Direct Approach to ABox Abduction in Description Logics

Mgr. Júlia Pukancová

Dissertation thesis

Faculty of Mathematics, Physics and Informatics
Comenius University in Bratislava

April 2018
Direct Approach to ABox Abduction in Description Logics

Dissertation thesis

study programme  Computer Science
field of study    Computer Science, Informatics
author           Mgr. Júlia Pukancová
advisor          doc. PhDr. Ján Šefránek, CSc.
consultant       RNDr. Martin Homola, PhD.
submitted        Bratislava, April 2018

Typeset in \LaTeX

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics
Department of Applied Informatics
Mlynská dolina, 842 48 Bratislava, Slovakia
We will deal with an abduction, 
there's no need of scared reaction. 
  No aliens, no kidnapping, 
only observed phenomenon explaining. 
  Did you observe the grass is wet 
and didn't find an explanation yet? 
  So let's start the journey together 
and you will love abduction forever!
**Name and Surname:** Mgr. Júlia Pukancová  
**Study programme:** Computer Science (Single degree study, Ph.D. III. deg., full time form)  
**Field of Study:** Computer Science, Informatics  
**Type of Thesis:** Dissertation thesis  
**Language of Thesis:** English  
**Secondary language:** Slovak  

**Title:** Direct Approach to ABox Abduction in Description Logics  
**Annotation:** Abduction is a specific reasoning task in which we look for an explanation of a certain observation that is captured as a formula that does not follow from a given knowledge base. We are looking for some hypothetical explanation that we could add to the knowledge base to support the observation. In diagnostics a certain system may be described by an ontology in description logics, and some observed condition of the system may be explained, particularly by ABox abduction. While applications for such a reasoning framework are known in domains such as medical diagnosis, manufacturing control, multimedia interpretation, etc., the development and implementation of specific reasoners for ABox abduction, particularly for more expressive description logics is yet rather new research area with a number of open problems.  

**Aim:** The aim of this thesis is to study the problem of abduction over ontologies represented in description logics with respect to intended applications in diagnostic reasoning. The thesis will feature theoretical goals including the development of abductive algorithms with specific properties relevant for the problem of diagnosis, but also implementation and experimental evaluation. Specific goals of the thesis include:  

* Abductive reasoning with description logics with greater expressivity than ALCI.  
* Focus on formal properties, especially soundness and completeness  
* To develop a working pilot implementation of the reasoner.  
* To empirically evaluate the implemented reasoner on experimental data sets.  

**Literature:**  


Tutor: doc. PhDr. Ján Šefránek, CSc.
Consultant: RNDr. Martin Homola, PhD.
Department: FMFI.KAI - Department of Applied Informatics
Head of department: prof. Ing. Igor Farkaš, Dr.

Assigned: 21.02.2014
Approved: 24.02.2016 prof. RNDr. Branislav Rovan, PhD.

Guarantor of Study Programme
ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Mgr. Júlia Pukancová
Študijný program: informatika (Jednoodoborové štúdium, doktorandské III. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: dizertačná
Jazyk záverečnej práce: anglický
Sekundárny jazyk: slovenský

Názov: Direct Approach to ABox Abduction in Description Logics
Priama metóda ABox abdukcie v deskripčnej logike

Anotácia: Abdukcia je špecifická inferenčná úloha, v ktorej hľadáme vysvetlenie pre určité pozorovanie v tvare formuly, ktorá nevyplýva z danej bázy znalostí. Hľadáme také hypotetické vysvetlenia, ktorých pridanie do bázy znalostí spôsobí vyplynutie pozorovania. V oblasti diagnostiky môžeme pre určitý systém popísaný ontológiou v deskripčnej logike hľadať vysvetlenia pre pozorované správanie systému, predovšetkým s využitím ABox abdukcie. Zatiaľ čo pre tento typ usudzovania poznáme aplikácie v doménach ako medicínska diagnostika, riadenie výroby, interpretácia multimedií a pod., vývoj a implementácia špecifických inferenčných softvérov pre ABox abdukciu, konkrétne pre expresívnejšie deskripčné logiky, je ešte pomerne nová výskumná oblasť s množstvom otvorených problémov.

Cieľ: Cieľom tejto práce je štúdium problému abdukcie nad ontológiami reprezentovanými v deskripčných logikách s ohľadom na využitie v diagnostickom usudzovaní. Práca zahŕňa jednak teoretické ciele vývoja abduktívnych algoritmov s konkrétnymi formálnymi vlastnosťami relevantnými pre problém diagnostiky, ale tiež implementáciu a experimentálnu evaluáciu. Konkrétne sa bude sústredovať na nasledujúce ciele:

* Abduktívne usudzovanie s expresívnejšimi logikami ako je ALCI.
* Dôraz na formálne vlastnosti algoritmu, predovšetkým zdravosť a úplnosť
* Navrhnúť fungujúcu pilotnú implementáciu inferenčného softvéru.
* Empirická evaluácia implementovaného inferenčného softvéru na experimentálnych dátach.


ACKNOWLEDGEMENTS

I would like to express my deep gratitude to Martin Homola for revealing to me the sense of doing this work and standing by me throughout the duration of the project. I exceptionally appreciate his willingness, helpfulness, patience, and guidance. No previous work catches up with the level of support I have received and knowledge I have gained.

I would also like to express my great appreciation to my supervisor Ján Šefránek for his valuable advice and suggestions. I am exceptionally grateful for introducing me to the beautiful world of logic.

Special thanks should be given to my family, friends, and colleagues, for their support and encouragement. In addition, I would also like to acknowledge all the organizers, attendees, and especially the reviewers of the conferences, doctoral consortia, and summer schools I attended, for their valuable feedback and inspiration. I am very thankful to my colleague and friend Ján Krička for his help and advice with the technical issues related to this work.

This work was supported by grant UK/426/2015 from Comenius University in Bratislava, VEGA projects 1/1333/12 and 1/0867/16, The Literature Fund of Slovakia, Slovak Society for Computer Science, and by extraordinary scholarships awarded by Faculty of Mathematics, Physics, and Informatics, Comenius University in Bratislava in 2016 and in 2017.
ABSTRACT

Abduction is a non-monotonic reasoning method used to derive explanations for an observed phenomenon. By enriching the theory with the explanation, the observation is entailed. More definitions of abduction problem exist according to the specific classes of the observations and the explanations, usually implied by the particular formalism. The family of description logics is a core formalism for representing ontologies, currently widespread because of applications in database management and querying, multi-agent systems, biomedical information systems, etc. This thesis is focused on abduction over description logics, as it recently earned an interest because of applications such as diagnostics, ontology debugging, semantic matchmaking, multimedia interpretation, etc. We focus on ABox abduction, since we are particularly interested in explanations at the extensional level, corresponding to the ABox in description logics. The main goal was to develop an ABox abduction algorithm addressing some open issues that are not completely addressed by the current related works. We have developed an ABox abduction algorithm for the description logic $\mathcal{ALC\!H\!O}$ based on the tableau algorithm for description logics and the minimal hitting set algorithm. We have formally proven soundness and completeness of the proposal. We have provided an implementation with focus on optimization techniques. An extensive empirical evaluation was conducted.
# CONTENTS

1 INTRODUCTION  
1.1 Ontologies and Description Logics  
1.2 Abductive Reasoning  
1.3 Goals of This Thesis  
1.4 Contribution of This Work  
1.5 Overview  

I STATE OF THE ART  

2 DESCRIPTION LOGICS  
2.1 $\mathcal{ALCH}$ Syntax  
2.2 $\mathcal{ALCH}$ Knowledge Base  
2.3 $\mathcal{ALCH}$ Semantics  
2.4 Decision Problems  
2.5 Tableau Algorithm for $\mathcal{ALCH}$  
2.6 Other Description Logics  
2.6.1 Role Transitivity  
2.6.2 Inverse Roles  
2.6.3 Role Functionality  
2.6.4 Number Restrictions  
2.6.5 Qualified Number Restrictions  
2.6.6 Other Extensions  

3 ABDUCTION  
3.1 Abduction in General  
3.2 Minimal Hitting Set Algorithm  
3.2.1 Conflict Sets  
3.2.2 Computing Hitting Sets  
3.3 Abduction in Description Logics  
3.3.1 Translation-based Approaches  
3.3.2 Direct Approaches  

II CONTRIBUTION  

4 BASIC IDEA  
4.1 Finding Explanations  
4.2 Minimal Hitting Sets and ABox Abduction  

5 ABOX ABDUCTION ALGORITHM  
5.1 Single Observation Algorithm  
5.2 Multiple Observations Algorithm  
5.2.1 Splitting into Subproblems  
5.2.2 Reduction  
5.3 Limited Length of Explanations
6 IMPLEMENTATION
6.1 Usage .............................................. 80
6.2 Example of Input and Output ................. 81
7 EVALUATION
7.1 Methodology ....................................... 83
7.2 Data Set ........................................... 85
7.3 Single Observation Experiment ............... 85
7.4 Different Samples of Single Observations .... 87
7.5 Multiple Observation Experiment ............. 87
7.6 Results ............................................. 93
  7.6.1 Exponential Growth of Time ............... 93
  7.6.2 Comparison of the Approaches ............. 94
  7.6.3 Number of the Explanations ............... 95
  7.6.4 Optimization of the HS-tree Search ....... 95
8 RELATED WORKS
8.1 Direct Approaches ................................. 101
8.2 Translation-Based Approaches ................ 102
9 CONCLUSIONS ......................................... 103

BIBLIOGRAPHY ........................................ 107
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>Peirce triangle</td>
<td>31</td>
</tr>
<tr>
<td>Figure 2</td>
<td>HS-tree for the set $M$ of Example 17</td>
<td>52</td>
</tr>
<tr>
<td>Figure 3</td>
<td>Subtree of HS-tree in Figure 2</td>
<td>53</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Depth vs. time: (a) without and (b) with loops</td>
<td>86</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Proportion of pruned nodes, reused models and TA calls from all HS-tree nodes in AAA for a single observation: (a) without and (b) with loops</td>
<td>88</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Execution times for different samples of single observations</td>
<td>89</td>
</tr>
<tr>
<td>Figure 7</td>
<td>Depth vs. time for multiple observations: (a) without and (b) with loops</td>
<td>90</td>
</tr>
<tr>
<td>Figure 8</td>
<td>Proportion of pruned nodes, reused models and TA calls from all HS-tree nodes in AAA for a multiple observation without loops</td>
<td>92</td>
</tr>
<tr>
<td>Figure 9</td>
<td>Proportion of pruned nodes, reused models and TA calls from all HS-tree nodes in AAA for a multiple observation with loops</td>
<td>92</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1</td>
<td>Interpretation of complex concepts</td>
<td>12</td>
</tr>
<tr>
<td>Table 2</td>
<td>Transformation to NNF</td>
<td>17</td>
</tr>
<tr>
<td>Table 3</td>
<td>Transformation rules of the tableau algorithm</td>
<td>18</td>
</tr>
<tr>
<td>Table 4</td>
<td>Transformation rules of the tableau algorithm with blocking</td>
<td>23</td>
</tr>
<tr>
<td>Table 5</td>
<td>DL constructors</td>
<td>27</td>
</tr>
<tr>
<td>Table 6</td>
<td>Other DL constructors</td>
<td>30</td>
</tr>
<tr>
<td>Table 7</td>
<td>Semantics of other DL constructors</td>
<td>30</td>
</tr>
<tr>
<td>Table 8</td>
<td>Parameters of the ontologies</td>
<td>86</td>
</tr>
<tr>
<td>Table 9</td>
<td>Average execution times in seconds (avg) and deviations in % (dev)</td>
<td>86</td>
</tr>
<tr>
<td>Table 10</td>
<td>Different samples of observations for single observation experiment</td>
<td>90</td>
</tr>
<tr>
<td>Table 11</td>
<td>Average execution times in seconds (avg) and deviations in % (dev) for multiple observations (splitting approach)</td>
<td>91</td>
</tr>
<tr>
<td>Table 12</td>
<td>Average execution times in seconds (avg) and deviations in % (dev) for multiple observations (reduction approach)</td>
<td>91</td>
</tr>
<tr>
<td>Table 13</td>
<td>Parameters of HS-trees for AAA without loops</td>
<td>97</td>
</tr>
<tr>
<td>Table 14</td>
<td>Parameters of HS-trees for AAA with loops</td>
<td>98</td>
</tr>
<tr>
<td>Table 15</td>
<td>Parameters of HS-trees for AAA for multiple observations without loops</td>
<td>99</td>
</tr>
<tr>
<td>Table 16</td>
<td>Parameters of HS-trees for AAA for multiple observations with loops</td>
<td>100</td>
</tr>
</tbody>
</table>
INTRODUCTION

1.1 ONTOLOGIES AND DESCRIPTION LOGICS

Ontologies are formal structures capturing hierarchies between classes relevant for a specific domain. They are used to describe semantics of data in many application domains: database management and querying, multi-agent systems, biomedical information systems, e-commerce applications, e-learning, etc. (Staab & Studer 2004).

Description logics (Baader et al. 2003) are a family of languages used as representation formalism for ontologies. The main advantage of description logics is that they enable to reason over ontologies, including checking for ontology consistency and derivation of consequences. In fact, description logics constitute a decidable fragment of first order logic.

In the research area of description logics, the standard reasoning problems that are most intensively investigated include consistency checking, subsumption checking, and instance retrieval. A number of well optimized reasoners were developed (Sirin et al. 2007, Shearer et al. 2008, Steigmiller et al. 2014, Haarslev et al. 2012, Horrocks 1998a,b). The description logics reasoners are mainly focused on these standard problems, and they are often highly optimized employing techniques such as tableau caching and incremental reasoning (Tsarkov et al. 2007).

1.2 ABDUCTIVE REASONING

Abductive reasoning, originally introduced by Peirce (1878), is a form of non-monotonic reasoning which is used to derive an explanation for an observed phenomenon. Given a theory \( \Gamma \) and an observation \( O \) that is not entailed by the theory (i.e. \( \Gamma \models \neg O \)), we are looking for an explanation \( \mathcal{E} \) such that \( \Gamma \cup \mathcal{E} \models O \). That means, with extending the theory \( \Gamma \) by the explanation \( \mathcal{E} \), the observation \( O \) is entailed. A classical example of abductive reasoning is with having the rule \textit{when it rains, the grass is wet} and the observation \textit{the grass is wet}. Intuitively, this observation is explained by the assumption that \textit{it rains}.

In the area of ontologies, abduction was investigated in context of a number of application domains. This type of reasoning is applied for example in diagnostic problems (Hubauer et al. 2011), ontology
debugging (Wei-Kleiner et al. 2014), semantic matchmaking (Colucci et al. 2005), multimedia interpretation (Petasis et al. 2013, Kaya et al. 2007).

While there is a number of applications for abduction, there is not yet a large number of works dedicated to abduction in description logics. Elsenbroich et al. (2006) established a categorization of the main abduction problems. In their work, they defined four types of abduction according to a class of observations and a class of explanations. Namely, the four types are concept abduction, knowledge base abduction, ABox abduction, and TBox abduction. Since the knowledge in the description logic knowledge base is split into the extensional part (ABox) and the intensional part (TBox), obviously in ABox abduction the class of observations and the class of explanations are restricted to the data, and in TBox abduction to TBox axioms.

Approaches to solve ABox abduction algorithmically are split in the two main groups. On the one hand, some works are based on a translation to another formalism. Klarman et al. (2011) utilizes standard techniques for translation of a description logic knowledge base to the modal logic and first order logic. The approach of Du et al. (2012) is built on the idea to use existing Prolog abduction solver, and so they dealt with a reduction to the logic programming. The works from the other group are based on solving the abduction problem through standard reasoning techniques for description logics (Halland & Britz 2012a, Ma et al. 2012).

1.3 GOALS OF THIS THESIS

Halland & Britz (2012a) proposed to solve abduction for description logics directly, by extending the standard tableau reasoning techniques (Baader et al. 2003) for description logics and exploiting the minimal hitting set algorithm (Reiter 1987). They assumed higher effectiveness thanks to avoiding a translation into another formalism and thanks to relying on the standard tableau optimization techniques. However they focused mainly on the required adjustment of the tableau algorithm and used the minimal hitting set algorithm as a black box. Completeness of their proposal was not proven and their proposal was also not implemented.

In this work, we build on top of the proposal of Halland and Britz, addressing the above mentioned issues. We focused on the following goals:

- increasing the expressivity of the underlying description logic and extending the classes of observations and explanations,
- formally proving both soundness and completeness,
1.4 CONTRIBUTION OF THIS WORK

We have proposed an ABox abduction solver for the description logic $\mathcal{ALC}$. Our approach is based on the tableau algorithm for description logics and on the minimal hitting set algorithm. The proposal utilizes the Reiter’s optimizations techniques for pruning of the search space. We have extended these techniques with other relevant for our purpose. Additionally, since the preferred explanations are the minimal ones, our algorithm enables to terminate when the explanations with all the desired maximal length are obtained.

We have proven soundness and completeness of the algorithm for the class of observations consisting of any type of $\mathcal{ALC}$ ABox assertions and for the class of explanations consisting of atomic and negated atomic $\mathcal{ALC}$ ABox assertions. The complexity was also analyzed. We have developed an implementation exploiting the Pellet reasoner (Sirin et al. 2007). We have implemented a number of optimizations in the minimal hitting set algorithm including Reiter’s pruning techniques, plus some of our own. We have conducted the empirical evaluation, that showed the necessity of the limitation on the length of the explanations. On the other hand, usually the high number of explanations were obtained already for the explanations of the length 1 and 2, which usually takes only a fraction of the overall time. An interesting lesson for the future work were learned thanks to the evaluation.

Summing up, the results presented in this thesis: a sound and complete ABox abduction algorithm for the description logic $\mathcal{ALC}$ with the implementation and the results from the empirical evaluation present a contribution that have not yet been addressed to this extent in the previous works.

The results presented in this thesis were published in the following publications:


- implementation using an existing tableau reasoner for description logics,
- empirical evaluation.

• Pukancová, J. & Homola, M. (2017), Tableau-based ABox abduction for the $\mathcal{ALCHO}$ description logic, in ‘Proceedings of the 30th International Workshop on Description Logics, Montpellier, France, July 18-21, 2017’.

The results were also presented at the following doctoral consortium and summer school:

• The 15th International Conference on Principles of Knowledge Representation and Reasoning Doctoral Consortium, April 2016,

• Third Nordic Logic Summer School (NLS), August 2017.

1.5 OVERVIEW

The thesis consists of the two main parts – State of the Art and Contribution. In the first part, description logics and abduction are presented. Chapter 2 deals with the syntax and semantics of the description logic $\mathcal{ALCHO}$, that is our main interest. The chapter includes the decision problems and the tableau algorithm. In the end of the chapter, the overview of more expressive description logics is given.

Chapter 3 studies the abductive reasoning firstly in general. Afterwards, the minimal hitting set algorithm as a tool for solving abduction problems is described. Next part is focused on abduction in description logics, specifically focusing on recent approaches to ABox abduction.

In the second part of the thesis, we firstly present the basic idea of our solution, in Chapter 4. The following chapter, Chapter 5 contains the proposal of our algorithm with all its parts. Also, the correctness of our algorithm is formally established in the terms of soundness and completeness, and its complexity is analyzed.

The implementation is described in Chapter 6. In Chapter 7 we concentrate on the evaluation and on the experiment results analysis. Chapter 8 focuses on the comparison with the related works. The conclusions and the future work are given in Chapter 9.
Part I

STATE OF THE ART
Description logics (DL) are decidable fragments of first-order logic (FOL). This representation formalism enables to model a class hierarchy, hierarchy of relations, and even more complex rules. Nowadays DLs are mainly used as a core formalism for ontologies – main representation formalism on the semantic web. According to a specific DL and the DL constructors it uses we can build knowledge bases with different expressivities.

For instance, the knowledge base in Example 1 describes the animal taxonomy, the rule that every human being has at least one parent that is also human, and the assertion of Lily being a human.

**Example 1.** Knowledge base $\mathcal{K}$ describing animal taxonomy:

$$\begin{align*}
\text{Chordate} & \sqsubseteq \text{Animal} \\
\text{Mammal} & \sqsubseteq \text{Chordate} \\
\text{Carnivore} & \sqsubseteq \text{Mammal} \\
\text{Human} & \sqsubseteq \text{Mammal} \\
\text{Human} & \sqsubseteq \exists \text{hasParent.Human} \\
\text{lily} & : \text{Human}
\end{align*}$$

In this chapter, we will first introduce the $\mathcal{ALCH}$ DL which will be the most relevant for our work. We will deal with its syntax, knowledge base, and semantics. In the next section we will focus on the decision problems studied in DL. Afterwards, we will introduce reasoning algorithms for DL $\mathcal{ALCH}$ with focus on the tableau algorithm. Finally, other DLs will be shortly described.

### 2.1 $\mathcal{ALCH}$ syntax

A vocabulary in every DL consists of individuals (specific instances, analogous to the constants in FOL), concepts (describing classes, analogous to the unary predicates in FOL) and roles (describing relations, analogous to the binary predicates in FOL).

**Definition 1** (Vocabulary). A DL vocabulary consists of three countable mutually disjoint sets:

- **set of individuals** $N_I = \{a, b, c, \ldots\}$
- **set of atomic concepts** $N_C = \{A, B, \ldots\}$
• set of roles \( N_R = \{ R, S, \ldots \} \)

For instance, in the Example 2 the vocabulary for the knowledge base in Example 1 is defined.

**Example 2.** Vocabulary for the knowledge base \( K \) from the Example 1:

\[
N_I = \{ \text{lily} \} \\
N_C = \{ \text{Chordate, Carnivore, Animal, Mammal, Human} \} \\
N_R = \{ \text{hasParent} \}
\]

In addition, from atomic symbols defined above we construct complex concepts. According to the specific DL, a particular set of constructors is used. For ALCHO there are following constructors: \( \neg \) (complement), \( \cap \) (concept intersection), \( \cup \) (concept union), \( \exists \) (existential restriction), and \( \forall \) (value restriction). Special types of concepts are nominal \( \{ a \} \) where \( a \) is an individual, top \( \top \) and bottom \( \bot \).

**Definition 2 (ALCHO concepts).** Concepts are recursively constructed as the smallest set of expressions of the forms:

\[
C, D ::= A | \neg C | C \cap D | C \cup D | \exists R.C | \forall R.C | \{ a \} | \top | \bot
\]

where \( A \in N_C \), \( R \in N_R \), \( a \in N_I \) and \( C, D \) are concepts.

Thanks to the complement constructor and concept intersection we can describe a class that actually corresponds to the set intersection and set complement. For example, all animals that are not carnivores are described with the following complex concept:

\[
\text{Animal} \cap \neg \text{Carnivore}.
\]

Similarly, concept union is used to express union of two classes. For instance, the class of carnivores and herbivores is as follows:

\[
\text{Carnivore} \cup \text{Herbivore}.
\]

Both existential and value restrictions are bound to a role \( R \). The value restriction \( \exists R.C \) determines a class of objects for which the existence of a relation \( R \) with an element belonging to the concept \( C \) is required. On the other hand, the value restriction \( \forall R.C \) determines a class of objects that are in the \( R \)-relation only with an element belonging to the class \( C \).

The following complex concept describes a class of everything, that has at least one human child and in the same time all of their children are human:

\[
\exists \text{hasChild.Human} \cap \forall \text{hasChild.Human}.
\]
A nominal \{a\} is a singleton class defined by an individual a. It is useful in such cases where we need to create a binding not to a whole class but only to an particular individual. For example, since South Africa is a particular country, the class of South-African animals is:

\[
\text{Animal} \sqcap \exists \text{livesIn}.\{\text{SouthAfrica}\}.
\]

The complex concept of nominals union \{a_1\} \sqcup \cdots \sqcup \{a_n\} can be shortly written as \{a_1, \ldots, a_n\}, where \{a_1, \ldots, a_n\} is an enumerated class of individuals \(a_1, \ldots, a_n\). Through this syntactic sugar we can simply describe a class of persons who like only South-African, Italian and French wine:

\[
\text{Person} \sqcap \forall \text{likeWine}.\forall \text{producedIn}.\{\text{SouthAfrica, Italy, France}\}.
\]

Sometimes the range in existential or value restricted complex concept does not have to be defined. For these cases special concepts top \top and bottom \bot can be used. The top concept \top is simply a class of everything. On the other hand, the bottom concept \bot is a class of nothing. For instance, the class of animals having a non-specific child and the class of animals having no child are described as follows:

\[
\text{Animal} \sqcap \exists \text{hasChild}.\top,
\]

\[
\text{Animal} \sqcap \forall \text{hasChild}.\bot.
\]

### 2.2 ALCDB Knowledge Base

As we are already able to define complex concepts, we can now describe meaningful rules, in DL called axioms. Through them the intensional knowledge is built. This part of knowledge base is called the TBox.

**Definition 3** (TBox). A TBox \(\mathcal{\Sigma}\) is a finite set of GCI (general concept inclusion) axioms \(\phi\) of the form:

\[
\phi ::= C \sqsubseteq D
\]

where \(C, D\) are any concepts.

Using GCIs it is possible to describe which class is more specific than the other – \(C \sqsubseteq D\) means that \(C\) is more specific than \(D\). It can also be read as an implication, that means each instance of \(C\) is also an instance of \(D\). The axioms from the Example 1 can serve as examples
of GCIs. Also more complex examples can be built, such as man and woman are humans, or aquatic animals live in water:

\[
\text{Man} \sqsubseteq \text{Woman} \sqsubseteq \text{Human}, \\
\text{AquaticAnimal} \sqsubseteq \exists \text{livesIn}.\text{Water}.
\]

Note that, C is also called a \textit{subconcept} of D or vice versa D is a \textit{superconcept} of C if and only if C \sqsubseteq D. To define that two concepts are \textit{equivalent}, C \equiv D is used as syntactic sugar for the two GCIs C \sqsubseteq D and D \sqsubseteq C.

In \textit{ALCHO} it is possible to describe role hierarchies as well. These are also kind of intensional knowledge, but are stored in the RBox.

**Definition 4** (RBox). An RBox \( \mathcal{R} \) is a finite set of RIAs (role inclusion axioms) \( \phi \) of the form:

\[
\phi ::= R \sqsubseteq S
\]

where R, S are any roles.

Actually, in \textit{ALCHO} all roles are atomic roles, i.e. an \textit{ALCHO} RIA is an axiom in form R \sqsubseteq S where R, S \in N_R. An RIA serves to express specificity between roles. For instance, to have a mother means to have a parent:

\[
\text{hasMother} \sqsubseteq \text{hasParent}.
\]

Additionally, R \sqsupseteq S expresses the transitive-reflexive closure over an RBox \( \mathcal{R} \), i.e. R \sqsupseteq S if and only if R = S or R \sqsubseteq R_1, \ldots, R_n \sqsubseteq S for any R, R_1, \ldots, R_n, S \in N_R.

The extensional knowledge, also called data, is stored in the ABox. It contains axioms that are called assertions, as they assert information about individuals – what concepts an individual belongs to or what are the relations between two individuals.

**Definition 5** (ABox). An ABox \( \mathcal{A} \) is a finite set of concept assertions \( \phi \) and role assertions \( \psi \) of the form:

\[
\phi ::= C(a) \\
\psi ::= R(b, c)
\]

where \( a, b, c \in N_I, R \in N_R, \text{and } C \text{ is any concept.} \)

In ABox assertions, individuals play a role. Either the assertions describe that an individual belongs to a class, e.g. Tom is a cat, or it
describes a relation between two individuals, e.g. Tom and Jerry are friends:

\[
\text{tom: Cat, } \quad \text{tom, jerry: hasFriend.}
\]

Note that negated role assertions are not usually considered in DL, however in \(\text{ALCHO}\) they can be easily introduced, as they can be encoded using nominals: \(\neg R(a, b)\) is equivalent to \(\forall R.\neg\{b\}(a)\) and equally to \(\{a\} \subseteq \forall R.\neg\{b\}\) (Horrocks et al. 2006). Every reasoner that supports \(\text{ALCHO}\) without negated role assertions can be also used to reason with them thanks to this reduction.

Also we will use \(\neg O\) for the negation of the assertion \(O\). That is, \(\neg O = \neg C(a)\) if \(O = C(a)\), \(\neg O = \neg R(a, b)\) if \(O = R(a, b)\), and \(\neg O = R(a, b)\) if \(O = \neg R(a, b)\), for any concept \(C\), any \(R \in \mathcal{N}_R\), and any \(a, b \in \mathcal{N}_I\). Finally, given any ABox \(A\), \(\neg A = \{\neg O | O \in A\}\).

Above mentioned intensional knowledge in form of a TBox and an RBox together with extensional data in form of an ABox form a DL knowledge base.

**Definition 6 (\(\text{ALCHO}\) knowledge base).** An \(\text{ALCHO}\) knowledge base \(K = (T, R, A)\) is a triple consisting of a TBox \(T\), an RBox \(R\), and an ABox \(A\).

### 2.3 \(\text{ALCHO}\) semantics

In order to provide semantics for \(\text{ALCHO}\), firstly we need to define an interpretation that consists of a non-empty domain \(\Delta^I\) and an interpretation function \(\cdot^I\), which provide meaning for each symbol and for the whole knowledge base. We need to specifically define an interpretation of the complex concepts.

**Definition 7 (Interpretation).** An interpretation for a given DL knowledge base \(K = (T, R, A)\) is a pair \(I = (\Delta^I, \cdot^I)\) which contains a non-empty domain \(\Delta^I \neq \emptyset\), and an interpretation function \(\cdot^I\) such that:

\[
\begin{align*}
\forall a \in \mathcal{N}_I & \quad a^I \in \Delta^I; \\
\forall A \in \mathcal{N}_C & \quad A^I \subseteq \Delta^I; \\
\forall R \in \mathcal{N}_R & \quad R^I \subseteq \Delta^I \times \Delta^I.
\end{align*}
\]

In addition, for any concept \(C, D\), any role \(R\), and any individual \(a\), the interpretation of complex concepts is recursively defined in Table 1.

