006.

[CGV06b] Willem Conradie, Valentin Goranko, and Dimiter Vakarelov. Algorithmic Correspondence and Completeness in Modal Logic. II. Polyadic and Hybrid Extensions of the Algorithm SQEMA. *Journal of Logic and Computation*, 16(5):579–612, 2006.

[CGV09] Willem Conradie, Valentin Goranko, and Dimiter Vakarelov. Algorithmic Correspondence and Completeness in Modal Logic. III. Extensions of the Algorithm SQEMA with Substitutions. *Fundam. Inform.*, 92(4):307–343, 2009.

[CGV10] Willem Conradie, Valentin Goranko, and Dimiter Vakarelov. Algorithmic correspondence and completeness in modal logic. V. Recursive extensions of SQEMA. *Journal of Applied Logic*, 8(4):319–333, 2010.

[Con06] Willem Conradie. On the strength and scope of DLS. *Journal of Applied Non-Classical Logics*, 16(3-4):279–296, 2006.

[Con09] Willem Conradie. Completeness and Correspondence in Hybrid Logic via an Extension of SQEMA. *Electronic Notes in Theoretical Computer Science*, 231:175–190, 2009.

[Cra57] William Craig. Three uses of the herbrand-gentzen theorem in relating model theory and proof theory. *Journal of Symbolic Logic*, 22(3):269–285, 1957.

[DBL14] *Automated Reasoning - 7th International Joint Conference, IJCAR 2014, Held as Part of the Vienna Summer of Logic, VSL 2014*, volume 8562 of *Lecture Notes in Computer Science*. Springer, 2014.

[DBL16] *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016*. IJCAI/AAAI Press, 2016.

[De 96] Giuseppe De Giacomo. Eliminating "Converse" from Converse PDL. *Journal of Logic, Language and Information*, 5(2):193–208, 1996.

[DH96] Giovanna D'Agostino and Marco Hollenberg. Uniform Interpolation, Automata and the Modal μ-Calculus. In *University of Utrecht*, page http://www.phil.ruu., 1996.

[DH00] Giovanna D'Agostino and Marco Hollenberg. Logical Questions Concerning The mu-Calculus: Interpolation, Lyndon and Los-Tarski. *Journal of Symbolic Logic*, 65(1):310–332, 2000.

[DL94a] Giuseppe De Giacomo and Maurizio Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. In *Proceedings of the 12th National Conference on Artificial Intelligence, Volume 1.*, volume 1, pages 205–212. AAAI Press, 1994.

[DL94b] Giuseppe De Giacomo and Maurizio Lenzerini. Concept Language with Number Restrictions and Fixpoints, and its Relationship with μ-calculus. In *11th European Conference on Artificial Intelligence, ECAI 1994*, pages 411–415. John Wiley and Sons, 1994.

[DL06] Giovanna D'Agostino and Giacomo Lenzi. On modal μ-calculus with explicit interpolants. *Journal of Applied Logic*, 4(3):256–278, 2006.

[DLL62] Martin Davis, George Logemann, and Donald W. Loveland. A machine program for theorem-proving. *Commun. ACM*, 5(7):394–397, 1962.

[DLS97] Patrick Doherty, Witold Lukaszewicz, and Andrzej Szalas. Computing Circumscription Revisited: A Reduction Algorithm. *J. Autom. Reasoning*, 18(3):297–336, 1997.

[DM96] Giuseppe De Giacomo and Fabio Massacci. Tableaux and Algorithms for Propositional Dynamic Logic with Converse. In *Automated Deduction - CADE-13, 13th International Conference on Automated Deduction, 1996, Proceedings*, volume 1104 of *Lecture Notes in Computer Science*, pages 613–627. Springer, 1996.

[DP60] Martin Davis and Hilary Putnam. A Computing Procedure for Quantification Theory. *J. ACM*, 7(3):201–215, 1960.

[dRV95] Maarten de Rijke and Yde Venema. Sahlqvist's theorem for Boolean algebras with operators with an application to cylindric algebras. *Studia Logica*, 54(1):61–78, 1995.

[EdC89] Patrice Enjalbert and Luis Fariñas del Cerro. Modal Resolution in Clausal Form. *Theor. Comput. Sci.*, 65(1):1–33, 1989.

