Semantically minimal ABox abduction in Description Logics

Merle Beaujon

Master's thesis Artificial Intelligence
Daily supervisor: ir. Michael van Bekkum (TNO)
Project supervisor: prof. dr. mr. Henry Prakken (Utrecht University)
Second examiner: prof. dr. Pinar Yolum Birbil (Utrecht University)
September 11, 2019

Abstract

Abduction is a reasoning method that can be used to derive explanations for unexpected observations. This paper aims to use abduction in Description Logics (DL) to find ABox assertions that can explain observations described in ABox assertions. This form of abduction is called ABox abduction and can be useful for domains where statistical inference is not possible, or not preferred. Several implementations of ABox abduction reasoners have already been built, however, not many implementations can select semantically minimal explanations. Therefore this paper is aimed at researching methods to find semantically minimal explanations in the DL $\mathcal{ALCHO}$, i.e. it is aimed at finding explanations that do not explain more than is necessary.

This study presents two algorithms: SEMAR, which is an algorithm that searches for semantically minimal explanations by making adjustments to a traditional Tableau Algorithm (TA), and the SMC algorithm, which is an algorithm that selects semantically minimal explanations from a set of found explanations. SEMAR can potentially find explanations more effectively than other implementations. However, further research is needed to develop a correct algorithm and a working implementation. This study provides a proof of correctness and a working implementation for the SMC algorithm. Empirical tests show that the implementation of the SMC algorithm is successful in selecting semantically minimal explanations from a set, and the implemented optimizing techniques have a positive effect on the performance.

Acknowledgements

First of all I would like to thank Michael van Bekkum for giving me the opportunity to research this topic at TNO. You always made sure I kept track of the bigger picture and did not get lost in this theoretical topic. Furthermore, I would like to thank Henry Prakken for his sharp attention to detail, your advise helped me to be precise.

Thank you to the whole Connected Business team of TNO, for all the advise, the philosophical discussions, and for tolerating my use of all the whiteboards. These past nine months you have welcomed me at TNO and were always willing to help me with any difficulties and doubts I encountered. Writing this thesis would have been a lot harder without the table-tennis matches, pub-quizzes and game nights!

Lastly, I would like to thank all my friends and family for supporting me. Every time I had to explain my thesis topic I could practice my examples and summaries, so thank you for asking! In special I would like to thank Michaela, Coen and Nils for getting me to this point and Hendrik for keeping me calm in during stressful times.

Contents

List of Algorithms

List of Figures

List of Tables

1 | Introduction

The ability to reason is one of the unique abilities of human intelligence. Pearl and Mackenzie (2018) even claim that the ability to question why is the main reason for the existence of our science- and technology based society. Reasoning can be used to understand what the effect of a certain action will be, but also to hypothesize about the causes of a certain observed effect. The act of generating hypotheses for the possible causes of a given effect can be described as *abductive reasoning* (Peirce 1878). While abductive reasoning is a rather unknown term, the following examples show that humans do reason abductively on a daily basis. When a weather man goes outside and sees that his lawn is wet, he usually assumes that is has rained, when a taxi driver is in a traffic jam and sees police cars racing by, he usually assumes that an accident has happened down the road, and when a dog owner comes home and finds a box in shreds, she usually assumes that her dog has chewed up the box. In all these situations the subject has reasoned about what the best explanation would be for the observations he or she has made.

Many scholars have tried to formally define the process of finding the best explanation to an observation (Lipton 2003; Aliseda-Llera 1997). With the increasing availability of data Bayesian networks entered as an effective method of finding the most probable explanation for a detected effect (Pearl and Mackenzie 2018). To use a Bayesian network, knowledge about the conditional probabilities of the system that one is reasoning about are needed. For domains that have enough data to calculate probabilities for each cause of an observed event, Bayesian networks are very powerful to find probable explanations for that observed effect. However, for domains that lack extensive data it can be hard, or impossible, to calculate conditional probabilities. In cases where calculating conditional probabilities is not feasible, Bayesian networks cannot

be used to search for explanations. Moreover, for calculating conditional probabilities based on frequencies of events happening, one needs to accept a (partial) closed-world assumption. For example, calculating conditional probabilities of every probable cause of a crime that could happen would be difficult, if not impossible. For such cases it could be helpful to abduct over a logical system that describes the knowledge of the given domain in order to retrieve plausible hypotheses (Elsenbroich et al. 2006).

For domains that require a lot of expert knowledge, it is interesting to express that knowledge in a Description Logic (DL). DLs are a family of logics that describe the structure of a domain, and form the basis of most ontologies. The connected business team of TNO implements ontologies for domains that could benefit from this organisational structure. These ontologies contain knowledge about the structure and relations within such a domain. While users can query the ontologies with the existing reasoners, such as HermiT, FaCT, Pellet and JFact, to retrieve knowledge that currently is present in the ontology, or can be deducted from that knowledge, it is impossible to use the knowledge that such an ontology holds to reason about surprising observations that have occurred. For example in an ontology that is designed to keep track of the status of an information system with the use of log files, it is possible to deduct that a system is not working properly due to the information in the log files. However, if it is observed that a system has crashed, while there is no information in the present log files that can be used to explain the crash, abduction can help to hypothesize about what information is missing to explain the crash. According to Elsenbroich et al. (2006) abduction over DLs can be split into four main forms of abduction; inferring concepts that are subsumed by a given concept (*concept abduction*), inferring instances that would entail a given assertion (*ABox abduction*), inferring meta-subsumptions that would entail a given subsumption (*TBox abduction*), and a generalization of ABox and TBox abduction (*Knowledge base abduction*). Especially ABox abduction is useful to detect missing information about instances, rather than about the ontology itself.

This paper aims to present an implementation of an ABox abduction algorithm, based on previous research. The implementation should be able to generate all assertions which serve as a useful explanation for an observed assertion. This paper is focused on investigating an

implementation that can find a minimal set of explanations regardless of the situation, such that future extensions can select the best explanations from that set by taking the situation at hand into account. To minimize the set of explanations, without having specific knowledge about the domain, every explanation should adhere to several constraints, one of which is described as semantic minimality. A semantically minimal solution is a solution such that it explains no other valid solution. This constraint ensures that an explanation never assumes more than is necessary. While several implementations have been built to search for explanations to an ABox abduction problem (Du, Wang, et al. 2014; Pukancová and Homola 2015; Pukancová and Homola 2017; Mrózek et al. 2018; Del-Pinto and Schmidt 2018), none of the implementations succeeded to generate semantically minimal explanations for both concept and role assertions. Therefore, this paper is specifically aimed at improving the existing implementations, such that only semantically minimal solutions are generated.

The subsequent chapter provides an overview of the existing literature on abduction and an in depth review of previous research on abduction in Description Logics. Besides, the chapter provides a short explanation of Description Logics and the relevant terms. After a base of the theory is established two different approaches to finding semantically minimal solutions are described in chapter 3. From the two approaches two algorithms are developed: *SEMAR* (section 3.1) and *SMC* (section 3.2). Both algorithms are evaluated, and a full proof of correctness and worst case complexity for SMC is provided end of chapter 3. SEMAR is only introduced as a theoretical algorithm, no working implementation of the algorithm could be built. Section 3.2.2 describes a working implementation of SMC and introduces a method for testing the performance of the implementation. The results of these performance tests are described in chapter 4. Lastly, chapter 5 summarizes the findings of this study and describes some limitations and ideas for future research.

2 | Background Theory

While the number of ABox abduction implementations is small, a lot of research has been conducted on abductive reasoning over other, more classic, logical systems. The next section provides a historical overview of the development of abductive reasoning in any logical system. After this introduction into abductive reasoning section 2.2 describes the relevant theory behind Description Logics. When basic theory about abduction and Description Logics is established, existing research on DL abduction, specifically ABox abduction, is discussed in section 2.3. From the existing ABox abduction methods the goal of this thesis is introduced and an overview of the possible additions is presented.

2.1 Abductive Reasoning

2.1.1 Philosophical background

The first description of abduction as a logical inference was presented by Pierce. In 1878 Pierce classified human reasoning into three types of reasoning: deduction, induction and abduction. While Pierce describes how three forms of reasoning form a natural triangle, many other philosophers do not agree there is a clear distinction between the different forms of reasoning (Mill 2011; Campos 2011). Still, a vast amount of the literature on abduction leads back to Pierce's work, thus understanding abduction starts by understanding Pierce's notion of it. Following Pierce's line of reasoning, *deduction* can be described as inferring new, concrete knowledge as a consequence of a known rule and a cause. This acquired knowledge is logically

entailed by the knowledge that one already has, thus must be true. Inductive reasoning on the other hand is concerned with inferring new, probable rules from observed samples. *Inductive reasoning* can be seen as learning the relations between cause and effect, based on the available observations. The third mode of reasoning, *abductive reasoning*, is used to hypothesize about the cause of an observed effect, by using given rules. Let us look at an example, based on the bean bag example of Aliseda-Llera (1997), to illustrate the differences between the three forms of reasoning. Imagine a box, called box 1, filled with coloured marbles. If we know a rule that "All marbles in box 1 are red", and we have a case where a marble is from box one, the result of the deduction must be that the drawn marble is red (example 1). In the case of induction the rule is unknown, but we have multiple cases where we know that the marbles are from box 1, and that those marbles are red. From this information it is probable that the rule about all marbles in box 1 being red holds (example 2). In the last example, the case of abduction (example 3), there is knowledge about the rule "All marbles in box 1 are red", and we know the result: the found marble is red. From this information one can abduct that the drawn marble might be from box 1.

Example 1. Deduction	**Example 2.** Induction	**Example 3.** Abduction
All marbles in box 1 are red,	These marbles are from box 1,	All marbles in box 1 are red,
This marble is from box 1	These marbles are red	This marble is red
This marble is red	All marbles in box 1 are red	This marble is from box 1

From this example some important differences between the forms or reasoning can be observed. Particularity important is that deduction leads to new information that can be accepted as true knowledge, while induction and abduction produce new knowledge that *can* be true. The marble in example 1 has to be red, at least if the two premises are considered to be true. In example 2, box 1 might contain both blue and red marbles even if both the premises are true. Just by chance, or because all the red marbles lay on top, all the drawn marbles where red. Pierce describes induction as a reasoning method both used for producing and validating rules. Abduction, on the other hand, merely presents a suggestion of what may have happened. In example 3, based on the given knowledge, it is certainly a possibility that the marble is from

box 1. However, there is no need to believe that box 1 is the only place with red marbles, there can be other boxes that contain red marbles as well. Another technique to recognize abduction is to take a look at the timeline of the line of reasoning. Abduction is mainly used when one makes a surprising observation and then reasons what has happened for this observation to occur. Deduction is used to predict what will happen in the (near) future if a premise is true, while the use of induction tries to explain how something currently works. An important note to remember is that abduction can also be used to reason about a past in the future, if a certain result is wanted in the future, and one reasons what has to happen in order to obtain this result, abductive reasoning is used.

Furthermore, induction is used to generalize given information, while deduction and abduction try to reason about specific cases. Peirce (1878) considers abduction as the only stage of reasoning to introduce truly new ideas, while deduction and induction predict and verify these ideas. This connects with the idea that abduction is the process of forming an explanatory hypothesis. However, Pierce also describes abduction as the process of choosing a hypothesis, which connects to selection rather than creating new ideas. This duality in his description is one of the reasons why other scholars were confused with Pierce's classification. Instead of using the term abduction, Harman (1965) introduced the concept of inference to the best explanation, which combines different types of reasoning to infer the best explanation for a given observation.

Harman (1965) introduced *inference to the best explanation* (IBE) as a term, as he sees abduction as a misleading term. He describes IBE as the process of forming the best explanation to an observation, by eliminating all other possible hypothesis. This form of IBE is strongly focused on selecting good hypotheses, instead of generating new, creative hypotheses. The concept of IBE has been revised by many scholars in an attempt to clarify what would be the best explanation (Lipton 2003). Barnes (1995) argued for differentiating between *inference to the loveliest explanation* and *inference to the likeliest explanations*. He described that while most scholars are concerned with finding an explanation that is the likeliest to occur in a given situation, there are explanations that are less likely to occur, but provide a better understanding about the situation. He described the search for explanations that would present the best description

of a situation as inference to the loveliest explanation, and the search for an explanation that is most likely to have occurred inference to the likeliest explanation. It is not the aim of this paper to refine the concept of abduction, however it is important to understand that different forms of reasoning can be intertwined to retrieve valid hypotheses to an observation. Readers interested in differences between Pierce's abduction and IBE are advised to read the discussion of Campos (2011) and the extensive background of Aliseda-Llera (1997).

While the ultimate goal of building an abductive reasoner for ontologies is to find the best explanation for observations, this paper focuses on the generation of all plausible hypotheses. When we refer to abduction as a process we want to describe the process to obtain a set of all plausible hypotheses. While some general constraints can be applied to limit the set of hypotheses, no ranking, or selection, of better hypotheses is incorporated. Abduction in this sense can be seen as the first phase of IBE, while both deduction and induction can be used to select the best explanation from the set of hypotheses. The result of abduction is considered to be a set of *hypotheses* for what has happened. In the body of this text a hypothesis might also be described as a possible *explanation* for the *observation*.

2.1.2 Significance of Abduction

As discussed in the previous section, abduction is a subject that many philosophers have investigated, yet why is it relevant to formalize abduction? By formalizing abduction in such a way that it can be used to reason over logical systems any system can create explanations for surprising observations. To give a few examples, this can be helpful for doctors to find possible diseases based on a patients symptoms, for detectives to find an explanation for what has happened based on forensic clues, or even for debugging to find what is missing from a piece of code in order to get the desired result.

One might argue that Bayesian networks are suited to take on this job, as they do not only calculate possible explanations, but inherently rank their explanations based on the conditional probabilities (Pearl 2009). The rise of Bayesian networks has indeed decreased the research into abduction significantly since the nineties. However, Bayesian networks are not the ultimate

solution for any case, as conditional probabilities are needed for them to work. Furthermore, they are focused on the chances of events happening concurrently, instead of the semantic value of an explanation. In the medical field this might not be a blocking issue, as most logical systems in medical research are based on use cases and thus have reliable probabilities. In the domain of law these probabilities are much harder, if not impossible, to decide on. Since the essay of Elsenbroich et al. (2006) research into abduction, specifically abduction in Description Logics, has increased. This paper continues to research abduction in formal logical systems, to develop a reasoner that can be used for domains that do not have the means to calculate or determine conditional probabilities.

The reason TNO started to look into abductive inference was to increase traceability for information systems. By building an abductive reasoner, the ontologies built by TNO can be used to hypothesize about what has caused a system failure, with the ontology providing the necessary background knowledge about the domain, yet missing the information of what has crashed the system. An abductive reasoner can be used on any domain that does not have the means to calculate conditional probabilities.

2.1.3 Abduction as Logical Inference

Since the introduction of abduction as a logical inference by Peirce (1878) many techniques to use abduction in logical systems are developed (Lucas 1997; Aliseda-Llera 1997). Many of the techniques focus on propositional logical systems. While the techniques differ on how correct hypotheses can be derived, there is general consensus on the desired result (Mayer and Pirri 1993; Lucas 1997; Aliseda-Llera 1997; Reiter 1987). Consider a background theory Γ, consisting of set of rules and premises written in the logical language $\mathcal{L}$. Given an observation Φ and the background theory Γ, a set of hypotheses $\mathcal{H}$ can be defined. The hypotheses may also be referred to as possible explanations, or *abducibles*. The observation Φ can be a set of different elements which are all observed, for example, "the grass is green" and "the sky is blue" are two elements of the observation "the grass is green and the sky is blue". Every explanation $\mathcal{E}$ has to entail the complete observation. Thus each element of the observation, when added to the

background theory, i.e. $\Gamma \cup \mathcal{E} \vDash \Phi$ for every $\mathcal{E} \subseteq \mathcal{H}$.

While only checking if the possible explanation joined by the background theory entails an observation results in explanations for the observation, there are several other constraints that are relatively arbitrary to add. Firstly, an explanation is *explanatory* when the observation is not entailed by the background theory itself. When the observation exists of multiple elements, the observation is not entailed by the background theory if not all elements of the observation are entailed by the background theory. Without the explanatory constraint anything can be a valid explanation, as the observation is already entailed. This constraint does not necessarily say anything about the explanation itself, thus the observation can be checked without even considering explanations. Definition 2.1 describes the constraints for an *abduction problem*.

Definition 2.1 (Abduction problem)**.** *Let Γ be a theory, $\mathcal{O}$ the set of all elements that can possibly be observed and $\Phi \subseteq \mathcal{O}$ a set of observed elements, dubbed the observation. A tuple $\langle \Gamma, \Phi \rangle$ can be described as a explanatory abduction problem if and only if:*

1. $\Gamma \nvDash \Phi$,
2. $\Gamma \cup \Phi \nvDash \bot$.

Secondly, an explanation together with the background theory should be *consistent*. To understand this constraint one should know that once there is an inconsistency in a logical system everything is entailed. Therefore, simply adding an explanation that is inconsistent with respect to the background theory would entail any observation. While it does entail the observation, such an explanation would not be useful to explain the observation. The third constraint, relevancy, is implemented to ensure that the explanation actually does provide some new information. An explanation is *relevant* when the explanation on itself does not entail the observation. An explanation that is not relevant does not use any previous knowledge, thus does not provide any new information. Definition 2.2 includes all three constraints to describe a useful explanation to an abduction problem.

Definition 2.2 (Abduction explanation)**.** *Let $\Phi \subseteq \mathcal{O}$ be an abduction problem, with theory Γ and observation $\Phi \subseteq \mathcal{O}$. A hypothesis $\mathcal{E} \subseteq \mathcal{H}$ can be described as an explanation of Φ with*

respect to Γ *if and only if:*

1. $\Gamma \cup \mathcal{E} \models \Phi$ *(plain)*
2. $\Gamma \not\models \Phi$ *(explanatory)*
3. $\Gamma \cup \mathcal{E} \not\models \bot$ *(consistent)*
4. $\mathcal{E} \not\models \Phi$ *(relevant)*

If an explanation does entail the observation, but does not respect the other constraints in definition 2.2, this explanation is said to be plain. While there are methods that only guarantee plain explanations, explanations that are not relevant, consistent or explanatory would not be accepted as a sensible explanations when processed by a human user. A fourth constraint that is not necessary to make sensible explanations, but does help to limit the set of hypotheses is *minimality*. Minimal explanations can be split into syntactically and semantically minimal explanations. First syntactically minimal explanations are defined.

Definition 2.3 (Syntactic minimality)**.** *Let* $\langle \Gamma, \Phi \rangle$ *be an abduction query, consisting of theory* Γ *and observation* $\Phi \subseteq \mathcal{O}$*, and* $\mathcal{E}, \mathcal{E}' \subseteq \mathcal{H}$ *be explanations to* $\langle \Gamma, \Phi \rangle$*.* $\mathcal{E}$ *is said to be syntactically smaller than* $\mathcal{E}'$ *if* $\mathcal{E} \subseteq \mathcal{E}'$*. If there is no other explanation to* $\langle \Gamma, \Phi \rangle$ *that is smaller than* $\mathcal{E}$ *, then* $\mathcal{E}$ *is said to be syntactically minimal.*

Syntactic minimality removes redundant information from the explanations. The explanations that are not syntactically minimal contain more information than needed. By removing these explanations from the set of hypotheses, uncovering the true cause for why a certain observation has occurred becomes easier. In contrast, semantic minimality limits the set of hypotheses to explanations that make no more assumptions than necessary to explain the observation. This means that the ultimate explanation can be excluded from the set of semantically minimal hypotheses. However, as the semantically minimal hypotheses will be entailed by the semantically non-minimal explanations, the ultimate explanation can be derived by an iterative application of abduction, with the semantically minimal explanations as new observations. Furthermore, iteratively searching for explanations to a semantically minimal explanation can form a chain of explanations, which does not only provide the original explanation, but gives insights on

how this explanation leads to the observation, by providing intermediate explanations. Let us first define semantic minimality, before proceeding with an example to illustrate the use of the different constraints.

Definition 2.4 (Semantic minimality)**.** *Let* $\langle \Gamma, \Phi \rangle$ *be an abduction query, consisting of theory* Γ *and observation* $\Phi \subseteq \mathcal{O}$*, and* $\mathcal{E}, \mathcal{E}' \subseteq \mathcal{H}$ *be explanations to* $\langle \Gamma, \Phi \rangle$*.* $\mathcal{E}$ *is said to be semantically stronger than* $\mathcal{E}'$ *(denoted by* $\mathcal{E} \preceq_\Gamma \mathcal{E}'$*) iff* $\Gamma \cup \mathcal{E} \vDash \mathcal{E}'$*. The explanation* $\mathcal{E}$ *is said to be semantically minimal if there is no other explanation* $\mathcal{E}'$ *to* $\langle \Gamma, \Phi \rangle$ *such that* $\mathcal{E} \preceq_\Gamma \mathcal{E}'$

To illustrate the formally defined constraint a familiar example is presented, based on the example of Pearl (2009).