From Table 1 it is clear, that concepts \(\top\) and \(\bot\) can be also introduced as syntactic sugar, since their semantics correspond to the complex concepts \(C \sqcup \neg C\) and \(C \sqcap \neg C\) for some concept \(C\) respectively.
Example 3 shows one of the possible interpretations for the particular knowledge base.

**Example 3.** Let $\mathcal{K}$ be a $\mathcal{ALCHO}$ knowledge base with the following axioms:

| $\text{Cat} \sqcup \text{Mouse} \sqsubseteq \text{Mammal}$ |
| $\text{Mammal} \sqsubseteq \text{Animal}$ |
| $\text{hasFriend} \sqsubseteq \text{knows}$ |
| $\text{Cat}(\text{tom})$ |
| $\text{hasFriend}(\text{tom}, \text{jerry})$ |

Then an interpretation $I$ for $\mathcal{K}$ is a pair $((\emptyset, \emptyset, \emptyset, \emptyset), ^3)$ where:

$\text{tom}^3 = \emptyset$

$\text{jerry}^3 = \emptyset$

$\text{Cat}^3 = \{\emptyset\}$

$\text{Mouse}^3 = \{\emptyset\}$

$\text{Mammal}^3 = \{\emptyset, \emptyset\}$

$\text{Animal}^3 = \{\emptyset, \emptyset, \emptyset, \emptyset\}$

$\text{hasFriend}^3 = \{(\emptyset, \emptyset), (\emptyset, \emptyset)\}$

$\text{knows}^3 = \{(\emptyset, \emptyset), (\emptyset, \emptyset), (\emptyset, \emptyset)\}$

Note that, the interpretation $I$ in Example 3 is just an example of an interpretation for $\mathcal{K}$. A knowledge base can be interpreted in many ways.

It is very important to be able to decide whether an interpretation satisfies an axiom or not. As there are more types of axioms in $\mathcal{ALCHO}$ knowledge base, for each type it must be defined when an interpretation satisfies it.

**Definition 8 (Satisfaction $\models$).** Given an axiom $\phi$, an interpretation $I = (\Delta^3, ^3)$ satisfies $\phi$ (denoted by $I \models \phi$) depending on its type:
\[ I \models C \subseteq D \iff C^3 \subseteq D^3 \]
\[ I \models R \subseteq S \iff R^3 \subseteq S^3 \]
\[ I \models C(a) \iff a^3 \in C^3 \]
\[ I \models R(a, b) \iff (a^3, b^3) \in R^3 \]

Although an axiom in form \( \neg R(a, b) \) is not listed in Definition 8 (since it was defined as syntactic sugar for \( \exists R, \{ b \}(a) \)) it is included in the case of \( C(a) \), one can deduce, that \( I \models \neg R(a, b) \) if and only if \( (a^3, b^3) \notin R^3 \).

Similarly, as \( C \equiv D \) is syntactic sugar for \( C \subseteq D \) and \( D \subseteq C \), it directly follows from Definition 8, that \( I \models C \equiv D \) if and only if \( C^3 \subseteq D^3 \) and \( D^3 \subseteq C^3 \), and consequently if and only if \( C^3 = D^3 \).

Once we are able to determine, whether an axiom is satisfied by an interpretation or not, we are able to determine, whether the whole knowledge base \( \mathcal{K} \), that is every axiom in \( \mathcal{K} \), is satisfied. In such a case, we call this interpretation a model of \( \mathcal{K} \).

**Definition 9** (Model). An interpretation \( I = (\Delta^3, \cdot^3) \) is a model of a DL knowledge base \( \mathcal{K} = (T, R, A) \) (denoted \( I \models \mathcal{K} \)) iff it satisfies every axiom in \( T \), every axiom in \( R \), and every assertion in \( A \).

An interpretation can be easily checked for satisfying a whole knowledge base through checking a satisfaction of all its axioms. This process is showed in the following example.

**Example 4** (Revisited). Considering the knowledge base \( \mathcal{K} \) and the interpretation \( I \) from Example 3, \( I \models \mathcal{K} \) iff:

\[
\begin{align*}
\text{Cat}^3 & \cup \text{Mouse}^3 \subseteq \text{Mammal}^3 \\
\text{Mammal}^3 & \subseteq \text{Animal}^3 \\
\text{hasFriend}^3 & \subseteq \text{knows}^3 \\
\text{tom}^3 & \in \text{Cat}^3 \\
(\text{tom}^3, \text{jerry}^3) & \in \text{hasFriend}^3
\end{align*}
\]

This is satisfied, because:

\[
\begin{align*}
\{\text{\textbullet}^3\} & \cup \{\text{\textbullet}^3\} \subseteq \{\text{\textbullet}^3, \text{\textbullet}^3\} \\
\{\text{\textbullet}^3, \text{\textbullet}^3\} & \subseteq \{\text{\textbullet}^3, \text{\textbullet}^3, \text{\textbullet}^3, \text{\textbullet}^3\} \\
\{\text{\textbullet}^3, \text{\textbullet}^3, \text{\textbullet}^3, \text{\textbullet}^3\} & \subseteq \{\text{\textbullet}^3, \text{\textbullet}^3, \text{\textbullet}^3, \text{\textbullet}^3, \text{\textbullet}^3\} \\
\text{\textbullet}^3 & \in \{\text{\textbullet}^3\} \\
(\text{\textbullet}^3, \text{\textbullet}^3) & \in \{\text{\textbullet}^3, \text{\textbullet}^3\}
\end{align*}
\]
Since I satisfies all axioms in K, I is a model of K.

Later on in our work, we will deal with models encoded by ABox assertions. In fact, such an encoding can be defined for any interpretation.

**Definition 10 (ABox Encoding of Interpretations).** The assertional encoding (also, the ABox encoding) of an interpretation I is an ABox \( M_i \) constructed as follows:

\[
M_i = \{ C(a) \mid J \models C(a), \ C \in \{A, \neg A\}, \ A \in N_C, \ a \in N_I \} \\
\cup \{ R(a, b) \mid J \models R(a, b), \ R \in N_R, \ a, b \in N_I \} \\
\cup \{ \neg R(a, b) \mid J \models \neg R(a, b), \ R \in N_R, \ a, b \in N_I \}.
\]

For each interpretation, there is exactly one ABox encoding, and it is a sufficient representation when concerned with the meaning of individuals that appear in the ABox in what regards to their membership to atomic concepts and roles, which follows directly from its construction.

**Observation 1.** The ABox encoding \( M_i \) of any interpretation I is uniquely determined. Moreover \( J \models C(a) \) (resp., \( J \models R(a, b) \)) iff \( C(a) \in M_i \) (resp., \( R(a, b) \in M_i \)) for \( a \in N_I \, C \in \{A, \neg A\}, \ A \in N_C, \) and \( R \in N_R \).

Example 5 demonstrates practicality, readability, and omission of the unnecessary data in ABox encoding of interpretations.

**Example 5.** The ABox encoding \( M_i \) for the interpretation I for \( K \) from Example 3 is as follows:

\[
M_i = \{ \text{Cat(tom), } \neg \text{Cat(jerry), Mouse(jerry), } \neg \text{Mouse(tom),} \\
\text{Mammal(tom), Mammal(jerry), Animal(tom), Animal(jerry),} \\
\text{hasFriend(tom,jerry), } \neg \text{hasFriend(jerry,tom),} \\
\text{knows(tom,jerry), } \neg \text{knows(jerry,tom)} \}
\]

2.4 Decision Problems

If it is possible to find such an interpretation that is a model of \( K \), then we call this knowledge base \( K \) consistent. This means, that the knowledge base does not contain any contradiction. On the other hand, if there is no model for a knowledge base, then this knowledge base is inconsistent. Inconsistent knowledge bases are useless in reasoning as everything is entailed from them.

**Definition 11 (Consistency).** A DL knowledge base \( K = (T, R, A) \) is consistent (also, \( A \) is consistent w.r.t. \( T \) and \( R \)) iff it has at least one model.
Example 6 (Revisited). Since the knowledge base $\mathcal{K}$ from Example 3 has a model $I$ as proved in Example 4, the knowledge base $\mathcal{K}$ is consistent.

Besides consistency checking there are more decision problems for a DL knowledge base. The problem of deciding whether a concept is meaningful w.r.t. a knowledge base $\mathcal{K}$ is called concept satisfiability. A concept $C$ is satisfiable if and only if there is a model $I$ of $\mathcal{K}$ such that it interprets $C$ into a non-empty set.

**Definition 12 (DL Concept Satisfiability).** Given a DL knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$, a concept $C$ is satisfiable w.r.t. $\mathcal{K}$ iff there is a model $I$ of $\mathcal{K}$ s.t. $C^I \neq \emptyset$.

If a concept $C$ is not satisfiable w.r.t. a knowledge base $\mathcal{K}$, i.e. there is no model $I$ of $\mathcal{K}$ s.t. $C^I \neq \emptyset$, then $C$ is unsatisfiable w.r.t. $\mathcal{K}$.

Example 7 (Revisited). There is a model $I$ stated in Example 4 with a non-empty interpretation of Mouse (Mouse$^I \neq \emptyset$). Therefore the concept Mouse is satisfiable w.r.t. $\mathcal{K}$ stated in Example 3.

Entailment is a problem of deciding whether an axiom $\phi$ is entailed by the knowledge base $\mathcal{K}$. It holds if and only if $\phi$ is satisfied in every model of $\mathcal{K}$.

For a GCI axiom, the entailment problem is called a subsumption problem, and it is a problem deciding whether a subsumption between two concepts in a knowledge base $\mathcal{K}$ always holds, i.e. in every model of $\mathcal{K}$ the GCI axiom must be satisfied. The task of deciding, whether Cat is a subconcept of Animal in the knowledge base $\mathcal{K}$ from Example 3 is the example of a subsumption problem.

For a concept assertion, the entailment problem is called an instance checking problem, and it is a problem of deciding whether an individual is always an instance of a concept in a knowledge base. It determines if a knowledge base $\mathcal{K}$ entails that an individual $a$ belongs to a class $C$. That holds if and only if for each model $I$ of a knowledge base $\mathcal{K}$ the interpretation of $a$ belongs to the interpretation of $C$. For instance, for Example 3 we can ask whether tom is an instance of Animal.

Analogously, for role assertion entailment, two individuals $a$ and $b$ are always related by $R$ w.r.t. a knowledge base $\mathcal{K}$ means that this relation follows from $\mathcal{K}$, i.e. the interpreted pair belongs to the interpreted set of $R$ in every model $I$ of $\mathcal{K}$. Similarly as in the instance checking, for the knowledge base $\mathcal{K}$ from Example 3 it can be determined, whether $\mathcal{K}$ entails that knows(tom, jerry).

**Definition 13 (DL Entailment).** Let $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ be a DL knowledge base. Given two concepts $C, D$, a role $R$, and individuals $a, b$, $\phi$ is entailed by $\mathcal{K}$ ($\mathcal{K} \models \phi$) according to its form:
\( \phi = C \subseteq D \quad \text{iff} \quad C^3 \subseteq D^3 \) in every model \( I \) of \( \mathcal{K} \)

(\( C \) is subsumed by \( D \) w.r.t. \( \mathcal{K} \))

\( \phi = C(a) \quad \text{iff} \quad a^3 \in C^3 \) in every model \( I \) of \( \mathcal{K} \)

(\( a \) is an instance of \( C \) w.r.t. \( \mathcal{K} \))

\( \phi = R(a, b) \quad \text{iff} \quad (a^3, b^3) \in R^3 \) in every model \( I \) of \( \mathcal{K} \)

The following lemmas are dealing with the reductions between decision problems. Particularly concept satisfiability, concept subsumption and instance checking are all reducible to the consistency checking.

**Lemma 1.** Given a knowledge base \( \mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A}) \), and some concept \( C \): \( C \) is satisfiable w.r.t. \( \mathcal{K} \) iff \( \mathcal{K}' = (\mathcal{T}, \mathcal{R}, \mathcal{A} \cup \{C(a)\}) \) is consistent, for some new individual \( a \) not appearing in \( \mathcal{K} \).

**Lemma 2.** Given a knowledge base \( \mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A}) \) and concepts \( C, D \): \( \mathcal{K} \vdash C \subseteq D \) iff \( C \cap \neg D \) is unsatisfiable w.r.t. \( \mathcal{K} \).

**Lemma 3.** Given a knowledge base \( \mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A}) \), an individual \( a \), and a concept \( C \): \( \mathcal{K} \models C(a) \) iff \( \mathcal{K}' = (\mathcal{T}, \mathcal{R}, \mathcal{A} \cup \{\neg C(a)\}) \) is inconsistent.

**Lemma 4.** Given a knowledge base \( \mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A}) \), the individuals \( a, b \), and a role \( R \): \( \mathcal{K} \models R(a, b) \) iff \( \mathcal{K}' = (\mathcal{T}, \mathcal{R}, \mathcal{A} \cup \{\neg R(a, b)\}) \) is inconsistent.

**Lemma 5.** Given a knowledge base \( \mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A}) \), the individuals \( a, b \), and a role \( R \): \( \mathcal{K} \models \neg R(a, b) \) iff \( \mathcal{K}' = (\mathcal{T}, \mathcal{R}, \mathcal{A} \cup \{R(a, b)\}) \) is inconsistent.

These reductions allow us to easily solve all four problems through an algorithm for consistency checking. The proofs can be found in the Description Logics Handbook (Baader et al. 2003).

### 2.5 Tableau Algorithm for ALCHO

In DL research, the tableau algorithm is the most common reasoning algorithm (Baader et al. 2003, Horrocks et al. 1999, 2000, Hladik & Model 2004, Horrocks & Sattler 2007, Horrocks et al. 2006). With the wide use of this reasoning method, focus on the optimization techniques (Tsarkov et al. 2007) and implementation is natural in the DL research area. Currently, there is a number of highly effective DL reasoners, such as FaCT (Horrocks 1998a, b), Pellet (Sirin et al. 2007), Hermit (Shearer et al. 2008), Konclude (Steigmüller et al. 2014), etc.

Therefore we will focus on the tableau algorithm for DL. Firstly each complex concept needa to be transformed into a normalized form. Particularly, a concept is transformed recursively into the negation normal form in which the symbol of concept complement stands only in front of an atomic concept.
Definition 14 (Negation normal form). A concept $C$ is in negation normal form (NNF) iff the complement constructor $\neg$ only occurs in front of atomic concept symbols in $C$.

It is always possible to find a concept in negation normal form for a concept $C$. For the complete proof, see Baader et al. (2003).

Lemma 6. For every concept $C$ there exists $C'$ in NNF such that $C \equiv C'$.

From now on, $\text{nnf}(C)$ represents a concept $C'$ equivalent with $C$ that is in negation normal form. To transform a concept to NNF we use de Morgan's rules and the usual rules for quantifies, as shown in Table 2.

<table>
<thead>
<tr>
<th>$C$</th>
<th>$\text{nnf}(C)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\neg(E \sqcap F)$</td>
<td>$\neg E \sqcup \neg F$</td>
</tr>
<tr>
<td>$\neg(E \sqcup F)$</td>
<td>$\neg E \sqcap \neg F$</td>
</tr>
<tr>
<td>$\neg \exists R. E$</td>
<td>$\forall R. \neg E$</td>
</tr>
<tr>
<td>$\neg \forall R. E$</td>
<td>$\exists R. \neg E$</td>
</tr>
</tbody>
</table>

Table 2: Transformation to NNF

To decide whether a knowledge base $\mathcal{K} = (T, R, A)$ is consistent or not, the tableau algorithm tries to find a model through constructing a consistent ABox. Firstly an ABox $A_0$ needs to be initialized according to the ABox $A$. If $A$ is empty, an assertion $\top (a)$, where $a$ is a new individual, is added to $A_0$. Otherwise all assertions from $A$ are copied to $A_0$. Consequently, the tableau rules listed in Table 3 are executed. In the end, when all rules are applied, $\mathcal{K}$ is consistent if and only if the final ABox is consistent. An ABox is consistent if and only if it does not contain a contradiction, in DL called clash.

Definition 15 (Clash). There is a clash in ABox $A$ iff for some individual $a$ and for some concept $C$: $A$ contains both $C(a)$ and $\neg C(a)$.

The tableau algorithm is presented in pseudocode in Algorithm 1. It uses tableau rules from Table 3.

In the case of o-rule, merging of two individuals is needed. This situation occurs because of the necessity of individual’s uniqueness. That is, because of the semantics of nominals ($\{a\}_I = \{a\}$), any two individuals having the same nominal assigned actually represent one individual. Therefore the tableau algorithm calls the procedure stated in Algorithm 2, that implements merging of two individuals $a$ and $b$ in an ABox $A_1$. Each class $a$ belongs to is reassigned to $b$, and all
<table>
<thead>
<tr>
<th>Rule</th>
<th>Condition</th>
<th>Action</th>
</tr>
</thead>
</table>
| $\cap$-rule | 1. $(C_1 \cap C_2)(a) \in A_i$
2. $C_1(a) \not\in A_i$ or $C_2(a) \not\in A_i$ | $A_i = A_i \cup \{C_1(a), C_2(a)\}$ |
| $\cup$-rule | 1. $(C_1 \cup C_2)(a) \in A_i$
2. $C_1(a) \not\in A_i$ and $C_2(a) \not\in A_i$ | $A_i = A_i \cup \{C_1(a)\}$
$A_{i+1} = A_i \cup \{C_2(a)\}$
$S = S \cup A_{i+1}$ |
| $\exists$-rule | 1. $(\exists R.C)(a) \in A_i$
2. there is no individual $c$ s.t.
$C(c) \not\in A_i$ and $R(a,c) \not\in A_i$ | $A_i = A_i \cup \{C(b)\}$
where $b$ is an individual name not occurring in $A_i$ |
| $\forall$-rule | 1. $(\forall R.C)(a) \in A_i$
2. $S(a,b) \in A_i$
3. $S \sqsubseteq R$
4. $C(b) \not\in A_i$ | $A_i = A_i \cup \{C(b)\}$ |
| $o$-rule | 1. $(o)(a) \in A_i$ and $(o)(b) \in A_i$
2. $a, b, o \in N_I$
3. and $a \neq b$ | Merge $(a, b)$ in $A_i$ |
| $\forall$-rule | 1. $C_1 \sqsubseteq A \in T$
2. $a$ is an individual occurring in $A_i$
3. $\text{nnf}(\lnot C_1 \cup C_2)(a) \not\in A_i$ | $A_i = A_i \cup \{\text{nnf}(\lnot C_1 \cup C_2)(a)\}$ |

Table 3: Transformation rules of the tableau algorithm

The occurrences of $a$ in role assertions are replaced with $b$. These processes result into no occurrences of the individual $a$ in the ABox $A_i$.

Example 8 illustrates the run of the tableau algorithm.

**Example 8.** Assume the following knowledge base $\mathcal{K}$:

$$\text{Cat} \sqsubseteq \exists \text{hunts.} \text{Mouse} \sqcap \forall \text{hunts.} \lnot \text{Cat}$$

(2)

$$\text{Mouse} \sqsubseteq \exists \text{hidesFrom.} \text{Cat}$$

(3)

$$\text{Cat} \sqsubseteq \exists \text{afraidOf.} \{\text{spike}\}$$

(4)

$$\text{Cat(tom)}$$

(5)

$$\text{Mouse(jerry)}$$

(6)

Using the tableau algorithm, we decide whether $\mathcal{K}$ is consistent or not through the following steps:
### Algorithm 1 Tableau algorithm($\mathcal{K}$)

Require: a knowledge base $\mathcal{K} = (T, R, A)$

Ensure: boolean value corresponding to the consistency of $\mathcal{K}$

1. $A_0 \leftarrow \{\}$
2. $S \leftarrow \{A_0\}$  \(\triangleright\) $S$ is the set of all generated ABoxes
3. if $\Lambda = \{\}$ then
4. $A_0 \leftarrow \{T(s_0)\}$, where $s_0$ is a new individual
5. else
6. for all individuals $a$ from $\Lambda$ do
7. $A_0 \leftarrow A_0 \cup \{(a)(a)\}$
8. end for
9. end if
10. while there is a rule $r$ from Table 3 applicable on $A_i \in S$ do
11. apply $r$ on $A_i$  \(\triangleright\) in case of $\sqcup$-rule: $S \leftarrow S \cup A_j$
12. end while
13. if there is a consistent ABox $A_i \in S$ then
14. return true
15. else
16. return false
17. end if

### Algorithm 2 Merge($A_i, a, b$)

Require: an ABox $A_i$, individuals $a$, $b$

1. for all $C(a) \in A_i$ do
2. $A_i \leftarrow (A_i \setminus \{C(a)\}) \cup \{C(b)\}$
3. end for
4. for all $R(a, c) \in A_i$ do
5. $A_i \leftarrow (A_i \setminus \{R(a, c)\}) \cup \{R(b, c)\}$
6. end for
7. for all $R(c, a) \in A_i$ do
8. $A_i \leftarrow (A_i \setminus \{R(c, a)\}) \cup \{R(c, b)\}$
9. end for
1. $A_0$ is initialized by copying all the ABox assertion from $K$: $A_0 = \{\text{Cat}(\text{tom}), \text{Mouse}(\text{jerry})\}$.

2. T-rule is applied on the individual tom in $A_0$: $A_0 = A_0 \cup \{\neg \text{Cat} \sqcup (\exists \text{hunts} \cdot \text{Mouse} \sqcap \forall \text{hunts} \cdot \neg \text{Cat})(\text{tom})\}$

3. Analogously, T-rule is applied three times on the individual jerry and another two times on the individual tom in $A_0$:

$$A_0 = A_0 \cup \{\neg \text{Cat} \sqcup (\exists \text{hunts} \cdot \text{Mouse} \sqcap \forall \text{hunts} \cdot \neg \text{Cat})(\text{jerry})\},$$
$$\neg \text{Mouse} \sqcup \exists \text{hidesFrom} \cdot \text{Cat}(\text{tom}),$$
$$\neg \text{Mouse} \sqcup \exists \text{hidesFrom} \cdot \text{Cat}(\text{jerry}),$$
$$\neg \text{Cat} \sqcup \exists \text{afraidOf}(\{\text{spike}\})(\text{tom}),$$
$$\neg \text{Cat} \sqcup \exists \text{afraidOf}(\{\text{spike}\})(\text{jerry})\}$$

4. By applying $\sqcup$-rule on $A_0$, $A_1$ is created and both $A_0$ and $A_1$ extended:

$$A_0 = A_0 \cup \{\neg \text{Cat}(\text{tom})\}$$
$$A_1 = A_0 \cup \{\exists \text{hunts} \cdot \text{Mouse} \sqcap \forall \text{hunts} \cdot \neg \text{Cat}(\text{tom})\}$$

5. As $A_0$ is not consistent any more, we will omit it from the next computations.

6. Next, $\sqcup$-rule is applied on all assertions added to $A_0$ in step 3. Similarly as in step 4, each application of $\sqcup$-rule doubles the number of ABoxes. As there are five assertions in form of a complex concept assertion containing $\sqcup$, there will be $2^5$ ABoxes. First of all, we omit all ABoxes $A_i$ s.t. $\neg \text{Mouse}(\text{jerry}) \in A_i$ and s.t. $\neg \text{Cat}(\text{tom}) \in A_i$, as these ABoxes contain a clash. There are still 8 consistent ABoxes. For simplicity, we will choose only one perspective of them, as the goal of the tableau algorithm is to find at least one consistent ABox. Let assume that it is $A_1$:

$$A_1 = A_1 \cup \{\neg \text{Cat}(\text{jerry}), \neg \text{Mouse}(\text{tom}), \exists \text{hidesFrom} \cdot \text{Cat}(\text{jerry}),$$
$$\exists \text{afraidOf}(\{\text{spike}\})(\text{tom})\}$$

7. As $(\exists \text{hunts} \cdot \text{Mouse} \sqcap \forall \text{hunts} \cdot \neg \text{Cat}(\text{jerry})) \in A_1$, $\sqcap$-rule is applied:

$$A_1 = A_1 \cup \{\exists \text{hunts} \cdot \text{Mouse}(\text{tom}), \forall \text{hunts} \cdot \neg \text{Cat}(\text{tom})\}$$
8. $\exists$-rule is applied three times:

\[
A_1 = A_1 \cup \{\text{Cat}(c), \text{hidesFrom}(\text{jerry}, c), \\
\{\text{spike}\}(s), \text{afraidOf}(\text{tom}, s), \\
\text{Mouse}(m), \text{hunts}(\text{tom, m})\}
\]

9. Let us focus now on the assertion $\text{Mouse}(m)$. We will again apply $T$-rule because of this assertion and axiom (3):

\[
A_1 = A_1 \cup \{\neg \text{Mouse} \cup \exists \text{hidesFrom}. \text{Cat}(m)\}
\]

Afterwards, $\sqcup$-rule is applied and only one consistent ABox is gained:

\[
A_1 = A_1 \cup \{\exists \text{hidesFrom}. \text{Cat}(m)\}
\]

Then $\exists$-rule is applied:

\[
A_1 = A_1 \cup \{\text{Cat}(c'), \text{hidesFrom}(m, c')\}
\]

10. Next, $T$-rule (axiom (2)), $\cup$-rule, and $\cap$-rule are applied, because of $\text{Cat}(c')$ (again, we ignore an ABox containing the assertion $\neg \text{Cat}(c')$, as this leads to a clash):

\[
A_1 = A_1 \cup \{\neg \text{Cat} \cup (\exists \text{hunts}. \text{Mouse} \cap \forall \text{hunts}. \neg \text{Cat})(c'), \\
(\exists \text{hunts}. \text{Mouse} \cap \forall \text{hunts}. \neg \text{Cat})(c'), \\
\exists \text{hunts}. \text{Mouse}(c'), \forall \text{hunts}. \neg \text{Cat}(c')\}
\]

11. The algorithm would continue with producing an infinite chain, as for each cat there is a mouse that the cat hunts and for each mouse there is a cat that the mouse is afraid of.

The tableau algorithm can be potentially executed infinitely, as Example 8 shows. Infinity can be caused only by the $\exists$-rule since this rule could produce a chain of descendants infinitely. Such infinite ABox created by the tableau algorithm given in Algorithm 1 would still be a model, but the algorithm would never stop. To assure termination of the algorithm and to represent the model in a finite form, we define blocking.

To avoid producing an infinite chain of related individuals (so called descendants, or in other direction ancestors), certain individuals are blocked and tableau rules can no longer be applied on them. An individual $b$ is blocked when it is a descendant of an individual $a$ while $b$ belongs to a subset of concepts of the set of concepts for $a$.

**Definition 16** (Ancestor, Descendant). An individual $a$ is an ancestor of an individual $b$ in an ABox $A_i$ iff $R_1(a, a_1), \ldots R_n(a_n, b) \in A_i$ for any
roles $\mathcal{R}_1, \ldots, \mathcal{R}_n \in \mathcal{N}_\mathcal{R}$ and any individuals $a, a_1, \ldots, a_n, b$. The individual $b$ is a descendant of the individual $a$.

**Definition 17** (Blocking). An individual $b$ is blocked by an individual $a$ in an ABox $\mathcal{A}$ iff:

1. $\{ C \mid C(b) \in \mathcal{A} \} \subseteq \{ D \mid D(a) \in \mathcal{A} \}$,
2. $a$ is ancestor of $b$,
3. $\{i\}(b) \notin \mathcal{A}$ and $\{i\}(c) \notin \mathcal{A}$ for any individuals $i, c$, where $c$ is an ancestor of $b$ and descendant of $a$.

Consequently, the tableau algorithm and the tableau rules need an update in such a way that blocking is applied. The tableau algorithm with blocking is stated in Algorithm 3 and the updated rules are listed in Table 4. The only change compared to Algorithm 1 is in line 11, where the table with updated rules is used.

**Algorithm 3** Tableau algorithm($\mathcal{K}$)

```latex
\textbf{Require:} a knowledge base $\mathcal{K} = (T, \mathcal{R}, \mathcal{A})$

\textbf{Ensure:} boolean value corresponding to the consistency of $\mathcal{K}$

1. $A_0 \leftarrow \{\}$
2. $S \leftarrow \{A_0\}$  \quad \triangleright \text{S is the set of all generated ABoxes}
3. if $\mathcal{A} = \{\}$ then
4. \hspace{1cm} $A_0 \leftarrow \{\top(s_0)\}$, where $s_0$ is a new individual
5. else
6. \hspace{1cm} $A_0 \leftarrow \mathcal{A}$
7. \hspace{1cm} for all individuals $a$ from $\mathcal{A}$ do
8. \hspace{2cm} $A_0 \leftarrow A_0 \cup \{\{a\}(a)\}$
9. \hspace{1cm} end for
10. end if
11. while there is a rule $r$ from Table 4 applicable on $A_i \in S$ do
12. \hspace{1cm} apply $r$ on $A_i$  \quad \triangleright \text{in case of \lor-rule: } S \leftarrow S \cup A_j$
13. end while
14. if there is a consistent ABox $A_i \in S$ then
15. \hspace{1cm} return true
16. else
17. \hspace{1cm} return false
18. end if
```

The tableau algorithm with blocking tries to find a consistent ABox $A_i$ containing possibly also blocked individuals. From now on, by the tableau algorithm we always mean the tableau algorithm with blocking.