[Eng96] T. Engel. Quantifier elimination in second-order predicate logic. Diplomarbeit, Fachbereich Informatik, Univ. des Saarlandes, Saarbrücken, Germany, October 1996.

[EW08] Thomas Eiter and Kewen Wang. Semantic forgetting in answer set programming. *Artificial Intelligence*, 172(14):1644–1672, 2008.

[FZ16] Hong Fang and Xiaowang Zhang. A tableau-based forgetting in ALCQ. In *Knowledge Graph and Semantic Computing: Semantic, Knowledge, and Linked Big Data - First China Conference, CCKS 2016*, volume 650 of *Communications in Computer and Information Science*, pages 110–116. Springer, 2016.

[GHSV04] V. Goranko, U. Hustadt, R. A. Schmidt, and D. Vakarelov. SCAN is complete for all sahlqvist formulae. In *Relational and Kleene-Algebraic Methods in Computer Science*, volume 3051 of *LNCS*, pages 149–162. Springer, 2004.

[GKV97] Erich Grädel, Phokion G. Kolaitis, and Moshe Y. Vardi. On the decision problem for two-variable first-order logic. *Bulletin of Symbolic Logic*, 3(1):53–69, 1997.

[GLW06] Silvio Ghilardi, Carsten Lutz, and Frank Wolter. Did I Damage My Ontology? A Case for Conservative Extensions in Description Logics. In Patrick Doherty, John Mylopoulos, and Christopher A. Welty, editors, *Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning*, pages 187–197. AAAI Press, 2006.

[GO92] Dov M. Gabbay and Hans Jürgen Ohlbach. Quantifier Elimination in Second-Order Predicate Logic. In Bernhard Nebel, Charles Rich, and William R. Swartout, editors, *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR'92). 1992.*, pages 425–435. Morgan Kaufmann, 1992.

[GOR99] Erich Grädel, Martin Otto, and Eric Rosen. Undecidability results on two-variable logics. *Arch. Math. Log.*, 38(4-5):313–354, 1999.

[Grä99] Erich Grädel. On The Restraining Power of Guards. *Journal of Symbolic Logic*, 64(4):1719–1742, 1999.

[GSS08] D. M. Gabbay, R. A. Schmidt, and A. Szałas. *Second-Order Quantifier Elimination: Foundations, Computational Aspects and Applications*, volume 12 of *Studies in Logic: Mathematical Logic and Foundations*. College Publications, 2008.

[Gus96] Joakim Gustafson. An implementation and optimization of an algorithm for reducing formulae in second-order logic. Technical report, University of Linköping, Sweden, 1996.

[GV06] Valentin Goranko and Dimiter Vakarelov. Elementary canonical formulae: extending Sahlqvist's theorem. *Ann. Pure Appl. Logic*, 141(1-2):180–217, 2006.

[GZ95] Silvio Ghilardi and Marek W. Zawadowski. Undefinability of propositional quantifiers in the modal system S4. *Studia Logica*, 55(2):259–271, 1995.

[Hen63] Leon Henkin. An Extension of the Craig-Lyndon Interpolation Theorem. *Journal of Symbolic Logic*, 28(3):201–216, 1963.

[Hen75] Sahlqvist Henrik. Completeness and correspondence in the first and second order semantics for modal logic. *Studies in Logic and the Foundations of Mathematics*, 82:110–143, 1975.

[HG97] Ian Horrocks and Graham Gough. Description Logics with Transitive Roles. In *Proceedings of the 1997 International Workshop on Description Logics*, volume 410 of *URA-CNRS*, 1997.

[HM08] Andreas Herzig and Jérôme Mengin. Uniform interpolation by resolution in modal logic. In *Logics in Artificial Intelligence, 11th European Conference, JELIA*, volume 5293 of *Lecture Notes in Computer Science*, pages 219–231. Springer, 2008.

[HP98] Ian Horrocks and Peter F. Patel-Schneider. Optimising Propositional Modal Satisfiability for Description Logic Subsumption. In Jacques Calmet and Jan A. Plaza, editors, *Artificial Intelligence and Symbolic Computation, International Conference AISC'98, Proceedings*, volume 1476 of *Lecture Notes in Computer Science*, pages 234–246. Springer, 1998.

[HS99] Ian Horrocks and Ulrike Sattler. A Description Logic with Transitive and Inverse Roles and Role Hierarchies. *Journal of Logic and Computation*, 9(3):385–410, 1999.