Example 4. Given an observation $\Phi = \{\mathsf{grass\_wet}\}$ and a propositional theory Γ with axioms $\{\mathsf{grass\_green}\}$ and rules:

$$\mathsf{rain} \rightarrow \mathsf{grass\_wet}$$
$$\mathsf{sprinklers} \rightarrow \mathsf{grass\_wet}$$
$$\mathsf{temperature\_high} \rightarrow \mathsf{sprinklers}$$

What are are the plain explanations? According to definition 2.2 the plain explanations for Φ, w.r.t. Γ are:

$$\mathcal{E}_1 = \{\mathsf{rain}\}$$
$$\mathcal{E}_2 = \{\mathsf{sprinklers}\}$$
$$\mathcal{E}_3 = \{\mathsf{temperature\_high}\}$$
$$\mathcal{E}_4 = \{\neg\mathsf{grass\_green}\}$$
$$\mathcal{E}_5 = \{\mathsf{grass\_wet}\}$$
$$\mathcal{E}_6 = \{\mathsf{rain}, \mathsf{sprinklers}\}$$
$$\dots$$

The given explanations are just a subset of all the plain explanations. The complete set

of plain explanations is infinite, for any trivial contradiction ($\mathcal{E}' \models \bot$), or any explanation that is a union of an excising explanation ($\mathcal{E}_n \subset \mathcal{E}'$), it holds that $\Gamma \cup \mathcal{E}' \models \Phi$, thus is a valid, plain, explanation. To illustrate how the constraints remove redundant explanations the set of explanations given above is sufficient.

According to definition 2.1 the abductive query is explanatory ($\Gamma \not\models \Phi$), thus all the explanations are explanatory. $\mathcal{E}_4$ entails Φ only because $\{\texttt{grass\_green}\} \cup \{\neg\texttt{grass\_green}\} \models \bot$, while the explanation has no relations to why the grass is wet. As $\{\texttt{grass\_green}\} \subseteq \Gamma$ and $\{\neg\texttt{grass\_green}\} \subseteq \mathcal{E}_4$ hold, $\Gamma \cup \mathcal{E}_4 \models \bot$ is true. Thus $\mathcal{E}_4$ is not consistent and can be removed from the explanations. All other explanations of the form $\mathcal{E}' \models \bot$ are removed by the consistency constraint as well. There is only one explanation that is not relevant, $\mathcal{E}_5$. This explanation is actually the same as the observation, thus does not add any new information, therefore it can be removed from the set.

Both $\mathcal{E}_1$ and $\mathcal{E}_2$ are syntactically smaller than $\mathcal{E}_6$, thus $\{\texttt{rain}, \texttt{sprinklers}\}$ is not syntactically minimal. If we already assume that it has rained, it would be redundant to also assume that the sprinklers are on. $\mathcal{E}_1$ and $\mathcal{E}_2$ cannot be compared as sets, $\{\texttt{rain}\} \not\subseteq \{\texttt{sprinklers}\}$ and $\{\texttt{sprinklers}\} \not\subseteq \{\texttt{rain}\}$, and are no subset of another explanation, thus are both syntactically minimal. By enforcing syntactic minimality there is only a finite set of explanations left as all infinite explanations of the form $\mathcal{E}_n \subset \mathcal{E}'$ are by definition not syntactically minimal. The only left explanations are $\mathcal{E}_1$, $\mathcal{E}_2$ and $\mathcal{E}_3$. From these explanations $\mathcal{E}_3$ is not semantically minimal as $\Gamma \cup \{\texttt{temperature\_high}\} \models \{\texttt{sprinklers}\}$, i.e. $\mathcal{E}_3 \preceq_\Gamma \mathcal{E}_2$. While the assumption that the temperature is high might be the ultimate explanation of why the grass is wet, giving it immediately as a solution leaves the knowledge that actually the sprinkles made the grass wet, and the knowledge that they turned on because of the high temperature, unknown. Using the same theory Γ with the new observation $\{\texttt{sprinklers}\}$ leads to the explanation $\{\texttt{temperature\_high}\}$, keeping all the relevant information.

By complying with all constraints only two explanations are left, either it has rained ($\mathcal{E}_1$) or the sprinklers were on ($\mathcal{E}_2$). To decide whether one of the explanations is better over the others future implementations should use other selection methods.

A fifth constraint is discussed by Lucas (1997): a diagnosis for a set of observed symptoms should only explain those symptoms that are observed, and should not entail any other possible symptoms. This constraint is based on the assumption that if a symptom is present, it is likely that this symptom has been observed. When a symptom is not observed, there still is a chance that the symptom is present, thus one cannot add the negation of this symptom to the set of observations. While Lucas (1997) is mainly focused on finding diagnoses for faulty systems, the same constraint could be applied for searching any kind of explanation: the constraint that an explanation for an observation can only explain the observation itself and does not entail additional information. In some situations this might be helpful, imagine the situation in example 4 enriched with the knowledge that when it rains the street will also be wet. In this situation the fifth constraint can help one reason that the sprinklers must have been on, because otherwise the street would also be wet, and that information is not in the observation. If you are standing outside this would be a valid line of reasoning. Yet, if somebody only told you the grass is wet, and the information whether the street is wet is simply missing, it does not seem valid to remove rain as a possible explanation. The fifth constraint is situation dependent, thus will not be used as a constraint in this paper. To avoid explanations that explain too much, negative observations can be added to the observation. To continue the example of the wet street, the street is *not* wet should be included in the observation to restrict the set of explanations. It would be interesting for future implementation to include an interactive system that can check for missing observations.

Based on the definitions given above, several algorithms have been developed to find explanations to an abductive query. Both Lucas (1997) as Aliseda-Llera (1997) provide a good overview of the different, classical techniques to obtain abductive explanations. Since this work focuses on abduction in description logics the next section introduces the basic terminology of description logics, before proceeding to introduce previously developed methods for ABox abduction.

2.2 Description Logics and Ontologies

Description logics (DLs) are knowledge representation languages that can structure knowledge about a certain domain. An ontology is a knowledge base in a DL based language, usually in the ontology language OWL. In an ontology information about the structure of a domain and of instances in that domain can be stored. The aim of this paper is to develop an implementation of abduction that can reason over various ontologies, however, no OWL terminology will be used. This section provides a compact overview of the terminology and reasoning techniques of DLs, based on the book of Baader et al. (2017). Appendix B contains some extra formal definitions that have been left out of this section for readability. For a more extensive overview of all DL terminology readers are advised to consult Baader et al. (2017).

2.2.1 The DL knowledge base

A description logic is a formal language that is used to store structured knowledge. Each DL is indicated by a set of calligraphic letters that represent the expressivity of the language. The basis for most DLs is formed by the concept language $\mathcal{ALC}$. This language consists of concepts, roles, and rules. Concept names can be seen as unary predicates, for example an office domain would have concepts $\mathtt{Employer}$, $\mathtt{Employee}$ and $\mathtt{Job}$. A role name represents a binary relation between two elements, for example the role $\mathtt{worksFor}$ represents a relation between an employer and employee. With the use of concept and role names *concept descriptions*, which are often referred to as concepts, can be formed (definition 2.5). Complex concepts are composed of different concepts which are combined by an operator. The semantics of every operator is described in appendix A. Individual names, such as $\mathtt{Mary}$ or $\mathtt{John}$, can be asserted to concepts and roles. A formula in the form $\mathtt{a} : \mathtt{C}$ is a concept assertion which indicates that the individual $\mathtt{a}$ is asserted to the concept $\mathtt{C}$. A formula in the form of $(\mathtt{a}, \mathtt{b}) : \mathtt{r}$ is a role assertion which indicates that the pair of individuals $(\mathtt{a}, \mathtt{b})$ is asserted to the role $\mathtt{r}$. For example a concept assertion $\mathtt{Mary} : \mathtt{Employer}$, would read as "Mary is an employer" and a role assertion $(\mathtt{John}, \mathtt{Mary}) : \mathtt{worksFor}$ would read as "John works for Mary". All the

assertions of a knowledge base together form the assertions part, called the *ABox*.

Definition 2.5 (Concept description)**.** *Given a set of concept names* $\mathbf{C}$*, and a set of role names* $\mathbf{R}$*, a concept description in* $\mathcal{ALC}$ *can be one of the following:*

1. *Any concept name* $\mathsf{C} \in \mathbf{C}$*,*
2. $\top$ *or* $\bot$*,*
3. $\mathsf{C} \sqcap \mathsf{D}$ *for any concept descriptions* C *and* D*,*
4. $\mathsf{C} \sqcup \mathsf{D}$ *for any concept descriptions* C *and* D*,*
5. $\neg\mathsf{C}$ *for any concept description* C*,*
6. $\exists\mathsf{r}.\mathsf{C}$ *for any concept description* C *and any role name* r*,*
7. $\forall\mathsf{r}.\mathsf{C}$ *for any concept description* C *and any role name* r*.*

An atomic concept description, or simple concept, consists of a single atom (case 1 or 2). A compound concept description, or complex concept, is constructed by at least one concept description and an operator (case 3, 4, 5, 6 or 7).

Rules provide structure as to how concepts are related to each other. This is done via *axioms*, also referred to as *is-a relations*, which are either *general concept inclusions* (GCIs) or *equivalence axioms*. A GCI is a formula $\mathsf{A} \sqsubseteq \mathsf{B}$, which indicates that every individual that is asserted to the concept A, must also be asserted to concept B. An equivalence axiom is a formula $\mathsf{A} \equiv \mathsf{B}$, which is an abbreviation for both formulas $\mathsf{A} \sqsubseteq \mathsf{B}$ and $\mathsf{B} \sqsubseteq \mathsf{A}$. Axioms make it possible to structure knowledge about a domain, without saying anything about the individuals in that domain. To continue the work example, a GCI $\exists\mathsf{worksFor}.\mathsf{Employer} \sqsubseteq \mathsf{Employee}$ could be constructed, meaning that every individual who works for an individual that is an employer, must be an employee. All GCIs together form the terminological part of a knowledge base, called the *TBox*.

Given a knowledge base consisting of an ABox and a TBox, an *interpretation* maps all concept, role, and individual names to an element (definition 2.6). An interpretation satisfies the ABox when all assertions in the ABox hold for all elements in the interpretation. Note that DLs do not necessarily hold a unique name assumption, thus two individuals in the knowledge base can have the same interpretation. An interpretation satisfies the TBox if all GCI's in the TBox hold for every element in the interpretation. When an interpretation satisfies both the ABox and TBox of a knowledge base it is said to be a *model* of that knowledge base (definition 2.7). For every model M there exists one *ABox encoding*, which is the set of all assertions which are entailed by the model M (definition 2.8).

Definition 2.6 (Interpretation)**.** *Given a set of concept names* $\mathbf{C}$*, a set of role names* $\mathbf{R}$*, and a set of individual names* $\mathbf{I}$*, an interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ *consists of a non-empty set* $\Delta^{\mathcal{I}}$*, called the interpretation domain, and a mapping* $\cdot^{\mathcal{I}}$ *that maps:*

- *every concept name* $\mathsf{A} \in \mathbf{C}$ *to a set* $\mathsf{A}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$*,*
- *every role name* $\mathsf{r} \in \mathbf{R}$ *to a binary relation* $\mathsf{r}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$*, and*
- *every individual name* $\mathfrak{a} \in \mathbf{I}$ *to an element* $\mathfrak{a}^{\mathcal{I}} \in \Delta^{\mathcal{I}}$*.*

The mapping of complex concepts, $\top$ *and* $\bot$ *is extended as follows:*

$$
\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}}, \\
\bot^{\mathcal{I}} &= \emptyset, \\
(\mathsf{C} \sqcap \mathsf{D})^{\mathcal{I}} &= \mathsf{C}^{\mathcal{I}} \cap \mathsf{D}^{\mathcal{I}}, \\
(\mathsf{C} \sqcup \mathsf{D})^{\mathcal{I}} &= \mathsf{C}^{\mathcal{I}} \cup \mathsf{D}^{\mathcal{I}}, \\
(\neg \mathsf{C})^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus \mathsf{C}^{\mathcal{I}}, \\
(\exists \mathsf{r}.\mathsf{C})^{\mathcal{I}} &= \{\mathfrak{d} \in \Delta^{\mathcal{I}} \mid \textit{there is an } \mathfrak{e} \in \Delta^{\mathcal{I}} \textit{ with } (\mathfrak{d}, \mathfrak{e}) \in \mathsf{r}^{\mathcal{I}} \textit{ and } \mathfrak{e} \in \mathsf{C}^{\mathcal{I}}\}, \\
(\forall \mathsf{r}.\mathsf{C})^{\mathcal{I}} &= \{\mathfrak{d} \in \Delta^{\mathcal{I}} \mid \textit{for all } \mathfrak{e} \in \Delta^{\mathcal{I}}, \textit{ if } (\mathfrak{d}, \mathfrak{e}) \in \mathsf{r}^{\mathcal{I}}, \textit{ then } \mathfrak{e} \in \mathsf{C}^{\mathcal{I}}\}.
\end{aligned}
$$

$\mathsf{C}^{\mathcal{I}}$ *is called the extension of* C *in* $\mathcal{I}$*, and* $\mathfrak{b} \in \Delta^{\mathcal{I}}$ *is called an* r*-filler of* $\mathfrak{a}$ *in* $\mathcal{I}$ *if* $(\mathfrak{a}, \mathfrak{b}) \in \mathsf{r}^{\mathcal{I}}$*.*

Definition 2.7 (Model)**.** *Given a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ with an ABox $\mathcal{A}$ and a TBox $\mathcal{T}$, an interpretation $\mathcal{I}$ is a model M for $\mathcal{K}$ iff:*

1. *The interpretation satisfies every assertion in the ABox, i.e.:*
   (a) $\mathfrak{a}^{\mathcal{I}} \in \mathsf{C}^{\mathcal{I}}$ *for every concept assertion* $\mathfrak{a} : \mathsf{C} \in \mathcal{A}$*, and*
   (b) $(\mathfrak{a}^{\mathcal{I}}, \mathfrak{b}^{\mathcal{I}}) \in \mathfrak{r}^{\mathcal{I}}$ *for every role assertion* $(\mathfrak{a}, \mathfrak{b}) : \mathfrak{r} \in \mathcal{A}$*.*
2. *The interpretation satisfies every GCI in the TBox, i.e.* $\mathsf{C}^{\mathcal{I}} \subseteq \mathsf{D}^{\mathcal{I}}$ *for every GCI* $\mathsf{C} \sqsubseteq \mathsf{D}$ *in* $\mathcal{T}$ *.*

If and only if there exists a model $\mathcal{M}$ for $\mathcal{K}$, then $\mathcal{K}$ can be called consistent.

Definition 2.8 (ABox encoding)**.** *Given the set of all concept names* $\mathbf{C}$*, all role names* $\mathbf{R}$*, and all individual names* $\mathbf{I}$*, the ABox encoding of a model* M *is an ABox* $\mathcal{B}_{\mathsf{M}}$ *constructed as follows:*

$$\begin{aligned}\mathcal{B}_{\mathsf{M}} =& \{\mathfrak{a} : \mathsf{C} | \mathfrak{a}^{\mathsf{M}} \in \mathsf{C}^{\mathsf{M}}, \mathsf{C} \in (\mathsf{A}, \neg\mathsf{A}), \mathsf{A} \in \mathbf{C}, \mathfrak{a} \in \mathbf{I}\} \\ & \cup \{(\mathfrak{a}, \mathfrak{b}) : \mathfrak{r} | (\mathfrak{a}^{\mathsf{M}}, \mathfrak{b}^{\mathsf{M}}) \in \mathfrak{r}^{\mathsf{M}}, \mathfrak{r} \in \mathbf{R}, \mathfrak{a}, \mathfrak{b} \in \mathbf{I}\} \\ & \cup \{(\mathfrak{a}, \mathfrak{b}) : \neg\mathfrak{r} | (\mathfrak{a}^{\mathsf{M}}, \mathfrak{b}^{\mathsf{M}}) \notin \mathfrak{r}^{\mathsf{M}}, \mathfrak{r} \in \mathbf{R}, \mathfrak{a}, \mathfrak{b} \in \mathbf{I}\}.\end{aligned}$$

If there exists at least one model for a knowledge base $\mathcal{K}$, then $\mathcal{K}$ is *consistent*, otherwise $\mathcal{K}$ is inconsistent. Almost every reasoning problem for DLs can be reduced to a problem of consistency. Finding a model of knowledge base proves the consistency of that knowledge base. Therefore, DL reasoners are built that are optimized to search for models of a knowledge base. As this thesis is focused on inferences, the main reasoning task is to check the entailment of assertions. An assertion is entailed by a knowledge base $\mathcal{K}$ if and only if the assertion must be true given the knowledge in $\mathcal{K}$, i.e. when all models of $\mathcal{K}$ contain the assertion. When an assertion $\mathfrak{a} : \mathsf{A}$ is entailed by a knowledge base $\mathcal{K}$ this is represented by the entailment symbol in the following form: $\mathcal{K} \models \mathfrak{a} : \mathsf{A}$. To check if an assertion $\mathfrak{a} : \mathsf{A}$ is entailed by $\mathcal{K}$, it is sufficient to check the consistency of $\mathcal{K}' = \mathcal{K} \cup \{\mathfrak{a} : \neg\mathsf{A}\}$. If there exists a model for $\mathcal{K}'$, then it is possible

that $\mathtt{a} : \neg\mathtt{A}$ is true in the context of $\mathcal{K}'$, thus $\mathtt{a} : \mathtt{A}$ is not entailed by $\mathcal{K}$. If there exists no model for $\mathcal{K}'$, then there is no possibility for $\mathtt{a} : \neg\mathtt{A}$ to be true, thus $\mathtt{a} : \mathtt{A}$ must be true and $\mathcal{K} \vDash \mathtt{a} : \mathtt{A}$ holds. Note that $\mathtt{a} : \mathtt{A}$ is always entailed by a knowledge base $\mathcal{K}$ when $\mathcal{K}$ is inconsistent on its own.

There are several methods to check the consistency of a knowledge base. The most common methods are resolution and tableau algorithms. As this paper is focused on an abduction method that uses the tableau algorithm, this method is explained in section 2.2.4. Dependent on the DL language that is used the tableau algorithm can vary. As the implementations of this thesis are written in the DL $\mathcal{ALCHO}$, additional knowledge is needed about nominals (indicated by the letter $\mathcal{O}$), and role inclusion (indicated by the letter $\mathcal{H}$), before explaining the tableau algorithm for the DL $\mathcal{ALCHO}$. Information about all different DL languages can be found in Baader et al. (2017).

2.2.2 Nominals

As described in the previous section, the rules in a TBox describe the relations between different concepts. However, in some cases, one wants to describe a concept with the use of an individual name. Take, for example, the case that TNO has a knowledge base that contains all information about its employees and departments. Now imagine one wants to add a concept for all employees in the department Data science, called $\mathtt{EmployeeOfDS}$. The department Data Science is already defined as an individual $\mathtt{DS}$ and a role $\mathtt{worksAt}$ exists to indicate at which department an employee works. In this case it is not possible to easily add a GCI that uses the present information to assert all correct employees to the concept $\mathtt{EmployeeOfDS}$. To use the present knowledge, one needs to express the concept $\mathtt{EmployeeOfDS}$ with the help of the individual $\mathtt{DS}$, which is why *nominals* were introduced.

An individual name $\mathtt{a}$ can be used as a nominal $\{\mathtt{a}\}$. The interpretation of a nominal $(\{\mathtt{a}\})^{\mathcal{I}}$ is equal to the interpretation of the individual $\{\mathtt{a}^{\mathcal{I}}\}$. By using an individual to form a nominal, the individual can be used as a concept and can therefore be used to define other concepts. To continue the example of employees of TNO, the individual $\mathtt{DS}$ can now be used as a nominal

$\{\mathsf{DS}\}$ to form the concept definition $\mathsf{EmployeeOfDS} \equiv \mathsf{Employee} \sqcap \exists \mathsf{worksAt}.\{\mathsf{DS}\}$. Now all the individuals that are asserted to the concept $\mathsf{Employee}$ and have been indicated to work at the department of DS, are asserted to the concept $\mathsf{EmployeeOfDS}$.

2.2.3 Role inclusion

The relations between concepts can be described via TBox axioms, yet relations between roles cannot be described in the DL $\mathcal{ALC}$. Still, a hierarchy between roles can exist in a domain. For example, if one wants to specify which employees work full-time at one department, the role $\mathsf{worksFulltimeAt}$ can be introduced. Intuitively one would like to infer that any employee that works full-time at a department, does in fact work at that department. With the DL $\mathcal{ALCO}$ one would have to separately assert all full-time employees both to the role $\mathsf{worksFulltimeAt}$ and the role $\mathsf{worksAt}$. This double assignment is not only redundant, it is also prone to errors if a user does add the assertion $\mathsf{worksFulltimeAt}$, yet forgets to add the assertion to $\mathsf{worksAt}$. Therefore, role inclusion axioms were introduced.

A *role inclusion axiom* (RIA) is a DL constructor that can form an implication between two roles, therefore allowing a hierarchy among roles. An RIA is an axiom in the form $\mathsf{r} \sqsubseteq \mathsf{s}$, where $\{\mathsf{r}, \mathsf{s}\}$ are roles, that indicates that every role assertion r must also be asserted to role s. The role inclusion axioms are added to the TBox of a knowledge base. By adding the role inclusion $\mathsf{worksFulltimeAt} \sqsubseteq \mathsf{worksAt}$ to the example given above, all fulltime employees of the department Data Science are automatically indicated to be employees of the department Data Science. To summarize role inclusion with multiple steps the symbol $\sqsubseteq^*$ can be used, i.e. $\mathsf{r} \sqsubseteq^* \mathsf{s}$ if and only if $\mathsf{r} \sqsubseteq \mathsf{r}_i, \ldots, \mathsf{r}_n \subseteq \mathsf{s}$ or $\mathsf{r} = \mathsf{s}$ for roles $\{\mathsf{r}, \mathsf{r}_i, \ldots, \mathsf{r}_n, \mathsf{s}\}$. For example, if everyone who works at a department is also an stakeholder of that department and $\mathsf{worksAt} \sqsubseteq \mathsf{stakeholderOf}$ is added to the TBox, then $\mathsf{worksFulltimeAt} \sqsubseteq^* \mathsf{stakeholderOf}$ holds.