The following example demonstrates blocking, by continuing the run of the tableau algorithm from Example 8.
<table>
<thead>
<tr>
<th>Rule</th>
<th>Condition</th>
<th>Action</th>
</tr>
</thead>
</table>
| ⊓-rule | 1. \(a\) is not blocked  
2. \((C_1 \cap C_2)(a) \in A_i\)  
3. \(C_1(a) \notin A_i\) or \(C_2(a) \notin A_i\) | \(A_i = A_i \cup \{C_1(a), C_2(a)\}\) |
| ⊔-rule | 1. \(a\) is not blocked  
2. \((C_1 \cup C_2)(a) \in A_i\)  
3. \(C_1(a) \notin A_i\) and \(C_2(a) \notin A_i\) | \(A_i = A_i \cup \{C_1(a)\}\)  
\(A_{i+1} = A_i \cup \{C_2(a)\}\)  
\(S = S \cup A_{i+1}\) |
| ∃-rule | 1. \(a\) is not blocked  
2. \((\exists R.C)(a) \in A_i\)  
3. there is no individual \(c\) s.t. \(C(c) \notin A_i\) and \(R(a,c) \notin A_i\) | \(A_i = A_i \cup \{C(b), R(a,b)\}\) where \(b\) is an individual name not occurring in \(A_i\) |
| ∀-rule | 1. \(a\) is not blocked  
2. \((\forall R.C)(a) \in A_i\)  
3. \(S(a,b) \in A_i\)  
4. \(S \equiv R\)  
4. \(C(b) \notin A_i\) | \(A_i = A_i \cup \{C(b)\}\) |
| o-rule | 1. \{o\}(a) \in A_i\) and \{o\}(b) \in A_i\)  
2. \(a, b, o \in N_I\)  
3. and \(a \neq b\) | Merge(a, b) in \(A_i\) |
| †-rule | 1. \(a\) is not blocked  
2. \(C_1 \subseteq C_2 \subseteq \mathcal{T}\)  
3. \(a\) is an individual occurring in \(A_i\)  
4. \(\text{nnf}(\neg C_1 \cup C_2)(a) \notin A_i\) | \(A_i = A_i \cup \{\text{nnf}(\neg C_1 \cup C_2)(a)\}\) |

Table 4: Transformation rules of the tableau algorithm with blocking

**Example 9.** Let us continue in Example 8.

1. After step 9, the condition for blocking is satisfied, as tom is an ancestor of \(c'\) and

\[
\{C \mid C(c') \in A_1\} \subseteq \{D \mid D(\text{tom}) \in A_1\}.
\]

That means, \(c'\) is blocked by the individual tom.
2. Similarly, for $\text{Cat}(c)$ $T$-rule, $\sqcup$-rule, $\sqcap$-rule, and finally $\exists$-rule are applied:

$$A_1 = A_1 \cup \{ \neg \text{Cat} \sqcup (\exists \text{hunts. Mouse} \sqcap \forall \text{hunts.} \neg \text{Cat})(c) \},$$

$$(\exists \text{hunts. Mouse} \sqcap \forall \text{hunts.} \neg \text{Cat})(c),$$

$$\exists \text{hunts. Mouse}(c), \forall \text{hunts.} \neg \text{Cat}(c),$$

$$\text{Mouse}(m'), \text{hunts}(c, m') \}$$

3. The individual $m'$ is blocked by the individual jerry, as jerry is an ancestor of $m'$ and

$$\{ C \mid C(m') \in A_1 \} \subseteq \{ D \mid D(jerry) \in A_1 \}.$$ 

4. $T$-rule is applied while possible on both the new non-blocked individuals $m$ and $c$.

$$A_1 = A_1 \cup \{ \neg \text{Cat} \sqcup (\exists \text{hunts. Mouse} \sqcap \forall \text{hunts.} \neg \text{Cat})(m),$$

$$\neg \text{Mouse} \sqcup \exists \text{hidesFrom. Cat}(c),$$

$$\neg \text{Cat} \sqcup \exists \text{afraidOf}((\text{spike}))(m),$$

$$\neg \text{Cat} \sqcup \exists \text{afraidOf}((\text{spike}))(c) \}$$

5. After applying $T$-rule, $\sqcup$-rule and $\exists$-rule are applied. Once again, let us consider only one of the consistent $A$Boxes:

$$A_1 = A_1 \cup \{ \neg \text{Cat}(m),$$

$$\neg \text{Mouse}(c),$$

$$\exists \text{afraidOf}((\text{spike}))(c)$$

$$\text{spike}(s'), \text{afraidOf}(c, s') \}$$

6. As there are two individuals that belong to the concept $\{ \text{spike} \}$, $o$-rule is applied:

$$A_1 = A_1 \setminus \{ \text{spike}(s'), \text{afraidOf}(c, s') \}$$

$$\cup \{ \text{afraidOf}(c, s) \}$$

7. Last rule that can still be applied is $\forall$-rule. There are two relevant assertions $\forall \text{hunts.} \neg \text{Cat}(\text{tom})$ and $\forall \text{hunts.} \neg \text{Cat}(c)$ (we have omitted the one for the individual $c'$ as it is blocked):

$$A_1 = A_1 \cup \{ \neg \text{Cat}(m), \neg \text{Cat}(m') \}.$$ 

Notice that $m'$ is still blocked by the individual jerry as the condition is still satisfied.
8. All tableau rules were now applied to the ABox $A_1$. $A_1$ is consistent, and so $A_1$ represents a model of the knowledge base $K$.

The tableau algorithm, which we have introduced and explained above, is correct as stated by the following theorem.

**Theorem 1** (Correctness). The tableau algorithm for checking consistency of a knowledge base always terminates and it is sound and complete. That is, if the algorithm found a consistent ABox then the knowledge base actually has a model. And vice versa, if the knowledge base has a model then the algorithm finds a consistent ABox.

To prove the soundness of the tableau algorithm, we need to prove that if the tableau algorithm finds a consistent ABox then the knowledge base has a model (i.e. the knowledge base is consistent). This can be proven by constructing a canonical model $I_{A_1}$ from the ABox $A_1$, as defined Definition 18.

**Definition 18** (Canonical interpretation). Canonical interpretation $I_A$ induced by $A$ is such interpretation that:

- the domain $\Delta^A$ of $I_A$ consists of all the individual names occurring in $A$ that are not blocked;
- for all atomic concepts $A$ we define $A^I_A = \{x \mid x$ is not blocked and $A(x) \in A\}$;
- for all roles $R$ we define $R^I_A = \{(x, y) \mid y$ is not blocked and $R(x, y) \in A\} \cup \{(x, z) \mid y$ is blocked by $z$ and $R(x, y) \in A\}$

The proof of the completeness of the tableau algorithm lies in proving that if the knowledge base has a model then the algorithm finds a consistent ABox. The termination of the tableau algorithm is assured by blocking. The proofs can be found in The Description Logic Handbook (Baader et al. 2003) and in the works of Horrocks & Sattler (2007) and Horrocks et al. (2000).

### 2.6 Other Description Logics

There are many description logics (DLs), each named according to the constructors it allows. The variety enables to use a particular DL for the specific problem. With increasing expressivity of the DL, the computational complexity increases as well. Therefore, usually for the specific practical application, the simplest possible DL is used.

There are less expressive DLs than the DL $\mathcal{ALC\Box}$ described above. One of the basic DLs is the Attributive Language $\mathcal{AL}$, that allows only atomic concept complement, concept intersection, existential and value restriction.
Definition 19 (\(\mathcal{AL}\) concepts). Concepts are recursively constructed as the smallest set of expressions of the forms:

\[
C, D := A | \neg A | C \cap D | \exists R . C | \forall R . C
\]

where \(A \in N_C\), \(R \in N_R\), \(a \in N_I\) and \(C, D\) are concepts.

By extending the DL \(\mathcal{AL}\) with the concept union (denoted by \(\cup\)), the DL \(\mathcal{ALU}\) is gained.

Definition 20 (\(\mathcal{ALU}\) concepts). Concepts are recursively constructed as the smallest set of expressions of the forms:

\[
C, D := A | \neg A | C \cap D | C \cup D | \exists R . C | \forall R . C
\]

where \(A \in N_C\), \(R \in N_R\), \(a \in N_I\) and \(C, D\) are concepts.

The DL \(\mathcal{AL}\) extended by unrestricted concept complement (i.e. a complement standing in front of any concept, not only atomic) is called \(\mathcal{ALE}\). Note that concept union can be expressed by intersection and the unrestricted complement, hence it is also contained in \(\mathcal{ALE}\) while the letter \(\cup\) is omitted from the name. \(\mathcal{ALE}\) is one of the most commonly studied DLs.

Definition 21 (\(\mathcal{ALE}\) concepts). Concepts are recursively constructed as the smallest set of expressions of the forms:

\[
C, D := A | \neg C | C \cap D | C \cup D | \exists R . C | \forall R . C
\]

where \(A \in N_C\), \(R \in N_R\), \(a \in N_I\) and \(C, D\) are concepts.

The definitions of \(\mathcal{AL}\), \(\mathcal{ALU}\), and \(\mathcal{ALE}\) knowledge bases are almost the same as for \(\mathcal{ALCHO}\) with the only difference. As there are no role hierarchies in the mentioned DLs, their knowledge base is simply a pair \((\mathcal{X}, A)\). There is also no need to explicitly define their interpretations, as they are less expressive than \(\mathcal{ALCHO}\). To gain the \(\mathcal{AL}\), \(\mathcal{ALU}\), and \(\mathcal{ALE}\) interpretation we can simply omit the interpretation of the nominal concept in Table 1 in Definition 7. More details of the DLs \(\mathcal{AL}\), \(\mathcal{ALU}\), and \(\mathcal{ALE}\) can be found in the work of Schmidt-Schauß & Smolka (1991).

The reader is now familiar with \(\mathcal{ALE}\). The name of the DL \(\mathcal{ALCHO}\), we are mainly focused on in this work, can be derived. \(\mathcal{ALE}\) extended with role hierarchies \(\mathcal{H}\) and nominals \(\mathcal{O}\) gives us the DL \(\mathcal{ALCHO}\).

To build DLs with higher expressivity we use constructors listed in Table 5, where also the respective symbol used in DL names is stated.

Additionally, there is a shortcut \(S\) for \(\mathcal{ALE}_{R^+}\), i.e. \(\mathcal{ALE}\) extended with role transitivity. The reason for the name \(S\) originate in a modal logic \(S\) with similar properties and restrictions. For instance, the DL
### 2.6 Other Description Logics

<table>
<thead>
<tr>
<th>Constructor name</th>
<th>Expression</th>
<th>Symbol</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concept union</td>
<td>⊔</td>
<td>⊔</td>
<td>Carnivore ⊔ Herbivore</td>
</tr>
<tr>
<td>Complex concept complement</td>
<td>¬</td>
<td>∈</td>
<td>¬Carnivore</td>
</tr>
<tr>
<td>Nominals</td>
<td>{α}</td>
<td>∈</td>
<td>∃livesIn.{SouthAfrica}</td>
</tr>
<tr>
<td>Role hierarchies</td>
<td>R₁ ⊑ R₂</td>
<td>⊑</td>
<td>hasFriend ⊑ knows</td>
</tr>
<tr>
<td>Role transitivity</td>
<td>Tra(R)</td>
<td>⊑</td>
<td>Tra(isPartOf)</td>
</tr>
<tr>
<td>Inverse roles</td>
<td>R⁻</td>
<td>⊑</td>
<td>∃hasChild⁻.Parent</td>
</tr>
<tr>
<td>Role functionality</td>
<td>Fun(R)</td>
<td>⊑</td>
<td>Fun(hasParent)</td>
</tr>
<tr>
<td>Number restrictions</td>
<td>⩽, ⩾</td>
<td>N</td>
<td>⩽ 2hasChild</td>
</tr>
<tr>
<td>Qualified number</td>
<td>⩽, ⩾</td>
<td>Q</td>
<td>⩽ 2hasChild.Human</td>
</tr>
</tbody>
</table>

Table 5: DL constructors

$SH$ uses all $ALC$ constructors plus role transitivity and role hierarchies.

#### 2.6.1 Role Transitivity

Some DLs feature a restriction on roles to be transitive. For a transitive role $R$, an axiom $\text{Tra}(R)$ is added to the RBox $R$. According to its semantics it will follow that for each triple $x, y, z \in \Delta^I$ if $x$ and $y$ are in relation $R$ and $y$ and $z$ are in relation $R$ as well, then also $x$ and $z$ are in relation $R$.

**Definition 22** (Role transitivity semantics). For an interpretation $I = (\Delta^I, \cdot^I)$ the axiom $\text{Tra}(R)$ where $R \in N_R$, is satisfied iff:

$$I \models \text{Tra}(R) \iff (x, y) \in R^I \text{ and } (y, z) \in R^I \text{ then } (x, z) \in R^I.$$  

An example of transitive roles is $\text{hasAncestor}$. When a knowledge base $\mathcal{K}$ contains an axiom $\text{Tra}(\text{hasAncestor})$ and also ABox assertions $\text{hasAncestor}(\text{charles}, \text{elisabeth})$ and $\text{hasAncestor}(\text{william}, \text{charles})$ then it implies that also $\text{william}$ has the ancestor $\text{elisabeth}$. That means, each model of $\mathcal{K}$ must also satisfy $\text{hasAncestor}(\text{william}, \text{elisabeth})$.

$ALC$ extended with role transitivity and role hierarchies is called $SH$. When allowing also nominals, the name $SHO$ is used (Horrocks et al. 1999).

#### 2.6.2 Inverse Roles

There is a way how to simply define a role inverse to a role $R \in N_R$. By notation $R^-$ we understand a set of such pairs that they are included in a relation $R$ in the other direction.
Definition 23 (Inverse roles semantics). For an interpretation $\mathcal{I} = (\Delta^I, \cdot^I)$ the interpretation of $R^-$ for any $R \in \mathcal{N}_R$ is:

$$R^- \mathcal{I} = \{(x, y) | (y, x) \in R^I \text{ and } x, y \in \Delta^I\}.$$

For example, a role hasChild describes the relation of having child. To describe the relation of having a parent it is not necessary to establish a new role, as we can simply use an inverse role hasChild$^-$. An example of DL allowing also inverse roles is $\mathcal{SHOIQ}$ (Horrocks et al. 1999, 2000).

2.6.3 Role Functionality

Another constraint is a functionality axiom, that asserts a role to be a partial function. It means, for each element at most one element in $R$-relation can be assigned. To restrict a role $R$ to be functional, i.e. to have each element in the $R$-relation at most once, an axiom $\text{Fun}(R)$ is added to an RBox.

Definition 24 (Role functionality semantics). For an interpretation $\mathcal{I} = (\Delta^I, \cdot^I)$ the axiom $\text{Fun}(R)$ where $R \in \mathcal{N}_R$, is satisfied iff:

$$\mathcal{I} \models \text{Fun}(R) \text{ if } (x, y) \in R^I \text{ and } (x, z) \in R^I \text{ then } y = z.$$

A typical example of functional role is hasMother. To restrict all individuals to be in the relation hasMother at most once, we use the axiom $\text{Fun(}\text{hasMother})$. Role functionality is included for example in the DL $\mathcal{SHOIF}$.

2.6.4 Number Restrictions

Relations in DL can be also restricted by a number. The complex concept $\leq n R$ (also called maximum cardinality restriction) determines a class for which elements must be hold that they are at most $n$ times in the relation $R$. Similarly, the complex concept $\geq n R$ (also called minimum cardinality restriction) determines a class for which elements must be hold that they are at least $n$ times in the relation $R$.

Definition 25 (Number restriction semantics). For an interpretation $\mathcal{I} = (\Delta^I, \cdot^I)$ the interpretations of $\leq n R$ and $\geq n R$ for any $R \in \mathcal{N}_R$ and any integer $n$ are:

$$\leq n R^I = \{x \in \Delta^I | \#\{y \in \Delta^I \mid (x, y) \in R^I\} \leq n\},$$

$$\geq n R^I = \{x \in \Delta^I | \#\{y \in \Delta^I \mid (x, y) \in R^I\} \geq n\}.$$
This serves to define number restricted classes such as a class of monogamists, i.e. \( \leq 1 \text{hasPartner} \), and a class of polygamists, i.e. \( \geq 2 \text{hasPartner} \). The DL \( \mathcal{SHO} \) can be extended with number restriction what leads to a DL \( \mathcal{SHOIN} \) (Horrocks & Sattler 2007).

### 2.6.5 Qualified Number Restrictions

Qualified number restrictions extend number restrictions with defining also the range of the role. That means, \( \leq n \text{R.C} \) restricts the class’ elements not only to have at most \( n \) relations R in arbitrary sense, but to have at most \( n \) relations R with the elements from the class C.

**Definition 26** (Qualified number restriction semantics). For an interpretation \( I = (\Delta^I, \cdot^I) \), the interpretations of \( \leq n \text{R.C} \) and \( \geq n \text{R.C} \) for any \( R \in \mathcal{N}_R \) and any concept C are:

\[
\leq n \text{R.C}^J = \{ x \in \Delta^I \mid |\{ y \in \Delta^J \mid (x, y) \in R^J \text{ and } y \in C^J \} \leq n \},
\]

\[
\geq n \text{R.C}^J = \{ x \in \Delta^I \mid |\{ y \in \Delta^J \mid (x, y) \in R^J \text{ and } y \in C^J \} \geq n \}.
\]

A class of those who have at most two female children is a complex concept in a form \( \leq 2 \text{hasChild.Female} \). A class of those who has at least two daughters is \( \geq 2 \text{hasChild.Female} \). Notice that \( =2 \text{hasChild.Female} \) is a syntactic sugar for \( \leq 2 \text{hasChild.Female} \cap \geq 2 \text{hasChild.Female} \) describing a class of those who has precisely two female children.

Qualified number restrictions extend number restrictions with an additional range in form of a concept. Hence, it is not necessary to list both of them in the name of the DL. Thus \( \mathcal{SHOIN} \) extended with qualified number restrictions is simply renamed to \( \mathcal{SHOIQ} \) (Horrocks & Sattler 2007).

### 2.6.6 Other Extensions

There are other constructors and axioms used for building different DLs. One of the most expressive DLs currently used is \( \mathcal{SROIQ} \) (Horrocks et al. 2006), that is \( \mathcal{SHOIQ} \) extended with self restriction, role chains, universal role, role disjointness, role reflexivity, role irreflexivity, role symmetry, and role asymmetry. These constructors are listed in Table 6 with their names, their syntax, and examples of their use. Semantics of each constructor or axiom is stated in Table 7. The DL \( \mathcal{SROIQ} \) corresponds to the Web Ontology Language OWL, more precisely to its current version OWL2.
### Table 6: Other DL constructors

<table>
<thead>
<tr>
<th>Constructor name</th>
<th>Expression</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self restriction</td>
<td>$\exists R.\text{Self}$</td>
<td>$\exists \text{like}.\text{Self}$</td>
</tr>
<tr>
<td>Role chains</td>
<td>$R_1 \cdot \ldots \cdot R_n \sqsubseteq R$</td>
<td>$\exists \text{hasSon} \cdot \exists \text{hasWife} \sqsubseteq \exists \text{hasDaughterInLaw}$</td>
</tr>
<tr>
<td>Universal role</td>
<td>$U$</td>
<td></td>
</tr>
<tr>
<td>Role disjointness</td>
<td>$\text{Dis}(R_1, R_2)$</td>
<td>$\text{Dis}(\text{hasMother}, \text{hasWife})$</td>
</tr>
<tr>
<td>Role reflexivity</td>
<td>$\text{Ref}(R)$</td>
<td>$\text{Ref}(\text{knows})$</td>
</tr>
<tr>
<td>Role irreflexivity</td>
<td>$\text{Irr}(R)$</td>
<td>$\text{Irr}(\text{gossipAbout})$</td>
</tr>
<tr>
<td>Role symmetry</td>
<td>$\text{Sym}(R)$</td>
<td>$\text{Sym}(\text{knows})$</td>
</tr>
<tr>
<td>Role asymmetry</td>
<td>$\text{Asy}(R)$</td>
<td>$\text{Asy}(\text{isTaller})$</td>
</tr>
</tbody>
</table>

### Table 7: Semantics of other DL constructors

<table>
<thead>
<tr>
<th>Expression</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\exists R.\text{Self}$</td>
<td>$\exists R.\text{Self}^I = { x \in I^J \mid (x, x) \in R^I }$</td>
</tr>
<tr>
<td>$R_1 \cdot \ldots \cdot R_n$</td>
<td>$R_1^I \cdot \ldots \cdot R_n^I = {(x_1, x_n) \mid (x_1, x_2) \in R_1^I, \ldots, (x_{n-1}, x_n) \in R_n^I }$</td>
</tr>
<tr>
<td>$U$</td>
<td>$U^I = {(x, y) \mid x, y \in I^J }$</td>
</tr>
<tr>
<td>$\text{Dis}(R_1, R_2)$</td>
<td>$I \models \text{Dis}(R_1, R_2)$ iff $R_1^I \cap R_2^I = \emptyset$</td>
</tr>
<tr>
<td>$\text{Ref}(R)$</td>
<td>$I \models \text{Ref}(R)$ iff ${(x, x) \mid x \in I^J } \subseteq R^I$</td>
</tr>
<tr>
<td>$\text{Irr}(R)$</td>
<td>$I \models \text{Irr}(R)$ iff ${(x, x) \mid x \in I^J } \cap R^I = \emptyset$</td>
</tr>
<tr>
<td>$\text{Sym}(R)$</td>
<td>$I \models \text{Sym}(R)$ iff if $(x, y) \in R^I$ then $(y, x) \in R^I$</td>
</tr>
<tr>
<td>$\text{Asy}(R)$</td>
<td>$I \models \text{Asy}(R)$ iff if $(x, y) \in R^I$ then $(y, x) \notin R^I$</td>
</tr>
</tbody>
</table>
ABDUCTION

In every-day life we meet situations when we use a reasoning in many forms. The most typical type of reasoning is deduction. Usually, we deduce from a known background theory that some fact follows. For example, let us assume that Tom is a five years old boy. Tom learns that every child has parents. When he makes a new friend Lilly, Tom deduces that Lilly has parents as well.

Another type of reasoning is induction. By this type we move from specific data to a general conclusion. For instance, Tom observes in a kindergarten that his girl mates Lilly, Jane and Penny have long hair. From this observations Tom concludes that all girls have long hair.

By abduction the observations are explained. According to previous Tom’s knowledge gained by deductive and inductive reasoning, he can now try to explain some observations. For instance, when he sees a child with long hair, he can hypothesize, that this child is a girl. Also, when he hears about somebody’s parents, he assumes, that he hears about child.

To sum up, three main types of reasoning differ in the type of premises and consequences (Peirce 1878, Van Harmelen et al. 2008). As you can see in Figure 1, the deduction derives effects from the theory and data; the induction derives a theory from the data and effects; the abduction derives data from the theory and effects.

![Peirce triangle](image)

In this chapter, we firstly formally define the abductive reasoning in general, including the notions of the abductive problem, what are the correct explanations, etc. Afterwards, we will introduce an approach for finding explanations, so called minimal hitting set algorithm (Reiter 1987). Then we will describe the particular problem of abductive
reasoning over description logics, and in the end we will review current approaches to this problem.

3.1 ABDUCTION IN GENERAL

Although reasoning with explanations (also called hypotheses) was researched by the philosophers many years ago, it was usually collapsed into one type of reasoning with induction. The American philosopher Peirce (1878) introduced abduction into modern logic and defined the problem formally.

As described in the previous example, to have an abduction problem we need to have a background theory $\Gamma$ and an observation $O$, where $O$ is a subset of some set of all possible observations $\emptyset$, i.e. $O \subseteq \emptyset$. To restrict the space for explanations, we define a set of all possible explanations (hypotheses) $\mathcal{H}$, so called abducibles. The main condition for each explanation is to entail, together with theory, the observation, i.e. $\Gamma \cup \mathcal{E} \models O$. Moreover, for an explanation to be meaningful, it must be consistent and the observation must not be entailed directly by the theory. This can be formally stated as follows (Aliseda-Llera 1997, Konolige 1990, Kakas et al. 1992, Mayer & Pirri 1993).

**Definition 27** (Abductive explanation). Let $\Gamma$ be a theory and $O \subseteq \emptyset$ be an observation. An hypothesis $\mathcal{E} \subseteq \mathcal{H}$ is an explanation of $O$ w.r.t. $\Gamma$ if and only if:

1. $\mathcal{E} \cup \Gamma$ is consistent,
2. $\Gamma \not\models O$,
3. $\Gamma \cup \mathcal{E} \models O$.

To show, how abductive reasoning works in a proper formalism, we will use an example from propositional logic adapted from Pearl (1987).

**Example 10.** Given $O = \{\text{wet\_road}, \text{wet\_grass}\}$, $\mathcal{H} = \{\text{rain, sun, irrigation, hot\_day}\}$, and a theory $\Gamma$:

- $\text{rain} \rightarrow \text{wet\_road}$
- $\text{rain} \rightarrow \text{wet\_grass}$
- $\text{sun} \leftrightarrow \neg\text{rain}$
- $\text{irrigation} \rightarrow \text{wet\_grass}$
- $\text{sun} \land \text{hot\_day} \rightarrow \text{irrigation}$
What are all possible explanations of $O = \{\text{wet} \_\text{grass}\}$? According to Definition 27, the explanations of $O$ are:

$$
\begin{align*}
\mathcal{E}_1 &= \{\text{rain}\} \\
\mathcal{E}_2 &= \{\text{irrigation}\} \\
\mathcal{E}_3 &= \{\text{sun}, \text{hot} \_\text{day}\} \\
\mathcal{E}_4 &= \{\text{rain}, \text{irrigation}\} \\
&\ldots
\end{align*}
$$

Notice that, there are infinitely many explanations, as for example for any set $\mathcal{E}'$ s.t. $\mathcal{E}_1 \subseteq \mathcal{E}'$ it holds $\Gamma \cup \mathcal{E}' \models O$.

Observe that, $\mathcal{E}_1, \mathcal{E}_2 \subseteq \mathcal{E}_4$. Also, we mentioned in the example, that there are infinitely many explanations. To limit the space of all possible explanations the desired explanations are usually the minimal ones. Thus, we define syntactic minimality.

**Definition 28 (Syntactic Minimality).** Let $\Gamma$ be a theory, $O \subseteq \mathcal{O}$ be an observation, and $\mathcal{E}, \mathcal{E}' \subseteq \mathcal{H}$ be the explanations of $O$ w.r.t. $\Gamma$. We say that $\mathcal{E}$ is (syntactically) smaller than $\mathcal{E}'$ if $\mathcal{E} \subseteq \mathcal{E}'$. We further say that $\mathcal{E}$ is syntactically minimal if there is no other explanation of $O$ w.r.t. $\Gamma$ that is smaller than $\mathcal{E}$.

Observe, that $\mathcal{E}_2 = \{\text{irrigation}\}$ and $\mathcal{E}_3 = \{\text{sun}, \text{hot} \_\text{day}\}$ are in a relation too, but in this case, in a semantic relation. Namely, $\mathcal{E}_2$ is entailed by $\mathcal{E}_3 \cup \Gamma$. In explaining, we usually do not want to hypothesize too much. In many applications, the more general an explanation is the better, as explanations are only hypotheses to explain some observed phenomenon. Thus, we define semantic minimality, to differentiate which explanation is semantically weaker, and hence semantically smaller.

**Definition 29 (Semantic Minimality).** Let $\Gamma$ be a theory, $O \subseteq \mathcal{O}$ be an observation, and $\mathcal{E}, \mathcal{E}' \subseteq \mathcal{H}$ be the explanations of $O$ w.r.t. $\Gamma$. We say that $\mathcal{E}$ is (semantically) stronger than $\mathcal{E}'$ (denoted by $\mathcal{E} \preceq \Gamma \mathcal{E}'$) iff $\Gamma \cup \mathcal{E} \models \mathcal{E}'$. We further say that $\mathcal{E}$ is semantically minimal if there is no other explanation $\mathcal{E}'$ of $O$ w.r.t. $\Gamma$ s.t. $\mathcal{E}' \preceq \Gamma \mathcal{E}$.

In this thesis, we focus on syntactic minimality, as this restriction is not only desired in most applications, but also necessary for finding a finite solution. Reiter (1987) proposed a minimal hitting set algorithm described in the next section to find all (syntactically) minimal explanations for an abduction problem.
3.2 MINIMAL HITTING SET ALGORITHM

Reiter (1987) in his work deals with computing diagnoses. He defines a system as a pair (SD, COMPONENTS), where SD is the system description in form of a set of first-order sentences and COMPONENTS are the system components in form of a finite set of constants. He also establishes a unary predicate AB, the meaning of which is abnormal. Intuitively, for every component $c_i$ that works properly it holds $\neg AB(c_i)$, and analogously for every faulty component $c_j$ it holds $AB(c_j)$.

**Example 11.** Consider the electric circuit of a simple cooker. Let us represent the cooker as a system with $\text{COMPONENTS} = \{p_1, p_2, b, s\}$ and the following system description $\text{SD}$:

\[
\begin{align*}
\text{supply}(X) \land \neg AB(X) \land \text{plate}(Y) \land \neg AB(Y) & \rightarrow \text{heats}(Y), \\
\text{plate}(X) \land \text{heats}(X) \land \text{bulb}(Y) \land \neg AB(Y) & \rightarrow \text{glows}(Y), \\
\text{plate}(p_1), & \\
\text{plate}(p_2), & \\
\text{bulb}(b), & \\
\text{supply}(s). &
\end{align*}
\]

The triple (SD, COMPONENTS, OBS) is a system (SD, COMPONENTS) with observations OBS in form of a finite set of first-order sentences.

Having the components $\{c_1, \ldots, c_n\}$ forming a system (SD, $\{c_1, \ldots, c_n\}$), then $\{\neg AB(c_1), \ldots, \neg AB(c_n)\}$ means that each component $c_i$ works properly. Assuming that OBS represents the observations of faulty behaviour of the system SD, than logically SD $\cup \{\neg AB(c_1), \ldots, \neg AB(c_n)\} \cup$ OBS is inconsistent. There are some faulty components, for that AB has to hold in order to restore consistency of the system and together with the observed phenomenon.