[HS07] Ian Horrocks and Ulrike Sattler. A Tableau Decision Procedure for $\mathcal{SHOIQ}$. *J. Autom. Reasoning*, 39(3):249–276, 2007.

[HST99] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical reasoning for expressive description logics. In *Logic Programming and Automated Reasoning, 6th International Conference, LPAR'99, 1999, Proceedings*, volume 1705 of *Lecture Notes in Computer Science*, pages 161–180. Springer, 1999.

[JLM+17] Jean Christoph Jung, Carsten Lutz, Mauricio Martel, Thomas Schneider, and Frank Wolter. Conservative Extensions in Guarded and Two-Variable Fragments. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017*, volume 80 of *LIPIcs*, pages 108:1–108:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

[KHM99] Jürg Kohlas, Rolf Haenni, and Serafín Moral. Propositional Information Systems. *Journal of Logic and Computation*, 9(5):651–681, 1999.

[KKS12] Yevgeny Kazakov, Markus Kroetzsch, and Frantisek Simancik. Practical Reasoning with Nominals in the EL Family of Description Logics. In Gerhard Brewka, Thomas Eiter, and Sheila A. McIlraith, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference, KR, 2012*. AAAI Press, 2012.

[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, The University of Manchester, UK, 2015.

[Kra07] Marcus Kracht. 8 modal consequence relations. *Studies in Logic and Practical Reasoning*, 3:491–545, 2007.

[KS13a] P. Koopmann and R. A. Schmidt. Uniform Interpolation of $\mathcal{ALC}$-Ontologies Using Fixpoints. In *Proc. FroCos'13*, volume 8152 of *LNCS*, pages 87–102. Springer, 2013.

[KS13b] Patrick Koopmann and Renate A. Schmidt. Forgetting concept and role

symbols in $\mathcal{ALCH}$-ontologies. In *Logic for Programming, Artificial Intelligence, and Reasoning - 19th International Conference, LPAR-19*, volume 8312 of *Lecture Notes in Computer Science*, pages 552–567. Springer, 2013.

[KS13c] Patrick Koopmann and Renate A. Schmidt. Implementation and Evaluation of Forgetting in ALC-Ontologies. In Chiara Del Vescovo, Torsten Hahmann, David Pearce, and Dirk Walther, editors, *Proceedings of the 7th International Workshop on Modular Ontologies co-located with the 12th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2013)*, volume 1081 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2013.

[KS13d] Patrick Koopmann and Renate A. Schmidt. Uniform Interpolation of $\mathcal{ALC}$-Ontologies Using Fixpoints. In *Frontiers of Combining Systems - 9th International Symposium, FroCoS 2013*, volume 8152 of *Lecture Notes in Computer Science*, pages 87–102. Springer, 2013.

[KS14] Patrick Koopmann and Renate A. Schmidt. Count and Forget: Uniform Interpolation of $\mathcal{SHQ}$ -Ontologies. In *Automated Reasoning - 7th International Joint Conference, IJCAR 2014, Held as Part of the Vienna Summer of Logic, VSL 2014* [DBL14], pages 434–448.

[KS15a] P. Koopmann and R. A. Schmidt. Uniform Interpolation and Forgetting for $\mathcal{ALC}$-Ontologies with ABoxes. In *Proc. AAAI'15*, pages 175–181. AAAI Press, 2015.

[KS15b] Patrick Koopmann and Renate A. Schmidt. LETHE: Saturation-Based Reasoning for Non-Standard Reasoning Tasks. In *Informal Proceedings of the 4th International Workshop on OWL Reasoner Evaluation (ORE-2015), 2015.*, volume 1387 of *CEUR Workshop Proceedings*, pages 23–30. CEUR-WS.org, 2015.

[KS15c] Patrick Koopmann and Renate A. Schmidt. Saturated-Based Forgetting

in the Description Logic SIF. In Diego Calvanese and Boris Konev, editors, *Proceedings of the 28th International Workshop on Description Logics, Athens,Greece, June 7-10, 2015.*, volume 1350 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015.

[KSH12] Markus Krötzsch, Frantisek Simancik, and Ian Horrocks. A Description Logic Primer. *CoRR*, abs/1201.4089, 2012.