2.2.4 Tableau algorithm

A tableau algorithm for description logics expands a knowledge base $\mathcal{K}$, such that all the information that is present in $\mathcal{K}$ is represented in simple assertions, while checking for conflicting assertions. By expanding all TBox axioms and ABox assertions to simple ABox assertions, a model can be found for a given knowledge base. As described in section 2.2.1, a knowledge base is consistent when there is at least one model for that knowledge base. If two conflicting assertions are present in a knowledge base no model is possible. This is indicated by saying there exists a clash in the knowledge base. A tableau algorithm applies expansion rules until there are no more rules to apply, or a clash is indicated. When a clash is indicated there is no possible model for the knowledge base, thus the tableau algorithm returns that the knowledge base is inconsistent. When there are no more rules to apply, and there is no clash, then there is a model for the knowledge base, and the knowledge base is consistent.

The expansion rules for a knowledge base in $\mathcal{ALCHO}$ are given in figure 2.1. Each expansion rule adds new assertions to the ABox, with the exception of the expansion rule for nominals, the {}-rule. The rule for nominals merges two individuals that both are assigned to one nominal. This rule replaces one of the individuals by the other individual in all assertions, as the interpretation of both individuals must be the same. While most rules add known concepts and roles to the ABox assertions, the or rule ($\sqcup$-rule) introduces a new concept, X, that can represent either side of the or operator. This rule introduces branching in the tableau algorithm, where one branch represents the left side of the or-operator and the other branch the right side. Every time the or rule is applied, two paths can be followed by the tableau algorithm. It can arbitrarily choose one of the paths to check for a model. If the first path leads to a model, consistency of the knowledge base is proven, and the second path does not have to be checked. If the first path leads to a clash, the second option should be checked to prove consistency or inconsistency.

The $\exists$-rule mentions a condition "$\mathtt{a}$ is not blocked". The blocking constraint was introduced to stop the algorithm in cases where existential concepts introduce an infinite chain of new individuals. Note that an individual $\mathtt{a}$ is an *ancestor* of an individual $\mathtt{b}$ if and only if there

Let $\mathcal{K} = (\mathcal{A}, \mathcal{T})$ be an $\mathcal{ALCHO}$ knowledge base with ABox $\mathcal{A}$ and TBox $\mathcal{T}$, $\{\mathfrak{a}, \mathfrak{b}, \mathfrak{c}, \mathfrak{d}\}$ be individual names, $\{\mathfrak{r}, \mathfrak{s}\}$ be role names, $\{\mathfrak{o}\}$ be a nominal, and $\{\mathsf{C}, \mathsf{D}\}$ be concepts (either simple or complex). The following rules can be applied to expand the knowledge base $\mathcal{K}$:

$\sqcap$-rule: if 1. $\mathfrak{a} : \mathsf{C} \sqcap \mathsf{D} \in \mathcal{A}$, and
2. $\{\mathfrak{a} : \mathsf{C}, \mathfrak{a} : \mathsf{D}\} \nsubseteq \mathcal{A}$
then $\mathcal{A} \longrightarrow \mathcal{A} \cup \{\mathfrak{a} : \mathsf{C}, \mathfrak{a} : \mathsf{D}\}$.

$\sqcup$-rule: if 1. $\mathfrak{a} : \mathsf{C} \sqcup \mathsf{D} \in \mathcal{A}$, and
2. $\{\mathfrak{a} : \mathsf{C}, \mathfrak{a} : \mathsf{D}\} \cap \mathcal{A} = \emptyset$
then $\mathcal{A} \longrightarrow \mathcal{A} \cup \{\mathfrak{a} : \mathsf{X}\}$ for some $\mathsf{X} \in \{\mathsf{C}, \mathsf{D}\}$.

$\exists$-rule: if 1. $\mathfrak{a} : \exists \mathfrak{r}.\mathsf{C} \in \mathcal{A}$,
2. there is no $\mathfrak{b}$ such that $\{(\mathfrak{a}, \mathfrak{b}) : \mathfrak{r}, \mathfrak{b} : \mathsf{C}\} \nsubseteq \mathcal{A}$, and
3. $\mathfrak{a}$ is not blocked,
then $\mathcal{A} \longrightarrow \mathcal{A} \cup \{(\mathfrak{a}, \mathfrak{d}) : \mathfrak{r}, \mathfrak{d} : \mathsf{C}\}$, where $\mathfrak{d}$ is new in $\mathcal{A}$.

$\forall$-rule: if 1. $\{\mathfrak{a} : \forall \mathfrak{r}.\mathsf{C}, (\mathfrak{a}, \mathfrak{b}) : \mathfrak{s}\} \subseteq \mathcal{A}$, and
2. $\mathfrak{b} : \mathsf{C} \notin \mathcal{A}$
3. $\mathfrak{s} \boxplus \mathfrak{r}$
then $\mathcal{A} \longrightarrow \mathcal{A} \cup \{\mathfrak{b} : \mathsf{C}\}$.

$\sqsubseteq$-rule: if 1. $\mathfrak{a} : \mathsf{C} \in \mathcal{A}, \top \sqsubseteq \mathsf{D} \in \mathcal{T}$, and
2. $\{\mathfrak{a} : \mathsf{D}\} \notin \mathcal{A}$
then $\mathcal{A} \longrightarrow \mathcal{A} \cup \{\mathfrak{a} : \mathsf{D}\}$.

{}-rule: if 1. $\{\mathfrak{a} : \{\mathfrak{o}\}, \mathfrak{b} : \{\mathfrak{o}\}\} \subseteq \mathcal{A}$,
2. $\mathfrak{a} \neq \mathfrak{b}$
then $\mathcal{A} \longrightarrow \mathcal{A} \setminus \{\mathfrak{a} : \mathsf{C}\} \cup \{\mathfrak{b} : \mathsf{C}\}$ for every $\mathfrak{a} : \mathsf{C} \in \mathcal{A}$,
$\mathcal{A} \longrightarrow \mathcal{A} \setminus \{(\mathfrak{a}, \mathfrak{c}) : \mathfrak{r}\} \cup \{(\mathfrak{b}, \mathfrak{c}) : \mathfrak{r}\}$ for every $(\mathfrak{a}, \mathfrak{c}) : \mathfrak{r} \in \mathcal{A}$, and
$\mathcal{A} \longrightarrow \mathcal{A} \setminus \{(\mathfrak{c}, \mathfrak{a}) : \mathfrak{r}\} \cup \{(\mathfrak{c}, \mathfrak{b}) : \mathfrak{r}\}$ for every $(\mathfrak{c}, \mathfrak{a}) : \mathfrak{r} \in \mathcal{A}$.

Figure 2.1: Expansion rules for $\mathcal{ALCHO}$ knowledge bases.

exist $\mathfrak{n}$ rules such that assertions $(\mathfrak{a}, \mathfrak{a}_1) : \mathfrak{r}_1, \ldots (\mathfrak{a}_n, \mathfrak{b}) : \mathfrak{r}_n$ are present in the ABox, with $\{\mathfrak{a}_1, \ldots, \mathfrak{a}_n\}$ being any individuals. In contrast, $\mathfrak{a}$ is a *descendant* of $\mathfrak{b}$, when $\mathfrak{b}$ is an ancestor of $\mathfrak{a}$. For an individual $\mathfrak{a}$ to be blocked by another individual $\mathfrak{b}$, $\mathfrak{a}$ must be a descendent of $\mathfrak{b}$, the set of concepts C that $\mathfrak{a}$ is asserted to must be a subset of the set of concepts $\mathfrak{b}$ is asserted to. The introduction of blocking ensures the termination of the algorithm. Proof of the soundness, completeness, and termination of the tableau algorithm for DL can be found in Baader et al. (2017).

2.3 Abductive Reasoning in DL: ABox Abduction

While abduction as a form of reasoning has a rich history, abduction over description logics is a relatively new concept. Elsenbroich et al. (2006) argued why abduction could be an interesting form of reasoning to use in ontologies. They classify abduction over DLs into the following four forms, which scholars have followed since:

- *Concept abduction*: The abduction problem consists of a concept Φ and a knowledge base in a DL as the theory Γ, which has to be solved by a concept E as an explanation, such that the knowledge base entails the implication that E leads to Φ, i.e. $\Gamma \models \mathsf{E} \sqsubseteq \Phi$.

  A special case of concept abduction is *Conditionalized concept abduction*. Instead of using the tuple $\langle \Gamma, \Phi \rangle$ as the abduction problem, a condition concept C is added. Only explanations for which the union with the condition implicates the observation are valid, i.e. $\Gamma \models \mathsf{E} \sqcap \mathsf{C} \sqsubseteq \Phi$.

- *ABox abduction*: The abduction problem consists of an assertion $\Phi = \mathsf{C}(\mathfrak{a})$ and a knowledge base in a DL as the theory Γ, which has to be solved by a set of assertions $\mathcal{E}$ as an explanation, such that the explanation together with the theory entails the observed assertion, i.e. $\Gamma \cup \mathcal{E} \models \Phi$.

- *TBox abduction*: The abduction problem consists of an general concept inclusion $\Phi = \mathsf{C} \sqsubseteq \mathsf{D}$ and a knowledge base in a DL as the theory Γ, which has to be solved by a set of GCI's $\mathcal{E}$ as an explanation, such that adding the explanation to the knowledge base implies the observation, i.e. $\Gamma \cup \mathcal{E} \models \mathsf{C} \sqsubseteq \mathsf{D}$.

- *Knowledge base abduction*: The abduction problem consists of either a general concept inclusion or an assertion Φ and a knowledge base in a DL as the theory Γ, which has to be solved by a set of GCI's and assertions $\mathcal{E}$ as an explanation, such that adding the explanation to the knowledge base implies the observation, i.e. $\Gamma \cup \mathcal{E} \models \Phi$.

Each form of abduction for DLs has its own unique goals. Concept abduction is used to reason about the relations between concepts. Finding an explanation for a simple concept name can

already be achieved by most existing DL reasoners. By using the concept forgetting technique of Zhao and Schmidt (2018) explanations for complex concept assertions can also be found easily. TBox abduction is more helpful for building and repairing ontologies as it is used to search for missing is-a relationships. A practical algorithm for finding TBox abduction explanations in expressive DLs is given in Du, Wan, et al. (2017). While the algorithm presented is practical in use, it lacks a formal proof of soundness and completeness. Both concept and TBox abduction are interesting forms of reasoning to research, yet they are used to reason about the ontologies itself rather than using the ontology to reason about individuals in the real world. Both ABox abduction and knowledge base abduction can reason about individuals and are therefore more interesting for the use case of TNO, where explanations are searched for specific incidents.

Research into knowledge base abduction is limited. One could think of knowledge base abduction as a combination of ABox and TBox abduction, as all solutions for an ABox/TBox assertion in ABox/TBox abduction will also be solutions in knowledge base abduction, This could be a reason that research is more focused on first solving ABox and TBox abduction, before attempting to create an implementation that can account for both. While the solution for knowledge base abduction contains all explanations that would be retrieved via ABox abduction, it can also contain explanations that include potential missing GCI's. In situations where one is using a tested ontology to abduct explanations for a certain event, explanations that suggest potential changes to the structure of an ontology might distract from the true cause. In such a situation, when one is confident about the structure of its ontology, ABox abduction can be more useful than knowledge base abduction. As this research is focused on finding explanations for surprising observations with the use of an established ontology, rather then searching for additions to a flawed ontology, this research is focused on improving the excising implementations for ABox abduction.

Definition 2.9 (ABox Abduction problem)**.** *Let* Γ *be a knowledge base in DL,* $\mathcal{O}$ *the set of all possible ABox assertions and* $\Phi \subseteq \mathcal{O}$ *an observation. A tuple* $\langle \Gamma, \Phi \rangle$ *can be described as an ABox abduction problem if and only if:*

1. $\Gamma \not\models \Phi$,

2. $\Gamma \cup \Phi \not\models \bot$.

Definition 2.10 (ABox Abduction explanation)**.** *Let* $\Phi \subseteq \mathcal{O}$ *be an ABox abduction problem, with knowledge base* Γ *and ABox assertion* $\Phi \subseteq \mathcal{O}$*. The set of solutions to an ABox abduction problem contains all valid explanations,* $\mathcal{S} = \{\mathcal{E} | \mathcal{E}$ *is an explanation for* $\langle \Gamma, \Phi \rangle\}$*. An explanation* $\mathcal{E} \subseteq \mathcal{H}$ *can be described as an explanation of* Φ *with respect to* Γ *if and only if:*

1. $\Gamma \cup \mathcal{E} \models \Phi$	*(plain)*
2. $\Gamma \not\models \Phi$	*(explanatory)*
3. $\Gamma \cup \mathcal{E} \not\models \bot$	*(consistent)*
4. $\mathcal{E} \not\models \Phi$	*(relevant)*
5. $\mathcal{E}' \not\subseteq \mathcal{E}$ *for every* $\mathcal{E}' \in \mathcal{S} \backslash \{\mathcal{E}\}$	*(syntactically minimal)*
6. $\Gamma \cup \mathcal{E}' \models \mathcal{E}$ *or* $\Gamma \cup \mathcal{E} \not\models \mathcal{E}'$ *for every* $\mathcal{E}' \in \mathcal{S} \backslash \{\mathcal{E}\}$	*(semantically minimal)*

2.3.1 Methods for ABox Abduction

Since Elsenbroich et al. (2006) argued for the use of abduction over an ontology several studies have been conducted on the topic. Following chronological order, Klarman et al. (2011) was the first to propose an ABox abduction method. Klarman et al. (2011) introduce an approach for ABox abduction in the DL $\mathcal{ALC}$, with observations and explanations in $\mathcal{ALE}$. To find explanations the knowledge base and abduction query are first translated to first-order logic and a modal structure. The approach is based on both regular connection tableau and resolution with set-of-support. The paper provides a proof of soundness (for plain solutions) and completeness for the proposed method, yet no practical implication is provided. A more practically oriented method is proposed in Du, Qi, et al. (2011), and further developed in Du, Wang, et al. (2014) and Du, Wang, et al. (2015). The proposed method translates $\mathcal{SHIQ}$ ontologies to Datalog, such that a Prolog abduction solver can be used to compute explanations, which are translated back to DL afterwards. The papers provide empirical proof to support their method, yet only soundness is formally proven. As the method is not proven to be complete, the method might not find all possible explanations to an abduction problem.

Both Klarman et al. (2011) and Du, Wang, et al. 2014 use translations to other logical systems for finding abductive explanations. By translating to another system, the abductive methods already developed in that specific system can be used. Yet, due to the translation to another system expressivity can be lost, and it is not possible to use the optimized techniques developed for DL anymore. Ma et al. (2012) propose to transform the abduction problem into a consistency problem of the knowledge base, which can be solved by using the traditional tableaux construction in the DL $\mathcal{ALCI}$. The abduction problem of searching an explanation $\mathcal{E}$ for an observation Φ, given the knowledge base Γ, is transformed into the consistency problem of finding any explanation $\mathcal{E}$, such that $\mathcal{E}$ is inconsistent when combining it with the knowledge base Γ and the negated observation, i.e. $\Gamma \cup \mathcal{E} \cup \{\neg\Phi\}$ is inconsistent. To find all explanations Ma et al. (2012) proposes to construct a tableau for the negation normal form of the set $\Gamma \cup \{\neg\Phi\}$. For every unclosed branch of the tableau, the negation of the last node is added to the explanation. The method of Ma et al. (2012) is illustrated with the use of an example, yet no complete implementation, nor a proof of soundness and completeness is included in the research.

In the same year another method which uses the tableau algorithm for DL, was presented by Halland and Britz (2012). Similarly, the abduction problem $\langle \Gamma, \Phi \rangle$ is transformed into the problem of finding all explanations $\mathcal{E}$ such that $\Gamma \cup \mathcal{E} \cup \{\neg\Phi\}$ is inconsistent. They use an extension of the tableau algorithm to find all models M for $\Gamma \cup \{\neg\Phi\}$. Subsequently, the *minimal hitting set* algorithm of Reiter (1987) is used to find explanations that can make every model M invalid. The minimal hitting set is only used as a tool and no alterations to the algorithm are made. Halland and Britz (2012) provide an implementation for the DL $\mathcal{ALC}$ that can compute explanations in $\mathcal{ALE}$. While the method is explained to be sound, Halland and Britz (2012) discuss that their approach is not complete. Surprisingly the semantically minimal explanations are the ones that are often missing from the set of explanations this implementation provides.

Building on the work of Halland and Britz (2012) other implementations were developed that used the minimal hitting set algorithm to search for useful explanations (Pukancová and Homola 2017; Pukancová and Homola 2018; Mrózek et al. 2018). In Pukancová and Homola (2017) an algorithm is presented that is sound and complete for finding explanations for single observations in the $\mathcal{ALCHO}$ description logic. Pukancová and Homola (2018) describe how an

extended version of this approach can be used for multiple observations. Furthermore Mrózek et al. (2018) describe how the implementation can be altered to use a reasoner of choice to call the tableau algorithm. Unlike the method of Halland and Britz (2012), these approaches do alter the hitting set algorithm, such that the tableau algorithm is called less frequently. An overview of how the minimal hitting set algorithm operates, and how it is used to find abductive explanations, is given in section 2.3.2.

The majority of the techniques for ABox abduction make use of the tableau algorithm, still Del-Pinto and Schmidt (2018) present an abduction method with the use of forgetting. *Forgetting* is a non-standard reasoning technique that uses resolution to remove a set of symbols from the knowledge base, while preserving all entailments (Koopmann and Schmidt 2015). The new ontology, formed by forgetting, is a *uniform interpolant*, which contains the strongest necessary entailment of the knowledge base. Del-Pinto and Schmidt (2018) show that computing the uniform interpolant over a conjunction of the knowledge base and the negated observation, by forgetting the concepts in the observation, results in a set of assertions, and that every resulting set is a semantically minimal explanation for the observation. With the use of traditional consistency checking, each explanation can be checked for the other constraints given in definition 2.2, such as consistency and syntactic minimality. The method is proven to be sound and complete in $\mathcal{ALC}$, yet it cannot reason with roles in the observation or explanations as the resolution rules for $\mathcal{ALC}$ cannot handle negated role assertions. As the forgetting technique is still in development, and the use of tableau algorithms for DLs is already widely used, this paper will focus on techniques that make use of the minimal hitting set algorithm, in combination with a tableau algorithm. Nonetheless, Del-Pinto and Schmidt (2018) present an interesting way of looking at the abduction problem, which showed that it is possible to develop a sound and complete method for extracting semantically minimal explanations.

2.3.2 Minimal Hitting Set Algorithm

As discussed in the previous section, the minimal hitting set algorithm forms a basis for promising ABox abduction techniques. To fully understand the technique this section explains the

minimal hitting set algorithm as developed by Reiter (1987), before proceeding to explain how it can be used for ABox abduction specifically. Reiter (1987) proposed an algorithm to run diagnosis in a first-order system. A *system* is described as a pair $(\mathsf{SD}, \mathsf{COMPONENTS})$, where SD, the *system description*, a set of first-order sentences and $\mathsf{COMPONENTS}$, the *system components*, is a finite set of constants. SD contains a unary predicate AB for each component, which indicates if a component is behaving abnormal. Provided an observation OBS a diagnosis for the triple $(\mathsf{SD}, \mathsf{COMPONENTS}, \mathsf{OBS})$ can be retrieved by searching for a minimal set of faulty components. In other words, a diagnosis assumes a minimal set of components to be working abnormal, while the rest of the components function properly

Definition 2.11 (Diagnosis)**.** *A diagnosis for (SD, COMPONENTS, OBS) is a minimal set* $\Delta \subseteq \mathsf{COMPONENTS}$ *such that:* $\mathsf{SD} \cup \mathsf{OBS} \cup \{\mathsf{AB}(c) | c \in \Delta\} \cup \{\neg \mathsf{AB}(c) | c \in \mathsf{COMPONENTS} - \Delta\}$ *is consistent.*

Example 5. Consider an webshop that has a system in place for handling orders. Whenever an order is placed by a client, and the client has enough credit on its account, the order is complete. When a complete order is placed, and the correct address in registered to the client then the order is send to that address. This system is formally represented by $(\mathsf{SD}, \mathsf{COMPONENTS})$ with $\mathsf{COMPONENTS} = \{o_1, o_2, c, a\}$ and SD:

$$\begin{aligned} \mathtt{order}(X) \wedge \mathtt{complete}(X) \wedge \mathtt{address}(Y) \wedge \neg \mathsf{AB}(Y) &\rightarrow \mathtt{send}(Y) \\ \mathtt{order}(X) \wedge \neg \mathsf{AB}(X) \wedge \mathtt{credit}(Y) \wedge \neg \mathsf{AB}(Y) &\rightarrow \mathtt{complete}(X) \\ \mathtt{credit}(c), \\ \mathtt{address}(a), \\ \mathtt{order}(o_1), \\ \mathtt{order}(o_2). \end{aligned}$$

Suppose, we observe that no package is sent to the client's address, $\mathsf{OBS} = \{\neg \mathtt{send}(a)\}$. Given definition 2.11 there are three proper diagnoses for $(\mathsf{SD}, \mathsf{COMPONENTS}, \mathsf{OBS})$:

$\Delta_1 : \{c\}, \Delta_2 : \{a\}$ and $\Delta_3 : \{o_1, o_2\}$. Either the client's credit, the client's address or both the orders are faulty. This is the result of the following consistent unions:

$$SD \cup OBS \cup \{AB(c)\} \cup \{\neg AB(a), \neg AB(o_1), \neg AB(o_2)\} \qquad \Delta_1$$

$$SD \cup OBS \cup \{AB(a)\} \cup \{\neg AB(c), \neg AB(o_1), \neg AB(o_2)\} \qquad \Delta_2$$

$$SD \cup OBS \cup \{AB(o_1), AB(o_2)\} \cup \{\neg AB(a), \neg AB(c)\} \qquad \Delta_3$$

To efficiently compute the diagnoses for a system Reiter (1987) proposed to use conflict sets. A *conflict set* is any subset of components such that assuming every component in this set to be normal, in conjunction with the observation, and the system description results in an inconsistent system (definition 2.12). He proposes that a diagnosis is any minimal set for which the complement of the diagnosis is no conflict set for a system. By definition 2.11 any complement set of the diagnosis has to be consistent with the system description and observation, therefore it cannot be a conflict set as defined by (2.12).