As Reiter correctly stated, the simplest way how to solve this problem is to assume, that all components are faulty. In such case SD $\cup \{AB(c_1), \ldots, AB(c_n)\} \cup$ OBS is consistent. Intuitively, this is not the right way how to find diagnoses, and so Reiter defined diagnosis as a minimal set of abnormal components that leads to a consistency with observations.

**Definition 30 (Diagnosis).** A diagnosis for (SD, COMPONENTS, OBS) is a minimal set $\Delta \subseteq \text{COMPONENTS}$ such that:

\[\text{SD} \cup \text{OBS} \cup \{AB(c) \mid c \in \Delta\} \cup \{\neg AB(c) \mid c \in \text{COMPONENTS} \setminus \Delta\}\]
In the following example, we illustrate the diagnoses for an observation for the system stated in Example 11.

**Example 12 (Continued).** Suppose, that we observe for the cooker, that the bulb does not glow. The observation is formally represented as follows:

\[ \text{OBS} = \{ \neg \text{glows}(b) \} \].

With this observation, \( SD \cup \text{OBS} \cup \{ \neg \text{AB}(p_1), \neg \text{AB}(p_2), \neg \text{AB}(b), \neg \text{AB}(s) \} \) is inconsistent, i.e. not all components work properly. There are three diagnoses for \( (SD, \text{COMPONENTS}, \text{OBS}) \): \{b\}, \{s\}, and \{p_1, p_2\}, i.e. the faulty component is either the bulb b, or the supply s, or both plates p_1 and p_2, as the following unions are consistent:

\[
\begin{align*}
SD \cup \text{OBS} \cup \{ \text{AB}(b) \} \cup \{ \neg \text{AB}(s), \neg \text{AB}(p_1), \neg \text{AB}(p_2) \}, \\
SD \cup \text{OBS} \cup \{ \text{AB}(s) \} \cup \{ \neg \text{AB}(b), \neg \text{AB}(p_1), \neg \text{AB}(p_2) \}, \\
SD \cup \text{OBS} \cup \{ \text{AB}(p_1), \text{AB}(p_2) \} \cup \{ \neg \text{AB}(s), \neg \text{AB}(b) \}.
\end{align*}
\]

### 3.2.1 Conflict Sets

In order to find all diagnoses according to Definition 30, Reiter established the notion of a conflict set. By this notion he describes the diagnosis as a minimal set of components such that when removing it from the set of all components, this set is not a conflict set. A conflict set is such a set of the components that their proper functioning together with the observation leads to inconsistency. That is, for example having two components that cause the faulty behavior of the system independently, both must be obtained in each conflict set, as with assuming at least one of them to work not properly, the behavior would be explained and the inconsistency lost.

**Definition 31 (Conflict set).** A conflict set for \( (SD, \text{COMPONENTS}, \text{OBS}) \) is a set \( \{c_1, \ldots, c_k\} \subseteq \text{COMPONENTS} \) such that:

\[ SD \cup \text{OBS} \cup \{ \neg \text{AB}(c_1), \ldots, \neg \text{AB}(c_k) \} \]

is inconsistent.

Note that, for a collection \( C \) containing all conflict sets for \( (SD, \text{COMPONENTS}, \text{OBS}) \), all the sets \( S \in C \) s.t. there is a set \( S' \in C \) and \( S' \subseteq S \) can be omitted, that leads us to a collection of all minimal conflict sets. The reason to omit all non-minimal conflict sets is implied by the purpose of the conflict sets – each conflict set includes at least one component from each faulty behavior of the system. That
is, there is no need to add the other components to the conflict set, as they would say nothing relevant for the conflict.

**Example 13** (Continued). Consider the previously described cooker and the observation \((SD, \{p_1, p_2, b, s\}, \{\neg \text{gloows}(b)\})\). The following collection \(\{(b, s, p_1), \{b, s, p_2\}\}\) is the collection of all minimal conflict sets for \((SD, \{p_1, p_2, b, s\}, \{\neg \text{gloows}(b)\})\), as

\[
\begin{align*}
SD \cup \text{OBS} \cup \{\neg \text{AB}(b), \neg \text{AB}(s), \neg \text{AB}(p_1)\}, \\
SD \cup \text{OBS} \cup \{\neg \text{AB}(b), \neg \text{AB}(s), \neg \text{AB}(p_2)\}
\end{align*}
\]

are inconsistent.

Reiter shows that finding the diagnoses actually amounts to finding hitting sets (Karp 1972) for the collection of conflict sets. A hitting set for a collection of sets \(C\) is such a set, that contains at least one element of each \(S \in C\). A minimal hitting set is then such hitting set, that no its subset is a hitting set.

**Definition 32** (Hitting set). Suppose \(C\) is a collection of sets. A hitting set for \(C\) is a set \(E \subseteq \bigcup_{S \in C} S\) such that \(E \cap S \neq \emptyset\) for each \(S \in C\). A hitting set for \(C\) is minimal iff no proper subset of it is a hitting set for \(C\).

Finally, Reiter results in his work into the theorem that \(\Delta \subseteq \text{COMPONENTS}\) is a diagnosis for \((\text{SD}, \text{COMPONENTS}, \text{OBS})\) iff \(\Delta\) is a minimal hitting set for the collection of minimal conflict sets for \((\text{SD}, \text{COMPONENTS}, \text{OBS})\).

**Theorem 2.** \(\Delta \subseteq \text{COMPONENTS}\) is a diagnosis for \((\text{SD}, \text{COMPONENTS}, \text{OBS})\) iff \(\Delta\) is a minimal hitting set for the collection of minimal conflict sets for \((\text{SD}, \text{COMPONENTS}, \text{OBS})\).

**Example 14** (Continued). Consider the cooker and the conflict sets \(C = \{(b, s, p_1), \{b, s, p_2\}\}\) described above. The collection of the minimal hitting sets for \(C\) is:

\[D = \{(b), \{s\}, \{p_1, p_2\}\}.
\]

The collection \(D\) corresponds to the set of all minimal diagnoses for \((\text{SD}, \{p_1, p_2, b, s\}, \{\neg \text{gloows}(b)\})\).

For all the details and complete proofs, see Reiter (1987).

### 3.2.2 Computing Hitting Sets

Reiter (1987) also proposed an efficient and optimized approach for computing hitting sets. With this approach, it is possible to find all minimal diagnoses for an abduction problem.
Actually, Reiter shows how to compute all minimal hitting sets for any collections of sets, but it can be used in a diagnostic setting, where the collection is the collection of conflict sets for (SD, COMPONENTS, OBS). He proposes to construct a tree with nodes labelled by elements of the collection $M$ (i.e., sets) and edges starting from any node $n$ labeled by elements from the label of $n$. The tree is constructed in such a way that hitting sets of $M$ are found on the paths from the root to the leaves.

**Definition 33 (HS-tree).** Suppose $M$ is a collection of sets. An edge-labelled and node-labelled tree $T = (V,E,L)$ is an HS-tree for $M$ iff it is a smallest tree with the following properties:

1. Its root $r$ is labelled by ✓, i.e. $L(r) = ✓$, if $M$ is empty. Otherwise, $L(r) = M$ for some $M \in M$.

2. If $n$ is a node of $T$, i.e. $n \in V$, define $E(n)$ to be the set of edge labels on the path in $T$ from the root node to $n$.

3. If $L(n) = ✓$, it has no successor nodes in $T$.

4. If $L(n) = M_n$ for some $M_n \in M$, then for each $\sigma \in M_n$, $n$ has a successor node $n_\sigma$ joined to $n$ by an edge labelled by $\sigma$, i.e. $L(n, n_\sigma) = \sigma$. The label of $n_\sigma$ is $L(n_\sigma) = M_{n_\sigma}$ for some $M_{n_\sigma} \in M$ such that $M_{n_\sigma} \cap H(n_\sigma) = \emptyset$ if such a set $M_{n_\sigma}$ exists. Otherwise, $L(n_\sigma) = ✓$.

A HS-tree $T$ constructed for $M$ w.r.t. Definition 33 contains all the minimal hitting sets of $M$, and possibly also some non-minimal. More precisely, a set of edge labels on the path from the root to a leaf in $T$ is a hitting set for $M$.

As we already mentioned we are specifically interested in minimal hitting sets (and so in minimal diagnoses, or minimal explanations). To compute only the minimal hitting sets, Reiter proposed to construct HS-tree by breadth-first search and to prune it as described below.

Pruned HS-tree for $M$ is constructed as follows:

1. Generate the HS-tree breadth-first, generating nodes at any fixed level in the tree in left-to-right order.

2. Reusing node labels: If node $n$ is labelled by the set $M \in M$, and if $n'$ is a node such that $E(n') \cap M = \emptyset$, label $n'$ by $M$. (We indicate that the label of $n'$ is a reused label by underlining it in the tree.) Such a node $n'$ requires no access to $M$.

3. Tree pruning:
   a) If node $n$ is labelled by ✓ and node $n'$ is such that $E(n) \subseteq H(n')$, close $n'$.
b) If node \( n \) has been generated and node \( n' \) is such that \( \mathcal{E}(n') = H(n) \), then close \( n' \). (We indicate a closed node in the tree by marking it with \( \times \).)

c) If nodes \( n \) and \( n' \) have been respectively labelled by sets \( M \) and \( M' \in \mathcal{M} \), and if \( M' \) is a proper subset of \( M \), then for each \( \alpha \in M \setminus M' \) mark as redundant the edge from node \( n \) labelled by \( \alpha \). A redundant edge, together with the subtree beneath it, may be removed from the HS-tree while preserving the property that the resulting pruned HS-tree will yield all minimal hitting sets for \( \mathcal{M} \).

In the end, thanks to the previous definitions, the construction of a pruned HS-tree ensures that each path from the root to the leaf labelled by \( \checkmark \) corresponds to a minimal hitting set.

**Theorem 3.** Let \( \mathcal{M} \) be a collection of sets, and \( T \) a pruned HS-tree for \( \mathcal{M} \), as previously described. Then \( \{ H(n) \mid n \text{ is a node of } T \text{ labelled by } \checkmark \} \) is the collection of minimal hitting sets for \( \mathcal{M} \).

For more details and the complete proofs, see Reiter (1987).

### 3.3 ABDUCTION IN DESCRIPTION LOGICS

In previous sections, we have defined abduction as a general problem for any formalism, and dealt with the examples in propositional and first-order logic. However, this thesis is focused on abduction in specific formalism – description logics. Let us illustrate a DL abduction problem in the following example.

**Example 15.** Having the following knowledge base, explain the observation about Jack being nervous: \( O = \text{Nervous(jack)} \).

\[
\begin{align*}
\text{Hungry} & \sqsubseteq \text{Nervous} \\
\text{Overworked} & \sqsubseteq \text{Nervous} \\
\text{Ill} & \sqsubseteq \text{Nervous}
\end{align*}
\]

There are three obvious explanations for Example 15: that Jack is hungry, or he is overworked, or he is ill, i.e. \( \mathcal{E}_1 = \{ \text{Hungry(jack)} \} \), \( \mathcal{E}_2 = \{ \text{Overworked(jack)} \} \), \( \mathcal{E}_3 = \{ \text{Ill(jack)} \} \). In fact, each combination of these explanations is an explanation as well.

According to Definition 27, \( \mathcal{E} \) is an explanation if and only if \( \mathcal{E} \) together with the knowledge base \( \mathcal{K} \) implies the observation \( O \), i.e. \( \mathcal{K} \cup \mathcal{E} \models O \). Different representation formalisms have different properties, and therefore sometimes specific abduction problems are studied. Elsenbroich et al. (2006) defined four main types of abduction in DL.
Firstly, they specified concept abduction, dealing with an observation in a form of a DL satisfiable concept, trying to find its subconcept, i.e. a solution for a concept abduction is a concept s.t. it is a subconcept of an observed concept.

**Definition 34** (DL concept abduction). Let $\mathcal{K}$ be a DL knowledge base, $C$ a DL concept satisfiable w.r.t. $\mathcal{K}$. A solution of a concept abduction problem $(\mathcal{K}, C)$ is any concept $D$ s.t. $\mathcal{K} \models D \sqsubseteq C$.

Concept abduction is a rather specific problem outside of our scope, it was studied by Colucci et al. (2003). We consider the other three types of DL abduction defined by Elsenbroich et al. (2006) as the main types of DL abduction. Actually, these types differentiate only in the form of the observation and the form of the explanation. The distinction between the intensional and the extensional knowledge, i.e., between the TBox and the ABox, is natural because we treat these two parts of knowledge separately.

**Definition 35** (Abduction problems in DL). An abduction problem is a pair $P = (\mathcal{K}, O)$ such that $\mathcal{K}$ is a knowledge base in DL, and $O$ a set of TBox axioms and ABox assertions. A solution of $P$ is any finite set $E$ of TBox axioms and ABox assertions such that $\mathcal{K} \cup E$ is consistent and $\mathcal{K} \cup E \models O$. In addition, $P$ is called:

- **TBox abduction problem**: if $E$ and $O$ are sets of TBox axioms.
- **ABox abduction problem**: if $E$ and $O$ are sets of ABox assertions.
- **Knowledge base abduction problem**: the general problem, i.e., if there are no restrictions on $E$ and $O$.

Originally, Elsenbroich et al. (2006) defined TBox abduction as an abduction problem with an observation in a form of a TBox axiom, and ABox abduction as an abduction problem with an observation in a form of a single ABox assertion. We generalize both problems to have an observation in form of a set of TBox axioms, or ABox assertions respectively.

In this work we are focused on ABox abduction, and therefore from now on the term abduction refers to ABox abduction. Also, as in case of ABox abduction $O$ represents a set of ABox assertions, by $O_1 \in O$ we will always mean an assertion from the set $O$.

To compute meaningful solutions, we need to restrict the explanations in more ways. Some of these requirements are already part of the general definition (Definition 27). Elsenbroich et al. (2006) require an explanation to be also relevant.

**Definition 36** (Explanation for ABox Abduction Problem). Given an ABox abduction problem $P = (\mathcal{K}, O)$ and its solution $E$ we say that:
1. $\mathcal{E}$ is consistent iff $\mathcal{E} \cup \mathcal{K} \not\vdash \bot$;

2. $\mathcal{E}$ is relevant iff $\mathcal{E} \not\vdash O_i$ for each $O_i \in O$;

3. $\mathcal{E}$ is explanatory iff $\mathcal{K} \not\vdash \mathcal{O}$. does not entail $\mathcal{O}$.

Consistency is very intuitive, it does not make sense to have inconsistent explanation, as from inconsistency everything is derived. Consistent explanation contains only consistent data and is consistent w.r.t. the knowledge base, as well. By relevance we mean that an explanation does not imply any of the observations without the need of the knowledge base.

The relevance must be satisfied for each observation separately, as in some cases, the condition $\mathcal{E} \not\vdash \mathcal{O}$ leads to undesired explanations, as it does not assure $\mathcal{E} \not\vdash O_1$ for each $O_1 \in O$. For example, having a knowledge base \{Professor $\sqsubseteq$ Teacher, Teacher $\sqsubseteq$ Human\} and the observation $O = \{O_1 = \text{Human}(\text{fred}), O_2 = \text{Teacher}(\text{fred})\}$, with checking the relevance only according the whole set $\mathcal{O}$, there would be two explanations $\mathcal{E}_1 = \{\text{Teacher}(\text{fred})\}$ and $\mathcal{E}_2 = \{\text{Professor}(\text{fred})\}$. Notice that, though $\mathcal{E}_1 \not\vdash \mathcal{O}$, $\mathcal{E}_1$ is a subset of $\mathcal{O}$ ($\mathcal{E}_1 \subset \mathcal{O}$), more precisely $\mathcal{E}_1 = O_1$, and consequently $\mathcal{E}_1 \models O_1$. As we consider this as undesired behavior of explanations, we define, that each explanation must be relevant to each observation from the set of observations. As Elsenbroich et al. (2006) dealt with observation in a form of a single ABox assertion, and not with a set of ABox assertions, they did not face this issue. They simply defined $\mathcal{E}$ as relevant if $\mathcal{E} \not\vdash \mathcal{O}$.

As we are looking for a solution of the abduction problem $\mathcal{P} = (\mathcal{K}, \mathcal{O})$, a solution $\mathcal{E}$ explains $\mathcal{O}$ together with $\mathcal{K}$. Similarly, the knowledge base should not directly entail the observation. In such a case, there would be nothing to explain. Hence an explanation must be explanatory.

Other usual requirements are syntactic (Definition 28) and semantic minimality (Definition 29). Syntactically minimal explanations are usually preferred. Semantically minimal are harder to find, and not always the desired ones, that is supported by the work of Petasis et al. (2013). In our work, we focus on syntactic minimality.

From now on, by an explanation of $\mathcal{P}$ we always mean a consistent, relevant, explanatory and syntactically minimal explanation of $\mathcal{P}$.

The approaches to DL abduction use either direct approach or approaches based on a translation. Both have some advantages and disadvantages. We will review the most important recent works on ABox abduction.
3.3.1 Translation-based Approaches

Klarman et al. (2011) proposes an approach for ABox abduction based on two reductions – a reduction to modal logic and a reduction to first-order logic. For both reductions standard techniques for translation are used.

For reduced modal theory, Klarman et al. use regular connection tableaux and for reduced first-order theory resolution with set-of-support. The main contribution of this work lies in finding the explanations extending the standard inference methods. Moreover, not only atomic explanations (in form of a literal), but also complex explanations are found.

As the space of explanations is not restricted, the proposal leads to non-termination. Klarman et al. ensured termination with some restrictions (e.g. local minimality).

The proposed solution works over the DL ALC with the class of observations and explanations in ALE. Klarman et al. proved soundness and completeness, whilst provide no implementation, thus no empirical evaluation exists.

Du et al. (2011, 2012) deal with ABox abduction through a reduction to logic programming. The main advantage of this approach is that state-of-the-art Prolog abduction solver that can be used to find the explanations, and the solution is then translated back to DL. Expressivity of the input DL is SHIQ. Du et al. proved soundness, but the completeness for SHIQ is not guaranteed. They described also the following restrictions for completeness – only simple roles are allowed in role assertions and the ontology is in Horn-SHIQg, which is roughly the Horn fragment of SHIQ (i.e. one that can be translated to a first-order logic theory consisting of Horn clauses, Horn 1951) without maximum cardinality restrictions or equality.

With respect to termination, they deal with the similar problem as the proposal of Klarman et al. – they assure that their algorithm terminates in case of explanations in form of assertions with literal concepts or atomic roles.

Du et al. provide experimental evaluation on thirteen ontologies, whilst there are four different basic ontologies, and other nine ontologies are only variations for the basic ones differentiating only in ABoxes. In their experiment they ran the algorithm separately for 40 observations (20 in form of an atomic concept assertion and 20 in form of a negated atomic concept assertion). For the abducibles in form of an atomic concept assertion, all of the four basic ontologies were handled successfully (within one hour). Consecutively they extended abducibles to include also assertions in form of negated atomic concept assertions. Only two basic ontologies were handled
successfully, and by the second two Prolog ran out of memory. They did not evaluate the algorithm with respect to observations, as they conjectured that this may lead in too many solutions. They also did not consider sets of multiple observations.

Based on their evaluation, Du et al. concluded that they proved feasibility of their method. However, they can guarantee completeness only on a restricted class of ontologies. Because of the memory problems with Prolog, they plan to investigate other options to handle the translated problem. In further work (Du et al. 2015) they extend their approach also to deal with inconsistent ontologies.

3.3.2 Direct Approaches

Halland & Britz (2012b,a) build their approach on the tableau algorithm for DL and Reiter’s minimal hitting set. They suppose that the tableau algorithm enables to utilize known optimization techniques such as back-jumping and tableau caching. Such a direct approach may therefore be more efficient compared to the translation-based approaches.

Their proposal is based on the fact, that $E$ is an explanation of the observation $O$ w.r.t. $K$ if and only if $K \cup E \models O$, i.e. $K \cup E \cup \neg O$ is inconsistent. All such explanations can be found by finding all models for $K \cup \neg O$ and consequently finding through the minimal hitting set algorithm all such $E$ that make these models inconsistent.

Halland and Britz proposed an abduction algorithm based on an extension of the tableau algorithm that not only checks for consistency, but also enumerates all models. For the collection of all models, they then find all minimal hitting sets using the Reiter’s minimal hitting set algorithm. They call this algorithm as a black box and do not provide any specific treatment for their particular case. After this procedure, they filter out all non-minimal, inconsistent, and irrelevant explanations. The class of explanations is in $ALE$, but algorithm works with a knowledge base and observations in $ALC$.

The proposed algorithm is sound, but not complete. As they state in conclusions, to work over more expressive DLs, some important changes need to be done in their approach to the tableau algorithm.

Ma et al. (2012) proposed an alternative for the Klarman et al.’s work. Similarly as Halland and Britz, instead of translation into an other formalism, they propose to make a tableau for the input knowledge base extended with the negation of the observation. After the tableau is saturated (no tableau rules can be applied), assertions that close all open branches are found. Such assertions correspond to the explanations. There is no claim about soundness and completeness,
and also no implementation is mentioned in this work.

Kaya et al. (2007), Peraldí et al. (2008) motivate their proposal for ABox abduction with multimedia interpretation. Particularly, they focus on what must be added to the theory to answer a conjunctive query true. They work over ontologies extended with DL-safe rules and abduction is based backward chaining of these rules. There are no proofs of soundness and completeness. The implementation is integrated in the RacerPro DL reasoner (Haarslev et al. 2012).
Part II

CONTRIBUTION
4

BASIC IDEA

In this chapter, the main idea of the ABox abduction algorithm will be described. The algorithm builds on top of the minimal hitting set (MHS) algorithm defined in Section 3.2.

We propose an ABox abduction algorithm based on the idea of Halland & Britz (2012a,b). They proposed a mechanism for finding solutions of the abduction problem based on the Minimal hitting set (MHS) algorithm defined by Reiter (1987) in Chapter 3.

4.1 FINDING EXPLANATIONS

Assuming a knowledge base $\mathcal{K}$ and an observation in the form of an ABox assertion, we want to find such a set of ABox assertions $\mathcal{E}$ that $O$ is entailed by $\mathcal{K}$ extended with $\mathcal{E}$, i.e. $\mathcal{K} \cup \mathcal{E} \models O$. That means, according to the reduction between entailment and consistency checking (Lemma 3), $\mathcal{K}$ extended with $\mathcal{E}$ and with $\neg O$ ($\mathcal{K} \cup \mathcal{E} \cup \{\neg O\}$) needs to be inconsistent. In other words, such $\mathcal{E}$ is needed to be found that it disables all models of $\mathcal{K} \cup \{\neg O\}$.

Let $\mathcal{M}$ be the collection of ABox encodings of all models of $\mathcal{K} \cup \{\neg O\}$ according to Definition 10, i.e. $\mathcal{M} = \{M_j \mid I \models \mathcal{K} \cup \{\neg O\}\}$. As observed by Reiter (1987), the set $\mathcal{E}$ can be found by finding a hitting set (Definition 32) of the collection $\mathcal{M}$ and then negating each assertion in the hitting set.

Actually, we can solve the problem homomorphically with the collection of negated ABox encodings of all models of $\mathcal{K} \cup \{\neg O\}$, i.e. $\mathcal{M} = \{-M_j \mid I \models \mathcal{K} \cup \{\neg O\}\}$. In this case, the set of hitting sets of $\mathcal{M}$ corresponds to the set of all explanations of $O$ and no further negations are needed.

The following example shows an example of the ABox abduction problem. Particularly, it introduces an instance of a knowledge base and an observation and a suitable explanation.

Example 16. (Revisited) Consider the extended knowledge base $\mathcal{K}$ and the observation $O = \{\text{Nervous}(\text{jack})\}$ from Example 15.

\[\text{Ill} \sqsubseteq \text{Nervous}\]
\[\text{Nervous} \sqsubseteq \text{Irritable}\]
\[\text{Overworked} \sqcap \text{Hungry} \sqsubseteq \text{Nervous}\]
To explain the observation $O$, we do the following steps:

1. **The collection of ABox encodings of all models for $K \cup \{\neg O\} = K \cup \{\neg \text{Nervous(jack)}\} is:**
   \[
   \left\{ \begin{array}{l}
   \{\neg \text{Nervous(jack)}, \neg \text{Hungry(jack)}, \neg \text{Overworked(jack)}, \neg \text{Ill(jack)}, \\
   \quad \quad \quad \quad \text{Irritable(jack)} \}, \\
   \{\neg \text{Nervous(jack)}, \neg \text{Hungry(jack)}, \neg \text{Overworked(jack)}, \neg \text{Ill(jack)}, \\
   \quad \quad \quad \quad \neg \text{Irritable(jack)} \}
   \end{array} \right. \]

2. **The collection of negations of ABox encodings of models is:**
   \[
   M = \left\{ \begin{array}{l}
   \{\text{Nervous(jack)}, \text{Hungry(jack)}, \text{Overworked(jack)}, \text{Ill(jack)}, \\
   \quad \quad \quad \quad \neg \text{Irritable(jack)} \}, \\
   \{\text{Nervous(jack)}, \text{Hungry(jack)}, \text{Overworked(jack)}, \text{Ill(jack)}, \\
   \quad \quad \quad \quad \text{Irritable(jack)} \}
   \end{array} \right. \]

3. **According to the Definition 32 and the Definition 28 the collection of all minimal hitting sets for $M$ is:**
   \[
   \{\{\text{Ill(jack)}\}, \\
   \quad \{\text{Overworked(jack)}, \text{Hungry(jack)}\} \}
   \]

The previous example shows a collection of all minimal hitting sets for the collection of sets corresponding to the negations of the models for the particular knowledge base. The example does not deal with a way how to compute this collection (actually the collection of the explanations for the problem). To find all explanations we need an algorithm to do it systematically. As we already explained, we will adopt the Reiter’s MHS algorithm for this purpose.

### 4.2 Minimal Hitting Sets and ABox Abduction

We will now adjust the minimal hitting set (MHS) algorithm described in Section 3.2 to the formalism of description logics. Given an input knowledge base $K$ and an observation $O$ in form of a single DL ABox assertion, the algorithm constructs a HS-tree for the collection $M$ of all models of $K \cup \{\neg O\}$ introducing some additional pruning besides for Reiter’s pruning conditions, as follows.

Let us recapitulate, that in a HS-tree, every node is labelled by a set from $M$, by $\checkmark$, or by $\times$ if the node is pruned. Every edge from a node $n$ is labelled by an element of the label of $n$. Moreover, there are constraints for labelling listed in Definition 33, such that the construction
of the HS-tree results into a tree where the set of labels on every path from the root to a leaf corresponds to a hitting set.

Let us introduce minor changes in HS-tree labelling. The nodes corresponding to minimal hitting sets will be labelled by $\{\}$, and the pruned nodes will be labelled either by $\times$ or by $\{\}$, which will be used to steer further pruning.

As described in Section 3.2, Reiter proposed to construct the HS-tree by breadth-first search and to prune it to find all minimal hitting sets. We simplify and extend pruning in the same time for our purpose, summarizing it in Definition 37. Recall that $H(n)$ represents a set of ABox assertions on the edges on the path from the root to the node $n$.

**Definition 37 (Pruned node).** A node $n \in V$ in a HS-tree $T = (V, E, L)$ for an ABox abduction problem $\mathcal{P} = (\mathcal{K}, \mathcal{O})$ is pruned if:

(a) either there is $n' \in V$ s.t. $H(n') \subseteq H(n)$ and $L(n') = \{\}$

(label $n$ by $\{\}$);

(b) or there is $n' \in V$ s.t. $H(n') = H(n)$ and $L(n') = M \in \mathcal{M}$

(label $n$ by $\times$);

(c) or $\{\neg O\} \cup H(n)$ is inconsistent (label $n$ by $\{\}$);

(d) or $H(n) \cup \mathcal{K}$ is inconsistent (label $n$ by $\{\}$);

The first condition (a) preserves Reiter’s proposal to prune a node $n$ if there is another node $n'$ with $H(n') \subseteq H(n)$ that corresponds to a minimal hitting set. In our case, such node $n'$ is labelled by $\{\}$. Additionally, a node $n$ is pruned, if there is another node $n'$ with $H(n') \subseteq H(n)$ that was previously pruned according to (c) or (d).

Secondly (condition (b)) $n$ is also pruned, but labelled by $\times$, if there is another node $n'$ with the same path from the root labelled by any set from the input collection. In this case, there is no need to continue in the same paths of the tree, and so one of them ($n$) is pruned.

The third condition (c) is new and prevents to build such tree branches that correspond to irrelevant explanations, i.e. such minimal hitting sets that implies the observation. Particularly, if $H(n) \cup \{\neg O\}$ is inconsistent, then $H(n) = O$, and so $n$ is pruned and $L(n) = \{\}$. Note that, each superset of $H(n)$ is also inconsistent with $\{\neg O\}$, and so the node must be pruned (assured by condition (a)).

The last condition (d) prevents to compute inconsistent explanations – if $H(n) \cup \mathcal{K}$ is inconsistent, then $n$ is pruned and $L(n) = \{\}$. Also, note that, in such a case each superset of $H(n)$ is inconsistent with $\mathcal{K}$, and so the node must be pruned (again assured by condition (a)).

A HS-tree is pruned if and only if all the pruned nodes are removed with their subtrees.
**Definition 38** (Pruned HS-tree). A pruned HS-tree is obtained from a HS-tree by removing all pruned nodes and their descendants.

Now, the consequence of the pruning is, that in the pruned HS-tree according to Definition 37 and Definition 38, all minimal hitting sets corresponding to minimal, explanatory, consistent, and relevant explanations of the problem \( \mathcal{P} = (\mathcal{X}, \mathcal{O}) \) are found.

**Theorem 4.** Let \( T = (V, E, L) \) be a pruned HS-tree for a collection of sets \( \mathcal{M} \) and an ABox abduction problem \( \mathcal{P} = (\mathcal{X}, \mathcal{O}) \). Then \( \{H(n) \mid n \in V, L(n) = \emptyset, \text{and } n \text{ is not pruned} \} \) is the collection of all explanatory, consistent, relevant, and minimal explanations of \( \mathcal{O} \) w.r.t. \( \mathcal{X} \).