[KWW08] Boris Konev, Dirk Walther, and Frank Wolter. The Logical Difference Problem for Description Logic Terminologies. In Alessandro Armando, Peter Baumgartner, and Gilles Dowek, editors, *Automated Reasoning, 4th International Joint Conference, IJCAR 2008*, volume 5195 of *Lecture Notes in Computer Science*, pages 259–274. Springer, 2008.

[KWW09] Boris Konev, Dirk Walther, and Frank Wolter. Forgetting and Uniform Interpolation in Large-Scale Description Logic Terminologies. In *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence*, pages 830–835, 2009.

[Lei17] João Leite. A Bird's-Eye View of Forgetting in Answer-Set Programming. In Marcello Balduccini and Tomi Janhunen, editors, *Logic Programming and Nonmonotonic Reasoning - 14th International Conference, LPNMR 2017*, volume 10377 of *Lecture Notes in Computer Science*, pages 10–22. Springer, 2017.

[LK13a] M. Ludwig and B. Konev. Towards practical uniform interpolation and forgetting for $\mathcal{ALC}$ TBoxes. In *Proc. DL'13*, volume 1014 of *CEUR Workshop Proceedings*, pages 377–389. CEUR-WS.org, 2013.

[LK13b] Michel Ludwig and Boris Konev. Towards Practical Uniform Interpolation and Forgetting for ALC TBoxes. In *Informal Proceedings of the 26th International Workshop on Description Logics*, volume 1014 of *CEUR Workshop Proceedings*, pages 377–389. CEUR-WS.org, 2013.

[LK14] Michel Ludwig and Boris Konev. Practical Uniform Interpolation and Forgetting for ALC TBoxes with Applications to Logical Difference. In

Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference, KR 2014. AAAI Press, 2014.

[LLM03] Jérôme Lang, Paolo Liberatore, and Pierre Marquis. Propositional Independence: Formula-Variable Independence and Forgetting. *Journal of Artificial Intelligence Research*, 18:391–443, 2003.

[LM02] Jérôme Lang and Pierre Marquis. Resolving inconsistencies by variable forgetting. In *Proceedings of the Eights International Conference on Principles and Knowledge Representation and Reasoning (KR-02)*, pages 239–250. Morgan Kaufmann, 2002.

[LR94] Fangzhen Lin and Ray Reiter. Forget It! In *Proc. AAAI Fall Symposium on Relevance*, pages 154–159. AAAI Press, 1994.

[LSW12] Carsten Lutz, Inanç Seylan, and Frank Wolter. An Automata-Theoretic Approach to Uniform Interpolation and Approximation in the Description Logic EL. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference, KR 2012*. AAAI Press, 2012.

[LW07] Carsten Lutz and Frank Wolter. Conservative Extensions in the Lightweight Description Logic EL. In Frank Pfenning, editor, *Automated Deduction - CADE-21, 21st International Conference on Automated Deduction*, volume 4603 of *Lecture Notes in Computer Science*, pages 84–99. Springer, 2007.

[LW10] Carsten Lutz and Frank Wolter. Deciding inseparability and conservative extensions in the description logic EL. *Journal of Symbolic Computation*, 45(2):194–228, 2010.

[LW11] Carsten Lutz and Frank Wolter. Foundations for uniform interpolation and forgetting in expressive description logics. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 989–995. IJCAI/AAAI Press, 2011.

[LWW07] Carsten Lutz, Dirk Walther, and Frank Wolter. Conservative Extensions in Expressive Description Logics. In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 453–458, 2007.

[MH09] Ralf Möller and Volker Haarslev. Tableau-Based Reasoning. In *Handbook on Ontologies*, International Handbooks on Information Systems, pages 509–528. Springer, 2009.

[MNS+12] Mark A. Musen, Natalya Fridman Noy, Nigam H. Shah, Patricia L. Whetzel, Christopher G. Chute, Margaret-Anne D. Storey, and Barry Smith. The national center for biomedical ontology. *JAMIA*, 19(2):190–195, 2012.

[Nik11] Nadeschda Nikitina. Forgetting in General EL Terminologies. In *Proceedings of the 24th International Workshop on Description Logics (DL 2011)*, volume 745 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2011.

[NR12] Nadeschda Nikitina and Sebastian Rudolph. On the (Non-)Succinctness of Uniform Interpolation in General ${\mathcal{EL}}$ Terminologies. In *Web Reasoning and Rule Systems - 6th International Conference, RR 2012*, pages 246–249, 2012.