Definition 2.12 (Conflict set)**.** *A conflict set for* $(SD, COMPONENTS, OBS)$ *is a set* $\{c_1, \ldots, c_k\} \subseteq COMPONENTS$ *such that* $SB \cup OBS \cup \{\neg AB(c_1), \ldots, \neg AB(c_k)\}$ *is inconsistent.*

A conflict set $\mathcal{C}$ *for* $(SD, COMPONENTS, OBS)$ *is minimal iff there is no conflict set* $\mathcal{C}'$ *for* $(SD, COMPONENTS, OBS)$ *such that* $\mathcal{C}' \subset \mathcal{C}$*.*

Definition 2.13 (Hitting set)**.** *Suppose* $\mathcal{S} = \{S_1, \ldots, S_i\}$ *is a collection of sets. A hitting set for* $\mathcal{S}$ *is a set* $\mathcal{H} \subseteq \bigcup_{S_i \in \mathcal{S}} S_i$ *such that* $\mathcal{H} \cap S_i \neq \emptyset$ *for each* $S_i \in \mathcal{S}$*.*

A hitting set $\mathcal{H}$ *for* $\mathcal{S}$ *is minimal iff there is no hitting set* $\mathcal{H}'$ *for* $\mathcal{S}$ *such that* $\mathcal{H}' \subset \mathcal{H}$*.*

Reiter (1987) shows that finding a diagnosis for a system can be solved by finding all minimal hitting sets for the collection of conflict sets for that system. As described in definition 2.13, a *hitting set* for a collection of sets contains at least one element of every set in the collection. Therefore, the complement of a hitting set for the collection all conflict sets cannot be a conflict set in itself. Given this knowledge, and the knowledge that any diagnosis is a minimal set such

that the complement is no conflict set, finding all diagnosis for a system amounts to finding minimal hitting sets for the collection of the conflict sets. While finding the hitting sets for the collection of all conflict sets is legitimate, it is sufficient to find the hitting sets for all minimal conflict sets. Any non-minimal conflict set has a minimal subset. If a hitting set contains an element of this subset, it also contains at least one element of the non-minimal conflict set. Therefore, the minimal hitting sets of all minimal conflict sets will be the same collection as the minimal hitting sets of all conflict sets.

Theorem 2.1. $\Delta \subseteq \mathsf{COMPONENTS}$ *is a diagnosis for* $(\mathsf{SD}, \mathsf{COMPONENTS}, \mathsf{OBS})$ *iff* Δ *is a minimal hitting set for the collection of minimal conflict sets for* $(\mathsf{SD}, \mathsf{COMPONENTS}, \mathsf{OBS})$.

Example 6 (Continued)**.** The order system and the observation described before, formulated as $(\mathsf{SD}, \{o_1, o_2, c, a\}, \{\neg \mathtt{send}(a)\})$ have two minimal conflict sets according to definition 2.12: $\{o_1, c, a\}$ and $\{o_2, c, a\}$, provided by the inconsistency of the following formulas respectively:

$$\mathsf{SD} \cup \mathsf{OBS} \cup \{\neg \mathsf{AB}(o_1), \neg \mathsf{AB}(c), \neg \mathsf{AB}(a)\}$$

and

$$\mathsf{SD} \cup \mathsf{OBS} \cup \{\neg \mathsf{AB}(o_2), \neg \mathsf{AB}(c), \neg \mathsf{AB}(a)\}.$$

Given the collection of minimal conflict sets $\mathcal{C} = \{\{o_1, c, a\}, \{o_2, c, a\}\}$ three minimal hitting sets can be found: $\{c\}, \{a\}$ and $\{o_1, o_2\}$. The minimal hitting sets correspond to the diagnoses (Δ_1, Δ_2 and Δ_3) computed in example 5.

Hitting sets can be used to compute diagnoses, but the given definitions do not present a effective way to identify minimal hitting sets yet. To effectively identify minimal hitting sets Reiter (1987) uses a hitting set tree (*HS-tree*). While an HS-tree can be used to find minimal hitting sets for any collection of sets, in the context of abduction HS-trees are used to compute the minimal hitting sets for a collection of conflict sets. To construct an HS-tree, the conflict

sets are used as nodes, and the edges consist of elements in the conflict set. An HS-tree is formally defined as:

Definition 2.14 (HS-tree)**.** *Suppose $\mathcal{S}$ is a collection of sets. An edge- and node-labeled tree T is an HS-tree for $\mathcal{S}$ iff it is a smallest tree with the following properties:*

1. *Its root is labeled by ✓ if $\mathcal{S} = \emptyset$. Otherwise, its root is labeled by some $\mathcal{C} \in \mathcal{S}$.*
2. *If $\mathfrak{n}$ is a node of T, define $\mathsf{H}(\mathfrak{n})$ to be the set of edge labels on the path in T from the root node to $\mathfrak{n}$.*
3. *If $\mathfrak{n}$ is labeled by ✓, it has no successor nodes in T.*
4. *If $\mathfrak{n}$ is labeled by some $\mathcal{C} \in \mathcal{S}$, then for each $\sigma \in \mathcal{C}$, $\mathfrak{n}$ has a successor node $\mathfrak{n}_\sigma$ joined to $\mathfrak{n}$ by an edge labeled by σ. The label for $\mathfrak{n}_\sigma$ is a set $\mathcal{C} \in \mathcal{S}$ such that $\mathcal{C} \cap \mathsf{H}(\mathfrak{n}_\sigma) = \emptyset$ if such a set $\mathcal{C}$ exists. Otherwise, $\mathfrak{n}_\sigma$ is labelled by ✓.*

By property 4 of definition 2.14 it is implied that the path from the root to a node, which is labelled by a check-mark, is a hitting set, as every set in the collection has at least one element that is already in the path. As the nodes have successor nodes for every element in their set, each minimal hitting set is in in the HS-tree. While the HS-tree does produce all minimal hitting sets, not all hitting sets in the tree are minimal. Therefore, Reiter (1987) introduces a pruning technique, such that the search for minimal hitting sets is optimized. With the following steps a a pruned HS-tree can be generated:

1. Generate the HS-tree breadth-first, thus generating nodes at any fixed level in the tree in left-to-right order.
2. Reusing node labels: if a node $\mathfrak{n}$ is labeled by the set $\mathcal{C} \in \mathcal{S}$ and if $\mathfrak{n}'$ is a node such that $\mathsf{H}(\mathfrak{n}') \cap \mathcal{C} = \emptyset$, $\mathfrak{n}'$ is labelled by $\mathcal{C}$. Underline the node to indicate that the label of $\mathfrak{n}'$ is a reused label. Such a node $\mathfrak{n}'$ requires no access to $\mathcal{S}$.
3. Tree pruning:

(a) If node $\mathfrak{n}$ is labeled by ✓and node $\mathfrak{n}'$ is such that $\mathsf{H}(\mathfrak{n}) \subseteq \mathsf{H}(\mathfrak{n}')$, close $\mathfrak{n}'$. Closing is done by marking the node with $\times$, i.e. do not compute a label for $\mathfrak{n}'$; do not generate any successors of $\mathfrak{n}'$.

(b) If node $\mathfrak{n}$ has been generated and node $\mathfrak{n}'$ is such that $\mathsf{H}(\mathfrak{n}) = \mathsf{H}(\mathfrak{n}')$, then close $\mathfrak{n}'$.

(c) If nodes $\mathfrak{n}$ and $\mathfrak{n}'$ have been respectively labeled by sets $\mathcal{C}$ and $\mathcal{C}'$ of $\mathcal{S}$, and if $\mathcal{C}' \subset \mathcal{C}$, then for each $\alpha \in \mathcal{C} - \mathcal{C}'$ mark the edge from node $\mathfrak{n}$ labeled by α as redundant by cutting the edge with)(. 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 $\mathcal{S}$.

Each path from the root to a node of the pruned HS-tree that is marked by a check-mark represents a minimal hitting set. Thus by constructing a pruned hitting set tree for the collection of conflict sets for a system, all diagnoses for that system are computed.

Theorem 2.2. *Let $\mathcal{S}$ be a collection of sets, and T a pruned HS-tree for $\mathcal{S}$, as previously described. Then $\{\mathsf{H}(\mathfrak{n}) | \mathfrak{n}$ is a node of T labeled by ✓$\}$ is the collection of minimal hitting sets for $\mathcal{S}$.*

2.3.3 ABox abduction via Minimal Hitting Set Algorithm

Reiter (1987) introduced the HS-tree to find minimal hitting sets for any collections of sets, specifically for finding conflict sets. For any plain explanation to an ABox abduction problem the following holds: $\Gamma \cup \mathcal{E} \models \Phi$. If $\Gamma \cup \mathcal{E} \models \Phi$ holds, then $\Gamma \cup \mathcal{E} \cup \{\neg\Phi\}$ must be inconsistent, as Φ must be true in every situation that $\Gamma \cup \mathcal{E}$ is true. Concurrently, as $\Gamma \not\models \Phi$ holds for every explanatory abduction problem, $\Gamma \cup \{\neg\Phi\}$ must be consistent. Thus, an ABox abduction problem can be transformed into the problem of finding any $\mathcal{E}$ for which $\Gamma \cup \mathcal{E} \cup \{\neg\Phi\}$ is inconsistent, yet $\Gamma \cup \{\neg\Phi\}$ is consistent. According to definition 2.7 a knowledge base $\mathcal{K}$ is consistent if and only if there exists at least one model M for $\mathcal{K}$. From these preliminaries it follows that there must be at least one model M for $\Gamma \cup \{\neg\Phi\}$ and, given that $\mathcal{M}$ is the set of all models for $\Gamma \cup \{\neg\Phi\}$, $\mathcal{E}$ must comprise of assertions such that every model $\mathsf{M} \in \mathcal{M}$ is not a

model for $\Gamma \cup \mathcal{E} \cup \{\neg\Phi\}$, in other words, an explanation must make every model for $\Gamma \cup \{\neg\Phi\}$ an invalid model for $\Gamma \cup \mathcal{E} \cup \{\neg\Phi\}$. Thus, for every model M for $\Gamma \cup \{\neg\Phi\}$, the explanation $\mathcal{E}$ must contain an assertion that can clash with an assertion in M. Therefore, Pukancová and Homola (2017) use an HS-tree to find the minimal hitting sets of the ABox encodings of every model for $\Gamma \cup \{\neg\Phi\}$.

While a standard HS-tree, as described in section 2.3.2, can be used to find all syntactically minimal explanations, Pukancová and Homola (2017) propose some changes in pruning and labeling to make the algorithm more efficient. An HS-tree, constructed by definition 2.14, with the collection of sets $\mathcal{S}$ corresponding to the ABox encodings of all models M for $\Gamma \cup \{\Phi\}$, can be pruned by the following definition:

Definition 2.15 (Pruned node)**.** *A node* $\mathfrak{n}$ *in an HS-tree* T *for an ABox abduction problem* $\langle \Gamma, \Phi \rangle$ *is pruned if:*

1. *either there exists an node* $\mathfrak{n}'$ *s.t.* $\mathsf{H}(\mathfrak{n}') \subseteq \mathsf{H}(\mathfrak{n})$ *and* $\mathfrak{n}'$ *is labelled by {} or* ✓ *(label* $\mathfrak{n}$ *by {});*
2. *or there exists an node* $\mathfrak{n}'$ *s.t.* $\mathsf{H}(\mathfrak{n}') = \mathsf{H}(\mathfrak{n})$ *and* $\mathfrak{n}'$ *is labelled by a model (label* $\mathfrak{n}$ *by* $\times$*);*
3. *or* $\{\neg\Phi\} \cup \mathsf{H}(\mathfrak{n})$ *is inconsistent (label* $\mathfrak{n}$ *by {});*
4. *or* $\mathsf{H}(\mathfrak{n}) \cup \mathcal{K}$ *is inconsistent (label* $\mathfrak{n}$ *by {}).*

By pruning a HS-tree with the rules above only consistent, relevant, and syntactically minimal explanations are retrieved. Due to condition 4 all paths that result in an inconsistent explanation are pruned immediately. Any node that contains all assertions of a path that has been pruned due to an inconsistency in its path will be pruned by condition 1, without having to check the consistency, as any super set of an inconsistent set will be inconsistent itself. In the same manner condition 3 immediately prunes all nodes that lead to explanations that do not add any new information.

While Halland and Britz (2012) adjust the tableau algorithm to compute the set of all models $\mathcal{M}$ for $\Gamma \cup \{\neg\Phi\}$ and then construct an HS-tree for $\mathcal{M}$, Pukancová and Homola (2017) only

compute one model M at the start and compute other models during the construction of the HS-tree. Furthermore, used models are stored in a in the set $\mathcal{M}$, such that they can be reused for other branches. The ABox encodings of a model are used as labels for the nodes in the HS-tree. The implementation used to find explanations to an ABox abduction problem $\langle \Gamma, \Phi \rangle$ uses the following order:

1. First the tableau algorithm is called to compute a model M for $\mathcal{K} = \Gamma \cup \{\neg\Phi\}$.
   (a) If there is no model, the abduction problem is not relevant and the algorithm stops.
2. An HS-tree is constructed, with the root node labeled by the ABox encoding $\mathcal{B}$ of model M.
3. For a node labelled by $\mathcal{B}$, a successor node is created for every assertion $\mathfrak{b} \in \mathcal{B}$, the path between the nodes is labelled by $\neg\mathfrak{b}$.
4. For each successor node $\mathfrak{n}$ the possibility of pruning is checked in the following order:
   (a) The path $\mathsf{H}(\mathfrak{n})$ is checked for clashes, as this will automatically violate both conditions 3 and 4. If a clash exists $\mathfrak{n}$ is pruned and labelled $\{\}$.
   (b) The path $\mathsf{H}(\mathfrak{n})$ is checked for minimality (condition 1).
   (c) The path $\mathsf{H}(\mathfrak{n})$ is checked for relevancy (condition 3).
   (d) The path $\mathsf{H}(\mathfrak{n})$ is checked for consistency (condition 4).
   (e) The path $\mathsf{H}(\mathfrak{n})$ is checked for duplicity (condition 2).
5. If the node should not be pruned, the algorithm checks whether there is a model N for $\mathcal{K} \cup \mathsf{H}(\mathfrak{n})$, in the following order:
   (a) If there is an $\mathsf{N} \in \mathcal{M}$ such that $\mathsf{H}(\mathfrak{n}) \subseteq \mathsf{N}$, the ABox encoding of model N is reused for the current node.
   (b) If there is no such model N, the tableau algorithm is called to search for a new model for $\mathcal{K} \cup \mathsf{H}(\mathfrak{n})$.

i. If such a model P is found, its ABox encoding is used for the current node and the model P is stored in $\mathcal{M}$ for later reuse.

ii. If such a model is not found, $\mathcal{K} \cup \mathsf{H}(\mathfrak{n})$ is inconsistent and $\mathsf{H}(\mathfrak{n})$ is a minimal hitting set. $\mathsf{H}(\mathfrak{n})$ is stored as explanation and $\mathfrak{n}$ is labelled by ✓.

6. If each node is labelled, the algorithm returns to step 3 to create and label successor nodes for the current level.

7. When all nodes are labelled, and no successor nodes can be created, the algorithm is done.

For observations consisting of a single concept or role assertion in $\mathcal{ALCHO}$ the algorithm is sound and complete for finding consistent, relevant and syntactically minimal explanations for an explanatory ABox abduction problem. Furthermore, the algorithm eventually terminates. Besides giving a proof of soundness and completeness, Pukancová and Homola (2017) explain that the worst case complexity of the algorithm is in ExpTime. As the algorithm implements the minimal hitting set algorithm, which is proven to be in NP (Reiter 1987), and calls the tableau algorithm at most once each step, the complexity of Pukancova's algorithm is equal to the complexity of the tableau algorithm. The tableau algorithm for $\mathcal{ALCHO}$ is in ExpTime, therefore the proposed algorithm is in ExpTime.

The algorithm presented above is built to deal with single ABox abduction observations. Pukancová and Homola (2017) therefore dubbed it the Single Observation Abduction (SOA) algorithm. To find explanations for multiple observations two methods are proposed in Pukancová and Homola (2018). The first approach is based on reduction to combine the multiple observations into one observation. All the assertions are first reduced to a disjunction, where the asserted individual is detached from the assertion and added as a nominal. The conjunction of all reduced assertions is then assigned to a dummy variable and used as input for the SOA algorithm. The complete implementation for multiple observations is dubbed as the ABox Abduction Algorithm (AAA). The first version of the algorithm, based on reduction, is indicated as $\mathrm{AAA_r}$.

A second approach to ABox abduction for multiple observations splits the ABox abduction problem into multiple sub problems (Pukancová and Homola 2018). The splitting approach (AAA_s) runs the SOA algorithm for each assertion in the observation separately, then adds any combination of the found explanations in the SOA algorithm as an explanation for the complete observation. As the observations can contain new individuals, which are not present in the knowledge base, the knowledge base is enriched with an assertion $\top(\mathfrak{a})$ for each individual $\mathfrak{a}$ that is present in one of the observations. After the extra individuals are added to the knowledge base, the SOA algorithm is called for each observation. The resulting explanations are stored in a general explanation set. After all the explanations are generated for each separate observation, all possible combinations of explanations are formed to compute the explanations for the complete observation. As the combination of explanations might result in some unwanted complete explanations, all the explanations are checked on consistency, relevancy and syntactic minimality again.

Both AAA_r and AAA_s are proven to terminate and to be sound and complete. Empirical proof in Pukancová and Homola (2018) shows that the reduction approach is faster in computing all explanations to a multiple observant abduction problem, while the splitting approach generates more explanations in a short amount of time. Still, as both approaches run in ExpTime, computing all explanations for a large ontology is not feasible in any application. Therefore, Pukancová and Homola (2018) introduced the possibility to search up to a certain depth in the HS-tree, by including a parameter $\mathfrak{l}$ in the algorithms. The SOA algorithm stops after an HS-tree has labelled all nodes and cannot produce any successor nodes, or has reached a full depth of $\mathfrak{l}$. From the tests that Pukancová and Homola (2017), Pukancová and Homola (2018), and Mrózek et al. (2018) conducted follows that a maximum depth of 3 is advisable for large ontologies to not let the implementations run out of memory.

2.4 Goal of this thesis

The overarching goal of this thesis is to create an application that can be used by TNO to find useful explanations for observations, using any of the ontologies they have developed, for example, to find explanations for surprising behavior of an information system, using an ontology with all the information about this information system. Ultimately an application that can generate explanations in a reasonable amount of time, and can select the best explanation (possibly with additional user input) is wanted. This is impossible in the scope of this thesis, nonetheless, it is the aim of this thesis to provide an implementation that can at least generate the possible explanations that adhere to all the constraints for ABox abduction explanations given in definition 2.10. Specifically, this thesis aims to answer the following four research questions:

1. How can the minimal hitting set algorithm be improved, such that it is sound and complete for generating semantically minimal solutions?

2. How can the tableau algorithm be optimized to generate models that lead to all and only semantically minimal solutions?

3. How can branches in a HS-tree be pruned, when its path contains parts of a semantically non-minimal solution?

4. How do the adjustments to the AAA implementation affect its performance?

The ABox abduction implementations of Pukancová and Homola (2018) and Mrózek et al. (2018) are applications to retrieve explanations to an abduction problem. However, the generated explanations are not necessarily semantically minimal. The aim of this paper is to adjust the implementation AAA of Pukancová and Homola (2018) such that it generates only semantically minimal solutions to an ABox abduction problem. One obvious solution to reach semantic minimality is to check for every explanation if it entails one of the other explanations, however, as this would mean calling the time intensive tableau algorithm $n \times (n-1)$ times

for $\mathfrak{n}$ explanations, this would not be a practical solution. Therefore, it is the aim of this paper to explore more efficient ways to detect non-minimal explanations within the algorithm of AAA, such that paths in the HS-tree that lead to semantically non-minimal explanations can be avoided or pruned.