**Proof.** Let us remind, that all the pruning conditions proposed by Reiter except the last one are included in our pruning. The Reiter’s last condition to prune deals with two nodes \( n_1 \) and \( n_2 \) labelled as follows \( L(n_1) = S_1 \in \mathcal{M} \) and \( L(n_2) = S_2 \in \mathcal{M} \), while \( S_1 \subset S_2 \). This situation never appears in our approach, as our collection \( \mathcal{M} \) contains only sets containing each atomic ABox assertion from \( \mathcal{X} \) either in positive way or its complement, and thus for no \( S_1, S_2 \in \mathcal{M} \) it can hold that \( S_1 \subset S_2 \). Since this condition never applies, and the other Reiter’s conditions are included in our approach, only minimal hitting sets are found. This follows from Theorem 4, as proven by Reiter (1987).

But obviously, our conditions (c) and (d) only filter from these all minimal hitting sets such sets, that are irrelevant and inconsistent. In addition, also supersets of such previously pruned sets are filtered out, as part of condition (a). That is, in our HS-tree, all minimal hitting sets corresponding to the relevant and consistent explanations are found.

Recall also that, an explanation \( \mathcal{E} \) for \( \mathcal{P} = (\mathcal{X}, \mathcal{O}) \) is explanatory if and only if \( \mathcal{X} \not\models \mathcal{O} \), i.e. \( \mathcal{X} \cup \{\neg \mathcal{O}\} \) has at least one model. Hence, if there is a HS-tree with non-empty collection of minimal hitting sets, then all these minimal hitting sets must correspond to explanatory explanations. \( \square \)

The following example shows how to find all minimal, explanatory, consistent, and relevant explanations for an ABox abduction problem through constructing a HS-tree. In Example 17, the description for the HS-tree in Figure 2 (continuing in Figure 3) is stated. Though, in this section, we established the labelling of each pruned node \( n \) and each node \( n \) s.t. \( H(n) \) is a minimal hitting set to be \( \emptyset \), let us draw the symbols \( \times \) and \( \checkmark \) in the trees to differentiate the pruned nodes from the minimal hitting set nodes (to apply the pruning from Definition 37 we assume each such node is labelled by \( \emptyset \)).

**Example 17.** (Revisited) Let us consider the knowledge base \( \mathcal{X} \), the observation \( \mathcal{O} \), and the collection of all negations of models \( \mathcal{M} \) from Example 16.
We will construct a pruned HS-tree $T = (V, E, L)$ according to the above definitions. The output tree is pictured in Figure 2. The steps are as follows (notice that, the tree is created w.r.t. breadth-first search):

1. Label a root node $\tau \in V$ of the tree $T$ with $M_1 \in M$: $L(\tau) = M_1$.

2. For each $\sigma \in L(\tau)$, create a child $n_\sigma$ of $\tau$, and label its edge with $\sigma$ (i.e. $L(\tau, n_\sigma) = \sigma$); particularly, there are five new nodes $n_{\text{Nervous}(\text{jack})}$, $n_{\text{Overworked}(\text{jack})}$, $n_{\text{Hungry}(\text{jack})}$, $n_{\text{Irritable}(\text{jack})}$, $n_{\text{Ill}(\text{jack})}$, with the respective edge-labels.

3. The node $n_{\text{Nervous}(\text{jack})}$ is pruned (i.e. labelled by $\times$), as

$$H(n_{\text{Nervous}(\text{jack})}) = \{\text{Nervous}(\text{jack})\} \models \text{O} = \{\text{Nervous}(\text{jack})\},$$

i.e. this hitting set cannot be a relevant explanation.

4. The node $n_{\text{Overworked}(\text{jack})}$ is labelled by a set $M_2 \in M$ s.t. $\neg M_2$ is a model of $\mathcal{K} \cup \neg \text{O} \cup H(n_{\text{Overworked}(\text{jack})})$. Afterwards, five children of this node are created for each $\sigma \in M_2$.

5. Similarly, the nodes $n_{\text{Hungry}(\text{jack})}$ and $n_{\text{Irritable}(\text{jack})}$ are labelled by $M_3, M_4 \in M$ s.t. $\neg M_3$ is a model of $\mathcal{K} \cup \neg \text{O} \cup H(n_{\text{Hungry}(\text{jack})})$, and $\neg M_4$ is a model of $\mathcal{K} \cup \neg \text{O} \cup H(n_{\text{Irritable}(\text{jack})})$. Afterwards, five children of each node are created for each $\sigma \in M_3$ and each $\sigma \in M_4$.

6. The node $n_{\text{Ill}(\text{jack})}$ is labelled by $\checkmark$, because $H(n_{\text{Ill}(\text{jack})}) = \{\text{Ill}(\text{jack})\} \cup \mathcal{K} \cup \neg \text{O}$ is inconsistent, i.e. $\{\text{Ill}(\text{jack})\} \cup \mathcal{K} \models \text{O}$ and so $\{\text{Ill}(\text{jack})\}$ is an explanatory, consistent, relevant, and minimal explanation of O.

7. On the third level of the tree, the first child $n_1$ of $n_{\text{Overworked}(\text{jack})}$ with the edge Nervous(jack) is pruned, because $H(n_1)$ is a superset of $H(n_1')$ for a pruned node $n_1'$. The child $n_2$ with the edge Ill(jack) is pruned, because $H(n_2)$ is a superset of a minimal hitting set $H(n_2')$. The child $n_3$ with the edge Overworked(jack) is pruned, because $H(n_3)$ contains clash (and so $\mathcal{K} \cup H(n_3)$ is inconsistent). The child $n_4$ with the edge Hungry(jack) corresponds to a minimal hitting set, as $\mathcal{K} \cup H(n_4) \cup \{\neg \text{O}\}$ is inconsistent. The node $n_5$ with the edge Irritable(jack) is labelled by $M_5$, s.t. $\neg M_5$ is a model of $\mathcal{K} \cup H(n_5)$.

8. There is only one remaining node $n_6$ that is not pruned, but labelled by $M_6$. All other nodes are pruned, as their paths from the root are not minimal, irrelevant or inconsistent. The node $n_7$ with $H(n_7) = \{\text{Irritable}(\text{jack}), \text{Hungry}(\text{jack})\}$ (a child of the node labelled by $M_4$) is pruned, because there is a node $n_8$ s.t. $H(n_7) = H(n_8)$ and $L(n_8) = M_6$ is satisfied.
$M_1 = \{\text{Nervous(jack)}, \text{Overworked(jack)}, \text{Hungry(jack)}, \text{Irritable(jack)}, \text{Ill(jack)}\}$

$M_2 = \{\text{Nervous(jack)}, \text{Overworked(jack)}, \text{Hungry(jack)}, \text{Irritable(jack)}, \text{Ill(jack)}\}$

$M_3 = \{\text{Nervous(jack)}, \text{Overworked(jack)}, \text{Hungry(jack)}, \text{Irritable(jack)}, \text{Ill(jack)}\}$

$M_4 = \{\text{Nervous(jack)}, \text{Overworked(jack)}, \text{Hungry(jack)}, \text{Irritable(jack)}, \text{Ill(jack)}\}$

$M_5 = \{\text{Nervous(jack)}, \text{Overworked(jack)}, \text{Hungry(jack)}, \text{Irritable(jack)}, \text{Ill(jack)}\}$
$M_3 = \{ \text{Nervous(jack)}, \text{Overworked(jack)}, \neg \text{Hungry(jack)}, \text{Irritable(jack)}, \text{Ill(jack)} \}$

$M_6 = \{ \text{Nervous(jack)}, \text{Overworked(jack)}, \neg \text{Hungry(jack)}, \neg \text{Irritable(jack)}, \text{Ill(jack)} \}$

Figure 3: Subtree of HS-tree in Figure 2
Our proposed algorithm solves the ABox abduction problem by constructing a HS-tree and finding the models using the tableau algorithm (TA).

We first present a basic version of the algorithm (called SOA), which computes explanations for a single observation. Afterwards, we extend this algorithm to handle a set of observations. We propose two approaches to deal with a set of observations. The first approach is called AAA\textsubscript{s}, and it is based on splitting the problem into subproblems, which means that for each observation from an input set the SOA algorithm is called separately, and the results are consequently combined. The second approach is called AAA\textsubscript{R}, and it is based on reducing the set of observation into a single complex concept assertion. Consequently, a modified version of SOA is used to compute the explanations for the reduced input.

Moreover, we propose a version with limited size of the explanations. This restriction can be applied on both of the listed approaches.

5.1 SINGLE OBSERVATION ALGORITHM

The algorithm for a single observation is given in a pseudocode in Algorithm 4. It takes a DL knowledge base \( \mathcal{K} \) and a single observation \( O \) in form of any ABox assertion (concept assertion, or role assertion, or negated role assertion) on the input.

Firstly, a model \( M \) of \( \mathcal{K}^\prime = \mathcal{K} \cup \{\neg O\} \) is computed using TA (lines 1–6). Note that, we assume that we extract the model from TA in form of ABox encoding (Definition 10). If \( \mathcal{K}^\prime \) has no models, it means that \( \mathcal{K} \cup \neg O \) is inconsistent, i.e. \( \mathcal{K} \models O \) and hence there is nothing to explain (line 4). Otherwise, the algorithm continues with constructing a HS-tree \( T = (V, E, L) \) (line 7).

The root \( r \in V \) of the tree \( T \) is created and labelled by \( \neg M \) (line 8). For each element \( \sigma \) from \( \neg M \) a successor of \( r \) is created and its edge is labelled by \( \sigma \) (line 9).

The root node \( r \) is now fully processed. We initialize the output set of explanations (i.e., minimal hitting sets) as \( S_\mathcal{E} = \{\} \) and traverse the remaining nodes in \( V \) by breadth-first search in a loop (lines 12–30). For each node \( n \), we compute a set \( M \) such that \( n \) will be labelled by \( \neg M \). There are three options for \( M \):
1. either $M$ will be set to $\{}$ if $n$ is a leaf node (i.e., it corresponds to a MHS) or if it can be pruned because of the minimality, relevance, or consistency;

2. or $M$ will be set to $\times$ if $n$ is pruned, because there is already the same path in the tree;

3. or $M$ must be a suitable model of $\mathcal{K} \cup \{\neg O\}$ s.t. $H(n) \subseteq M$.

To implement this effectively, the algorithm first checks the pruning conditions for the node $n$ (line 13):

1. if there is a clash in $H(n)$, then this path cannot lead to a consistent MHS and thus prune the node (note that, this case could be included in the last condition, but as it is much simpler to check a clash within $H(n)$ – a set of ABox assertions, we do this first),

2. if there is a node $n' \in V$ with $H(n') \subseteq H(n)$ and $L(n') = \{}$ then prune $n$ according to Definition 37 (a),

3. if $H(n) \cup \{\neg O\}$ is inconsistent, then prune $n$ according to Definition 37 (c),

4. if $H(n) \cup \mathcal{K}$ is inconsistent, then prune $n$ according to Definition 37 (d).

In all these cases node $n$ is labelled by $\{}$. One additional pruning condition is checked in line 15:

5. if there is a node $n' \in V$ that is already labelled and $H(n') = H(n)$, then prune $n$ according to Definition 37 (b) and label it by $\times$.

If pruning cannot be applied to the node, another optimization is considered, which is model reuse. Since calling TA to compute new models is expensive, we try to do it as least as possible. Every computed model is stored in $MS$ and reused when possible. So we test if there already is $N \in MS$ that contains every assertion from $H(n)$ and if yes, we set $M = N$ (lines 17–18).

If neither pruning nor model reuse can be applied, TA is called and a new model $M$ for $\mathcal{K}' \cup H(n)$ is computed (line 20). If such a model $M \neq \{}$ is indeed found, it is stored in MS for later reuse (lines 21–22).

In the other case, $M = \{}$ and $\mathcal{K}' \cup H(n)$ is inconsistent. This means that $\mathcal{K} \cup H(n) \models O$, in other words we have found a new MHS which represents an explanation for $O$, and so $H(n)$ is stored in $SE$ (line 24).

Finally, the node $n$ is now labelled with $\neg M$ (which is $\{}$ in case when $M = \{}$ and $\times$ in case when $M = \times$), its successors are created, their edges are labelled and the loop is reiterated.
The main loop in lines 12–30 is repeated until there is no other unexplored node. The output of the algorithm is the set of all minimal hitting sets $S_E$, which corresponds to all minimal explanations of the observation $O$ w.r.t. the knowledge base $\mathcal{K}$.

The algorithm SOA is sound and complete, and always terminates.

**Theorem 5.** Let $\mathcal{K}$ be an $\text{ALCHO}$ knowledge base, and let $O$ be an observation in form of an $\text{ALCHO}$ ABox assertion. The SOA algorithm, initialized with $\mathcal{K}$ and $O$: (a) outputs precisely all consistent, relevant, explanatory, and subset-minimal explanations of $\mathcal{P} = (\mathcal{K}, O)$ of the form $E \subseteq \{A(a), \neg A(a), R(a, b), \neg R(a, b) \mid A \in N_C, R \in N_R, a, b \in N_I\}$; and (b) it eventually terminates.

**Proof.** The soundness, completeness, and termination of the SOA algorithm is directly implied by Corollary 1 on page 73, where this is proven for the algorithm with length of explanations limited to $l$. The algorithm SOA$^l$ corresponds to the case when $l = \infty$.

As SOA is based on the tableau algorithm for $\text{ALCHO}$ and on the minimal hitting set algorithm, its complexity is implied by these two algorithms. Particularly it is implied by the higher of these two, that is by the consistency checking of an $\text{ALCHO}$ knowledge base.

**Theorem 6.** The worst case complexity of the SOA algorithm is ExpTime.

**Proof.** Reiter’s minimal hitting set algorithm is in NP (Reiter 1987, Karp 1972). The SOA algorithm implements this algorithm and calls TA at most once in each step. Since TA for $\text{ALCHO}$ is in ExpTime (Hladik & Model 2004), the overall complexity of SOA is also ExpTime.

### 5.2 Multiple Observations Algorithm

We will now explain how the algorithm is extended to handle multiple observations. As this is the overall algorithm, we call it the ABox Abduction Algorithm (AAA). In fact, we explore two ways how to compute explanations for multiple explanations, resulting into two versions of the algorithm. The first is based on calling SOA for each observation separately (the splitting into subproblems approach), the other is based on reducing the set of observations into a single observation and then calling SOA (the reduction approach).

#### 5.2.1 Splitting into Subproblems

We will now explain how the algorithm is extended to handle multiple observations by splitting the problem into subproblems. Particularly, an ABox abduction problem $\mathcal{P} = (\mathcal{K}, \{O_1, \ldots, O_n\})$ can be solved
Algorithm 4 SOA(\(\mathcal{K}, O\)): Single Observation Abduction

Require: knowledge base \(\mathcal{K}\), observation \(O\)
Ensure: set of all explanations \(S_E\)

1: \(\mathcal{K}' \leftarrow \mathcal{K} \cup \{\neg O\}\)
2: \(M \leftarrow\) call TA with input \(\mathcal{K}'\) \hspace{1cm} \triangleright \) TA returns a model of \(\mathcal{K}'\)
3: if \(M = \{\}\) then
4:    return "nothing to explain"
5: end if
6: \(MS \leftarrow \{M\}\)
7: create new HS-tree \(T = (V, E, L)\) with root \(r\)
8: \(L(r) \leftarrow \neg M\)
9: for each \(\sigma \in \neg M\) create a successor \(n_\sigma\) of \(r\) and label the resp. edge by \(\sigma\)
10: \(S_E \leftarrow \{\}\)
11: \(n \leftarrow\) next node w.r.t. \(r\) in \(T\) by breadth-first search
12: while \(n \neq \) null do
13:    if (clash in \(H(n)\)) \hspace{1cm} \triangleright \) \(H(n)\) – set of edge-labels on path \(r-n\)
14:       or (\(n' \in V\) and \(H(n') \subseteq H(n)\) and \(L(n') = \{\}\))
15:       or (\(H(n) \cup \{\neg O\}\) is inconsistent)
16:       or (\(H(n) \cup \mathcal{K}\) is inconsistent) then
17:       \(M \leftarrow \{\}\\) \hspace{1cm} \triangleright \) prune the path
18:    else if \(n' \in V\) and \(H(n) = H(n')\) and \(L(n') \neq\) null then
19:       \(M \leftarrow \times\)
20:    else if \(N \in MS\) and \(H(n) \subseteq N\) then
21:       \(M \leftarrow N\) \hspace{1cm} \triangleright \) reuse model
22:    else
23:       \(M \leftarrow\) call TA for \(\mathcal{K}' \cup H(n)\)
24:       if \(M \neq \{\}\) then
25:          \(MS \leftarrow MS \cup \{M\}\) \hspace{1cm} \triangleright \) store the model for later reuse
26:       else
27:          \(S_E \leftarrow S_E \cup \{H(n)\}\) \hspace{1cm} \triangleright \) the case \(H(n)\) explains \(O\)
28:       end if
29:    end if
30: end while
31: return \(S_E\)
by solving \( n \) subproblems \( \mathcal{P}_1, \ldots, \mathcal{P}_n \), where each \( \mathcal{P}_i = (\mathcal{K}', \mathcal{O}_i) \). Notice that the input knowledge base changed. The reason is, it is not enough to solve \( n \) subproblems where \( \mathcal{P}_i = (\mathcal{K}, \mathcal{O}_i) \), whilst in such a case, some explanations of the form \( R(a, b) \) or \( \neg R(a, b) \), where \( a, b \) are individuals occurring in \( \mathcal{O}_i \) and \( \mathcal{O}_j \) for some \( j \neq i \), may be missing.

To include also these types of explanations, the input knowledge base for \( \mathcal{P}_i \) is simply gained as \( \mathcal{K} \) extended with all individuals occurring in all observations \( \mathcal{O}_1, \ldots, \mathcal{O}_n \), i.e. \( \mathcal{K}' = \mathcal{K} \cup \{ \top | a \in \mathcal{O}_i \}^* \). Note that when an assertion \( \top(a) \) is added to \( \mathcal{K} \) it does not change the meaning of \( \mathcal{K} \), but it merely adds an individual to the domain.

In the following example we show that without this extension some obvious explanations would be missed.

**Example 18.** Find all explanations of \( \mathcal{P} = (\mathcal{K}, \mathcal{O} = \{\mathcal{O}_1, \mathcal{O}_2\}) \) for the following knowledge base \( \mathcal{K} \) and the observations \( \mathcal{O}_1 = \text{HappyCat}(\text{tom}) \) and \( \mathcal{O}_2 = \text{CatLover}(\text{mammy}) \):

\[
\exists \text{owns}.\text{Cat} \sqsubseteq \text{CatLover}
\]

\[
\exists \text{ownedBy}.\text{CatLover} \sqsubseteq \text{HappyCat}
\]

One of the explanations for \( \mathcal{O}_1 \) is

\[
\mathcal{E}_1 = \{\text{CatLover}(\text{mammy}), \text{ownedBy}(\text{tom}, \text{mammy})\},
\]

and similarly for \( \mathcal{O}_2 \) is

\[
\mathcal{E}_2 = \{\text{owns}(\text{mammyTwoShoes}, \text{tom}), \text{Cat}(\text{tom})\}.
\]

But \( \mathcal{E}_1 \) is not a solution for \( \mathcal{P}_1 = (\mathcal{K}, \mathcal{O}_1) \) and \( \mathcal{E}_2 \) is not a solution for \( \mathcal{P}_2 = (\mathcal{K}, \mathcal{O}_2) \), as the individuals \( \text{mammy} \) and \( \text{tom} \) do not occur in \( \mathcal{K} \). But, when considering each \( a \in \mathcal{O}_i \) in each \( \mathcal{P}_i \), then \( \mathcal{E}_1 \) is an explanation of \( \mathcal{P}'_1 = (\mathcal{K} \cup \{\top(\text{mammy}, \top(\text{tom}))\}, \mathcal{O}_1) \), and \( \mathcal{E}_2 \) is an explanation for \( \mathcal{P}'_2 = (\mathcal{K} \cup \{\top(\text{tom})\}, \mathcal{O}_2) \).

Let us now have a look on the algorithm. First we need to update the SOA algorithm that will be used to solve the partial problems. The new version is given in Algorithm 5. It has one additional input parameter – the set of all observations \( S_\mathcal{O} \). The reason for this new input parameter is to optimize the computation. According to Definition 36, each explanation must be relevant with respect to each \( \mathcal{O}_i \in S_\mathcal{O} \). The relevance can be checked in the end, but by checking it during construction of each HS-tree, some computation can be saved.

---

* By the notation \( a \in \mathcal{O}_i \) we mean that \( a \) is an individual and \( \mathcal{O}_i = C(a) \), or \( \mathcal{O}_i = \neg C(a) \) for some concept \( C \), or \( \mathcal{O}_i = R(a, b) \), or \( \mathcal{O}_i = \neg R(a, b) \), or \( \mathcal{O}_i = R(b, a) \), or \( \mathcal{O}_i = \neg R(b, a) \) for some role \( R \) and some individual \( b \).

† In case of an observation in the form of a single ABox assertion, the input parameter \( S_\mathcal{O} \) is simply set to \( \{\} \).
Although in SOA only the explanations for a subproblem are computed, and in the end of AAA$_S$ they are composed with other subproblem’s solutions, the irrelevance can be checked already in SOA, as it is preserved in supersets, as shown by Observation 2. Thus, the third pruning condition on line 13 in SOA (Algorithm 5) is changed to check relevance for each $O_i \in S_O$.

Observation 2. If $E$ is irrelevant with respect to an observation $O$, then any $E' \subseteq E$ is irrelevant with respect to $O$.

AAA$_S$ receives a knowledge base $K$ and a set of observations $O = \{O_1, \ldots, O_n\}$ as inputs. Before the explanations for each observation $O_i \in O$ are computed, the input knowledge base $K$ is extended into $K'$ by adding all individuals from all input observations in line 2.

The loop in lines 3–10 is executed for each $O_i \in O$. Firstly, a set of explanations $S_{E_i}$ for $O_i$ is computed using SOA, i.e. by calling Algorithm 4 with the inputs $K'$ and $O_i$ (line 4). If $O_i$ cannot be explained, then $S_{E_i}$ is empty. Moreover, since one of the observations from $O$ cannot be explained, then also the whole set of observations $O$ cannot be explained. In that case the algorithm terminates and returns an empty set of explanations (line 6).

If SOA returned $S_{E_i} =$ "nothing to explain" for $O_i$, then $K' \models O_i$, $O_i$ is excluded from the set of observations $O$ (line 7) and algorithm continues in the loop for another $O_j$. As shown by Observation 3, this leads to a correct solution.

Observation 3. Given a set of observations $O = \{O_1, \ldots, O_n\}$ and given $O' = O \setminus \{O_i \in O \mid K \models O_i\}$, we have $K \cup E \models O$ if and only if $K \cup E \models O'$. Hence $P = (K, O)$ has the same set of explanations as $P' = (K', O')$ where $K' = K \cup \{\top(a) \mid a \in O_i, K \models O_i\}$

Otherwise, if SOA returned a nonempty set of explanations $S_{E_i}$, it is accumulated in the collection (line 8).

After executing the loop for each $O_i \in O$, the emptiness of $\Sigma$ is checked (line 11). If $\Sigma$ is empty there is nothing to explain, as this would happen only if for each $O_i$ SOA returned "nothing to explain" in line 7.

If $\Sigma$ is non-empty, all combinations of all explanations are computed and stored in $S_\Sigma$, by combining one explanation $E_i$ from each $S_{E_i}$ stored in $\Sigma$ (line 14).

Finally, the collection of these combinations is filtered to omit those that are not minimal, consistent, or relevant (line 15) and returns it as the output of AAA$_S$ (line 17). This additional filtration is necessary because with composing the explanations, the minimality, consistency, and relevance may be lost.
Algorithm 5 SOA(\(\mathcal{K}, O, S_O\)): Single Observation Abduction (Updated)

**Require**: knowledge base \(\mathcal{K}\), observation \(O\), set of observations \(S_O\)

**Ensure**: set of all explanations \(S_E\)

1: \(\mathcal{K}' \leftarrow \mathcal{K} \cup \{\neg O\}\)
2: \(M \leftarrow \text{call TA with input } \mathcal{K}' \quad \triangleright \text{TA returns a model of } \mathcal{K}'\)
3: if \(M = \emptyset\) then
4: return "nothing to explain"
5: end if
6: \(MS \leftarrow \{M\}\)
7: create new HS-tree \(T = (V, E, L)\) with root \(r\)
8: \(L(r) \leftarrow \neg M\)
9: for each \(\sigma \in \neg M\) create a successor \(n_\sigma\) of \(r\) and label the resp. edge by \(\sigma\)
10: \(S_E \leftarrow \emptyset\)
11: \(n \leftarrow \text{next node w.r.t. } r \text{ in } T \text{ by breadth-first search}\)
12: while \(n \neq \text{null}\) do
13: if (clash in \(H(n)\)) \(\triangleright \text{H}(n) \text{ - set of edge-labels on path } r-\neg M-n\)
14: \(M \leftarrow \emptyset\) \(\quad \triangleright \text{prune the path}\)
15: else if \(n' \in V \text{ and } H(n) \subseteq H(n') \text{ and } L(n') = \emptyset\)
16: \(M \leftarrow \times\)
17: else if \(N \in MS \text{ and } H(n) \subseteq N\) then
18: \(M \leftarrow N\) \(\quad \triangleright \text{reuse model}\)
19: else
20: \(M \leftarrow \text{call TA for } \mathcal{K}' \cup H(n)\)
21: if \(M \neq \emptyset\) then
22: \(MS \leftarrow MS \cup \{M\}\) \(\quad \triangleright \text{store the model for later reuse}\)
23: else
24: \(S_E \leftarrow S_E \cup \{H(n)\}\) \(\quad \triangleright \text{the case } H(n) \text{ explains } O\)
25: end if
26: end if
27: \(L(n) \leftarrow \neg M\)
28: for each \(\sigma \in \neg M\) create a successor \(n_\sigma\) of \(n\) and label the resp. edge by \(\sigma\)
29: \(n \leftarrow \text{next node in } T \text{ w.r.t. } n \text{ by breadth-first search}\)
30: end while
31: return \(S_E\)
In case of minimality, the combining of explanations may lead to producing two explanations, while one is a subset of the other. That is the minimality must be verified after composing the explanations. For example, there can be an explanation $E = \{\text{Female}(\text{eva})\}$ for each $P_i$. It means, $E$ is also the solution of $P$. But composing $E$ with all the other possible explanations for each $P_i$ would also result into supersets of $E$, and such explanations are not minimal, i.e. they must be filtered out.

Also, combining multiple consistent partial explanations may result into an inconsistent explanation, and this explanation must be omitted. For example, consider an explanation $E_1 = \{\text{Female}(\text{alex})\}$ for $O_1$ and an explanation $E_2 = \{\neg\text{Female}(\text{alex})\}$ for $O_2$, both consistent w.r.t. a knowledge base $\mathcal{K}$. Their composition $E_1 \cup E_2$ results into an inconsistent explanation, and so cannot be the solution of $P = (\mathcal{K}, \{O_1, O_2\})$.

Observe that while irrelevance is preserved in supersets, relevance is not. Therefore relevance must also be verified after combining the partial explanations (whose relevance was already verified by SOA). For example, having $O_i = \{\text{Female} \land \text{Professor}(\text{eva})\}$, an explanation $E_j = \{\text{Female}(\text{eva})\}$ for any $O_j$ for $j \neq i$ and an explanation $E_k = \{\text{Professor}(\text{eva})\}$ for any $O_k$ for $k \neq i \neq j$, both $E_j, E_k$ are relevant with respect to $O_i$, but $E_j \cup E_k$ is not (i.e. $E_j \cup E_k \models O_i$). Thus, each combination of explanations must be tested to be relevant with respect to each $O_i \in O$.

The filtered set $S_{E}^{\min}$ is returned as the output of AAA$_S$ in line 17.

The following example illustrates, what AAA$_S$ returns for a concrete input.