[NR14] Nadeschda Nikitina and Sebastian Rudolph. (Non-)Succinctness of uniform interpolants of general terminologies in the description logic EL. *Artificial Intelligence*, 215:120–140, 2014.

[NS98] A. Nonnengart and A. Szałas. A Fixpoint Approach to Second-Order Quantifier Elimination with Applications to Correspondence Theory. In *Logic at work: essays dedicated to the memory of Helena Rasiowa*, volume 24, pages 307–328. Physica Verlag, 1998.

[Ohl96] Hans Jürgen Ohlbach. SCAN - Elimination of Predicate Quantifiers. In *Automated Deduction - CADE-13, 13th International Conference on Automated Deduction, 1996, Proceedings*, volume 1104 of *Lecture Notes in Computer Science*, pages 161–165. Springer, 1996.

[PGJ11] Heather S. Packer, Nicholas Gibbins, and Nicholas R. Jennings. An On-Line Algorithm for Semantic Forgetting. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence, IJCAI 2011*, pages 2704–2709. IJCAI/AAAI Press, 2011.

[Pit92] Andrew M. Pitts. On an Interpretation of Second Order Quantification in First Order Intuitionistic Propositional Logic. *Journal of Symbolic Logic*, 57(1):33–52, 1992.

[PRZ16] Jeff Z. Pan, Yuan Ren, and Yuting Zhao. Tractable approximate deduction for OWL. *Artificial Intelligence*, 235:95–155, 2016.

[PT07] Jeff Z. Pan and Edward Thomas. Approximating OWL-DL Ontologies. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, 2007*, pages 1434–1439. AAAI Press, 2007.

[Rob63] John Alan Robinson. Theorem-Proving on the Computer. *J. ACM*, 10(2):163–174, 1963.

[RPZ10] Yuan Ren, Jeff Z. Pan, and Yuting Zhao. Soundness Preserving Approximation for TBox Reasoning. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010*. AAAI Press, 2010.

[Sat96] Ulrike Sattler. A Concept Language Extended with Different Kinds of Transitive Roles. In *KI-96: Advances in Artificial Intelligence, 20th Annual German Conference on Artificial Intelligence, 1996, Proceedings*, volume 1137 of *Lecture Notes in Computer Science*, pages 333–345. Springer, 1996.

[Sch91] Klaus Schild. A Correspondence Theory for Terminological Logics: Preliminary Report. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence. 1991*, pages 466–471. Morgan Kaufmann, 1991.

[Sch94] Klaus Schild. Terminological cycles and the propositional μ-calculus. In *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning (KR'94). 1994.*, pages 509–520, 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.

[Sha03] Vladimir Yurievich Shavrukov. *Adventures in diagonalizable algebras*. ILLC Publications, 2003.

[Sim94] Harold Simmons. The Monotonous Elimination of Predicate Variables. *Journal of Logic and Computation*, 4(1):23–68, 1994.

[SS91] Manfred Schmidt-Schauß and Gert Smolka. Attributive Concept Descriptions with Complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[Str82] Robert S. Streett. Propositional Dynamic Logic of Looping and Converse Is Elementarily Decidable. *Information and Control*, 54(1/2):121–141, 1982.

[Sza93] Andrzej Szalas. On the Correspondence between Modal and Classical Logic: An Automated Approach. *Journal of Logic and Computation*, 3(6):605–620, 1993.

[Sza02] Andrzej Szalas. Second-order quantifier elimination in modal contexts. In *Logics in Artificial Intelligence, European Conference, JELIA 2002, Cosenza, Italy, September, 23-26, Proceedings*, volume 2424 of *Lecture Notes in Computer Science*, pages 223–232. Springer, 2002.

[Sza06] Andrzej Szalas. Second-order reasoning in description logics. *Journal of Applied Non-Classical Logics*, 16(3-4):517–530, 2006.

[Tob00] Stephan Tobies. The Complexity of Reasoning with Cardinality Restrictions and Nominals in Expressive Description Logics. *Journal of Artificial Intelligence Research*, 12:199–217, 2000.

[Var85] Moshe Y. Vardi. The Taming of Converse: Reasoning about Two-way Computations. In *Logics of Programs, Conference, Brooklyn College, 1985, Proceedings*, volume 193 of *Lecture Notes in Computer Science*, pages 413–423. Springer, 1985.