3 | Approaches for finding Semantically Minimal ABox Abduction Explanations

One of the possible approaches for finding semantically minimal ABox abduction explanations to explore is to use an adjusted tableau algorithm. While Pukancová and Homola (2018) do optimise the minimal hitting set (MHS) algorithm to call the tableau algorithm the fewest number possible, they use the tableau algorithm as a black-box. The tableau algorithm they use has been optimised for deductive reasoning techniques, yet during the expansion of a tableau, clashes with decedents of the observation, in combination with backtracking to the last or-branch, can indicate which elements of the model lead to a direct clash with (a part of) the observation. Only the elements in a model that lead to a direct clash with a part of the observation are interesting for forming semantically minimal solutions. Another important feature of semantically minimal explanation is that it should be possible to retrieve at least one of the observations after applying at most one TBox rule to the explanation. If two TBox rules are applied to retrieve at least one observation, then the set of assertions retrieved after applying the first TBox rule would be entailed by the original set of assertions, thus the original set cannot be semantically minimal according to definition 2.4. By marking the elements that directly lead to a clash during the tableau algorithm, and only used one TBox axiom, the number of potential explanations returned by the TA algorithm could be significantly reduced. Let's call this approach the *one-axiom approach*, as the approach is based on the theory that only one TBox axiom is needed to explain at least a part of an explanation. Section 3.1 works

out the details of this approach and presents an algorithm based on this approach.

While the one-axiom approach has the potential to find semantically minimal explanations without finding all explanations first, it is an adjustment to the TA algorithm and does not use the MHS algorithm developed by Pukancová and Homola (2018). An alternative approach that builds on the algorithm of Pukancová and Homola (2018) is presented in section 3.2. This approach is aimed at optimising the process of finding semantically minimal explanations from an existing set of explanations, instead of finding explanations on its own. Therefore this approach is referred to as the *optimising approach*.

Both approaches hold a similar structure. First, a theoretical basis of the algorithm should be described. Secondly, potential proofs of soundness and completeness are provided. Lastly, an implementation of the algorithm is described. For the one-axiom approach no proof of completeness could be provided as there are some theoretical loopholes that where not possible to be solved within the time limit of this thesis. Furthermore, no working implementation could be formed of the one axiom approach due to the large number of adjustments to the optimised TA implementations that were needed. The details of the non-working implementation are described at the end of section 3.1. The optimising approach successfully introduces an algorithm, and proofs of soundness and completeness. Furthermore, an implementation of the algorithm is presented in section 3.2.2. Next to a description of the implementation this section describes some tests that can be done to compare the performance of the new implementation.

3.1 One Axiom Approach

The tableau algorithms are optimized to prove consistency, thus are optimized to find a model of a knowledge base $\mathcal{K}$. However, to find an abductive solution we are especially interested in the clashes, as a clash with the negated observation would possibly lead to an explanation. Pukancová and Homola (2018) use the minimal hitting set to find which assertions in a model can cause a clash with the observation. While this approach can stop the tableau algorithm after it has found a model, it does call the tableau algorithm numerous times, at least once for

each element in the ABox encoding of the first model. Only when no model can be found via the tableau algorithm, no elements are added to the solution and no new tableau algorithms are called. When searching for all possible explanations, every assertion that is possible in a model should be tested as an explanation, however in this paper we focus on semantically minimal explanations. A semantically minimal explanation does imply only what is necessary and not more (definition 2.4), thus should be able to explain the observation without the intermediate use of other assertions. This means that only the assertions that are closest to a clash with the negated observation are candidates for a semantically minimal explanation. Testing assertions that are not directly related to a clash are not interesting to explore. Therefore, instead of using a minimal hitting set with models found by using a tableau algorithm, this section is focused on using the clashes found during a tableau algorithm to identify semantically minimal explanations.

To identify all semantically minimal explanations during the expansion of the tableau algorithm it is important to distinguish between knowledge base clashes and observational clashes. A clash within the ABox is an observational clash if and only if one of the clashing assertions is an observation derived assertion (ODA).

Definition 3.1 (Observation derived assertion)**.** *An assertion in an extended ABox is considered to be an observation derived assertion iff:*

1. *the assertion is the negated observation* $\neg\Phi$ *itself, or*
2. *the assertion is the product of an expansion rule that is not the* $\sqcup$*- or* $\sqsubseteq$*- rule, and at least one of the parent assertions is an observation derived assertion.*

Any set of assertions that can force an observational clash can be considered a plain explanation to the observation, as every possible model for $\Gamma \cup \mathcal{E} \cup \neg\Phi$ is avoided by a forced clash (definition 2.10). However, these sets might not be minimal (semantically or syntactically), and they can even be inconsistent or irrelevant. To check for the additional constraints new tableau algorithms could be called on the sets of assertions that form a potential explanation. Still there is already information in the original tableau algorithm that can be used to determine

if a set of assertions adheres to the constraints. To understand how a tableau algorithm can be used to identify minimal and relevant explanations, let us first take a closer look at how the expansion rules introduce new assertions to the ABox and which rule can contribute to a relevant explanation. As explained in section 2.2.4, there are six expansion rules that can be used to expand a $\mathcal{ALCHO}$ knowledge base. Each rule introduces at least one new assertion when it is applied to an ABox and can therefore introduce a clash. When an observational clash occurs during a tableau algorithm for a knowledge base $\mathcal{K} \cup \Phi$, then the path that leads to this observational clash can be forced by adding a set of assertions $\mathcal{E}$ that ensure that any other path in the tableau of $\mathcal{K} \cup \Phi \cup \mathcal{E}$ leads to a clash. Which assertions should be added to $\mathcal{E}$ is dependent on the expansion rules that are used in the path from the root of the tableau to the observational clash. When backtracking from an observational clash to the root of the tableau, one could add the following assertions to an explanation to ensure that the observation is implied:

$\sqcap$-rule The conjunction rule adds two new assertion from one parent assertion. Both new assertions are necessary, therefore adding the parent assertion of an $\sqcap$-rule that leads to an observational clash to the knowledge base will implies both new assertions and therefore imply the observation. Note that constructing a new knowledge base with only the parent assertion will also imply the observation in this case, $\mathcal{E} \vDash \Phi$, thus these explanations are not relevant. Any set of assertions that implies the parent assertion, will also imply the observation.

$\sqcup$-rule The disjunction rule introduces a new branch in the tableau, each with a new assertion. If one of these new assertions leads to an observational clash, then the tableau should be forced to take that path. In other words, the other path should inevitably lead to a clash. Let us illustrate the situation with a parent assertion $\mathsf{I} : \mathsf{D} \sqcup \mathsf{E}$ on which a $\sqcup$-rule can be applied. If the assertion $\mathsf{I} : \mathsf{E}$ leads to an observational clash, then the easiest way to ensure any tableau takes the branch to the observational clash is by adding the negation of the root assertion of the branch that is closest to the explanation, thus $\mathsf{I} : \neg\mathsf{D}$. By adding both the parent assertion $\mathsf{I} : \mathsf{D} \sqcup \mathsf{E}$ and the negated root assertion $\mathsf{I} : \neg\mathsf{D}$ to

the knowledge base the observation is implied. Note that due to the parent assertion the explanation itself already explains the observation without any knowledge from the knowledge base. Therefore, the parent assertion should be implied instead of added to the explanation itself in order for the explanation to be relevant.

∃-rule The existential quantifier has one parent assertion and adds a new individual to the ABox. As the ∃-rule only adds new individuals it can never lead directly to an observational clash. In fact the only time that an existential quantifier will lead to an observational clash is when the the observation itself contains a quantifier. If the ∃-rule indirectly leads to an observational clash, and the parent assertion is not an observation derived assertion in itself, then the parent assertion would form an irrelevant explanation to the observation.

∀-rule The universal quantifier has two parent assertions and adds only one new assertion. If this assertion leads to an observational clash, then both parent assertions should be implied by $\mathcal{K} \cup \mathcal{E}$ for the observation to be implied as well. One of the parent assertions is a role assertion and the other is a concept assertion of the form $\mathsf{I} : \forall \mathsf{r}.\mathsf{C}$. As the only new roles that are introduced via the tableau algorithm must have a new individual as their subject, and the ∀-rule only asserts new concepts to the subject of the role parent assertion, a universal quantifier can only introduce an observational clash when the role parent assertion is already implied by $\mathcal{K}$. Thus the parent assertion of the form $\mathsf{I} : \forall \mathsf{r}.\mathsf{C}$ can be added to the explanation. There is only one case in which this explanation is not relevant, which is the case if the observation is equal to $\mathsf{I} : \forall \mathsf{r}.\mathsf{C}$, else the explanation given is relevant.

{}-rule A nominal replaces one individual name with another individual name as they represent the same individual if they are both assigned to the same nominal. This will only lead to an observational clash if one of the parent individuals is the observed individual itself. If the {}-rule is applied to the observed individual and this rule leads to a clash, then the assertion $\mathsf{I} : \{\mathsf{J}\}$ must be added to the explanation where I is the observed individual name. The assertion that causes an observational clash $\mathsf{J} : \mathsf{C}$ should still be checked for explanations.

$\sqsubseteq$-rule The GCI rule assigns the TBox rules to every individual in the ABox. As every individual in the ABox must adhere to every TBox rule, the new assertions are implied by the knowledge base. If a $\sqsubseteq$-rule inevitably leads to an observational clash, then the observation itself is already implied by the knowledge base and the abduction query is not relevant. If the $\sqsubseteq$-rule only leads to an observational clash under certain conditions then those conditions together form a complete explanation. There is no need to add to an explanation beyond a $\sqsubseteq$-rule, as is will only lead to non-minimal explanations.

When dissecting the influence of each rule as done above, some clear explanations come through when applying a tableau algorithm. As the $\sqsubseteq$-rule introduces assertions that are implied by the knowledge base, they can end a search path. However this only works if there are no open assertions before the $\sqsubseteq$-rule is applied. By only applying a $\sqsubseteq$-rule when no other expansion rule is possible one ensures that the expansion rules that are applied after the $\sqsubseteq$-rule use information presented by this rule, because when the expansion rule did not need any information of the new assertion, then the expansion rule could have already been applied before applying the $\sqsubseteq$-rule.

Another interesting rule is the nominal rule, as it can merge two individuals thereby removing and adding an unknown number of assertions. Because of the unknown number of assertions it is difficult to pinpoint which new individual led to the observational clash, when a nominal rule caused an observational clash. This especially holds in cases where a nominal is part of an GCI, thus can be introduced during the expansion of an ABox as presented in example 7.

Example 7. For an observation $\Phi = \{\mathsf{Mary} : \mathsf{Human}\}$, an ABox $\mathcal{A} = \{(\mathsf{John}, \mathsf{Judy}) : \mathsf{hasChild}\}$, and a TBox $\mathcal{T} = \{\mathsf{C} \sqsubseteq \{\mathsf{Judy}\}, \mathsf{Human} \sqsubseteq \forall \mathsf{hasChild}.\mathsf{Human}\}$ there are several valid, minimal explanations:

- The explanation $\mathsf{E}_1 = \{\mathsf{John} : \mathsf{Human}, (\mathsf{John}, \mathsf{Mary}) : \mathsf{hasChild}\}$ might be the most intuitive explanation, adding a new relation and assuming that John is human.
- The explanation $\mathsf{E}_2 = \{\mathsf{John} : \forall \mathsf{hasChild}.\mathsf{Human}, \mathsf{Mary} : \{\mathsf{Judy}\}\}$ is less intuitive, it assumes that Mary and Judy are the same person. Given that it is already known that

Judy is a child of John, it is not necessary abstract to the assertion that John is human, the assumption that John only has human children produces already is a relevant solution.

- The explanation $\mathsf{E}_3 = \{\mathtt{Judy} : \mathtt{Human}, \mathtt{Mary} : \mathsf{C}\}$ assumes that Mary if of the type C and that Judy is human. As Mary must be the same individual as Judy, indicated by the first TBox rule, Mary must be human too. This explanation is both relevant and minimal.

The second explanation requires two applications of a $\sqsubseteq$-rule, which makes it different from the standard methods of retrieving explanations.

The situation in example 7 is theoretically possible but practically rarely used. Using a GCI in the form of $\mathsf{C} \sqsubseteq \{\mathrm{I}\}$ means that every instance of concept C must be the individual I, which makes the concept C more like a property of the individual than that it acts as a concept. As a GCI in de form $\mathsf{C} \sqsubseteq \{\mathrm{I}\}$ would rarely be used in a well modelled ontology, and as it causes difficult situations, let us first look at an algorithm that can be used to find explanations on knowledge bases that have no nominals included in any TBox axiom.

3.1.1 Adjusted Tableau Algorithm without nominals GCI's

Whereas the GCI rule only adds implied assertions, the other rules can apply assertions that are not necessarily entailed by the knowledge base. The information that these rules entail is therefore interesting to add as an explanation to the observation. To find semantically minimal explanations with the use of the tableau algorithm an adjusted tableau algorithm can be used (algorithm 1).

The Adjusted Tableau Algorithm (ATA) applies expansion rules that are formulated in figure 2.1, yet unlike a standard tableau algorithm it only applies a $\sqsubseteq$-rule when no other rules are applicable. Furthermore, the ATA keeps track of all the observational derived assertions, such that relevant clashes can be identified. To check for clashes algorithm 2 is used. Because the clash check is done each time an expansion rule is applied, we know that an ABox is consistent

Algorithm 1: Adjusted Tableau Algorithm (ATA)

Data: Knowledge base $\mathcal{K} = \mathcal{A} \cup \mathcal{T}$
Observation Φ
Result: A set of semantically minimal explanations $\mathcal{E}$

```
Initiate tableau with A ∪ ¬Φ at the root;
while no model is found do
    if ⊓-, ⊔-, ∃-, ∀-, or {}-rule can be applied then
        Apply rule;
        if one of the parent assertions is an ODA then
            Mark the new assertions as observational derived assertions;
        Check for clashes;
    else if ⊑-rule can be applied then
        Apply rule;
        Check for clashes;
    else
        Close the path and mark as model;
while not at the root node do
    Backtrack to the nearest non-closed alternative path;
    if ⊓-, ⊔-, ∃-, ∀-, or {}-rule can be applied then
        Apply rule;
        if one of the parent assertions is an ODA then
            Mark the new assertions as observational derived assertions;
        Check for clashes;
    else
        Close path;
```

before the application of an expansion rule. Therefore, it is only necessary to test the new assertions for clashes. For each new assertion it should be checked whether the negation of that assertion is already in the ABox. When it is, then it should be determined whether it is a normal clash or an observational clash. When an observational clash is found, rules that are applied to get to this clash are examined. Both a $\sqcup$-rule, and a $\forall$-rule can add assertions to the explanation. Only a $\forall$-rule or a $\sqsubseteq$-rule can close an explanation. When an explanation is closed, then the set of assertions that form an explanation are returned. Note that this can be an empty set when there are no assertions added to the explanation. In this case there was no interesting explanation to be found, either because there only was a non-observational clash, or because the observational clash was entailed. If no clash is found the function returns false.

Both ATA and the Clash check have conditional statements that check if a model is already found. When no model can be found for $\mathcal{K} \cup \neg\Phi$, then the observation is already entailed, thus a model has to be found for any explanation to be explanatory (definition 2.10). Once a model is found, the ABox cannot be expanded any further, otherwise it would not have been a model (definition 2.7). The $\sqsubseteq$-rule ensures that every individual in the ABox is asserted to every GCI in the TBox, thus when no $\sqsubseteq$-rule can be applied, then every individual in the ABox is already asserted to every GCI of the TBox. As semantically minimal explanations only can be found up to the closest $\sqsubseteq$-rule. Alternative paths to a model that do not lead to an observational clash without the application of a new $\sqsubseteq$-rule will not lead to a new semantically minimal solution, as the same rule will already be applied somewhere in the path to the model. Therefore, alternative paths to a model only have to be expanded with ABox expansion rules. As soon as the $\sqsubseteq$-rule is the only rule that can be applied to an alternative path, this path can be left unexplored.

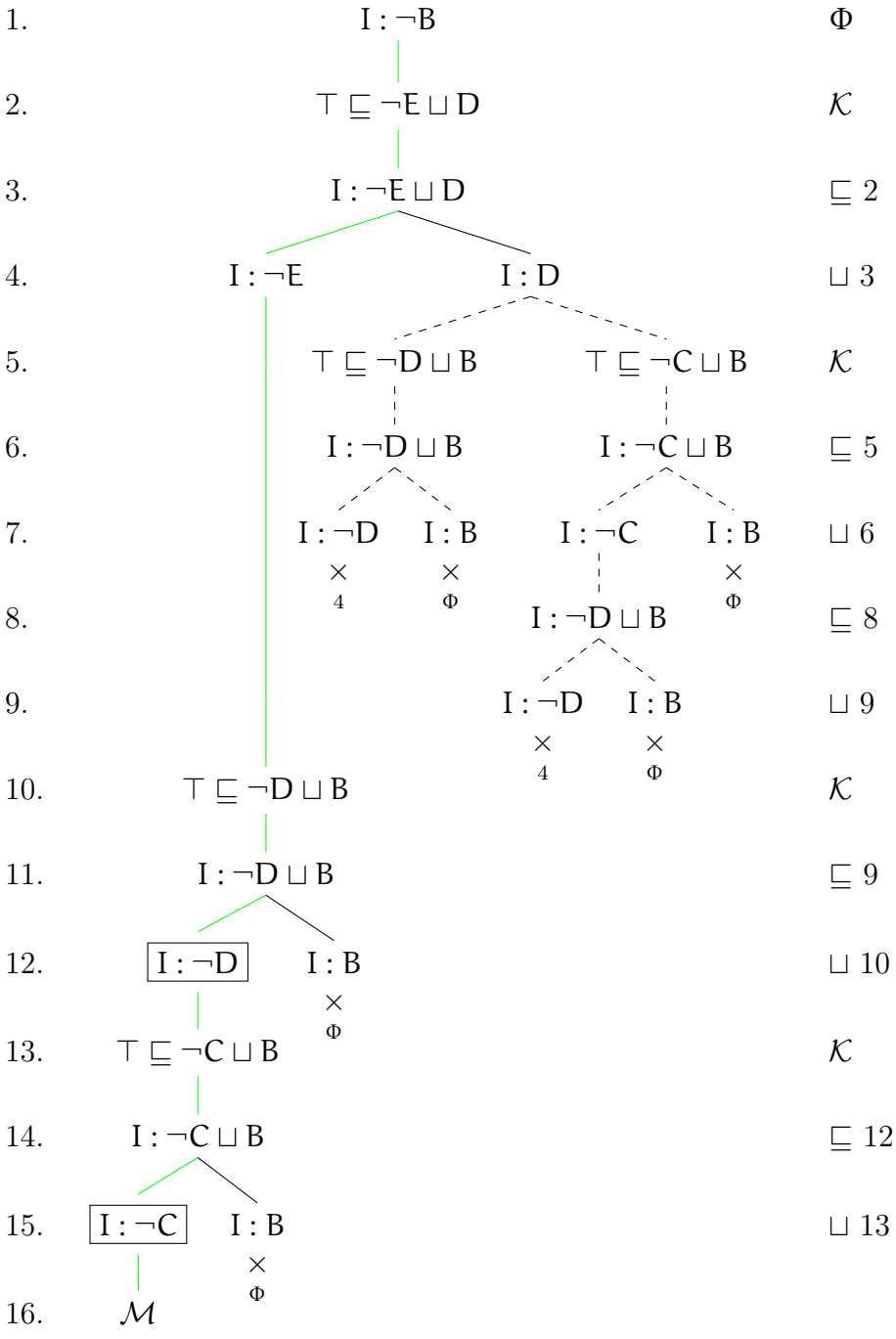
In example 8 this is illustrated by drawing out all the possible paths of the ABox extensions for the given knowledge base. The green base line is a path to a model (in this case the only model). Branches from the green lines are alternative paths that are introduced by disjunctions. From line 5 to line 9 the alternatives are shown in dashed line when one would still allow new GCI axioms to be applied. While the dashed paths do encounter other observational clashes, no new explanations are found as they represent similar clashes as those found on line 12 and

Algorithm 2: Clash check

Data: ABox $\mathcal{A}$
Set of new assertions $\mathcal{A}^+$
List of visited notes N
Observation Φ
Result: A possible explanation E

```
1  foreach assertion I : C ∈ A+ do
2      if I : ¬C ∈ A then
3          if I : C or I : ¬C is an ODA then
4              while current node is not the root note do
5                  if last rule is ⊓-, {}- or ∃-rule then
6                      Move one node up;
7                  else if last rule is ⊔-rule then
8                      if alternative path leads to a model then
9                          E ⟶ E ∪ I : ¬C, s.t. I : C is the root assertion of alternative path;
10                     Move one node up;
11                 else if last rule is ∀-rule then
12                     if parent assertion of the form I : ∀r.C ∈ {Φ ∪ ¬Φ} then
13                         Move one node up;
14                     else
15                         if alternative path leads to a model then
16                             E ⟶ E ∪ I : ∀r.C;
17                         Return E;
18                 else if last rule is ⊑-rule then
19                     Return E;
20             Return E;
21         else
22             Normal clash found, return ∅;
23 No clash is found, return ⊥;
```

Example 8. For an observation $\Phi = \{I : B\}$, and a knowledge base $\mathcal{K}$ where $\mathcal{A} = \{I : A\}$ and $\mathcal{T} = \{C \sqsubseteq B, D \sqsubseteq B, E \sqsubseteq D\}$, the following tableau can be constructed:



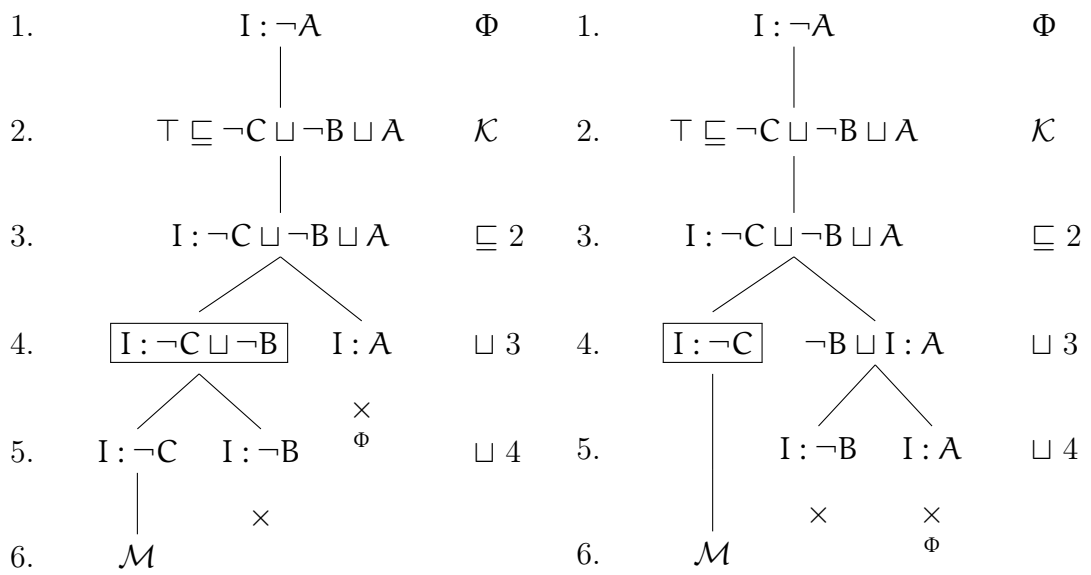
15.