**Example 19.** Consider the following knowledge base $\mathcal{K}$:

\[
\exists \text{owns}.\text{Cat} \sqsubseteq \text{CatLover} \\
\exists \text{chase}.\text{Mouse} \sqcap \text{Cat} \sqsubseteq \text{HappyCat} \\
\text{Cat} \sqcup \text{Mouse} \sqsubseteq \text{Animal} \\
\text{Cat} \sqsubseteq \neg\text{Mouse} \\
\text{Cat(\text{tom})}
\]

Let us run AAA$_S$ with the input parameters as follows: the knowledge base $\mathcal{K}$ and a set of observations $O$:

\[
O = \{\text{CatLover}(\text{mammy}), \text{HappyCat}(\text{tom}), \text{Animal}(\text{jerry})\}.
\]

For each observation $O_i \in O$, SOA is called and the set of explanations $S_{E_i}$ is returned:
Algorithm 6 AAA₅ (K, O): ABox Abduction Algorithm with Splitting

Require: knowledge base K, set of observations O
Ensure: set of all minimal explanations S₅

1: Σ ← \{\} \triangleright collection of the sets of explanations for all observations
2: \( \mathcal{K}' = \mathcal{K} \cup \{ T(a) \mid a \in O \} \)
3: for all \( O_i \in O \) do
4: \( S_{E_i} \leftarrow \text{SOA}(\mathcal{K}', O_i, O) \)
5: if \( S_{E_i} = \{\} \) then
6: return \( \{\} \)
7: else if \( S_{E_i} \neq \text{"nothing to explain"} \) then \( \mathcal{K} \models O_i \) – exclude \( O_i \)
8: \( \Sigma \leftarrow \Sigma \cup \{S_{E_i}\} \)
9: end if
10: end for
11: if \( \Sigma = \{\} \) then
12: return \"nothing to explain\"
13: else
14: \( S_E \leftarrow \{E_1 \cup \ldots \cup E_m \mid E_i \in S_{E_i}, S_{E_i} \in \Sigma, m = |\Sigma|\} \triangleright \) all combinations of the expl.
15: \( S_{E}^{\min} \leftarrow \{E \mid E \in S_E \text{ and } \forall E' \in S_E : E' \not\subseteq E \text{ and } E \text{ is consistent and relevant}\} \)
16: end if
17: return \( S_{E}^{\min} \)

\( O_1 = \{\text{CatLover(mammy)}\} \quad \quad S_{E_1} = \{\{\text{owns(mammy, tom)}\}\}, \)

\( O_2 = \{\text{HappyCat(tom)}\} \quad \quad S_{E_2} = \{\{\text{Mouse(jerry), chase(tom, jerry)}\}, \)
\( \quad \quad \{\text{Mouse(mammy), chase(tom, mammy)}\}\}, \)

\( O_3 = \{\text{Animal(jerry)}\} \quad \quad S_{E_3} = \{\{\text{Cat(jerry)}\}, \{\text{Mouse(jerry)}\}\}. \)

Afterwards, all combinations for each \( E_i \in S_{E_i} \) are composed:

\( S_E = \{\)
\( \{\text{owns(mammy, tom), Mouse(jerry), chase(tom, jerry), Cat(jerry)}\}, \)
\( \{\text{owns(mammy, tom), Mouse(jerry), chase(tom, jerry)}\}, \)
\( \{\text{owns(mammy, tom), Mouse(mammy), chase(tom, mammy)}\}, \)
\( \text{Cat(jerry)}\}, \)
\( \{\text{owns(mammy, tom), Mouse(mammy), chase(tom, mammy)}\}, \)
\( \text{Mouse(jerry)}\} \}

From \( S_E \) all non-minimal, inconsistent, and irrelevant explanations must be filtered. As you can see, the first explanation \{\text{owns(mammy, tom), Mouse(jerry), chase(tom, jerry), Cat(jerry)}\} is a superset of the second \{\text{owns(mammy, tom), Mouse(jerry), chase(tom, jerry)}\} (moreover, the first is inconsistent with \( \mathcal{K} \) because of the axiom \( \text{Cat} \subseteq \neg \text{Mouse} \)). Thus, the first explanation is omitted and the output of AAA₅ is as follows:
The algorithm $\text{AAA}_S$ is sound and complete, and always terminates.

**Theorem 7.** Let $\mathcal{K}$ be an $\mathcal{ALCHO}$ knowledge base, and let $O$ be an observation in form of a set of $\mathcal{ALCHO}$ ABox assertions. The $\text{AAA}_S$ algorithm, initialized with $\mathcal{K}$ and $O$: (a) outputs precisely all consistent, relevant, explanatory, and subset-minimal explanations of $P = (\mathcal{K}, O)$ of the form $E \subseteq \{A(a), \neg A(a), R(a, b), \neg R(a, b) \mid A \in N_C, R \in N_R, a, b \in N_I\}$; and (b) it eventually terminates.

**Proof.** The soundness, completeness, and termination of the $\text{AAA}_S$ algorithm is directly implied by Corollary 2 on page 75, where this is proven for the algorithm with length of explanations limited to $l$. The algorithm $\text{AAA}_S$ corresponds to the case when $l = \infty$. □

The complexity of $\text{AAA}_S$ is simply entailed by the complexity of SOA, which is run several times and the results are combined.

**Theorem 8.** The worst case complexity of the $\text{AAA}_S$ algorithm is ExpTime.

**Proof.** According to Theorem 6 the SOA algorithm is ExpTime. The $\text{AAA}_S$ algorithm calls SOA $n$ times for the input knowledge base and the observation $O = \{O_1, \ldots, O_n\}$, and in the end it computes all the combinations of the explanations obtained from each run of SOA, which is a polynomial process. So $\text{AAA}_S$ is also ExpTime. □

### 5.2.2 Reduction

Instead of splitting an ABox abduction problem $P = (\mathcal{K}, O)$, where $O = \{O_1, \ldots, O_n\}$ into subproblems $P_1 = (\mathcal{K}', O_1), \ldots, P_n = (\mathcal{K}', O_n)$ as described in the previous section, one could propose to represent a set of problems $\{O_1, \ldots, O_n\}$ as a single observation joining $O_1, \ldots, O_n$ into one observation.

Notice that, e.g. first-order logic enables to join formulae representing the observations into one observation through conjunction. But in the DL case, each $O_i$ from an ABox abduction problem $P = (\mathcal{K}, \{O_1, \ldots, O_n\})$ is an ABox assertion (i.e. it is either a concept assertion $C(a)$, or a role assertion $R(a, b)$, or a negated role assertion $\neg R(a, b)$), and it is not possible to directly join two ABox assertions
with any DL constructor into any form of DL axiom (neither ABox assertion, nor GCI, nor RIA).

Therefore we need to come up with a concept $C^R$ that will replace the whole set of observations, more precisely $\mathcal{E}$ will be an explanation of $C^R(s_0)$ for some new individual $s_0$ if and only if $\mathcal{E}$ is an explanation of $O = \{O_1, \ldots, O_n\}$. Note that, the concept $C^R$ is not equivalent to $O$ in any means, and the construction is highly dependent on a presence of nominals. The reduction is introduced in Definition 39. From now on, according to the definition, $SO(O) = C^R(s_0)$ is a reduced observation gained from a set of observations $O$.

**Definition 39** (Reduced observation). Let $O$ be a set of observations:

$$O = \{C_1(a_1), \ldots, C_n(a_n),
\quad R_1(b_1, c_1), \ldots, R_m(b_m, c_m),
\quad \neg R_{m+1}(b_{m+1}, c_{m+1}), \ldots, \neg R_k(b_k, c_k)\}.$$

The reduced observation $SO(O)$ is defined as follows:

$$SO(O) = (\neg \{a_1\} \cup C_1) \cap \cdots \cap (\neg \{a_n\} \cup C_n)
\cap (\neg \{b_1\} \cup \exists R_1(.c_1)) \cap \cdots \cap (\neg \{b_m\} \cup \exists R_m(.c_m))
\cap (\neg \{b_{m+1}\} \cup \forall R_{m+1}.\neg(c_{m+1})) \cap \cdots
\cdots \cap (\neg \{b_k\} \cup \forall R_k.\neg(c_k))(s_0).$$

Although not directly apparent, this reduction leads to a preserving the set of explanations. That is, $\mathcal{E}$ is an explanation of an ABox abduction problem $P = (\mathcal{K}, O)$ if and only if $\mathcal{E}$ is an explanation of a reduced ABox abduction problem $P' = (\mathcal{K}, SO(O))$ with the exception of explanations involving $s_0$. This fact is captured in the following lemma.

**Lemma 7.** Let $P = (\mathcal{K}, O)$ and $P' = (\mathcal{K}, SO(O))$ be ABox abduction problems. Let $\mathcal{E}$ be a set of atomic and negated atomic ABox assertions, s.t. $s_0$ does not occur in any $E \in \mathcal{E}$. Then $\mathcal{E}$ is an explanation of $P$ if and only if it is an explanation of $P'$.

**Proof.** Only-if part. By contradiction, assume that $\mathcal{E}$ is an explanation of $P$, but not of $P'$. Then $\mathcal{K} \cup \mathcal{E} \not\models SO(O)$, hence $\mathcal{K} \cup \mathcal{E} \cup \neg C^R(s_0)$ has a model $\mathcal{J}$. Note that

$$\neg C^R = (\{a_1\} \cap \neg C_1) \cup \cdots \cup (\{a_n\} \cap \neg C_n)
\cup (\{b_1\} \cap \forall R_1.\neg(c_1)) \cup \cdots \cup (\{b_m\} \cap \forall R_m.\neg(c_m))
\cup (\{b_{m+1}\} \cap \exists R_{m+1}.(c_{m+1})) \cup \cdots
\cdots \cup (\{b_k\} \cap \exists R_k.(c_k)).$$
This means that $s_0^j$ either belongs to $\{a_i \cap \neg C_i \}$ for some $i \in [1..n]$ and hence also $a_i^j \in \neg C_i^j$ and thus $a_i^j \notin C_i^j$; or $s_0^j$ belongs to $\{b_j \} \cap \forall R_j, \neg \{c_j \}$ for some $j \in [1..m]$ and hence also $b_j^0 \in \neg \exists R_j, \{c_j \}$ (because $\forall R_j, \neg \{c_j \} = \neg \exists R_j, \{c_j \}$) and thus $(b_j^0, c_j^0) \notin R_j$; or $s_0^j$ belongs to $\{b_1 \} \cup \exists R_1, \{c_1 \}$ for some $l \in [m+1..k]$ and hence also $b_l^0 \in \exists R_1, \{c_1 \}$ and thus $(b_l^0, c_l^0) \in R_1^0$. In the first case we have $J \not\models C_i(a_i)$; in the second case we have $J \not\models R_j(b_j, c_j)$; and in the third case we have $J \not\models \neg R_l(b_1, c_1)$. Either case contradicts $E$ being an explanation of $O$.

If part. By contradiction, assume that $E$ is an explanation of $\mathcal{P}'$, but not of $\mathcal{P}$. Then either $\mathcal{K} \cup \mathcal{E} \not\models C_i(a_i)$ for some $i \in [1..n]$, or $\mathcal{K} \cup \mathcal{E} \not\models R_j(b_j, c_j)$ for some $j \in [1..m]$, or $\mathcal{K} \cup \mathcal{E} \not\models \neg R_l(b_1, c_1)$ for some $l \in [m+1..k]$.

In the first case, $\mathcal{K} \cup \mathcal{E} \cup \{\neg C_i(a_i)\}$ has a model $J$. Let $J'$ be $J$ extended with $s_0^j = a_i^j$. Now, $J'$ is a model of $\mathcal{K} \cup \mathcal{E} \cup \{a_i \cap \neg C_i(s_0)\}$ and thus also of $\mathcal{K} \cup \mathcal{E} \cup \{\neg C_i^R(s_0)\}$. Thus $\mathcal{K} \cup \mathcal{E} \not\models C_i^R(s_0)$, that is, $\mathcal{K} \cup \mathcal{E} \not\models \text{SO}(O)$ which is a contradiction.

In the second case, $\mathcal{K} \cup \mathcal{E} \cup \{\neg R_j(b_j, c_j)\}$ has a model $J$. Let $J'$ be $J$ extended with $s_0^j = b_j^0$. Now, $J'$ is a model of $\mathcal{K} \cup \mathcal{E} \cup \{b_j \cap \forall R_j, \neg \{c_j \}(s_0)\}$ and thus also of $\mathcal{K} \cup \mathcal{E} \cup \{\neg C_i^R(s_0)\}$. Thus again $\mathcal{K} \cup \mathcal{E} \not\models C_i^R(s_0)$, that is, $\mathcal{K} \cup \mathcal{E} \not\models \text{SO}(O)$ which is a contradiction.

In the third case, $\mathcal{K} \cup \mathcal{E} \cup \{R_l(b_1, c_1)\}$ has a model $J$. Let $J'$ be $J$ extended with $s_0^j = b_l^0$. Now, $J'$ is a model of $\mathcal{K} \cup \mathcal{E} \cup \{b_1 \cap \exists R_1, \{c_1 \}(s_0)\}$ and thus also of $\mathcal{K} \cup \mathcal{E} \cup \{\neg C_i^R(s_0)\}$. Thus again $\mathcal{K} \cup \mathcal{E} \not\models C_i^R(s_0)$, that is, $\mathcal{K} \cup \mathcal{E} \not\models \text{SO}(O)$ which is a contradiction. $\square$

The following example shows an ABox abduction problem with the reduced observation and the solution.

**Example 20.** Let us remind the knowledge base from Example 19:

- $\exists\text{owns.Cat} \sqsubseteq \text{CatLover}$
- $\exists\text{chase.Mouse} \sqcap \text{Cat} \sqsubseteq \text{HappyCat}$
- $\text{Cat} \sqcap \text{Mouse} \sqsubseteq \text{Animal}$
- $\text{Cat} \sqsubseteq \neg\text{Mouse}$
- $\text{Cat}(\text{tom})$,

and the observations $O$:

$$O = \{\text{CatLover(\text{mammy})}, \text{HappyCat(\text{tom})}, \text{Animal(\text{jerry})}\}.$$  

Then the reduced $\text{SO}(O)$ is as follows:

$$\text{SO}(O) = \{((\neg\{\text{mammy}\} \cup \text{CatLover}) \sqcap (-\{\text{tom}\} \cup \text{HappyCat}) \sqcap (-\{\text{jerry}\} \cup \text{Animal}))\}(s_0)\).$$
The following collection $S_E$ is the solution of $P' = (K, SO(O))$ ignoring the assertions containing the individual $s_0$, and so $S_E$ is the solution of $P = (K, O)$.

$$S_E = \begin{cases} 
\{\text{owns(mammy, tom), Mouse(jerry), chase(tom, jerry)}\}, \\
\{\text{owns(mammy, tom), Mouse(mammy), chase(tom, mammy)}\}, \\
\text{Cat(jerry)} \\
\{\text{owns(mammy, tom), Mouse(mammy), chase(tom, mammy)}\}, \\
\text{Mouse(jerry)} 
\end{cases}$$

Notice that, each $E \in S_E$ explains $SO(O)$, i.e. $K \cup E \cup \{\neg SO(O)\}$, and so $K \cup E \models SO(O)$.

Let us consider an explanation $E' = \{\text{owns(mammy, tom), Mouse(s_0), chase(tom, s_0)}\}$, that explains $P' = (K, SO(O))$, but not $P = (K, O)$. There are many of these explanations containing the individual $s_0$, but they are obviously undesired, and so excluded from the output set of explanations.

In Algorithm 7 the ABox abduction algorithm based on a reduction of observations is given. An input set of observations $O$ is rewritten to a single complex concept observation $O' = SO(O)$ according to Definition 39 in line 1.

The algorithm continues in the steps very similar as in SOA (Algorithm 4). There are only two changes. The first change is an addition of line 7 and line 23, where the ABox assertions containing the individual $s_0$, established only because of the reduction, are removed from a model $M$. The reason is, not to include the ABox assertions containing this new individual $s_0$ in explanations, as illustrated in Example 20. The undesired explanations could be filtered also in the end of the algorithm, but in this way it is much more efficient.

The second change (line 15, second condition) lies in checking relevance of each $H(n)$ (i.e. potential explanation). Since we are computing the explanations for the original problem $P = (K, O)$ with a set of observations $O = \{O_1, \ldots, O_n\}$ we also need to check relevance with respect to $O$ and not with respect to the reduced observation $O'$.

In line 34 a set of all minimal, consistent, relevant, and explanatory explanations is returned. For more details, see Algorithm 4.

The algorithm $\text{AAA}_R$ is sound and complete, and always terminates.

**Theorem 9.** Let $K$ be an $\mathcal{ALCE}I\!O$ knowledge base, and let $O$ be an observation in form of a set of $\mathcal{ALCE}I\!O$ ABox assertions. The $\text{AAA}_R$ algorithm, initialized with $K$ and $O$: (a) outputs precisely all consistent, relevant, explanatory, and subset-minimal explanations of $P = (K, O)$ of the form $E \subseteq \{\lambda(a), \neg \lambda(a), r(a, b), \neg r(a, b) \mid \lambda \in N_C, r \in N_R, a, b \in N_I\}$; and (b) it eventually terminates.
Algorithm 7 AAA\textsubscript{R} (K, O): ABox Abduction Algorithm with Reduction

Require: knowledge base K, set of observations O
Ensure: set of all minimal explanations $S_E$

1: $O' \leftarrow SO(O)$  
   \hspace*{1em} $\triangleright$ $O'$ is in form $C^R(s_0)$
2: $K' \leftarrow K \cup \{\neg O'\}$
3: $M \leftarrow$ call TA with input $K'$
4: if $M = {}$ then
5: \hspace*{1em} return "nothing to explain"
6: end if
7: $M \leftarrow M \setminus \{A \mid A$ is an ABox assertion and $s_0 \in A\}$
8: $MS \leftarrow \{M\}$
9: create new HS-tree $T = (V, E, L)$ with root $r$
10: $L(r) \leftarrow \neg M$
11: for each $\sigma \in \neg M$ create a successor $n_\sigma$ of $r$ and label the resp. edge by $\sigma$
12: $S_E \leftarrow \{\}$
13: $n \leftarrow$ next node w.r.t. $r$ in $T$ by breadth-first search
14: while $n \neq$ null do
15: \hspace*{1em} if (clash in $H(n)$) 
16: \hspace*{2em} or ($n' \in V$ and $H(n') \subseteq H(n)$ and $L(n') = {}$) 
17: \hspace*{2em} or ($H(n) \cup \{\neg O_i\}$ is inconsistent for some $O_i \in O$) 
18: \hspace*{2em} or ($H(n) \cup K$ is inconsistent) then
19: \hspace*{3em} $M \leftarrow {}$
20: else if $n' \in V$ and $H(n) = H(n')$ and $L(n') \neq$ null then
21: \hspace*{3em} $M \leftarrow \times$
22: else if $N \in MS$ and $H(n) \subseteq N$ then
23: \hspace*{3em} $M \leftarrow N$
24: else
25: \hspace*{3em} $M \leftarrow$ call TA for $K' \cup H(n)$
26: \hspace*{4em} if $M = {}$ then
27: \hspace*{5em} $MS \leftarrow MS \cup \{M\}$
28: \hspace*{4em} else  
29: \hspace*{5em} $\triangleright$ the case $H(n)$ explains $O'$
30: \hspace*{5em} $S_E \leftarrow S_E \cup \{H(n)\}$
31: end if
32: end if
33: $L(n) \leftarrow \neg M$
34: for each $\sigma \in \neg M$ create a successor $n_\sigma$ of $n$ and label the resp. edge by $\sigma$
35: \hspace*{1em} $n \leftarrow$ next node in $T$ w.r.t. $n$ by breadth-first search
36: end while
37: return $S_E$
Proof. The soundness, completeness, and termination of the AAA\textsubscript{R} algorithm is directly implied by Corollary 3 on page 76, where this is proven for the algorithm with length of explanations limited to \( l \). The algorithm AAA\textsubscript{R} corresponds to the case when \( l = \infty \).

The complexity of AAA\textsubscript{R} is ExpTime, as it actually runs SOA once on a reduced input.

Theorem 10. The worst case complexity of the AAA\textsubscript{R} algorithm is ExpTime.

Proof. According to Theorem 6 the SOA algorithm is ExpTime. The AAA\textsubscript{R} algorithm reduces the input set of observations into one observation. The reduction is polynomial (in fact it is linear). For the reduced observation, SOA is called. That means, AAA\textsubscript{R} is ExpTime as well.

5.3 Limited length of explanations

The run of SOA, AAA\textsubscript{S}, and AAA\textsubscript{R} can potentially produce very complex HS-tree. Its size depends on more parameters of a knowledge base, such as number of individuals \(|N_I|\), concepts \(|N_C|\), and roles \(|N_R|\). Observe that – in the worst case, when no pruning is applied – each node-label has the size \(|N_C| \cdot |N_I| + |N_R| \cdot |N_I|^2\), which leads to a construction of \(|N_C| \cdot |N_I| + |N_R| \cdot |N_I|^2\) successors. As this holds for every node of HS-tree, the size of HS-tree grows exponentially with each level, and so in the depth \( n \), there are approximately \((|N_C| \cdot |N_I| + |N_R| \cdot |N_I|^2)^n\) of nodes.

On the other hand, SOA, AAA\textsubscript{S}, and AAA\textsubscript{R} try to find minimal explanations. Therefore, many of the desired explanations are found in one of the first levels of HS-tree. Moreover, shorter explanations may be seen as more preferred, as in some intuitive way, the simpler explanation the better. Usually, the longer paths more likely lead to non-minimality, inconsistency, or irrelevancy. Hence, we propose an approach, the explanations are computed only to certain limited length.

That is, the algorithms are terminated in the particular depth of a HS-tree. The depth is one of the input parameters of each algorithm. SOA extended with the possibility to stop in the particular depth, is given in Algorithm 8, called SOAL with the input parameters a knowledge base \( \mathcal{K} \), an observation \( O \), and a maximal length of the explanations \( l \) (i.e. \( \text{SOAL}(\mathcal{K}, O, l) \)). Notice that, the termination is assured by adding a simple condition in the while cycle in line 12. As \( H(n) \) is a set of all edge-labels on the path from the root of the HS tree to a node \( n \), and each edge-label is a distinct ABox assertion, the
cardinality of $H(n)$ precisely corresponds to the depth the node $n$ is in. Also, the construction of the tree is done w.r.t. breadth-first search. That means, while $|H(n)|$ is less or equal to the required length, the loop is executed.

Algorithm SOA$^L$ is sound and complete w.r.t. the length of explanations, and it always terminates.

Lemma 8 (Soundness). Let $\mathcal{K}$ be an ALCIO knowledge base, let $O$ be an observation in form of an ALCIO ABox assertion, and let $l \geq 1$. Let $S_\mathcal{E}$ be the output of the SOA$^L$ algorithm initialized with $\mathcal{K}$, $O$, and $l$ on the input. Then each $\mathcal{E} \in S_\mathcal{E}$ is a consistent, relevant, explanatory, and subset minimal explanation of the abduction problem $P = (\mathcal{K}, O)$.

Proof. If SOA$^L$ returned "nothing to explain", it must have terminated in line 4, and this was because $\mathcal{K} \cup \{\neg O\}$ was inconsistent, which is the same as $\mathcal{K} \models O$, and in such a case there are no explanations.

In the other case SOA$^L$ returned a set $S_\mathcal{E}$. Let $\mathcal{E} \in S_\mathcal{E}$. In such a case $E = H(n)$ for some node $n$ and it was added to $S_\mathcal{E}$ in line 24. However in this case we have also called TA on $\mathcal{K} \cup \{\neg O\} \cup \mathcal{E}$ in line 20 and it returned no model (as we tested in line 21). Hence $\mathcal{K} \cup \mathcal{E} \models O$, i.e., $\mathcal{E}$ is an explanation of $P$. In addition, $\mathcal{E}$ is consistent and relevant, because we have tested this (in line 13). It is also explanatory because in the other case the algorithm returned "nothing to explain" and terminated already in line 4 as described above.

The minimality of $\mathcal{E}$ follows from the fact that we only add such $\mathcal{E} = H(n)$ into $S_\mathcal{E}$ in line 24 which correspond to paths from root to a leaf which are not pruned in the HS-tree, and as showed by Reiter (1987), in a pruned HS-tree all such paths correspond to minimal hitting sets. This can be verified by observing the HS-tree is constructed breadth-first, that is, when $\mathcal{E} = H(n)$ is considered as an explanation, all nodes $n'$ corresponding to smaller explanations ($H(n') \subseteq H(n)$) are already labelled by $\{\}$. Consequently if there is such node, the if-condition in line 13 is evaluated as true and hence the assignment of $\mathcal{E}$ into $S_\mathcal{E}$ in line 24 is not executed.

Lemma 9 (Completeness). Let $\mathcal{K}$ be an ALCIO knowledge base, let $O$ be an observation in form of an ALCIO ABox assertion, and let $l \geq 1$. Let $\mathcal{E} \subseteq \{A(a), \neg A(a), R(a,b), \neg R(a,b) \mid A \in N_C, R \in N_R, a,b \in N_I\}$ be a consistent, relevant, explanatory, and subset minimal explanation of the abduction problem $P = (\mathcal{K}, O)$ with size $|\mathcal{E}| \leq l$. Then the SOA$^L$ algorithm, initialized with $\mathcal{K}$, $O$, and $l$ on the input, produces $\mathcal{E}$ as one of its outputs.

Proof. Given an abduction problem $P = (\mathcal{K}, O)$, let $S_\mathcal{E}$ be an output of SOA$^L$ for $P$. Let $\mathcal{E}$ be a consistent, relevant, explanatory, and subset minimal explanation of $P$. 

Algorithm 8 SOA\(^L\) (K,O,S\(_O\),l): Single Observation Abduction with Explanations of Limited Length

**Require:** knowledge base K, observation O, set of observations S\(_O\), max length of an explanation l

**Ensure:** set \(\mathcal{S}\_E\) of all minimal explanations up to length l

1. \(K' \leftarrow K \cup \neg O\)
2. M ← call TA with input \(K'\)
3. if \(M = \{\}\) then
   4. return "nothing to explain"
5. \(MS \leftarrow \{M\}\)
6. create new HS-tree \(T = (V, E, L)\) with root \(r\)
7. \(L(r) \leftarrow \neg M\)
8. for each \(\sigma \in \neg M\) create a successor \(n_\sigma\) of \(r\) and label the resp. edge by \(\sigma\)
9. \(S_E \leftarrow \{\}\)
10. \(n \leftarrow\) next node w.r.t. \(r\) in \(T\) by breadth-first search
11. while \(n \neq \text{null}\) and \(H(n) \leq l\) do
12.   if (clash in \(H(n)\)) or \((n' \in V \text{ and } H(n') \subseteq H(n) \text{ and } L(n') = \{\})\) or \((H(n) \cup \neg O_i\) is inconsistent for some \(O_i \in S_O\)) then
13.     \(M \leftarrow \{\}\)
14.   else if \(n' \in V \text{ and } H(n) = H(n') \text{ and } L(n') \neq \text{null}\) then
15.     \(M \leftarrow \times\)
16.   else if \(N \in MS \text{ and } H(n) \subseteq N\) then
17.     \(M \leftarrow N\)
18.   else
19.     \(M \leftarrow\) call TA for \(K' \cup H(n)\)
20.     if \(M \neq \{\}\) then
21.         \(MS \leftarrow MS \cup \{M\}\)
22.     else
23.         \(S_E \leftarrow S_E \cup \{H(n)\}\)
24.     end if
25. end if
26. end while
27. \(L(n) \leftarrow \neg M\)
28. for each \(\sigma \in \neg M\) create a successor \(n_\sigma\) of \(n\) and label the resp. edge by \(\sigma\)
29. \(n \leftarrow\) next node in \(T\) w.r.t. \(n\) by breadth-first search
30. end while
31. return \(S_E\)
As $\mathcal{E}$ is explanatory, $\mathcal{K} \cup \{-O\}$ has at least one model. Hence the root $r$ of HS-tree $T$ constructed by SOA$^L$ is labelled by $\neg M$, where $M$ is a model of $\mathcal{K} \cup \{-O\}$ (line 8). Note that, from the construction of $M$ (Definition 10) it follows that $\varphi \in \neg M$ or $\neg \varphi \in \neg M$ for every atomic ABox assertion $\varphi$.

It is clear that $\mathcal{K} \cup M \cup \{-O\}$ is consistent and so $\mathcal{E} \not\subseteq M$, i.e. there is an ABox assertion $\sigma_1 \in \mathcal{E}$ s.t. $\sigma_1 \not\subseteq M$. Hence $\neg \sigma_1 \in M$, and so $\sigma_1 \in \neg M$, and also $L(r,n_{\sigma_1}) = \sigma_1$ for some successor $n_{\sigma_1}$ of $r$, from line 9.

The rest of the proof is by induction. Let us assume that SOA$^L$ extended $T$ until there is a node $n_{\sigma_k}$ s.t. $H(n_{\sigma_k}) \subseteq \mathcal{E}$ and $|H(n_{\sigma_k})| = k$. We will show that (+) either $H(n_{\sigma_k}) = \mathcal{E}$ or there is some $\sigma_{k+1} \in \mathcal{E} \setminus H(n_{\sigma_k})$ which will become the label of some new edge leading from $n_{\sigma_k}$. Observe that none of the pruning conditions in line 13 applies on $n_{\sigma_k}$: $\mathcal{E}$ is consistent, and hence $H(n_{\sigma_k})$ does not contain a clash; there is no $H(n) \subseteq H(n_{\sigma_k})$ s.t. $L(n) = \{\}$, because $\mathcal{E}$ is minimal, relevant (and so no its subset can be irrelevant), and consistent (and so no its subset can be inconsistent). And if there is some other node $n$ in $T$ such that $H(n) = H(n_{\sigma_k})$ we can assume w.l.o.g. that $n_{\sigma_k}$ is the one which is visited first and hence it is not pruned in line 15.

Next we distinguish two cases. In the first case $\mathcal{K} \cup H(n_{\sigma_k}) \cup \{-O\}$ is inconsistent, and so $H(n_{\sigma_k}) = \mathcal{E}$, $L(n_{\sigma_k}) = \{\}$ and therefore SOA$^L$ adds $H(n_{\sigma_k}) = \mathcal{E}$ into $S_\mathcal{E}$. In the second case $\mathcal{K} \cup H(n_{\sigma_k}) \cup \{-O\}$ is consistent, i.e. it has a model $M_k$ and $L(n_{\sigma_k}) = \neg M_k$. It is clear that $H(n_{\sigma_k}) \subseteq M_k$ and that $\mathcal{K} \cup M_k \cup \{-O\}$ is consistent and so $\mathcal{E} \not\subseteq M_k$, i.e. there is $\sigma_{k+1} \in \mathcal{E} \setminus H(n_{\sigma_k})$ s.t. $\sigma_{k+1} \not\subseteq M_k$, and so $\sigma_{k+1} \in \neg M_k$. Therefore SOA$^L$ consequently creates a node $n_{\sigma_{k+1}}$ with $L(n_{\sigma_k}, n_{\sigma_{k+1}}) = \sigma_{k+1}$.

Since we have proved (+) for any $k$, by induction SOA$^L$ will eventually create a node $n_{\sigma_m}$ s.t. $H(n_{\sigma_m}) = \mathcal{E}$ and $L(n_{\sigma_m}) = \{\}$. That means that SOA$^L$ has added $H(n_{\sigma_m}) = \mathcal{E}$ into $S_\mathcal{E}$. Also, as we construct the HS-tree breadth-first and since $|\mathcal{E}| = |H(n_{\sigma_m})| \leq l$ and, the while loop in line 12 surely does not terminate before the node $n_{\sigma_m}$ is visited and fully processed.