[vB75] Johan van Benthem. A Note on Modal Formulae and Relational Properties. *Journal of Symbolic Logic*, 40(1):55–58, 1975.

[vB76] Johan van Benthem. Modal reduction principles. *Journal of Symbolic Logic*, 41(2):301–312, 1976.

[vB89] Johan van Benthem. Notes on Modal Definability. *Notre Dame Journal of Formal Logic*, 30(1):20–35, 1989.

[vBDMP97] Johan van Benthem, Giovanna D'Agostino, Angelo Montanari, and Alberto Policriti. Modal Deduction in Second-Order Logic and Set Theory - I. *Journal of Logic and Computation*, 7(2):251–265, 1997.

[vdHdR95] Wiebe van der Hoek and Maarten de Rijke. Counting Objects. *Journal of Logic and Computation*, 5(3):325–345, 1995.

[Vis96] Albert Visser. *Bisimulations, Model Descriptions and Propositional Quantifiers.* Logic Group Preprint Series. Department of Philosophy, Utrecht Univ., 1996.

[WNS+11] Patricia L. Whetzel, Natalya Fridman Noy, Nigam H. 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(Web-Server-Issue):541–545, 2011.

[WWT+09] Kewen Wang, Zhe Wang, Rodney W. Topor, Jeff Z. Pan, and Grigoris Antoniou. Concept and role forgetting in $\mathcal{ALC}$ ontologies. In *The Semantic Web - ISWC 2009, 8th International Semantic Web Conference, ISWC 2009*, volume 5823 of *Lecture Notes in Computer Science*, pages 666–681. Springer, 2009.

[WWT+14] Kewen Wang, Zhe Wang, Rodney W. Topor, Jeff Z. Pan, and Grigoris Antoniou. Eliminating concepts and roles from ontologies in expressive descriptive logics. *Computational Intelligence*, 30(2):205–232, 2014.

[WWTP08] Zhe Wang, Kewen Wang, Rodney W. Topor, and Jeff Z. Pan. Forgetting Concepts in DL-Lite. In *The Semantic Web: Research and Applications, 5th European Semantic Web Conference, ESWC 2008*, volume 5021 of *Lecture Notes in Computer Science*, pages 245–257. Springer, 2008.

[WWTP10] Zhe Wang, Kewen Wang, Rodney W. Topor, and Jeff Z. Pan. Forgetting for knowledge bases in DL-Lite. *Annals of Mathematics and Artificial Intelligence*, 58(1-2):117–151, 2010.

[WWTZ10] Zhe Wang, Kewen Wang, Rodney W. Topor, and Xiaowang Zhang. Tableau-based forgetting in $\mathcal{ALC}$ ontologies. In *ECAI 2010 - 19th European Conference on Artificial Intelligence,*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, pages 47–52. IOS Press, 2010.

[ZFW05] Yan Zhang, Norman Y. Foo, and Kewen Wang. Solving Logic Program Conflict through Strong and Weak Forgettings. In *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, 2005*, pages 627–634. Professional Book Center, 2005.

[Zho14] Yi Zhou. Polynomially Bounded Forgetting. In *PRICAI 2014: Trends in Artificial Intelligence - 13th Pacific Rim International Conference on Artificial Intelligence, 2014*, volume 8862 of *Lecture Notes in Computer Science*, pages 422–434. Springer, 2014.

[ZS15] Yizheng Zhao and Renate A. Schmidt. Concept forgetting in $\mathcal{ALCOI}$-ontologies using an ackermann approach. In *The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference*, 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 $\mathcal{ALCOIH}\mu^{+}(\triangledown, \sqcap)$-ontologies. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016* [DBL16], pages 1345–1353.

[ZS17] Yizheng Zhao and Renate A. Schmidt. Role forgetting for $\mathcal{ALCOQH}(\triangledown)$-ontologies using an ackermann-based approach. In *Proceedings of the*

Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, pages 1354–1361. AAAI/IJCAI Press, 2017.

[ZZ10] Yan Zhang and Yi Zhou. Forgetting revisited. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Twelfth International Conference, KR 2010*. AAAI Press, 2010.

[ZZ11] Yi Zhou and Yan Zhang. Bounded Forgetting. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011*. AAAI Press, 2011.