Using the adjusted tableau algorithm most semantically minimal explanations are found. Still there are some cases in which the adjusted tableau algorithm fails to find the correct explanations. The next sections provide some extra adjustments to the algorithm to ensure that every semantically minimal explanation is found and that every explanation that is found is in fact a semantically minimal explanation. All the simple assertions produced by the ATA are in fact semantically minimal explanations. However there are situations where complex assertions found via the ATA are not minimal explanations. Section 3.1.2 analyses the problem with complex assertions as explanations. Furthermore, due to the use of quantifiers some extra explanations can be found which are semantically minimal, but do not emerge from the ATA, section 3.1.4 and section 3.1.5 focus on finding all explanations for cases with universal and existential quantifiers respectively.

3.1.2 Non-minimal explanations

The adjusted tableau algorithm guarantees that the presented explanations are explanatory, relevant and minimal in the degree that it does not assume more information than is needed to explain the observations. For simple assertions this is enough to grantee that the given explanation is both semantically and syntactically minimal. However, complex assertions can still be semantically or syntactically non-minimal, as information that is entailed by the knowledge base can minimize a complex assertion that is presented as explanations by the ATA. As illustrated by example 9 there are complex assertions returned as an explanation that contain a conjunction, which are not necessarily syntactically minimal, as part of the assertion might be already explained by the knowledge base itself. A crude yet effective way to ensure that all the complex assertions that contain a conjunction are syntactically minimal is to check if one of conjuncts is already entailed by the knowledge base.

Example 9. For an observation $\Phi = \{\mathsf{I} : \mathsf{A}\}$, and a knowledge base $\mathcal{K}$ where $\mathcal{A} = \{\mathsf{I} : \mathsf{B}\}$ and $\mathcal{T} = \{\mathsf{C} \sqcap \mathsf{B} \sqsubseteq \mathsf{A}\}$, the following tableaux could be constructed:

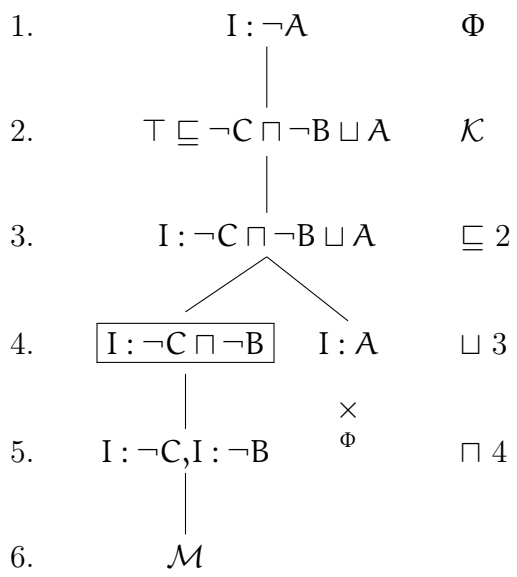


If the ABox is expanded as displayed in the second tableau, a simple concept assertion $I : C$ is given as an explanation by the ATA. However, the algorithm can also expand in the order displayed by the first tableau. In this case the complex assertion $I : C \sqcap B$ is returned as an explanation. The explanation $I : C \sqcap B$ is syntactically bigger than the explanation $I : C$, $\{I : C\} \subset \{I : C, I : B\}$, thus the assertion $I : C \sqcap B$ is not a syntactically minimal solution according to definition 2.10. As the ATA does not require a particular order for the expansion rules to be applied, both tableaux can be produced.

3.1.3 Redundant disjuncts

Another interesting situation is when a disjunction is given, while one of the disjuncts is an inconsistent explanation. In this situation the given explanation would seem non-minimal with regards to the consistent explanation that can be formed by one of the disjuncts. However, given the formal minimality constraints given in definition 2.10 the complete assertion with disjunction is considered to be a minimal explanation. Let us look at the following example to understand the issue with this explanation.

Example 10. For an observation $\Phi = \{I : A\}$, and a knowledge base $\mathcal{K}$ where $\mathcal{A} = \{I : \neg B\}$ and $\mathcal{T} = \{C \sqcup B \sqsubseteq A\}$, the following tableau is constructed:



In this case one explanation is given by the ATA, $\mathcal{E}_1 = \{I : C \sqcup B\}$. As all explanations can be written as one explanation where all assertions are joint via a disjunction, it seems intuitive that every explanation that contains an assertion with a disjunction can be written as two separate explanations. While this is the case in most situations, in this particular example only one explanation would be consistent, $\mathcal{E}_2 = \{I : C\}$. The explanation $I : B$ is inconsistent because of the ABox assertion $\mathcal{E}_3 = \{I : \neg B\}$. One might intuitively assume that this makes $\{I : C \sqcup B\}$ non-minimal, however by the definitions of semantic and syntactic minimality $\{I : C \sqcup B\}$ cannot be excluded on the basis of non-minimality. As the explanations do not contain the same elements they are not syntactically comparable, i.e. $\mathcal{E}_1 \nsubseteq \mathcal{E}_2$ and $\mathcal{E}_2 \nsubseteq \mathcal{E}_1$. As $\mathcal{E}_1 \vDash \mathcal{E}_2$ it is certainly true that $\mathcal{K} \cup \mathcal{E}_1 \vDash \mathcal{E}_2$, which means that the explanation $\{I : C \sqcup B\}$ is semantically minimal according to definition 2.10. In fact a disjunction is always semantically weaker then one of its elements as it allows for more uncertainty and thus assumes less. In this particular situation it also holds that $\mathcal{K} \cup \mathcal{E}_2 \vDash \mathcal{E}_1$, meaning that $\mathcal{E}_1$ and $\mathcal{E}_2$ are semantically equal.

This example actually shows an even deeper problem with semantically minimal explanations, as the disjunction of an explanation with any concept that is negatively asserted in the knowledge base is a semantically minimal explanation, i.e. given a set of semantically minimal explanations $\mathcal{E}$, for every $\{I : C\} \in \mathcal{E}$ and every $C \neq D$, if $\{I : \neg D \in \mathcal{K}\}$ then $\{I : C \sqcup D\} \in \mathcal{E}$. Furthermore a disjunction of all the explanations together is semantically minimal, or at least semantically equal, to all other explanations because $\{I : C\} \vDash \{I : C \sqcup D\}$. This means that

for each abduction problem exactly one semantically minimal explanation can be given: the disjunction of all explanations. However, a portion of the disjuncts might not contain relevant information as it is inconsistent with the knowledge base on its own. While the proposed method does not return explanations with random negated concepts, formally they should be accounted for. Therefore, a fifth constraint should be added to our definition of an explanation.

Definition 3.2 (Independent explanation)**.** *Let $\Phi \subseteq \mathcal{O}$ be an ABox abduction problem, with knowledge base Γ and ABox assertion $\Phi \subseteq \mathcal{O}$. The set of solutions to an ABox abduction problem contains all valid explanations, $\mathcal{S} = \{\mathcal{E} | \mathcal{E}$ is an explanation for $\langle \Gamma, \Phi \rangle\}$.*

An explanation $\mathcal{E} \subseteq \mathcal{H}$ can be described as an independent explanation of Φ with respect to Γ if and only if $\Gamma \nvDash \neg \mathsf{C}_i$ for each C_i in $\mathcal{E} = \mathrm{I} : \mathsf{C}_1 \sqcup \cdots \sqcup \mathsf{C}_n$.

The proposed method might produce some distinctions that do not follow the independence constraint as presented in example 10. To check if an explanation consists of only independent disjuncts each disjunct on its own should be tested for consistency instead of the explanation as a whole. As $\{\mathrm{I} : \mathsf{C}\} \vDash \{\mathrm{I} : \mathsf{C} \sqcup \mathsf{D}\}$ and $\{\mathrm{I} : \mathsf{D}\} \vDash \{\mathrm{I} : \mathsf{C} \sqcup \mathsf{D}\}$ hold, the assertion $\mathcal{K} \cup \{\mathrm{I} : \mathsf{C} \sqcup \mathsf{D}\}$ must be consistent iff $\mathcal{K} \cup \{\mathrm{I} : \mathsf{C}\}$ or $\mathcal{K} \cup \{\mathrm{I} : \mathsf{D}\}$ is consistent. Therefore checking the consistency of the whole conjunction is not necessary, as it can be deduced from the consistency of the disjuncts. It is necessary to check all the disjuncts for their consistency, as a disjunct should be removed from the explanation when it is not consistent, following the independence constraint (definition 3.2).

3.1.4 The universal quantifier

As shown at the introduction of our algorithm, in section 3.1.1, the universal quantifier takes two assertions as input to introduce a new assertion in the ABox. The ATA, as it is currently described, can only find explanations that contain universal quantifiers which are present in the TBox. This means that both roles as an explanation, and new universal quantifiers as an explanation are not found. To discover all explanations that can be retrieved via an universal

quantifier some additional tests should be done. First let us look at the missing explanations that contain a new role.

When adding a role assertion leads to an observational clash, this role is (part of) an explanation to the observation. A role assertion in an explanation is only relevant when an existential quantifier is encountered during the expansion of an ABox. Adding both a role and the corresponding existential quantifier to an explanation would form explanations that are not relevant as they explain the observation without using any information in the knowledge base. For example the observation $\Phi = \{I : A\}$ could be explained by $\mathcal{E} = \{J : \forall r.A, (J, I) : r\}$, regardless of the information in a knowledge base $\mathcal{K}$. However, when an assertion with existential quantifier can be generated by the knowledge base $\mathcal{K} \cup \{J : D\} \vDash \{J : \forall r.C\}, D \neq \forall r.C$ such that $I : C \vDash I : A$, then an explanation $\mathcal{E} = \{(J, I) : r, J : D\}$ is relevant. Moreover, when $\mathcal{K} \vDash \{J : \forall r.C\}$ such that $I : C \vDash I : A$, then the relevant explanation $\mathcal{E} = \{(J, I) : r\}$ can be formed.

As the ATA finds at least one model for a consistent $\mathcal{K}$, and every assertion that is entailed by $\mathcal{K}$ must be in every model for $\mathcal{K}$, the assertion $J : \forall r.C$ must be expanded somewhere during the ATA for $\mathcal{K}$ if $\mathcal{K} \vDash \{J : \forall r.C\}$. Furthermore, as the ATA checks every alternative branch to the found model, up to a new TBox rule, all assertions $J : \forall r.C$ that can be generated by a TBox rule in $\mathcal{K}$ will be encountered during the ATA. To check if adding a role can lead to an observational clash an alternative branch can be created. In this alternative branch the role $(J, I) : r$ is added to the ABox, with J being the individual that is assigned to the concept containing the universal quantifier $\forall r.C$ and I being the individual of the observation. Adding this alternative path is redundant when the role $(J, I) : r$ already is implied by the knowledge base. As there is no rule that can add new role assertions to existing individuals in the ABox (figure 2.1) except for the nominal rule, and we have decided to ignore ontologies that contain GCI's of the form $C \sqsubseteq \{I\}$, we can assume that all the role assertions that are entailed by $\mathcal{K}$ are already in the ABox when the first TBox rule is applied. Therefore, an alternative path that adds the role assertion $(J, I) : r$, should be opened whenever an assertion $J : \forall r.C$ is encountered and the assertion $(J, I) : r$ is not present in the ABox. This extra expansion rule is illustrated in figure 3.1.

$\forall_2$-rule: if 1. $\{a : \forall r.C\} \subseteq \mathcal{A}$, and
2. $b : C \notin \mathcal{A}$
3. $(a, b) : r \notin \mathcal{A}$
then $\mathcal{A}' \longrightarrow \mathcal{A} \cup \{b : C, (a, b) : r\}$.

Figure 3.1: Extra $\forall$ expansion rules for abduction via tableau.

When the alternative branch leads to an observational clash, then the added role assertion is part of a potential explanation. The explanation can be checked similar to the traditional explanation check, with the exception that the added role assertion is added to the explanation instead of the assertion that contains the universal quantifier, and that the explanation is not closed after the $\forall$-rule. The branch can be expanded until a TBox rule has to be applied, if there is no observational clash found before that point the potential explanation is not semantically minimal.

To find the explanations that contain an assertion of the form $J : \forall r.C$, that cannot be generated by the knowledge base, all the roles in the ABox should be checked. Only if there is a role $(J, I) : r$ entailed by the ABox, where I is the individual the the observation, then an assertion $J : \forall r.C$ can be given as explanation, where C is the concept that I is assigned to in the observation. Again, because roles cannot be asserted to existing individuals, except with the use of a nominal, all the roles that are in any model for $\mathcal{K}$ are in every model for $\mathcal{K}$ and all the roles that are not in a model for $\mathcal{K}$ are not in any model for $\mathcal{K}$. Therefore, the ABox encoding of the model formed during the ATA can be used to select all roles $(J, I) : r$, where I is the individual of the observation. For every role of this form an extra explanation $J : \forall r.C$ can be added to the set of explanations, if and only if there does not already exist an explanation $J : \forall r.C$, with C being the concept the individual I is asserted to in the observation.

3.1.5 Observations with existential quantifiers

The adjusted tableau algorithm finds all explanations that contain the same individual. However, when a quantifier is present in the observation, an explanation can also be provided

by asserting individuals to the object of the quantifier. This is only the case for existential quantifiers, as universal quantifiers require every individual to have a certain assertion. For the existential quantifier any individual that can be added as an object to the quantifier's role and that is still consistent with the knowledge base can be added as an explanation. For example, the observation $\Phi = \mathsf{Mary} : \exists \mathsf{hasPet.Cat}$ added to a knowledge base with $\mathcal{A} = \mathsf{Felix} : \mathsf{Cat}, (\mathsf{Mary}, \mathsf{Bello}) : \mathsf{hasPet}, \mathsf{Whiskers} : \mathsf{Striped}$ and $\mathcal{T} = \mathsf{CatLady} \sqsubseteq \exists \mathsf{hasPet.Cat}$ cannot only be explained by the assertion $\mathsf{Mary} : \mathsf{CatLady}$, yet can also be explained by $(\mathsf{Mary}, \mathsf{Felix}) : \mathsf{hasPet}$, $\mathsf{Bello} : \mathsf{Cat}$ or even $\{\mathsf{Whiskers} : \mathsf{Cat}, (\mathsf{Mary}, \mathsf{Whiskers}) : \mathsf{hasPet}\}$. As the tableau method never adds new roles to existing individuals, they have to be found in a different way. Furthermore, the tableau algorithm checks whether Φ is entailed by the knowledge base, therefore we know there is no individual X such that $\mathcal{K} \models \{(\mathsf{I}, \mathsf{X}) : \mathsf{r}, \mathsf{X} : \mathsf{C}\}$ given an observation of the form $\mathsf{I} : \exists \mathsf{r.C}$. In other words, all individuals $\mathsf{X} \in \mathbf{I}$ should be checked whether can be added as a potential object. To check whether an individual can be used to form an explanation to $\mathsf{I} : \exists \mathsf{r.C}$, and which form the explanation should have, follow these steps for each individual $\mathsf{X} \in \mathbf{I}$:

1. First consistency of adding the role should be checked. Therefore, check if $\mathcal{K} \cup \{(\mathsf{I}, \mathsf{X}) : \mathsf{r}\}$ is consistent. If it is consistent check if it is not entailed by checking the consistency of $\mathcal{K} \cup \{(\mathsf{I}, \mathsf{X}) : \neg \mathsf{r}\}$.

   (a) If $\mathcal{K} \cup \{(\mathsf{I}, \mathsf{X}) : \mathsf{r}\}$ is inconsistent there is no possible explanation that is formed by the individual X, return false.

   (b) If $\mathcal{K} \cup \{(\mathsf{I}, \mathsf{X}) : \neg \mathsf{r}\}$ is inconsistent the role $(\mathsf{I}, \mathsf{X}) : \mathsf{r}$ is already entailed by the knowledge base, continue from step 3.

   (c) If both $\mathcal{K} \cup \{(\mathsf{I}, \mathsf{X}) : \mathsf{r}\}$ and $\mathcal{K} \cup \{(\mathsf{I}, \mathsf{X}) : \neg \mathsf{r}\}$ are consistent, an explanation that includes $(\mathsf{I}, \mathsf{X}) : \mathsf{r}$ is possible, continue from step 2.

2. Check if $\mathsf{X} : \mathsf{C}$ is already entailed by the knowledge base. Iff $\mathcal{K} \cup \{\mathsf{X} : \neg \mathsf{C}\}$ is inconsistent $\mathsf{X} : \mathsf{C}$ is entailed, return $(\mathsf{I}, \mathsf{X}) : \mathsf{r}$ as a minimal explanation. If $\mathsf{X} : \mathsf{C}$ is not entailed by $\mathcal{K}$ $\mathsf{X} : \mathsf{C}$, or a part of $\mathsf{X} : \mathsf{C}$, should be added to the explanation to present a valid solution.

To check whether $X : C$ can be minimized and does not introduce inconsistencies continue from step 3.

3. The assertion $X : C$ can contribute to a possible explanation iff adding $X : C$ to the knowledge base does not result in an inconsistency, as an inconstant knowledge base cannot become constant by adding something and every explanation should be consistent with the knowledge base (definition 2.10). If $\{X : C\} \cup \mathcal{K}$ is consistent, continue to check for minimisation of $X : C$. If $\{X : C\} \cup \mathcal{K}$ is inconsistent $X : C$ is cannot contribute to a potential explanation, return false.

4. Iff C is of the form $C \sqcup E$ or $C \sqcap E$ there is a possible smaller explanation than $\{X : C, \dots\}$. For all other forms of C, return $\{X : C, \dots\}$ as minimal explanation.

   (a) If C is of the form $D \sqcup E$ either $X : \neg D$ or $X : \neg E$ has to be entailed by $\mathcal{K}$ for a smaller explanation to exist. When $\mathcal{K} \vDash X : \neg D$ then $\mathcal{K} \cup X : (D \sqcup E) \vDash X : E$, as $\{X : \neg D\} \cup \{X : (D \sqcup E)\} \vDash X : E$. Therefore, $X : E$ is semantically equal to $X : (D \sqcup E)$ if $\mathcal{K} \vDash X : \neg D$ (definition 2.4). However, $X : (D \sqcup E)$ cannot be defined as independent (definition 3.2).

      i. Iff $\mathcal{K} \cup \{X : D\}$ is inconsistent, then $X : E$ is smaller than $X : C$. Check for sub-assertions with $X : E$ as input (step 4).

      ii. Iff $\mathcal{K} \cup \{X : E\}$ is inconsistent, then $X : D$ is smaller than $X : C$. Check for sub-assertions with $X : D$ as input (step 4).

      iii. If both $\mathcal{K} \cup \{X : D\}$ and $\mathcal{K} \cup \{X : E\}$ are consistent there is no smaller explanation than $X : C$, return $X : C$ as explanation.

   (b) If C is of the form $D \sqcap E$ either $X : D$ or $X : E$ has to be entailed by $\mathcal{K}$ for a smaller explanation to exist.

      i. Iff $\mathcal{K} \cup \{X : \neg D\}$ is inconsistent, then $X : E$ is smaller than $X : C$. Check for sub-assertions with $X : E$ as input (step 4).

      ii. Iff $\mathcal{K} \cup \{X : \neg E\}$ is inconsistent, then $X : D$ is smaller than $X : C$. Check for sub-assertions with $X : D$ as input (step 4).

iii. If both $\mathcal{K} \cup \{\mathsf{X} : \neg\mathsf{D}\}$ and $\mathcal{K} \cup \{\mathsf{X} : \neg\mathsf{E}\}$ are consistent there is no smaller explanation than $\mathsf{X} : \mathsf{C}$, return $\mathsf{X} : \mathsf{C}$ as explanation.

Algorithm 3 implements these checks to find all explanations for an observation that contains an existential quantifier. Note that it is first checked if there is already an explanation that is implied by the observation itself. This is an important step, as $\mathsf{I} : \exists\mathsf{r}.\mathsf{C}$ cannot entail an assertion about another existing individual. If there is an explanation E such that $\mathcal{K} \cup \{\mathsf{I} : \exists\mathsf{r}.\mathsf{C}\} \models \mathsf{E}$, then every obtained explanation E' must entail E by transitivity. As $\mathcal{K} \cup \{\mathsf{I} : \exists\mathsf{r}.\mathsf{C}\} \models \mathsf{E}'$ is not possible for any E' containing an individual $\mathsf{J} \in \mathbf{I}, \mathsf{J} \neq \mathsf{I}$, $\mathcal{K} \cup \mathsf{E} \not\models \mathsf{E}'$ holds, which means E' cannot be semantically minimal. Furthermore, the minimisation steps introduced above ensure syntactic minimality and independence. However, as this should be checked for every returned explanation, it can be left out of this.