Summing up the two lemmas, the algorithm is correct in the sense that finds exactly all desired explanations up to the given length limitation $l$, and that it eventually terminates. The termination follows from the observation that the construction of the HS-tree is depth-bound by $l$ and that it branches finitely, as there are only finitely many ABox assertions that may serve as labels of edges starting at any node of the tree.

**Theorem 11.** Let $\mathcal{K}$ be an $\mathcal{ALCHo}$ knowledge base, and let $O$ be an observation in form of an $\mathcal{ALCHo}$ ABox assertion. The SOA$^L$ algorithm, initialized
with $K$, $O$, and $l \geq 1$: (a) outputs precisely all consistent, relevant, explanatory, and subset-minimal explanations of $P = (K, O)$ of the form $E \subseteq \{\text{A}(a), \neg \text{A}(a), \neg \text{R}(a, b), \text{R}(a, b) | A \in NC, R \in NR, a, b \in NI\}$ such that $|E| \leq l$; and (b) it eventually terminates.

In addition, if we remove the depth limitation (i.e., we set it to $\infty$), the algorithm still terminates as the depth of the HS-tree is also bound by the number of possible ABox expressions. Thus if the depth limitation is removed, the algorithm finds all desired explanations of the input ABox abduction problem.

**Corollary 1.** Let $K$ be an $ALCHO$ knowledge base, let and $O$ be an observation in form of an $ALCHO$ ABox assertion. The $SOAL$ algorithm, initialized with $K$, $O$, and $l = \infty$: (a) outputs precisely all consistent, relevant, explanatory, and subset minimal explanations of $P = (K, O)$ of the form $E \subseteq \{\text{A}(a), \neg \text{A}(a), \neg \text{R}(a, b), \text{R}(a, b) | A \in NC, R \in NR, a, b \in NI\}$; and (b) it eventually terminates.

The complexity of the $SOAL$ algorithm is precisely the same as the complexity of the SOA algorithm. The reason is that $SOAL$ is only updated to not compute the whole HS-tree but to terminate in the particular length.

**Theorem 12.** The worst case complexity of the $SOAL$ algorithm is ExpTime.

**Proof.** According to Theorem 6 the SOA algorithm is ExpTime. As $SOAL$ is the SOA algorithm updated to compute the explanations up to the particular length, the worst case complexity is the same as for SOA, i.e. $SOAL$ is ExpTime. \hfill $\Box$

Similarly to $SOAL$, since $AAA_R$ is a simple extension of SOA, Algorithm 10 called $AAA_RL$ ($K, O, l$) has a condition of checking the depth of the tree in line 14.

An updated version of $AAA_S$ with termination in a depth of $l$ is given in Algorithm 9, called $AAA_SL$ ($K, O, l$). $AAA_SL$ simply passes the input parameter of length $l$ to $SOAL$ in line 4. But, as $SOAL$ computes all explanations with maximal length $l$ for each subproblem passed from $AAA_SL$, obviously after their composition in $AAA_SL$, also the explanations of length greater than $l$ are gained. From this point of view, as $AAA_SL$ does some overwork, $AAA_RL$ is more appropriate to solve the ABox abduction problem with an input set of observations and limited explanation length. Nonetheless, there is no guarantee, that $AAA_RL$ computes the solution in better time than $AAA_SL$. Thus, we test both approaches and state the results in Chapter 7.

Algorithm $AAA_SL$ is sound and complete w.r.t. the length of explanations, and it always terminates.
Algorithm 9 AAA$_L$($\mathcal{K},O,l$): ABox Abduction Algorithm with Explanations of Limited Length

Require: knowledge base $\mathcal{K}$, set of observations $O$, max length of an explanation $l$
Ensure: set $\mathcal{E}_{\text{min}}$ of all minimal explanations up to length $l$ (and possibly some longer)

1: $\Sigma \leftarrow \{\}$
2: $\mathcal{K}' = \mathcal{K} \cup \{T(a) \mid a \in O_l\}$
3: for all $O_i \in O$ do
4:   $\mathcal{E}_{\text{min}} \leftarrow \text{SOA}^L(\mathcal{K}, O_i, O, l)$ \triangleright set of explanations up to length $l$ for the subproblem $O_i$
5:   if $\mathcal{E}_{\text{min}} = \{\}$ then
6:     return $\{\}$
7:   else if $\mathcal{E}_{\text{min}} \neq \text{"nothing to explain"}$ then $\mathcal{K} \models O_i$ – exclude $O_i$
8:     $\Sigma \leftarrow \Sigma \cup \{\mathcal{E}_{\text{min}}\}$
9:   end if
10: end for
11: if $\Sigma = \{\}$ then
12:   return "nothing to explain"
13: else
14:   $\mathcal{E} \leftarrow \{\mathcal{E}_1 \cup \ldots \cup \mathcal{E}_m \mid \mathcal{E}_i \in \mathcal{E}_{\text{min}}, \mathcal{E}_i \in \Sigma, m = |\Sigma|\}$
15:   $\mathcal{E}_{\text{min}} \leftarrow \{\mathcal{E} \mid \mathcal{E} \in \mathcal{E} \text{ and } \forall \mathcal{E}' \in \mathcal{E} : \mathcal{E}' \notin \mathcal{E} \text{ and } \mathcal{E} \text{ is consistent and relevant}\}$
16: end if
17: return $\mathcal{E}_{\text{min}}$

Lemma 10 (Soundness). Let $\mathcal{K}$ be an ALCIO knowledge base, let $O$ be an observation in form of a set of ALCIO ABox assertions, and let $l \geq 1$. Let $\mathcal{E}$ be the output of AAA$_L$ initialized with $\mathcal{K}$, $O$, and $l$ on the input. Then each $\mathcal{E} \in \mathcal{E}$ is a consistent, relevant, explanatory, and subset minimal explanation of the abduction problem $\mathcal{P} = (\mathcal{K}, O)$.

Proof. If the algorithm returned "nothing to explain", it was because in line 11 collection of the sets of all explanations $\Sigma$ was empty. This can only be the case when SOA$^L$ returned "nothing to explain" for each $O_i$ (line 7), that is, according to Observation 3, $O' = \{\}$. This means that $\mathcal{K} \models O_i$ for each $O_i$, and so $\mathcal{K} \models O$.

In the other case the algorithm returned a set $\mathcal{E}_\Sigma$. Let $\mathcal{E} \in \mathcal{E}_\Sigma$. From lines 14–15 it is apparent that $\mathcal{E} = \mathcal{E}_1 \cup \ldots \cup \mathcal{E}_m$ where $\mathcal{E}_i \in \mathcal{E}_{\text{min}}$ and each $\mathcal{E}_i \in \Sigma$ is the set of minimal explanations for $O_i$ returned by SOA$^L$ in line 4.

From Lemma 8 we have $\mathcal{K} \cup \mathcal{E}_i \models O_i$ for all $\mathcal{E}_i \in \mathcal{E}_{\text{min}}$. Observe, that $\Sigma$ collects $\mathcal{E}_i$ for all those $O_i$, for which $\mathcal{K} \not\models O_i$ (lines 7–8), hence from Observation 3 we have $\mathcal{K} \cup \mathcal{E} \models O$, that is $\mathcal{E}$ is an explanation of $\mathcal{P} = (\mathcal{K}, O)$. Moreover, subset minimality, consistency, and relevancy of each $\mathcal{E} \in \mathcal{E}$ is consecutively verified in line 15. $\mathcal{E}$ is also explana-
tory, as otherwise $\mathcal{K} \models O$, i.e., $\mathcal{K} \models O_i$ for all $i$, and thus $\Sigma = \{\}$. In such a case the algorithm already terminates in line 12.

**Lemma 11 (Completeness).** Let $\mathcal{K}$ be an $\mathcal{ALCHO}$ knowledge base, let $O$ be an observation in form of a set of $\mathcal{ALCHO}$ ABox assertions, and let $l \geq 1$. Let $E \subseteq \{A(a), \neg A(a), R(a, b), \neg R(a, b) \mid A \in N_C, R \in N_R, a, b \in N_1\}$ be a consistent, relevant, explanatory, and subset minimal explanation of the abduction problem $P = (\mathcal{K}, O)$ with size $|E| \leq l$. Then $\text{AAA}_S^L$, initialized with $\mathcal{K}$ and $O$ on the input, produces $E$ as one of its outputs.

**Proof.** Given an abduction problem $P = (\mathcal{K}, O)$, let $S^\text{min}_E$ be the output of the algorithm $\text{AAA}_S^L$ for $P$. Let $E$ be a consistent, relevant, explanatory, and subset minimal explanation of $P$.

As $\mathcal{K} \cup E \models O = \{O_1, \ldots, O_n\}$, then also $\mathcal{K} \cup E \models O_i$ and hence $E$ explains each $P_i = (\mathcal{K}', O_i)$.

Let $E_i$ be a smallest subset of $E$ that explains $P_i$. For some $i$’s $E_i$ may equal to $\{\}$ but not for all, as then $\mathcal{K} \models O$ which is not the case. If $E_i \neq \{\}$ then $E_i$ is a minimal explanation of $P_i$, otherwise it would not be a smallest subset of $E$ that explains $P_i$. It is trivially explanatory, and it is also consistent and relevant w.r.t. $P_i$ (because whole $E$ is). Also trivially $|E_i| \leq 1$ due to $E_i \subseteq E$. In addition $E = E_1 \cup \cdots \cup E_n$ because $E_1 \cup \cdots \cup E_n$ explains all $O_1, \ldots, O_n$ hence otherwise $E$ would not be minimal.

Now, the algorithm obtained the set of explanations $S_{E_i}$ for each $P_i$ by calling $\text{SOA}^L$. From Lemma 9 we have that $S_{E_i}$ contains all minimal, consistent, relevant, and explanatory explanations of $P_i$ up to the length $l$. Hence on line 14 $E$ was surely added into $S_{E_i}$. But since $E$ is minimal, consistent, relevant, and explanatory, it was also added to $S^\text{min}_E$ on line 15.

We have already established the termination of $\text{SOA}^L$, hence it is trivial to observe that the combined algorithm also terminates on every input. Putting this and the two lemmas above together we obtain the following correctness results for the bounded and for the unbounded case.

**Theorem 13.** Let $\mathcal{K}$ be an $\mathcal{ALCHO}$ knowledge base, let and $O$ be an observation in form of a set of $\mathcal{ALCHO}$ ABox assertions. $\text{AAA}_S^L$ initialized with $\mathcal{K}$, $O$, and $l \geq 1$: (a) outputs all consistent, relevant, explanatory, and subset-minimal explanations of $P = (\mathcal{K}, O)$ of the form $E \subseteq \{A(a), \neg A(a), R(a, b), \neg R(a, b) \mid A \in N_C, R \in N_R, a, b \in N_1\}$ such that $|E| \leq l$, plus possibly some of greater length; and (b) it eventually terminates.

**Corollary 2.** Let $\mathcal{K}$ be an $\mathcal{ALCHO}$ knowledge base, let and $O$ be an observation in form of a set of $\mathcal{ALCHO}$ ABox assertions. $\text{AAA}_S^L$ initialized with $\mathcal{K}$, $O$, and $l = \infty$: (a) outputs precisely all consistent, relevant, explanatory,
and subset minimal explanations of \( \mathcal{P} = (\mathcal{K}, \mathcal{O}) \) of the form \( \mathcal{E} \subseteq \{ A(a), 
abla A(a), R(a, b), \nabla R(a, b) \mid A \in \mathcal{N}_C, R \in \mathcal{N}_R, a, b \in \mathcal{N}_I \} \); and (b) it eventually terminates.

The complexity of the AAA\(_S\) algorithm is ExpTime, as it is actually the AAA\(_S\) algorithm that additionally enables to terminate in the particular depth of the HS-tree.

**Theorem 14.** The worst case complexity of the AAA\(_S\) algorithm is ExpTime.

**Proof.** According to Theorem 8 the AAA\(_S\) algorithm is ExpTime. As AAA\(_S\) is the AAA\(_S\) algorithm updated to compute the explanations up to the particular length, the worst case complexity is the same as for AAA\(_S\), i.e. AAA\(_S\) is ExpTime.

Algorithm AAA\(_R\) is sound and complete w.r.t. the length of explanations, and it always terminates.

**Theorem 15.** Let \( \mathcal{K} \) be an ALCHO knowledge base, let and \( \mathcal{O} \) be an observation in form of a set of ALCHO ABox assertions. AAA\(_R\) initialized with \( \mathcal{K}, \mathcal{O} \), and \( l \geq 1 \): (a) outputs precisely all consistent, relevant, explanatory, and subset-minimal explanations of \( \mathcal{P} = (\mathcal{K}, \mathcal{O}) \) of the form \( \mathcal{E} \subseteq \{ A(a), 
abla A(a), R(a, b), \nabla R(a, b) \mid A \in \mathcal{N}_C, R \in \mathcal{N}_R, a, b \in \mathcal{N}_I \} \) such that \( |\mathcal{E}| \leq l \); and (b) it eventually terminates.

The theorem is a direct consequence of Lemma 7 and the construction of Algorithm 7 AAA\(_R\) which calls SOA\(_L\) with an observation reduced into a single assertion, and features only minor modifications whose only effect is filtering out assertions involving the individual \( s_0 \) from the candidate explanations.

**Corollary 3.** Let \( \mathcal{K} \) be an ALCHO knowledge base, let and \( \mathcal{O} \) be an observation in form of a set of ALCHO ABox assertions. AAA\(_R\) initialized with \( \mathcal{K}, \mathcal{O} \), and \( l = \infty \): (a) outputs precisely all consistent, relevant, explanatory, and subset minimal explanations of \( \mathcal{P} = (\mathcal{K}, \mathcal{O}) \) of the form \( \mathcal{E} \subseteq \{ A(a), 
abla A(a), R(a, b), \nabla R(a, b) \mid A \in \mathcal{N}_C, R \in \mathcal{N}_R, a, b \in \mathcal{N}_I \} \); and (b) it eventually terminates.

The complexity of the AAA\(_R\) algorithm is ExpTime, as it is actually the AAA\(_R\) algorithm that additionally enables to terminate in the particular depth of the HS-tree.

**Theorem 16.** The worst case complexity of the AAA\(_R\) algorithm is ExpTime.

**Proof.** According to Theorem 10 the AAA\(_R\) algorithm is ExpTime. As AAA\(_R\) is the AAA\(_R\) algorithm updated to compute the explanations up to the particular length, the worst case complexity is the same as for AAA\(_R\), i.e. AAA\(_R\) is ExpTime.
Algorithm 10 AABR^L (K, O, l): ABox Abduction Algorithm with Reduction and Explanations of Limited Length

**Require:** knowledge base $K$, set of observations $O$, max length of an explanation $l$

**Ensure:** set $\mathcal{E}$ of all minimal explanations up to length $l$

1: $O' \leftarrow SO(O)$
2: $K' \leftarrow K \cup \{\neg O'\}$
3: $M \leftarrow$ call TA with input $K'$
4: if $M = \emptyset$ then
5:   return "nothing to explain"
6: end if
7: $M \leftarrow M \setminus \{A \mid A$ is an ABox assertion and $s_0 \in A\}$
8: $MS \leftarrow \{M\}$
9: create new HS-tree $T = (V, E, L)$ with root $r$
10: $L(r) \leftarrow \neg M$
11: for each $\sigma \in \neg M$ create a successor $n_\sigma$ of $r$ and label the resp. edge by $\sigma$
12: $S_E \leftarrow \{\}$
13: $n \leftarrow$ next node w.r.t. $r$ in $T$ by breadth-first search
14: while $n \neq \text{null}$ and $H(n) \leq l$ do
15:   if (clash in $H(n)$) or $(n' \in V$ and $H(n') \subseteq H(n)$ and $L(n') = \emptyset$) or $(H(n) \cup \{\neg O_i\}$ is inconsistent for some $O_i \in O$ or $(H(n) \cup K$ is inconsistent) then
16:     $M \leftarrow \emptyset$
17:   else if $n' \in V$ and $H(n) = H(n')$ and $L(n') \neq \text{null}$ then
18:     $M \leftarrow \times$
19:   else if $N \in MS$ and $H(n) \subseteq N$ then
20:     $M \leftarrow N$
21:   else
22:     $M \leftarrow$ call TA for $K' \cup H(n)$
23:     $M \leftarrow M \setminus \{A \mid A$ is an ABox assertion and $s_0 \in A\}$
24:   if $M \neq \emptyset$ then
25:     $MS \leftarrow MS \cup \{M\}$
26:   else
27:     $S_E \leftarrow S_E \cup \{H(n)\}$
28: end if
29: end if
30: $L(n) \leftarrow \neg M$
31: for each $\sigma \in \neg M$ create a successor $n_\sigma$ of $n$ and label the resp. edge by $\sigma$
32: $n \leftarrow$ next node in $T$ w.r.t. $n$ by breadth-first search
33: end while
34: return $S_E$
IMPLEMENTATION

Our algorithm is implemented in Java. It is based on our implementations described in previous works (Pukancová & Homola 2016, 2017). Knowledge base consistency is verified by the Pellet reasoner (Sirin et al. 2007) (version 2.3.1). Pellet is an optimized tableau-based reasoner.

Current version of the implementation corresponds to Algorithm 9 and Algorithm 10, where one of the input parameters from the user is, if the computation should use reduction (i.e. Algorithm 10 is called) or not (i.e. Algorithm 9 is called). To sum up, the implementation solves an ABox abduction problem \( P = (\mathcal{K}, O) \), where \( \mathcal{K} \) is a knowledge base in a form of OWL ontology and \( O \) is a set of ABox assertions in any form (including complex concept assertion, role assertion, negated role assertion). Besides an input OWL ontology and a set of observations, the required maximal length of explanations \( l \) is entered, where \( l \) is a positive integer. The output of the implementation is the set of all explanations \( E \) for the problem \( P \) with length up to \( l \) (i.e. \( |E| \leq l \)). In case of Algorithm 9 also some explanations with greater length are included, as explained in Section 5.3. Each explanation is in a form of a set of atomic ABox assertions.

We will now briefly describe the most important implementation details. Firstly, OWL ontology and a set of ABox assertions representing the observations are loaded as input. If the reduction is chosen by the user, the set of observations is transformed into a single complex concept assertion, while a new unique individual is generated according to Definition 39. In case of splitting into subproblems, the following process is executed for each observation in a cycle. An observation \( O \) and the ontology are passed to Pellet, and Pellet constructs a knowledge base \( \mathcal{K} \) from the ontology. If the knowledge base \( \mathcal{K} \) is inconsistent, the run is terminated, as there is no reason to explain an observation w.r.t. inconsistent knowledge base. If \( \mathcal{K} \) is consistent, then it is extended with the negated observation \( \neg O \), i.e. \( \mathcal{K}' = \mathcal{K} \cup \{\neg O\} \) and again, consistency is checked. If \( \mathcal{K}' \) is inconsistent, then the observation follows from the ontology \( (\mathcal{K} \models O) \), and the run is terminated with the announcement „nothing to explain”.

After a successful consistency check, we obtain the ABox \( A \) from \( \mathcal{K}' = (T, R, A) \) using the getABox() method and construct \( M \), that actually corresponds to the ABox encoding of the model of \( \mathcal{K}' \) found by Pellet. Particularly, through the getNodes() method all individu-
als from $A$ are obtained. Consequently, we obtain all the concepts an individual belongs to (using the method `getTypes()`) and all its relations an individual is in (through `getOutEdges()`). Next, we compute the completion of $M$ by adding the negation of all atomic assertions which are not already in $M$.

Our implementation uses Pellet for consistency checking and for model construction. All other features, from initializing the algorithm through constructing the HS-tree to answering the set of all minimal hitting sets for the input set of observations, are executed by our own implementation. All optimizations presented in this thesis, such as HS-tree pruning or model reusing, are implemented.

Our implementation is available for download at: \url{http://dai.fmph.uniba.sk/~pukancova/aaa/}.

6.1 Usage

Let us explain the input for the implemented algorithm. To run the algorithm the following input parameters need to be defined:

- `-i` input file
- `-out` output file
- `-uri` URI for the individuals and concepts in the observation
- `-obs` observation in the form `<individual1>:<concept1>;<...;<individualn>:<concepn>;<ind1,ind2>:<role1>,...,<indm,indp>:<rolem`
- `-d` maximum length of an explanation
- `-l` allow loops in relations amongst explanations -- OPTIONAL
- `-r` solve the problem using reduction -- OPTIONAL

The first parameter `-i` defines a path to the input ontology. According to the requirements of Pellet reasoner, a format of the ontology must be RDF/XML. The next parameter is output file (`-out`), where the explanations, and statistic informations about the ontology and the computation are written. It is necessary to define the prefix uri (`-uri`) for the individuals, concepts, and roles in the observation.

A set of observations is stated after `-obs` parameter as a list of ABox assertions separated by the “;” character. The following syntax is required for the concept assertions:

- `not(C)` for any concept C (syntax in DL: $\neg C$
- `and(C,D)` for any concepts C,D (syntax in DL: $C \cap D$
- `or(C,D)` for any concepts C,D (syntax in DL: $C \cup D$
- `some(R,C)` for any role R and any concept C (syntax in DL: $\exists R.C$
- `all(R,C)` for any role R and any concept C (syntax in DL: $\forall R.C$)
In case of a role assertion, the input is either $a, b: R$ or $a, b: \text{not}(R)$ for any individuals $a, b$ and a role $R$.

After an observation (-obs), the depth of HS-tree – the maximal length of the explanations follows (-d).

Finally, there is an optional parameter determining, whether the reduction should be used (-r). By adding a parameter -l, the loops on relations would be allowed. In other words, a reflexive assertion can occur in an explanation. That means, the explanations in form $R(a, a)$ would be allowed.

### 6.2 Example of Input and Output

Let us show an example of usage. We want the algorithm to compute the explanations for the observations jack:Employee;a:Publication over the ontology LUBM.owl. More specifically, we are looking for the explanations up to length 2, we would like the algorithm to use the reduction approach, and we do not want reflexive relations amongst explanations. We run the following command:

```
java -jar AAA.jar -i LUBM.owl -uri "http://swat.cse.lehigh.edu/onto/univ-bench.owl" -out "LUBM-2-r.txt" -obs "jack:Employee; a:Publication" -d 2 -r
```

The output file contains three parts. The first part sums up the input configuration and describes the parameters of the input ontology. Particularly, the following information about the inputs and about the LUBM ontology are included:

Reduction: true
Avoid loops: true
Observation: jack:Employee;a:Publication
-----------------ONTOLOGY DETAILS-----------------
Number of concepts: 43
Number of roles: 25
Number of individuals: 3
Number of TBox axioms: 45
Number of ABox assertions: 3

The second part describes the parameters of the computation, more precisely of the HS-tree constructed by our algorithm:

-----------------MHS DETAILS-----------------
Depth of HS-tree: 2
Number of nodes: 18633
Number of TA calls: 6341
Number of reused models: 3272
Number of pruned nodes: 9020
Number of explanations: 320
The third part contains the output in the form of a list of explanations. Each explanation is printed as a set of ABox assertions. For our example input, the following explanations are listed (note that, these are only the first six of them):

All explanations: {{a: JournalArticle, jack: Lecturer},
{a: some(publicationDate,{jack}), jack: SystemsStaff},
{a: some(publicationResearch,{jack}), jack: ResearchAssistant},
{a: some(softwareDocumentation,{jack}), jack: Lecturer},
{a: some(softwareVersion,{jack}), jack: some(teacherOf,{a}}),
{a: some(softwareVersion,{jack}), a: some(advisor,{jack}}),
...

Note that, an assertion a: some(R,b) corresponds to a complex concept ABox assertion $\exists R.b(a)$, which is precisely the same as a role ABox assertion $R(a,b)$. 
To evaluate our approach, we have run the implementation of AAA with different input ontologies. The main goal lies on the one hand in studying properties of the computation according to different ontologies of different sizes, on the other hand in measuring the time of the run depending on length of the explanations.

In this chapter, let us refer to the overall algorithm as AAA. Note that, for each problem the respective algorithm is called according to the form of an observation (SOA for a single observation and AAAS or AAAR for a multiple observation). Additionally, we do not consider the algorithm without the restriction on the length of explanations, because the maximal length of explanations is always inputed in our evaluation.

All experiments were done on a 6-core 3.2 GHz AMD Phenom™ II X6 1090T Processor, 8 GB RAM, running Ubuntu 17.10, Linux 4.13.0, while the maximum Java heap size was set to 4GB. We have used the GNU time utility to measure the CPU time consumed by AAA while running in user mode, summed over all threads.

7.1 Methodology

Our evaluation method is split into two main experiments. In the first experiment we have a single observation as input and in the second experiment a set of observations is an input. In both cases, the following properties are recorded:

Properties of the knowledge base:

- number of concepts,
- number of roles,
- number of individuals,
- number of axioms (including TBox axiom and ABox assertions),

Properties of the computation:

- depth of HS-tree,
- avoiding/allowing the loops,
- splitting approach/reduction approach (in case of the multiple observation experiment)
• number of the nodes in the HS-tree,
• number of TA calls,
• number of pruned nodes,
• number of reused models,
• number of explanations,
• time of execution.

As we described in Section 5.3, to get the explanations of the particular length, the algorithm can be simply terminated in the particular depth of the HS-tree. As AAA usually does not compute whole HS-tree in acceptable time, we ran AAA iteratively with rising maximal length of the explanations.

Another, in many applications very meaningful restriction, is to forbid reflexive assertions in explanations. Many times a reflexive assertion, i.e. \( R(a, a) \) for any role \( R \) and any individual \( a \), is not a desired explanation. As these role assertions make the search space bigger, excluding them from the search space is another parameter of the experiment. We have also compared the results with respect to this parameter. From now on, we will refer to reflexive assertions as loops.

In our work, we are focused on optimality. The base for our proposal and implementation is MHS algorithm with pruning. As the number of the nodes in a HS-tree grows exponentially with each level of the tree, the optimization by pruning useless branches of the tree is very important. Since calling TA is also very expensive, another optimization lies in reusing already computed models. Thus we record the number of the nodes for which TA was called, of the nodes for which a model was reused, and of the nodes that were pruned. Consequently we compute the proportion of these nodes.

Three ontologies described in the next section were chosen for the evaluation. To obtain as much objective results as possible, we firstly ran AAA for each ontology and single selected observation ten times (iteratively for the depths from one to five) disallowing loops. We compute the average values and the standard deviation (expressed in the percentage value). As the deviation represents the distance between the values from the individual experiments, as closer to zero the more uniform the individual results are. If the deviation is low enough, also one experiment would be sufficiently representative.

In case of sufficiently low deviation from the previously described part of evaluation, similar experiment but with the loops is executed for each ontology and an observation only once.

Analogously as for the single observations, an evaluation for multiple observations is conducted. Firstly, we focus on experiments with-
out allowing loops. AAA is processed with each of the three ontologies and a set of observations (consisting of the three ABox assertions), iteratively for the depths from one to three. These inputs do not allow computation in bigger depths, as in such cases the algorithm usually runs out of memory (which is allocated to 4GB).

To sum up, we deal with the following classes of experiments for each ontology:

1. single observation, no loops, depth 1–5 (10 runs),
2. single observation, loops, depth 1–5 (1 run),
3. multiple observation, no loops, depth 1–3 (10 runs),
4. multiple observation, loops, depth 1–3 (1 run).

From now on, by the experiment setting we mean the combination of the ontology, the observation, the maximal length of explanations (the maximal depth of the HS-tree), the avoiding or allowing loops, and in the case of multiple observations the particular approach.

7.2 DATA SET

For the empirical evaluation, a set of different ontologies were chosen. First of them is LUBM – the Lehigh University Benchmark (Guo et al. 2005). LUBM is currently a popular benchmark ontology used in the DL research area. This ontology was also used by Du et al. (2011). The second ontology is Coffee ontology, which was published by Carlos Mendes on Github*.

For the purpose to evaluate our algorithm AAA, we developed Family ontology. It covers the simple structure of family relations. Compared to LUBM and Coffee ontology, Family ontology is smaller, hence is suitable for testing inputs, that cause memory exceeding in combination with LUBM and Coffee ontology. All three ontologies are available at http://dai.fmph.uniba.sk/~pukancova/aaa/ont/.

From now on, by the size of an ontology we mean the size in means of number of axioms, concepts, roles, and individuals. That is, Coffee ontology is the biggest ontology amongst all ontologies in our data set, as we can see according to the ontology parameters listed in Table 8.

7.3 SINGLE OBSERVATION EXPERIMENT

Our first experiment is for a single observation as an input. It is split into two experiments – one with avoiding loops and one with accept-
ing loops. Firstly, we focus on the experiment with avoiding loops. In Figure 4 (a) the execution times for each ontology and each maximal depth of the HS-tree from one to five are showed. Note that, the plot is shown in a logarithmic scale.

To understand how much the times from the individual runs actually differ, we calculated the standard deviations in percentage for each ontology and its execution time, as listed in Table 9.

The parameters of the HS-trees for the experiment with single observation without loops are shown in Table 13. The values are obtained from 10 experiments as the average value, and so also the standard deviation in percentage is stated.

<table>
<thead>
<tr>
<th>Depth</th>
<th>LUBM</th>
<th>Coffee</th>
<th>Family</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>avg</td>
<td>dev</td>
<td>avg</td>
</tr>
<tr>
<td>1</td>
<td>7.490</td>
<td>1.41</td>
<td>5.940</td>
</tr>
<tr>
<td>2</td>
<td>23.588</td>
<td>5.93</td>
<td>62.272</td>
</tr>
<tr>
<td>3</td>
<td>106.472</td>
<td>1.89</td>
<td>175.877</td>
</tr>
<tr>
<td>4</td>
<td>1139.938</td>
<td>0.47</td>
<td>645.671</td>
</tr>
<tr>
<td>5</td>
<td>28897.682</td>
<td>1.01</td>
<td>2878.653</td>
</tr>
</tbody>
</table>

Table 9: Average execution times in seconds (avg) and deviations in % (dev)
The proportion of the nodes for which TA was called, the nodes for which a model was reused, and the nodes that were pruned plotted against the particular depth is shown in Figure 5. Note that, the sum of these nodes gives us the total number of all nodes created by AAA for each depth, but not all nodes that would be generated without pruning.

The experiment with allowing loops was conducted for each experiment setting only once. The execution times are pictured in the plot in Figure 4 (b). The recorded properties of the constructed HS-trees for the ontologies are shown in Table 14. Note that, in case when no numbers are stated, AAA exceeded memory. In Figure 5 (b), the proportion of the pruned nodes, the nodes labelled by reused models, and the nodes for which a consistency check was done (TA calls) are pictured.

7.4 DIFFERENT SAMPLES OF SINGLE OBSERVATIONS

The values obtained from experiments with different samples of observations are shown in Figure 6. For each ontology, experiments with 5 different single observations were conducted. The maximal depth was set to 3. The respective observations with the number of explanations obtained in the depth 3 are shown in Table 10. The indices of the observations in the table correspond to the the indices in the legends in the figure. In the case of LUBM ontology, sample 4, and depth 3, the memory was exceeded.

7.5 MULTIPLE OBSERVATION EXPERIMENT

Similarly as in the single observation experiment, experiments for multiple observations are executed in two phases – without loops and with accepting loops. Let us start with the experiments without loops.

For each ontology, AAA is executed with a set of observations 10 times. Moreover, each experiment is conducted two times, once using the splitting approach (the AAA$^L_S$ algorithm), and once using the reduction approach (the AAA$^L_R$ algorithm). The average execution times for the particular depths and for both approaches are plotted in Figure 7. In the legend, S stands for the splitting approach, and R stands for the reduction approach.

For multiple observations, the depths only range between 1 and 3, as even by the depth of 3, in cases of LUBM ontology and Coffee ontology the memory exceeded (in both, the splitting and the reduction approach). Observe that, in all cases, the time grows exponentially.
Figure 5: Proportion of pruned nodes, reused models and TA calls from all HS-tree nodes in AAA for a single observation: (a) without and (b) with loops.
Figure 6: Execution times for different samples of single observations.
Table 10: Different samples of observations for single observation experiment

Figure 7: Depth vs. time for multiple observations: (a) without and (b) with loops
Table 11: Average execution times in seconds (avg) and deviations in % (dev) for multiple observations (splitting approach)

<table>
<thead>
<tr>
<th>Depth</th>
<th>LUBM avg</th>
<th>dev</th>
<th>Coffee avg</th>
<th>dev</th>
<th>Family avg</th>
<th>dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>32.078</td>
<td>5.80</td>
<td>23.461</td>
<td>1.73</td>
<td>4.542</td>
<td>1.37</td>
</tr>
<tr>
<td>2</td>
<td>421.210</td>
<td>1.60</td>
<td>826.740</td>
<td>1.08</td>
<td>21.783</td>
<td>1.29</td>
</tr>
<tr>
<td>3</td>
<td>–</td>
<td>–</td>
<td>332177.08</td>
<td>5.74</td>
<td>1169017</td>
<td>5.74</td>
</tr>
</tbody>
</table>

Table 12: Average execution times in seconds (avg) and deviations in % (dev) for multiple observations (reduction approach)

<table>
<thead>
<tr>
<th>Depth</th>
<th>LUBM avg</th>
<th>dev</th>
<th>Coffee avg</th>
<th>dev</th>
<th>Family avg</th>
<th>dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9.111</td>
<td>1.12</td>
<td>11.853</td>
<td>1.24</td>
<td>2.773</td>
<td>1.64</td>
</tr>
<tr>
<td>2</td>
<td>229.168</td>
<td>0.68</td>
<td>503.723</td>
<td>0.69</td>
<td>13.628</td>
<td>1.10</td>
</tr>
<tr>
<td>3</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>79.679</td>
<td>4.74</td>
</tr>
</tbody>
</table>

Again, the times in Figure 7 are computed as the average values from 10 values. Table 11 and Table 12 contain the standard deviations in percentage for each ontology, while the first table contains the data for the splitting approach and the second for the reduction approach. In cases when average time was not computed, only the time when AAA exceeds memory, the deviation can not be computed, and so no value is stated. The case of Coffee ontology and the splitting approach is an exception, as in this case the memory was not exceeded, but because of a high overall time (almost four days) we choose not to repeat this run multiple times. Therefore no deviation is recorded.

The recorded parameters of the HS-trees are listed in Table 15, where again the average values together with their deviations are stated (unless AAA exceeds memory). As stated above, in case of Coffee ontology and the splitting approach there are no deviations because this experiment was only run once. For each ontology, explanations were computed through both approaches; in the table S stands for the splitting approach and R for the reduction approach.

In Figure 8, the proportion of the pruned nodes, the nodes labelled by reused models and the nodes for which a consistency check was done (TA calls) are pictured.

In the end, the same experiment was executed with loops only once because the deviations in case of the experiment without loops were sufficiently low (Tables 11,12). The execution times are plotted in Figure 7, the parameter of the HS-trees are shown in Table 16, and finally, the proportion of the pruned nodes, the reused models, and TA calls for each experiment setting is pictured in Figure 9.
Figure 8: Proportion of pruned nodes, reused models and TA calls from all HS-tree nodes in AAA for a multiple observation without loops

Figure 9: Proportion of pruned nodes, reused models and TA calls from all HS-tree nodes in AAA for a multiple observation with loops
7.6 RESULTS

In this section, we will focus on the results from the evaluation. We will point out the most important characteristics of the results. Each of the following sections deals with some interesting feature.

7.6.1 Exponential Growth of Time

In all experiments, the expectation about exponential growth of time with the growth of the depth of the HS-tree (in other words, with the growth of the maximal length of explanations) was confirmed (see Figures 4, 7). Whilst the single observation experiment was processed up to the depth of 5, the multiple observation experiment was processed only up to the depth of 3, as even in this depth the algorithm ran out of memory in case of two of three ontologies (both without and with loops). On the other hand, the search up to the length of 1 or 2 takes a fraction of the time required for greater lengths, and as we will discuss below, usually even for the length of 1 quite high number of explanations is found, which could be sufficient in many applications.

In regards to runs without and with loops, the time for experiment with loops is almost always higher than the one without loops (for the particular experiment setting). This is implied by the fact that the search space including also the reflexive assertions is bigger than the one without them. The only exception is in case of the LUBM ontology and the depth 3, when the time for an experiment with loops was lower.

The experiment comparing the times with different samples of single observations was conducted for all three ontologies, for the maximal depths 1–3 and without loops (see Figure 6). Our hypothesis was that there will be no significant differences between the times for different samples for the same ontology. This hypothesis was mostly confirmed. The only significant difference was shown by observations in form of role assertions. In these cases, the time was much higher than the times for observations in other forms. A likely reason is that in case of role observations two new individuals are involved instead of just one. This increases the search space, and thus the execution time. Note that, in case of LUBM and a role observation, the algorithm exceeded the memory.

The following section focus on the comparison of our two approaches, including also the comparison in regards to time.
7.6.2 Comparison of the Approaches

In Figure 7 we can see, that the time is always higher for the splitting approach than for the reduction approach. Let us again ignore the results from the experiments where the memory was exceeded. Note that, for the particular experiment with a set of observations of the size $n$, the search space in the splitting approach is approximately $n$ times bigger than the search space in the reduction approach. Hence naturally, the time is also higher.

On the other hand, in case of the splitting approach, a big number of solutions are yielded even for the depth 1. Note that, this is caused by the character of the splitting approach, where the length limitation is applied for each subproblem separately. Thus, also longer explanations are computed (as compositions of the separate results obtained for each subproblem).

With respect to the theoretical point of view, the reduction approach is the reliable one, as for the maximal length $l$ it always assures all the explanations up to $l$ and in our experiments it always reaches lower time than the splitting approach for the same maximal length $l$ (Figure 7). On the other hand, we observed from the experiments that in most cases a high number of the explanations was obtained through the splitting approach already for the maximal length 1 (Tables 15,16).

However we cannot state the precise class of the obtained explanations regarding the maximal length. It is not assured that the solution contains all the explanations, not even all the explanations of any length higher than $l$ (as it is complete only for $l$). Despite this, we assume this approach to be suitable for applications, where completeness is not a high priority, and the main requirement is to compute as much explanations as possible in the shortest possible time.

For example, let us focus on the multiple observation experiment without loops, specifically on Family ontology. In Table 15 we can see that the number of explanations for depth 1 and splitting approach is already 7. Whilst the number of explanations in the reduction approach for depths 1 and 2 is 0, only for the depth 3 the first explanations are found – all 9 explanations of length 3. Focus now on Figure 7 or Tables 11,12 where the respective times can be compared. More precisely, the time for the splitting approach and the depth 1 is 4.542 s, while the time for the reduction approach for the depth 3 is 79.679 s. Note that, although both recorded times are still short, the difference is quite big and may be significant for bigger ontologies.

Let us consider a rather theoretical case when each observation from the input set of observations is independent. In other words, each observation does not have any common or even partly overlap-
ping explanation with some other observation. Then for the desired explanations of the maximal length \( l \), the splitting approach could be simply run to the depth \( \lceil l/n \rceil \), where \( n \) is the number of observations. Of course, this cannot be stated in general, but only in cases when the independence is assured. For example, there can be an application where a set of \( n \) independent observations is given while each observation needs to be explained by one separate assertion. By running the splitting approach into the depth 1, explanations with maximal length \( n \) are obtained from a set of separate explanations of the length 1 of each observation.

### 7.6.3 Number of the Explanations

The above described results support the importance of the restriction for the length of explanations discussed in Section 5.3. Moreover, focus on Tables 13, 14, 15, 16, where the numbers of the explanations for the particular problem with respect to the particular maximal length of explanations are shown. Since the numbers of explanations are quite high already in the small depths, the computation of even more explanations (of a greater length) may not be desired.

Specifically, from the single observation experiment without loops (Table 13) we see that in all cases the number of all explanations stabilizes at the latest at the maximal length 2. We cannot make a general conclusion, but in the results we have recorded that each explanation contains at most two ABox assertions, and so there is no need to compute the whole HS-tree. Similarly, in the single observation experiment with loops (Table 14), the number of explanations stabilizes at the latest at the length of 3. Of course, these results may vary according to the different characters of the ontologies (e.g. when an ontology captures not only hierarchy between atomic classes, but also more complex relations, the number of longer explanations may rise).

In results of the multiple observation experiment (Table 15, Table 16) the stabilization of the number of explanations is harder to follow, since half of the experiments gave us the results only up to the depth of 2. On the other hand, as we already discussed, the number of the explanations is so high, that from the practical point of view it does not make sense to compute bigger solutions.

### 7.6.4 Optimization of the HS-tree Search

As mentioned many times before, this thesis is focused on ABox abduction algorithm with respect to the optimization techniques in the minimal hitting set algorithm. In the experiments, we have recorded
also the proportion of the nodes for which TA was called, the nodes for which a model was reused, and the nodes that were pruned. Focus on the Figures 5, 8, 9. In all experiment settings, the proportion of the pruned nodes tends to rise with the increasing depth of the HS-tree. On the other hand, the proportion of the reused models tends not to change with the increasing depth of the HS-tree.

We consider the decreasing proportion of the TA calls to be a very positive result. With (potentially) exponential growth of the nodes, it is very important to minimize the number of TA calls as much as possible. The reason is that TA is a very expansive procedure.

In Tables 13, 14, 15, 16 the numbers of nodes for the respective maximal depth in each experiment setting are shown. Regarding the exponential growth of the nodes, it is important to point out, that in these tables we can observe how the HS-tree pruning minimizes the exponential growth of the nodes with the increasing depth of the HS-tree.
<table>
<thead>
<tr>
<th>Ontology</th>
<th>Observation</th>
<th>Depth</th>
<th>Nodes</th>
<th>$d_{nodes}$</th>
<th>TA calls</th>
<th>$d_{TA}$</th>
<th>Reused models</th>
<th>$d_{models}$</th>
<th>Pruned nodes</th>
<th>$d_{pruned}$</th>
<th>Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUBM</td>
<td>Person(jack)</td>
<td>1</td>
<td>44.00</td>
<td>0.00</td>
<td>39.00</td>
<td>0.00</td>
<td>5.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>990.00</td>
<td>0.00</td>
<td>244.10</td>
<td>0.12</td>
<td>74.90</td>
<td>0.40</td>
<td>671.00</td>
<td>0.00</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>10923.00</td>
<td>0.00</td>
<td>1581.20</td>
<td>0.06</td>
<td>1049.20</td>
<td>0.28</td>
<td>8292.60</td>
<td>0.04</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>84018.70</td>
<td>0.19</td>
<td>9901.70</td>
<td>1.10</td>
<td>8023.30</td>
<td>1.65</td>
<td>66093.70</td>
<td>0.23</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>518495.00</td>
<td>0.42</td>
<td>50401.70</td>
<td>0.58</td>
<td>57770.10</td>
<td>0.96</td>
<td>410323.20</td>
<td>0.34</td>
<td>20</td>
</tr>
<tr>
<td>Coffee</td>
<td>Macchiato(a)</td>
<td>1</td>
<td>42.00</td>
<td>0.00</td>
<td>32.80</td>
<td>1.22</td>
<td>9.20</td>
<td>4.35</td>
<td>0.00</td>
<td>0.00</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>1600.00</td>
<td>0.00</td>
<td>1297.50</td>
<td>0.37</td>
<td>108.50</td>
<td>4.39</td>
<td>194.00</td>
<td>0.00</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>7627.00</td>
<td>0.00</td>
<td>5929.00</td>
<td>0.24</td>
<td>607.70</td>
<td>1.59</td>
<td>1090.30</td>
<td>0.76</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>30997.00</td>
<td>2.01</td>
<td>23507.70</td>
<td>1.56</td>
<td>2482.70</td>
<td>5.92</td>
<td>5006.60</td>
<td>2.59</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>108519.80</td>
<td>4.26</td>
<td>81649.90</td>
<td>4.07</td>
<td>7783.90</td>
<td>6.11</td>
<td>19086.00</td>
<td>4.47</td>
<td>2</td>
</tr>
<tr>
<td>Family</td>
<td>Mother(jane)</td>
<td>1</td>
<td>9.00</td>
<td>0.00</td>
<td>6.00</td>
<td>0.00</td>
<td>3.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>57.00</td>
<td>0.00</td>
<td>32.00</td>
<td>0.00</td>
<td>12.00</td>
<td>0.00</td>
<td>13.00</td>
<td>0.00</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>137.00</td>
<td>0.00</td>
<td>71.00</td>
<td>0.00</td>
<td>16.00</td>
<td>0.00</td>
<td>50.00</td>
<td>0.00</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>177.00</td>
<td>0.00</td>
<td>86.00</td>
<td>0.00</td>
<td>18.00</td>
<td>0.00</td>
<td>73.00</td>
<td>0.00</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>193.00</td>
<td>0.00</td>
<td>91.00</td>
<td>0.00</td>
<td>18.00</td>
<td>0.00</td>
<td>84.00</td>
<td>0.00</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 13: Parameters of HS-trees for AAA without loops
<table>
<thead>
<tr>
<th>Ontology</th>
<th>Observation</th>
<th>Depth</th>
<th>Nodes</th>
<th>TA calls</th>
<th>Reused models</th>
<th>Pruned nodes</th>
<th>Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUBM</td>
<td>Person(jack)</td>
<td>1</td>
<td>69</td>
<td>62</td>
<td>7</td>
<td>0</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>2245</td>
<td>454</td>
<td>188</td>
<td>1603</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>36245</td>
<td>4324</td>
<td>3432</td>
<td>28489</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>414869</td>
<td>38878</td>
<td>39557</td>
<td>336434</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Coffee</td>
<td>Macchiato(α)</td>
<td>1</td>
<td>48</td>
<td>41</td>
<td>7</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>1928</td>
<td>1516</td>
<td>137</td>
<td>275</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>12550</td>
<td>9673</td>
<td>909</td>
<td>1968</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>59456</td>
<td>44681</td>
<td>4697</td>
<td>10078</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>256762</td>
<td>190168</td>
<td>19778</td>
<td>46816</td>
<td>2</td>
</tr>
<tr>
<td>Family</td>
<td>Mother(jane)</td>
<td>1</td>
<td>10</td>
<td>6</td>
<td>4</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>73</td>
<td>37</td>
<td>15</td>
<td>21</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>208</td>
<td>92</td>
<td>33</td>
<td>83</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>379</td>
<td>149</td>
<td>40</td>
<td>190</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>442</td>
<td>167</td>
<td>40</td>
<td>235</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 14: Parameters of HS-trees for AAA with loops
| Ontology  | Observation                  | Approach | Depth | Nodes | \(d_{\text{nodes}}\) | TA calls | \(d_{\text{TA}}\) | Reused
|-----------|------------------------------|----------|-------|-------|---------------------|----------|-----------------| models
|           |                              |          |       |       |                     |          | Pruned
|           |                              |          |       |       |                     |          | nodes | \(d_{\text{pruned}}\) | models
|           |                              |          |       |       |                     |          | Explanations |       |
| LUBM      | \{ Person(jack), Employee(jack), Publication(a) \} | S        | 1     | 411.00| 0.00    | 342.10   | 0.50  | 68.90 | 2.47 | 0.00 | 0.00 | 320     |
|           |                              |          | 2     | 39756.30| 8.46  | 13346.00 | 18.34 | 6445.90 | 25.19 | 19964.40 | 0.78 | 320     |
|           |                              |          | 3     | -     | -       | -       | -     | -     | -    | -    | -    | -      |
| R         |                              |          | 1     | 137.00| 0.00    | 111.00   | 0.00  | 26.00 | 0.00 | 0.00 | 0.00 | 0      |
|           |                              |          | 2     | 18633.00| 0.00  | 6462.40  | 0.37  | 3149.80 | 0.49  | 9020.80 | 0.10 | 320     |
|           |                              |          | 3     | -     | -       | -       | -     | -     | -    | -    | -    | -      |
| Coffee    | \{ Milk(a), Coffee(b), Pure(c) \} | S        | 1     | 480.00| 0.00    | 393.00   | 0.00  | 87.00 | 0.00 | 0.00 | 0.00 | 12     |
|           |                              |          | 2     | 41303.00| 0.30  | 30176.80 | 0.36  | 6529.50 | 1.59  | 4696.70 | 1.10 | 12     |
|           |                              |          | 3     | 2052996| -      | 1393400  | -     | 193125 | -     | 466471 | -    | -      |
| R         |                              |          | 1     | 160.00| 0.00    | 129.00   | 0.00  | 31.00 | 0.00 | 0.00 | 0.00 | 0      |
|           |                              |          | 2     | 25441.00| 0.00  | 12928.30 | 0.35  | 3461.60 | 1.25  | 9051.10 | 0.04 | 0      |
|           |                              |          | 3     | -     | -       | -       | -     | -     | -    | -    | -    | -      |
| Family    | \{ Father(jack), Mother(eva), Person(fred) \} | S        | 1     | 93.00 | 0.00    | 54.00    | 0.00  | 39.00 | 0.00 | 0.00 | 0.00 | 7      |
|           |                              |          | 2     | 1545.00| 0.00   | 588.00   | 0.00  | 546.00 | 0.00  | 411.00 | 0.00 | 144    |
|           |                              |          | 3     | 17072.60| 1.48  | 5710.60  | 0.75  | 5149.40 | 2.21  | 6212.60 | 1.88 | 813    |
| R         |                              |          | 1     | 31.00 | 0.00    | 17.00    | 0.00  | 14.00 | 0.00 | 0.00 | 0.00 | 0      |
|           |                              |          | 2     | 931.00| 0.00    | 235.00   | 0.00  | 335.00 | 0.00  | 361.00 | 0.00 | 0      |
|           |                              |          | 3     | 13951.00| 0.00  | 3665.00  | 0.00  | 4382.00 | 0.00  | 5904.00 | 0.00 | 9      |

Table 15: Parameters of HS-trees for AAA for multiple observations without loops
<table>
<thead>
<tr>
<th>Ontology</th>
<th>Observation</th>
<th>Approach</th>
<th>Depth</th>
<th>Nodes</th>
<th>TA calls</th>
<th>Reused models</th>
<th>Pruned nodes</th>
<th>Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUBM</td>
<td>{Person(\textit{jack}), Employee(\textit{jack}), Publication(\textit{a})}</td>
<td>S</td>
<td>1</td>
<td>561</td>
<td>488</td>
<td>73</td>
<td>0</td>
<td>550</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>71521</td>
<td>26533</td>
<td>10251</td>
<td>34737</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td>R</td>
<td>1</td>
<td>187</td>
<td>158</td>
<td>29</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>34783</td>
<td>12723</td>
<td>5093</td>
<td>16967</td>
<td>598</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Coffee</td>
<td>{Milk(\textit{a}), Coffee(\textit{b}), Pure(\textit{c})}</td>
<td>S</td>
<td>1</td>
<td>534</td>
<td>448</td>
<td>86</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>48129</td>
<td>35816</td>
<td>6660</td>
<td>5653</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>2536664</td>
<td>1721600</td>
<td>218995</td>
<td>596069</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td></td>
<td>R</td>
<td>1</td>
<td>178</td>
<td>150</td>
<td>28</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>29383</td>
<td>15601</td>
<td>3740</td>
<td>10042</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Family</td>
<td>{Father(\textit{jack}), Mother(\textit{eva}), Person(\textit{fred})}</td>
<td>S</td>
<td>1</td>
<td>102</td>
<td>67</td>
<td>35</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>1872</td>
<td>764</td>
<td>586</td>
<td>522</td>
<td>225</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>23810</td>
<td>7338</td>
<td>6888</td>
<td>9584</td>
<td>1404</td>
</tr>
<tr>
<td></td>
<td></td>
<td>R</td>
<td>1</td>
<td>34</td>
<td>20</td>
<td>14</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>1123</td>
<td>274</td>
<td>391</td>
<td>458</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>18481</td>
<td>4366</td>
<td>5800</td>
<td>8315</td>
<td>9</td>
</tr>
</tbody>
</table>

Table 16: Parameters of HS-trees for AAA for multiple observations with loops
In Section 3.3 we have reviewed existing approaches to ABox abduction. In this chapter, we will discuss the differences with our approach. Particularly, we have presented a direct approach based on existing reasoning techniques for DLs and Reiter’s minimal hitting set algorithm. The distinctive contribution of our work is that at the same time we provide the proofs of soundness and completeness of our algorithm, the implementation, and empirical evaluation. Additionally, we focus on the optimal HS-tree construction involving Reiter’s pruning proposals, and some additional techniques.

8.1 Direct Approaches

Our proposal builds on the work of Halland & Britz (2012b,a). Both proposals, ours and theirs, are based on the minimal hitting set algorithm (Reiter 1987) and the tableau algorithm for description logics. Our approach is both sound and complete. While the approach of Halland and Britz is purely theoretical, we also provide an implementation and empirical evaluation.

On the one hand fully exploits Reiter’s optimization techniques, on the other hand it extends his proposal with some additional optimizations relevant for our case. On the contrary, Halland and Britz mention the Reiter’s algorithm as a black box in their proposal, and did not give much attention to the details.

Moreover, compared to Halland and Britz we avoid to compute all models of the knowledge base in preprocessing. Instead of this expensive process, we compute the models on the fly, during the construction of the HS-tree. Additionally, each time an already computed model can be reused, it is reused and calling the tableau algorithm is spared. Also, the restrictions for explanations (consistency, relevance, and minimality) are taken into account during the HS-tree search, and the undesired explanations are filtered out earlier than in the end of the algorithm, that also leads to a reduction of unnecessary branches. Last but not least, Halland and Britz proposed an algorithm for the description logic $\mathcal{ALC}$, whilst we proposed and developed an algorithm for the more expressive description logic $\mathcal{ALCHO}$.

Ma et al. (2012) proposed an approach to ABox abduction over $\mathcal{ALC}$. They based their proposal on the work of Klarman et al. (2011), but proposed to avoid the translation into another formalism and in-
stead extend the tableau algorithm for DL. They did not prove soundness or completeness, and they do not mention any implementation, or empirical evaluation.

Among other known direct approaches, Kaya et al. (2007) and Peraldi et al. (2008) propose an ABox abduction method for DL knowledge bases enriched with DL-safe rules. They describe an implementation, which is integrated into the Racer reasoner. However, they do not provide any evaluation, and in addition their algorithm deals with a rather specific application, namely multimedia interpretation.

8.2 TRANSLATION-BASED APPROACHES

These approaches are based on translation to a different formalism. The proposal of Klarman et al. (2011) is based on the translation to a theory in the modal logic and to a theory in the first order logic. The main contribution lies in the extending known inference methods for the mentioned formalism with finding the explanations. Although soundness and completeness were proven, termination is assured only for a specific class of explanations. They also provide no implementation and no empirical evaluation.

Du et al. (2011, 2012) proposed an ABox abduction algorithm based on the translation to the logic programming. They work on top of the SHIQ DL. Their approach is sound, but it is complete only w.r.t. a restricted case of the Horn fragment of SHIQ with some additional restrictions. They have also implemented the algorithm and provide rather extensive evaluation. In the experiments they only worked with single observations while we have also considered multiple observations.
CONCLUSIONS

In this thesis, we have focused on ABox abduction over description logics. We have proposed a sound and complete algorithm for the description logic $\mathcal{ALC}H(\Delta)$, based on the minimal hitting set algorithm and the tableau algorithm for description logics. Our algorithm features a number of optimizations in the minimal hitting set algorithm. We have also provided an implementation and an empirical evaluation. The implementation exploits the Pellet reasoner, which belongs to the number of efficient reasoners for description logics.

Our approach addressed the issues spread among the current solutions – to our best knowledge, no other approach have in the same time proposed, developed, implemented, and empirically evaluated a sound and complete direct ABox abduction algorithm for the class of observations of any ABox assertions and the class of explanations of any atomic or negated atomic assertions for the description logic $\mathcal{ALC}H(\Delta)$.

The empirical evaluation of our algorithm was provided for three ontologies in six experiments. First we tested single observations without and with reflexive assertions amongst the explanations (i.e. the $\text{SOA}^L$ algorithm was evaluated). Then we tested multiple observations without and with reflexive assertions amongst the explanations through both approaches – the splitting approach (i.e. the $\text{AAA}_S^L$ algorithm was evaluated) and the reduction approach (i.e. the $\text{AAA}_R^L$ algorithm was evaluated).

A particular strength of our proposal is in its completeness w.r.t. any given length of explanations. It means, when the algorithm runs up to the depth $l$, it assures to find all the explanations with the maximal length $l$. Usually, a high number of explanations were found already for a low maximal length of explanations. We assume, that in many such cases computing the explanations with higher maximal length may not even be desired, as the number of explanations would simply be too high. Moreover, minimal explanations are generally considered as preferred one. The algorithm finds the explanations iteratively with increasing the length, so naturally most of the minimal explanations are found in the first steps.

The evaluation supported the necessity of the limitation on the length of explanations also regarding to execution time and search space. In a case of single observations, the maximal length was set to 5, in a case of the multiple observations, the maximal length was set
to 3. The size of the search space grows exponentially with increasing the length of explanations. Thus, also the time grows exponentially, which was showed by the evaluation. The Java heap space memory was exceeded already for the length 3 in half of the multiple observations experiments. However, our results from the evaluation also show how the search space is reduced thanks to the optimizations in the minimal hitting set algorithm.

For the future work, we plan to extend our algorithm for more expressive description logics. The class of explanations is currently restricted to atomic and negated atomic ABox assertions, and so we would like to consider also complex assertions. As in such a case the search space would be infinite, this class of explanations needs to be restricted in some way. One of the most common ways to restrict explanations in abduction is to define abducibles, i.e. a set of assertions, that are potential explanations. We assume this would allow to operate with the size of the search space, and so more interesting experiments can be conducted. The restriction on abducibles seems to be important also from the point of view of practical use, as indicated also by our evaluation. In our opinion, this restriction is a realistic constraint that can help to boost effectivity as many times the user can rule out a number of uninteresting assertions with respect to the desired explanations.

Currently, our approach computes the minimal explanations regarding to subset minimality. We plan to consider also other types of preferences amongst explanations, such as semantic minimality. Semantic minimality compares explanations with respect to entailment, namely one explanation should not imply the other. We would like to implement also this restriction on our class of explanations, as we consider this to be an interesting extension.

We would also like to exploit other optimization techniques. We will investigate the implemented optimization techniques amongst existing effective reasoners for description logics. One of the most relevant optimization techniques for our algorithm is incremental reasoning (Kazakov & Klinov 2013, Cuenca Grau et al. 2010). Its relevance lies in the fact that our algorithm works with the input knowledge base extended with the observation, and the consistency is checked iteratively after adding and removing assertions. The reuse of the existing tableau is non-trivial (mainly in case of the assertion removing) and special techniques need to be applied. Incremental reasoning deals exactly with this problem.

Our implementation can be extended with other reasoners for description logics by exploiting OWL API (Horridge & Bechhofer 2011). Consecutively, an evaluation with the focus on the comparison of the particular reasoners can be conducted.
Interesting for our work is also to propose an ABox abduction algorithm for less expressive description logics. The lower expressivity is sufficient in many applications, and also highly efficient reasoners are implemented for these description logics, e.g. the ELK reasoner (Kazakov et al. 2014). Our algorithm can be also exploited with these reasoners.

From the technical point of view, the splitting approach, in which an ABox abduction problem with an observation \( O = \{O_1, \ldots, O_n\} \) into \( n \) is split into independent subproblems, opens space for parallelization. The solution for each \( O_i \) can be computed in parallel.
BIBLIOGRAPHY