Algorithm 3: Extra explanations for Observations with Existential Quantifier (OEQ-check)

Data: Knowledge base $\mathcal{K}$
An observation $\{\mathsf{I} : \exists\mathsf{r}.\mathsf{C}\}$
A set of found explanations $\mathcal{E}_{\text{input}}$
Result: A set of semantically minimal explanations $\mathcal{E}$

```
foreach E ∈ E_input do
    if K ∪ {I : ∃r.C} ⊨ E then
        return E ;
foreach Individual X ∈ I do
    if K ∪ {(I, X) : r} is consistent then
        if K ∪ {(I, X) : ¬r} is consistent then
            if K ∪ {X : ¬C} is inconsistent then
                E ← E ∪ {(I, X) : r};
                Continue to next individual;
            if There exists an X : D s.t. K ∪ {X : D} ⊨ X : C then
                E ← E ∪ {(I, X) : r, X : D} Continue to next individual;
        if K ∪ {X : ¬C} is consistent then
            E ← E ∪ X : C;
return E
```

3.1.6 Minimizing consistency checks with use of found models

The method for checking explanations that contain an $\exists$ concept does at least one and a maximum of $4+\mathfrak{x}$ consistency checks per individual, where $\mathfrak{x}$ is the number of atomic concepts in the object's concept. Consistency checks are relatively expensive to do, as they have a maximum of ExpTime. Especially when there are many individuals in the knowledge base this could delay the algorithm. To minimize the number of consistency checks one can first do a rough consistency check. Following the definition of a model a knowledge base $\mathcal{K}$ is consistent if and only if there exists a model for $\mathcal{K}$ (definition 2.7). For the normal explanations search we have already found at least one and possibly more models for $\mathcal{K} \cup \neg\Phi$. As $\mathcal{K} \cup \neg\Phi$ only adds assertions to the knowledge base we can say that $\mathcal{K} \subseteq \mathcal{K} \cup \neg\Phi$. For any $\mathcal{K}_{\mathtt{and}}$, such that $\mathcal{K} \subseteq \mathcal{K}_{\mathtt{and}}$, every assertion in the ABox of $\mathcal{K}$ is in the ABox of $\mathcal{K}_{\mathtt{and}}$, and every GCI in the TBox of $\mathcal{K}$ is in the TBox of $\mathcal{K}_{\mathtt{and}}$. According to definition 2.7 any interpretation that satisfies both the ABox and TBox of a knowledge base is a model. Thus, any model for $\mathcal{K}_{\mathtt{and}}$ must be a model for $\mathcal{K}$.

If $X^{\mathcal{M}} \in F^{\mathcal{M}}$ for any model $\mathcal{M}$ of $\mathcal{K}$, then $X : F$ is consistent. A consistency check on $\mathcal{K} \cup \{X : F\}$ has to be done iff $X^{\mathcal{M}} \in F^{\mathcal{M}}$ is not true for any model $\mathcal{M}$ of $\mathcal{K}$. For a simple concept F this can easily be checked by checking if $X : \neg F$ is in the intersection of the ABox encoding of every model for $\mathcal{K}$, i.e. $X : \neg F \in \bigcap_{M \in \mathcal{M}} \mathcal{B}_M$. If $X : \neg F \in \bigcap_{M \in \mathcal{M}} \mathcal{B}_M$, then $X : F$ is not consistent, else $X : F$ is consistent. As we are not certain whether we computed every model for $\mathcal{K}$ (there can exist models for $\mathcal{K}$ that are no model for $\mathcal{K}_{\mathtt{and}}$) only a rough consistency check can be done with the set of models that were already found $\mathcal{M}$. If $X : \neg F \in \bigcap_{M \in \mathcal{M}} \mathcal{B}_M$ then a classical consistency check is still required to determine with certainty if $X : F$ is consistent or not. If $X : \neg F \notin \bigcap_{M \in \mathcal{M}} \mathcal{B}_M$ then we already know that $X : F$ must be consistent and we do not have to do an additional consistency check.

As the ABox encoding of a model does not contain all complex concepts (definition 2.8), a complex concept F should be decomposed to use the ABox encoding for a rough consistency check. As role assertions cannot be complex, a rough role assertion check can be done by

checking $(\mathsf{I}, \mathsf{X}) : \neg \mathsf{r} \in \bigcap_{\mathsf{M} \in \mathcal{M}} \mathcal{B}_{\mathsf{M}}$. If $(\mathsf{I}, \mathsf{X}) : \neg \mathsf{r} \notin \bigcap_{\mathsf{M} \in \mathcal{M}} \mathcal{B}_{\mathsf{M}}$, then $(\mathsf{I}, \mathsf{X}) : \mathsf{r}$ is consistent, else do a traditional consistency check. Algorithm 4 can be followed to do a rough consistency check on $\mathsf{X} : \mathsf{F}$ with the use of $\mathcal{M}$, which represents the set of all found models M for any $\mathcal{K}_{\mathtt{and}}$.

If the algorithm returns $\mathsf{X} : \mathsf{F}$ is consistent, no additional consistency check has to be done on $\mathsf{X} : \mathsf{F}$. If the algorithm return "false", then a traditional consistency check should be done on $\mathsf{X} : \mathsf{F}$. A good thing to remember is that when the algorithm returns false, then $\mathsf{X} : \neg \mathsf{F}$ must be consistent, regardless of the outcome of the traditional consistency check on $\mathsf{X} : \mathsf{F}$. If one has to check if $\mathsf{X} : \mathsf{F}$ consistent, yet not entailed, this is relevant information that can be saved.

It can be seen that the intersection of the ABox encodings of all models $\mathsf{M} \in \mathcal{M}$, i.e. $\bigcap_{\mathsf{M} \in \mathcal{M}} \mathcal{B}_{\mathsf{M}}$ is used on multiple occasions. It would be wasteful to calculate this intersection every time a check has to be done, as one has to go through all the models in $\mathcal{M}$. Instead the intersection can be taken at the first check and only has to be updated when a new model has been found, i.e. when a traditional consistency check has been done.

3.1.7 Complete algorithm

To integrate the extra checks that have been introduced in the previous sections, a complete algorithm is formed. For future reference this algorithm is dubbed SEMAR, short for SEMantically Minimal ABox Abduction Reasoner. Algorithm 5 has added extra checks for consistency of the explanations. By adding the consistency checks, the minimality of explanations that contain a conjunction is secured. Furthermore, the consistency checks on explanations that contain a disjunction ensures that all explanations are independent. To find all explanations for observations that contain an existential quantifier the extra explanation search OEQ-check is called on line 37. Lastly, the extra check for potential explanations formed by universal quantifiers is added at line 32.

Algorithm 4: Pre-check for consistency

Data: Knowledge base $\mathcal{K}$
An ABox assertion $\mathsf{X} : \mathsf{F}$
A set $\mathcal{B}_{\mathcal{M}}$ that contains the ABox encodings of every found model M for $\mathcal{K}_{\mathtt{and}}$
A set $\mathcal{B}_{\cap}$ that represents the intersection of $\mathcal{B}_{\mathcal{M}}$
Result: Decision on consistency for $\mathsf{X} : \mathsf{F}$

1 **if** F *is of the form* $\mathsf{D} \sqcup \mathsf{E}$ **then**
2 **if** D *and* E *are simple concepts* **then**
3 **if** $\mathsf{X} : \neg\mathsf{D} \notin \mathcal{B}_{\cap}$ **then return** $\mathsf{X} : \mathsf{F}$ is consistent;
4 **if** $\mathsf{X} : \neg\mathsf{E} \notin \mathcal{B}_{\cap}$ **then return** $\mathsf{X} : \mathsf{F}$ is consistent;
5 **return** false
6 **if** *only* D *is a simple concept* **then**
7 **if** $\mathsf{X} : \neg\mathsf{D} \notin \mathcal{B}_{\cap}$ **then return** $\mathsf{X} : \mathsf{F}$ is consistent;
8 **return** Pre-check on $\mathsf{X} : \mathsf{E}$
9 **if** D *and* E *are complex concepts* **then**
10 **if** *Pre-check on* $\mathsf{X} : \mathsf{D}$ *returns consistent* **then return** $\mathsf{X} : \mathsf{F}$ is consistent;
11 **return** Pre-check on $\mathsf{X} : \mathsf{E}$
12 **if** F *is of the form* $\mathsf{D} \sqcap \mathsf{E}$ **then**
13 **if** D *and* E *are simple concepts* **then**
14 **foreach** $\mathcal{B}_{\mathsf{N}} \in \mathcal{B}_{\mathcal{M}}$ **do**
15 **if** $\{\mathsf{X} : \mathsf{D}\} \cap \{\mathsf{X} : \mathsf{E}\} \in \mathcal{B}_{\mathsf{N}}$ **then return** $\mathsf{X} : \mathsf{F}$ is consistent;
16 **return** false
17 **if** *only* D *is a simple concept* **then**
18 **foreach** $\mathcal{B}_{\mathsf{N}} \in \mathcal{B}_{\mathcal{M}}$ **do**
19 **if** $\{\mathsf{X} : \mathsf{D}\} \in \mathcal{B}_{\mathsf{N}}$ **then**
20 **if** *Pre-check on* $\mathsf{X} : \mathsf{E}$ *with* $\mathcal{B}_{\mathcal{M}} \leftarrow \mathcal{B}_{\mathsf{N}}$ *and* $\mathcal{B}_{\cap} \leftarrow \mathcal{B}_{\mathsf{N}}$ *returns consistent* **then return** $\mathsf{X} : \mathsf{F}$ is consistent;
21 **return** false
22 **if** D *and* E *are complex concepts* **then return** false ;
23 **if** F *is of the form* $\exists\mathsf{r}.\mathsf{D}$ **then**
24 **foreach** *individual* $\mathsf{Y} \in \mathbf{I}$ **do**
25 **if** $(\mathsf{X}, \mathsf{Y}) : \neg\mathsf{r} \notin \mathcal{B}_{\cup}$ **then**
26 **foreach** $\mathcal{B}_{\mathsf{N}} \in \mathcal{B}_{\mathcal{M}}$ **do**
27 **if** $\{(\mathsf{X}, \mathsf{Y}) : \mathsf{r}, \mathsf{Y} : \mathsf{D}\} \in \mathcal{B}_{\mathsf{N}}$ **then return** $\mathsf{X} : \mathsf{F}$ is consistent;
28 **return** false
29 **if** F *is of the form* $\forall\mathsf{r}.\mathsf{D}$ **then**
30 **foreach** $\mathcal{B}_{\mathsf{N}} \in \mathcal{B}_{\mathcal{M}}$ **do**
31 **foreach** $(\mathsf{X}, \mathsf{Y}) : \mathsf{r} \in \mathcal{B}_{\mathsf{N}}$ **do**
32 **if** $\mathsf{Y} : \mathsf{D} \notin \mathcal{B}_{\mathsf{N}}$ **then** Skip to next model;
33 **return** $\mathsf{X} : \mathsf{F}$ is consistent
34 **return** false
35 **if** F *is of the form* C, $\neg\mathsf{C}$ *or* $\{\mathrm{I}\}$ *for* $\mathsf{C} \in \mathbf{C}$ *and* $\mathrm{I} \in \mathbf{I}$ **then**
36 **if** $\mathsf{C} : \neg\mathsf{F} \notin \mathcal{B}_{\cup}$ **then return** $\mathsf{X} : \mathsf{F}$ is consistent;
37 **else return** false;

Algorithm 5: Semantically Minimal ABox Abduction Reasoner (SEMAR)

Data: Knowledge base $\mathcal{K} = \mathcal{A} \cup \mathcal{T}$
Observation $\Phi = \mathrm{I} : \mathsf{C}$
Result: A set of semantically minimal explanations $\mathcal{E}$

```
Initiate tableau with A ∪ ¬Φ at the root;
while no model is found do
    if ⊓-, ⊔-, ∃-, ∀-, ∀2-, or {}-rule can be applied then
        Apply rule;
        if one of the parent assertions is an ODA then
            Mark the new assertions as observational derived assertions;
        if Clash check for SEMAR is not false then
            Add return of clash check to E_pre;
            else if There is an open alternative path then
                Backtrack to the nearest non-closed alternative path;
            return ∅
    else if ⊑-rule can be applied then
        Apply rule;
        if Clash check for SEMAR is not false then
            Add return of clash check to E_pre;
            else if There is an open alternative path then
                Backtrack to the nearest non-closed alternative path;
            return ∅
    else
        Close the path and mark as model;
Backtrack to the nearest non-closed alternative path;
while not at the root node do
    if ⊓-, ⊔-, ∃-, ∀-, ∀2-, or {}-rule can be applied then
        Apply rule;
        if one of the parent assertions is an ODA then
            Mark the new assertions as observational derived assertions;
        if Clash check for SEMAR is not false then
            Add return of clash check to E_pre;
            Backtrack to the nearest non-closed alternative path;
    else
        Close path;
foreach (X, I) : r in the intersection of all found ABox encodings do
    if X is an exsisting individual and K ∪ {(X : I : ¬r} is inconsistent then
        if {X : ∀r.C} ∉ E_pre then
            E_pre ← X : ∀r.C};
if Φ is of the form I : ∃r.C then
    E_pre = OEQ-check with K , Φ and E_pre as input;
E = Post-check for SEMAR with E_pre as input;
return E
```

Algorithm 6: Clash check for SEMAR

Data: ABox $\mathcal{A}$
Set of new assertions $\mathcal{A}^+$
List of visited notes N
Observation Φ
Result: A possible explanation E

```
foreach assertion I : C ∈ A⁺ do
    if I : ¬C ∈ A then
        if I : C or I : ¬C is an ODA then
            while current node is not the root note do
                if last rule is ⊓-, {}- or ∃-rule then
                    Move one node up;
                else if last rule is ⊔-rule then
                    E ⟶ E ∪ I : ¬C, s.t. I : C is the root assertion of alternative path;
                    Move one node up;
                else if last rule is ∀-rule then
                    if parent assertion of the form I : ∀r.C ∈ {Φ ∪ ¬Φ} then
                        Move one node up;
                    else
                        E ⟶ E ∪ I : ∀r.C;
                        Return E;
                else if last rule is ∀₂-rule then
                    E ⟶ E ∪ (I, J) : r s.t. (I, J) : r is one of the parrent assertions;
                else if last rule is ⊑-rule then
                    Return E;
            Return E;
        else
            Normal clash found, return ∅;
No clash is found, return ⊥;
```

Algorithm 7: Post-check for SEMAR

Data: A set of possible explanations $\mathcal{E}_{\text{pre}}$
Result: A set of semantically minimal explanations $\mathcal{E}$

foreach $\mathsf{E} \in \mathcal{E}_{\text{pre}}$ **do**
  if E *contains an assertion* $\mathsf{I} : \mathsf{C}$ *s.t.* C *is of the form* $\mathsf{D} \sqcap \mathsf{E}$ **then**
    if $\mathcal{K} \cup \{\mathsf{I} : \neg\mathsf{D}\}$ *is inconsistent* **then**
      $\mathsf{E} \leftarrow \mathsf{E} \setminus \{\mathsf{I} : \mathsf{C}\} \cup \{\mathsf{I} : \mathsf{E}\}$;
    if $\mathcal{K} \cup \{\mathsf{I} : \neg\mathsf{E}\}$ *is inconsistent* **then**
      $\mathsf{E} \leftarrow \mathsf{E} \setminus \{\mathsf{I} : \mathsf{C}\} \cup \{\mathsf{I} : \mathsf{D}\}$;
  else if E *contains an assertion* $\mathsf{I} : \mathsf{C}$ *s.t.* C *is of the form* $\mathsf{D} \sqcup \mathsf{E}$ **then**
    if $\mathcal{K} \cup \{\mathsf{I} : \mathsf{D}\}$ *is inconsistent* **then**
      if $\mathcal{K} \cup \{\mathsf{I} : \mathsf{E}\}$ *is inconsistent* **then**
        Continue with next E ;
      $\mathsf{E} \leftarrow \mathsf{E} \setminus \{\mathsf{I} : \mathsf{C}\} \cup \{\mathsf{I} : \mathsf{E}\}$;
    else if $\mathcal{K} \cup \{\mathsf{I} : \mathsf{E}\}$ *is inconsistent* **then**
      $\mathsf{E} \leftarrow \mathsf{E} \setminus \{\mathsf{I} : \mathsf{C}\} \cup \{\mathsf{I} : \mathsf{D}\}$;
    $\mathcal{E} \leftarrow \mathcal{E} \cup \mathsf{E}$;
    Continue with next E;
  if $\mathcal{K} \cup \mathsf{E}$ *is inconsistent* **then**
    Continue with next E;
  $\mathcal{E} \leftarrow \mathcal{E} \cup \mathsf{E}$;
foreach $\mathsf{E} \in \mathcal{E}$ **do**
  foreach $\mathsf{F} \in \mathcal{E}$ *s.t.* $\mathsf{F} \neq \mathsf{E}$ **do**
    if $\mathcal{K} \cup \mathsf{E} \cup \neg\mathsf{F}$ *is inconsistent* **then**
      if $\mathcal{K} \cup \mathsf{F} \cup \neg\mathsf{E}$ *is consistent* **then**
        $\mathcal{E} \leftarrow \mathcal{E} \setminus \mathsf{E}$;
return $\mathcal{E}$

3.1.8 Evaluation of SEMAR

The one axiom approach started from two simple premises: every semantically minimal explanation can use at most one TBox axiom to infer a simple observation, and only clashes with a descendant of the observation can lead to an explanation to that observation. However, sections 3.1.2 to section 3.1.5 show that there are many small exceptions and details that ask for additional consistency checks. Before looking at the limitations that these adjustments bring with them, let us look at the soundness and completeness of SEMAR.

Lemma 3.1 (Soundness)**.** *Given a knowledge base $\mathcal{K}$ in $\mathcal{ALCHO}$ containing no TBox axioms that contain nominals, and a single observation in negation normal form (NNF) Φ that contains no conjunctions, SEMAR initialized with $\mathcal{K}$ and Φ produces a set $\mathcal{E}$. Each explanations $\mathsf{E}_\mathsf{i} \in \mathcal{E}$ is a explanatory, consistent, relevant, independent, syntactically minimal, and semantically minimal ABox abduction explanation to the ABox abduction problem $\langle \mathcal{K}, \Phi \rangle$.*

Proof. First let us check when an explanation E can be added as an explanation. There are three options when an explanation for an ABox abduction problem $\langle \mathcal{K}, \Phi \rangle$, $\Phi = \{\mathsf{I} : \mathsf{C}\}$ can be added to $\mathcal{E}_{\text{pre}}$ by SEMAR :

1. during the Clash check algorithm, when adding a set of assertions E inevitably leads to a clash with an observation derived assertion (ODA), and $\mathsf{E} \not\models \Phi$, then E is added as potential explanation.

2. during the loop on line 32. For any $(\mathsf{X}, \mathsf{I}) : \mathsf{r}$ s.t. X is an existing individual, I is the individual of the observation Φ and $\mathcal{K} \models \{(\mathsf{X}, \mathsf{I}) : \mathsf{r}\}$, an explanation $\mathsf{X} : \forall \mathsf{r}.\mathsf{C}$ is formed.

3. during the OEQ-check an explanation can be added when the observation is of the form $\mathsf{I} : \exists \mathsf{r}.\mathsf{C}$, there exists no explanation E s.t. $\Phi \cup \mathcal{K} \models \mathsf{E}$. There are three types of explanations that OEQ-check can return for any existing individual X:

   (a) an explanation of the form $\{(\mathsf{X}, \mathsf{I}) : \mathsf{r}\}$, s.t. $\mathcal{K} \models \{\mathsf{X} : \mathsf{C}\}$

(b) an explanation of the form $\{(\mathsf{X},\mathsf{I}):\mathsf{r},\mathsf{X}:\mathsf{D}\}$, s.t. $\mathcal{K}\not\models\{\mathsf{X}:\mathsf{C}\}$, $\mathcal{K}\cup\{\mathsf{X}:\mathsf{D}\}\models\{\mathsf{X}:\mathsf{C}\}$, and $\{\mathsf{X}:\mathsf{D}\}\not\models\{\mathsf{X}:\mathsf{C}\}$

(c) an explanation of the form $\{\mathsf{X}:\mathsf{D}\}$, s.t. $\mathcal{K}\models\{(\mathsf{X},\mathsf{I}):\mathsf{r}\}$, $\mathcal{K}\cup\{\mathsf{X}:\mathsf{D}\}\models\{\mathsf{X}:\mathsf{C}\}$, and $\mathcal{K}\cup\{\mathsf{X}:\mathsf{C}\}\models\{\mathsf{X}:\mathsf{D}\}$

For the first situation, where explanations are added because they cause a clash with an ODA, $\mathcal{K}\cup\mathsf{E}\models\Phi$ must be true. As the observation Φ is in NNF and does not contain any conjunctions, the negation of Φ cannot contain a disjunction. The only expansion rule that could introduce a disjunction to an assertion is the $\sqsubseteq$-rule. By definition an ODA is a descendent of the observation Φ by any expansion rule except the $\sqcup$- or $\sqsubseteq$-rule (definition 3.1). Furthermore, the only expansion rule that introduces branching in the original set of expansion rules (figure 2.1) is the $\sqcup$-rule. Thus, for any ODA O a tableau algorithm in $\mathcal{ALCHO}$ will find no model for $\mathcal{K}\cup\neg\mathsf{O}\cup\neg\Phi$, i.e. $\mathcal{K}\cup\neg\mathsf{O}\models\Phi$. Any explanation E causes a clash with at least one ODA, thus $\mathcal{K}\cup\mathsf{E}\models\neg\mathsf{O}$. By transitivity the explanation E must entail the observation Φ. Any explanation E returned by the Clash check algorithm must be relevant, as $\mathsf{E}\not\models\Phi$ (definition 2.10).

For the second case, an explanation $\mathsf{E}=\{\mathsf{X}:\forall\mathsf{r}.\mathsf{C}\}$ is added if, and only if X is an existing individual, and $\mathcal{K}\models\{(\mathsf{X},\mathsf{I}):\mathsf{r}\}$. As $\{(\mathsf{Y},\mathsf{Z}):\mathsf{s},\mathsf{Y}:\forall\mathsf{r}.\mathsf{D}\}\models\{\mathsf{Z}:\mathsf{D}\}$, for any individuals $\mathsf{Y},\mathsf{Z}\in\mathbf{I}$, role $\mathsf{s}\in\mathbf{R}$, and concept D, the entailment $\{(\mathsf{X},\mathsf{I}):\mathsf{r},\mathsf{X}:\forall\mathsf{r}.\mathsf{C}\}\models\{\mathsf{I}:\mathsf{C}\}$ must be true. As $\mathcal{K}\cup\mathsf{E}\models\{(\mathsf{X},\mathsf{I}):\mathsf{r},\mathsf{X}:\forall\mathsf{r}.\mathsf{C}\}$ is true, $\mathcal{K}\cup\mathsf{E}\models\Phi$ must be true by transitivity. Furthermore, $\{\mathsf{Y}:\forall\mathsf{r}.\mathsf{D}\}\not\models\{\mathsf{Z}:\mathsf{D}\}$ holds for any $\mathsf{Y},\mathsf{Z}\in\mathbf{I}$, role $\mathsf{s}\in\mathbf{R}$, and concept D. Therefore, $\{\mathsf{X}:\forall\mathsf{r}.\mathsf{C}\}\not\models\{\mathsf{I}:\mathsf{C}\}$, which means that the explanation E is relevant by definition 2.10.

Lastly, the explanations generated by the OEQ-check algorithm are explanations to an observation $\mathsf{I}:\exists\mathsf{r}.\mathsf{C}$ because $\{\mathsf{X}:\mathsf{C},(\mathsf{X},\mathsf{I}):\mathsf{r}\}\models\mathsf{I}:\exists\mathsf{r}.\mathsf{C}$, and for all three options $\mathcal{K}\cup\mathsf{E}\models\{\mathsf{X}:\mathsf{C},(\mathsf{X},\mathsf{I}):\mathsf{r}\}$. The explanations are relevant as well as $\{(\mathsf{X},\mathsf{I}):\mathsf{r}\}\not\models\{\mathsf{I}:\exists\mathsf{r}.\mathsf{C}\}$, $\{\mathsf{X}:\mathsf{D}\}\not\models\{\mathsf{I}:\exists\mathsf{r}.\mathsf{C}\}$, and $\{(\mathsf{X},\mathsf{I}):\mathsf{r},\mathsf{X}:\mathsf{D}\}\not\models\{\mathsf{I}:\exists\mathsf{r}.\mathsf{C}\}$ because $\{\mathsf{X}:\mathsf{D}\}\not\models\{\mathsf{X}:\mathsf{C}\}$.

If an explanation E is not explanatory, then no model for $\mathcal{K}\cup\neg\Phi$ can be found. SEMAR returns an empty set when no model can be found (line 11 and 18). Therefore, all explanations $\mathsf{E}\in\mathcal{E}_{\text{pre}}$ returned by SEMAR must be explanatory.

For the explanations found in the first two situations the post check algorithm performs an extra check to ensure that the explanations are consistent, independent, syntactically minimal, and semantically minimal. Any explanation $\mathsf{E} \in \mathcal{E}_{\text{pre}}$ is added to the final set of explanations $\mathcal{E}$ if and only if:

1. $\mathcal{K} \cup \mathsf{E}$ is consistent, and
2. $\mathcal{K} \cup \{\mathsf{I} : \neg\mathsf{D}\}$ is consistent for any D when $\mathsf{E} = \{\mathsf{I} : \mathsf{D} \sqcap \mathsf{E}\}$, and
3. $\mathcal{K} \cup \{\mathsf{I} : \mathsf{D}\}$ is consistent for any D when $\mathsf{E} = \{\mathsf{I} : \mathsf{D} \sqcup \mathsf{E}\}$, and
4. $\mathcal{K} \cup \mathsf{E} \cup \neg\mathsf{F}$ or $\mathcal{K} \cup \mathsf{F} \cup \neg\mathsf{E}$ is consistent for any $\mathsf{F} \in \mathcal{E}$ s.t. $\mathsf{F} \neq \mathsf{E}$.

By definition conditions 1 and 4 ensure that every explanation is consistent and semantically minimal respectively (definition 2.10). Condition 2 ensures syntactic minimality for every explanation that contains a conjunction. As explanations that do not contain a conjunction, cannot be split in a multiple assertion explanation, it cannot be syntactically non-minimal. Therefore, only checking the explanations that contain a conjunction is sufficient. Lastly, every explanation that contains a disjunction is checked for independence. By definition only explanations that contain a disjunction can be dependent (definition 3.2). Therefore, any explanation E returned by the post check algorithm must be consistent, independent, syntactically minimal, and semantically minimal.

In conclusion all explanations $\mathsf{E} \in \mathcal{E}$ that are generated by SEMAR must be in fact an explanation that is explanatory, consistent, relevant, independent, syntactically minimal and semantically minimal by definition 2.2 and definition 3.2. □

While a proof of soundness of the algorithm is given above, a proof of completeness was not found within the time limit of this thesis. The main issue in proving completeness was found due to the use of complex concepts in the observation. Furthermore, the use of nominals make some extra explanations possible as two individuals can be merged. While it is not that common to have an ontology that forces individuals to merge in practice, this could be possible in

theory. To prove completeness of the algorithm, a way to handle nominals asserted to different individuals, e.g. $\mathrm{I} : \{\mathrm{J}\}$, must be included in the algorithm, or it must be defined what specific explanations are not possible to be found by SEMAR.

Next to the limitation that nominals pose on the algorithm, SEMAR is not sound for observations that contain a conjunction, when in NNF. The conjunction posed an issue as the algorithm is based on finding explanations by closing branches during the expansion of the ABox. When a conjunction is present in the observation, then the negated observation contains a disjunction, meaning it can be split over multiple branches. Closing only one branch is not enough to be an actual explanation to the observation. While this is a constraint of the algorithm, such an observation can be split in multiple observations. Then SEMAR can be used to find explanations for each sub-observation. To make SEMAR compatible for multiple observations, thus observations that contain a conjunction in NNF using a method inspired by the splitting method of Pukancová (2018) would be interesting for future research.

3.1.9 Implementation

As no proof of completeness could be formulated for SEMAR within the scope of this thesis, an implementation of the algorithm would be helpful to empirically test the algorithm. At the first attempt for an implementation a new Java application was written that uses the algorithms presented in this chapter as the base. The Jena Apache framework was used to import ontologies to the application. Unfortunately it was not possible to build an application that implemented SEMAR and did not run out of memory for the test ontology Family.

A second attempt at a working implementation of SEMAR included the open source, tableau based, reasoner Pellet 2.0. By using the optimisation techniques that Pellet has implemented a working implementation of SEMAR might be possible without memory issues. However, lazy unfolding, which is one of the major optimizing techniques that Pellet uses, makes the foundations of SEMAR weak. Lazy unfolding only applies the $\sqsubseteq$-rule when the premise is already present in an ABox, while SEMAR is aimed at finding those premises that are not yet present in the ABox. An idea that could integrate lazy unfolding rules in SEMAR, would

be to use lazy unfolding on all individuals except the individual of the observation. However, implementing the lazy unfolding and understanding the complete syntax of Pellet was not possible within the time limit of this research. Therefore a decision was made to move on with to the optimising approach, which proved to be more fruitful.

3.2 Optimising Approach

Alternatively to searching for semantically minimal solutions using an adjusted tableau algorithm, the minimal hitting set (MHS) of Pukancová and Homola (2015) can be used to search for all explanations, after which an extra algorithm can be used to determine which of those explanations are semantically minimal. Let us first check if integrating a semantic minimality check in the MHS algorithm is effective. During the MHS algorithm there are branches with potential explanations, if it can be proven that a branch containing a potential explanation cannot lead to a semantically minimal explanation, that branch can be pruned. Therefore, we want to be able to check incomplete explanations for their potential to become a complete, semantically minimal explanation.

Checking if an incomplete explanation implies another incomplete explanation is not useful, as it gives no grantees about the potential of this branch to lead to a complete, semantically minimal explanation. Given two incomplete explanations $\mathcal{E}_{p1}$ and $\mathcal{E}_{p2}$, that can potentially lead to the complete explanations $\mathcal{E}_1$ and $\mathcal{E}_2$ respectively, one can check if $\mathcal{K} \cup \mathcal{E}_{p1} \models \mathcal{E}_{p2}$, yet the check provides no knowledge that can ensure whether $\mathcal{K} \cup \mathcal{E}_1 \models \mathcal{E}_2$ is true or not. If $\mathcal{E}_{p1}$ does indeed imply $\mathcal{E}_{p2}$, there can still be an assertion $\mathsf{I} : \mathsf{C} \in \mathcal{E}_2$ that is not implied by $\mathcal{E}_1$. Therefore the branch of $\mathcal{E}_{p1}$ cannot be pruned based on semantic minimality. If $\mathcal{E}_{p1}$ does not imply $\mathcal{E}_{p2}$, there can be assertions in $\mathcal{E}_1$ that make it so that $\mathcal{E}_1$ implies $\mathcal{E}_2$. Thus even if $\mathcal{E}_{p1}$ is semantically minimal, that gives no grantee that $\mathcal{E}_1$ is as well. To illustrate imagine a knowledge base that contains the following two GCI's:

1. $\mathsf{Man} \sqcap \exists \mathsf{hasChild.Parent} \sqsubseteq \mathsf{Grandfather}$, and
2. $\mathsf{Grandfather} \sqsubseteq \mathsf{Father}$.

] If the MHS is used to find explanations for $\mathtt{John} : \mathtt{Father}$, two explanations will be found,

1. $\mathcal{E} = [\{\mathtt{John} : \mathtt{Grandfather}\}$, and
2. $\{\mathtt{John} : \mathtt{Man}, \mathtt{John} : \exists \mathtt{hasChild.Parent}\}]$.

It is clear that the second explanation implies the first explanation, thus is not semantically minimal, however only the complete explanation implies $\{\mathtt{John} : \mathtt{Grandfather}\}$. Checking $\mathcal{K} \cap \{\mathtt{Jonn} : \mathtt{Man}\} \vDash \{\mathtt{John} : \mathtt{Grandfather}\}$ results in the knowledge that $\{\mathtt{John} : \mathtt{Man}\}$ does not imply $\{\mathtt{John} : \mathtt{Grandfather}\}$, however this knowledge is useless when anything is added to $\{\mathtt{John} : \mathtt{Man}\}$, as the check has to be done again with the new set of assertions.

If there is an incomplete explanation $\mathcal{E}_{p1}$ that implies another complete explanation $\mathcal{E}_2$, this is useful information, as any explanation that contains the assertions $I : A \in \mathcal{E}_{p1}$ will also imply $\mathcal{E}_2$. This means a branch can be pruned if a consistency check proves that there is another complete explanation $\mathcal{E}_i$, that is implied by the current set of assertions. However, when the current set of assertions $\mathcal{E}_{p1}$ does not imply any complete explanation no relevant knowledge is gained, as in the next step an assertion can be added that makes it so that there is an $\mathcal{E}_i$ s.t. $\mathcal{K} \cup \mathcal{E}_{p1} \vDash \mathcal{E}_i$. Doing a consistency check is expensive, and has to be conducted every time a new assertion is added to a potential explanation, for every complete explanation, until a potential explanation does imply another complete explanation, or the branch delivers a new complete explanation. Therefore, doing a consistency check every step can prune a branch in an early stage, however it greatly increases the maximum complexity. Furthermore, if a complete explanation is found and that explanation does not imply any other explanation that has been found, it has to be tested for semantic minimality to every new complete explanation that has been found.

It should be noted that the pre-check for consistency checking explained in section 3.1.6, which checks the presence of an assertion in the found ABox assertions, can be used to minimize the amount of calls to the tableau algorithm. When an explanation $\mathcal{E}_1$ is not semantically minimal, it has to imply at least one other explanation $\mathcal{E}_i, i \neq 1$. An explanation $\mathcal{E}_1$ implies another explanation $\mathcal{E}_i$ if and only if every assertion $I : C \in \mathcal{E}_i$ is true in any model $\mathcal{M}$ for

$\mathcal{K} \cup \mathcal{E}_1$. Therefore, if an assertion $I : C \in \mathcal{E}_i$ is not present in an ABox encoding of an arbitrary model $\mathcal{M}$ for $\mathcal{K} \cup \mathcal{E}_1$, then $\mathcal{E}_i$ does not imply $\mathcal{E}_i$. As this check can only conclude if a potential explanation does not imply any other explanation it cannot be used to prune branches due to semantic minimality without using an extra call to the tableau algorithm.

While the pre-check cannot be used to prune the MHS tree, it can be used after the MHS algorithm has run to check the proposed explanations for semantic minimality. One extra benefit of using this check with the ABox abduction Algorithm (AAA) is that AAA only produces explanations containing simple concepts. Therefore, the explanations won't have to be broken down to simple concepts as done in section 3.1.6. On every node of the MHS the consistency of the current set of assertions $\mathcal{E}_{p1}$ is tested. If the current set is consistent there exists a model $\mathcal{M}$ for $\mathcal{K} \cup \mathcal{E}_{p1}$. If $\mathcal{E}_{p1}$ is in fact an explanation, then it would be useful to store the ABox encoding of $\mathcal{M}$ to use for the pre-check for semantic minimality.

After all the explanations $\mathcal{E}_i$ have been found and at least one ABox encoding for $\mathcal{K} \cup \mathcal{E}_i$ has been stored the check for semantic minimality can be done. Algorithm 8 shows how the explanations can be checked for semantic minimality. In the rest of this paper this algorithm is referred to as the *Semantic minimality Checker (SMC)*.

Line 4. The first check to be done is to check if all the elements of the control explanation, $\mathcal{E}_j$, are present in the intersection of all ABox encodings for the explanation to be checked, with regards to the knowledge base, $\mathcal{E}_i \cup \mathcal{K}$. When one of the elements is not in all of the ABox encodings, then the explanation $\mathcal{E}_j$ cannot be entailed by $\mathcal{E}_i$, therefore the tableau algorithm does not have to be called.

Line 6. When the explanation $\mathcal{E}_i$ entails another explanation $\mathcal{E}_j$ it can only be semantically minimal if the explanation $\mathcal{E}_j$ also entails $\mathcal{E}_i$, therefore the entailment check is done in reverse. Only if $\mathcal{E}_i$ is not entailed by $\mathcal{E}_j$ it can be disregarded as semantically minimal.

Line 7. As the current explanation $\mathcal{E}_i$ entails the control explanation $\mathcal{E}_j$, any other explanation that entails $\mathcal{E}_i$ will also entail $\mathcal{E}_j$. As the other explanations will be checked against $\mathcal{E}_j$, the check against $\mathcal{E}_i$ is redundant. Therefore it can be removed form the check set.

Algorithm 8: Semantic minimality check for AAA

Data: Knowledge base $\mathcal{K}$
Finite set of pairs $\mathcal{E}$ containing an explanation $\mathsf{E_i}$ and the intersection $\mathcal{B}_\mathsf{i}$ of all found ABox encodings for $\mathcal{K} \cup \mathsf{E_i}$
Result: A set of semantic minimal explanations $\mathcal{E}^{\mathsf{sem}}$

```
1  E^check ← E
2  foreach Pair E_i =< E_i, B_i >, E_i ∈ E do
3      foreach Pair E_j =< E_j, B_j >, E_j ∈ E^check, E_j ≠ E_i do
4          if E_j ∩ B_i = E_j then
5              Call the tableau algorithm on E_i ∪ K ∪ ¬E_j
6              if tableau returns no model then
7                  E^check = E^check \ E_i
8                  if E_i ∩ B_j = E_i then
9                      Call the tableau algorithm on E_j ∪ K ∪ ¬E_i
10                     if tableau returns model then
11                         Continue to next pair in E
12                     else
13                         Store the ABox encoding of found model in B
14                         B_j = B ∩ B_j
15                 else
16                     Continue to next pair in E
17             else
18                 Store the ABox encoding of found model in B
19                 B_i = B ∩ B_i
20     E^sem ← E_i
```

Line 10. It has been proven that the explanation $\mathcal{E}_\mathsf{i}$ is not semantically minimal. As the explanation only has to be semantically bigger then one of the other explanations to be semantically non-minimal, the rest of the explanations do not have to be checked. Therefore the algorithm continues to test the next explanation for semantic minimality.

Line 19. Any model for $\mathcal{K} \cup \mathcal{E}_\mathsf{i} \cup \mathsf{X}$ for an arbitrary set of ABox assertion X must also be a model for $\mathcal{K} \cup \mathcal{E}_\mathsf{i}$. As the pre-check is stricter when the intersection of ABox encodings is smaller adding a new ABox encoding is beneficial for the computation time. Therefore the found model for $\mathcal{K} \cup \mathcal{E}_\mathsf{i} \cup \neg\mathcal{E}_\mathsf{j}$ is first encoded to a set of ABox assertion, and then the intersection of the current set with the new ABox encoding is taken to minimize the intersection set.

3.2.1 Evaluation of SMC

Given that algorithm 8 is written separately from the MHS algorithm it is also possible to use it as a semantic minimality checker (SMC) on any other set of explanations. Therefore the constraints of AAA are not necessarily binding for the the semantic minimality checker. However in this thesis the algorithm is written to select all semantically minimal explanations from a set of explanations produced by the AAA implementation of Pukancová (2018). Therefore it is limited by the limitations of the output produced by AAA: The explanations are in $\mathcal{ALCHO}$ and consist of simple concepts. Furthermore for practical reasons the AAA implementation might be set to a certain depth to complete in reasonable time. When a depth $\mathtt{d}$ is set on AAA it unfolds the MHS up to a depth of $\mathtt{d}$ nodes for each branch. For AAA_r setting a depth $\mathtt{d}$ means that every explanation entail at most $\mathtt{d}$ assertions. For AAA_s setting a depth $\mathtt{d}$ means that only the explanations that entail at most $\mathtt{d}$ assertions are guaranteed to be found. As the SMC does not find explanations on its own, but only tests the set of found explanations for semantic minimality, the proofs in the remainder of this section only hold for explanations that have a maximum depth of $\mathtt{d}$ are in $\mathcal{ALCHO}$ and consist of simple concepts. As these are constrains of the input and not of the algorithm itself they are not explicitly stated in the formal proofs.

Lemma 3.2 (Soundness)**.** *Given a knowledge base $\mathcal{K}$, an observation Φ, a finite set of explanations $\mathcal{E}$ produced by an algorithm A, and a finite set $\mathcal{B}$ consisting of elements $\mathcal{B}_\mathsf{i}$, that represent the intersection of ABox encodings for $\mathcal{K} \cup \mathsf{E}_\mathsf{i}$, the SMC algorithm initialized with $\mathcal{K}$, $\mathcal{E}$ and $\mathcal{B}$ as input produces a set $\mathcal{E}^{\text{sem}}$. Each explanation $\mathsf{E}_\mathsf{i} \in \mathcal{E}^{\text{sem}}$ is a semantically minimal ABox abduction explanation to the ABox abduction problem $\langle \mathcal{K}, \Phi \rangle$, within the limitations of algorithm A.*

Proof. According to definition 2.10 an explanation E is semantically minimal if and only if E is not semantically stronger than any other explanation E' to the abduction problem, i.e. $\mathcal{K} \cup \mathsf{E}' \models$ or $\mathcal{K} \cup \mathsf{E} \not\models \mathsf{E}'$. Therefore, if every explanation $\mathsf{E}_\mathsf{i} \in \mathcal{E}^{\text{sem}}$ is not semantically stronger than any other explanation $\mathsf{E}_\mathsf{j} \in \mathcal{E}$, then every explanation $\mathsf{E}_\mathsf{i} \in \mathcal{E}^{\text{sem}}$ is semantically minimal, assuming

$\mathcal{E}$ are all the explanations to the abduction problem $\langle \mathcal{K}, \Phi \rangle$.

Only on line 20 of SMC an explanation E_i is added to the set of semantically minimal explanations $\mathcal{E}^{sem}$. As this line is on the end of the loop that checks every explanations of the input set $\mathcal{E}$, only explanations that are not thrown out of the loop are added to $\mathcal{E}^{sem}$. Thus any explanation $E_i \in \mathcal{E}$ is added to $\mathcal{E}^{sem}$ if for each other explanation $E_j \in \mathcal{E}^{check}$:

1. the intersection of the explanation E_j and all the found ABox encodings for the explanation E_i are not equal to E_j (line 4), or
2. a tableau algorithm called on $\mathcal{K} \cup E_i \cup \neg E_j$ returns a model (line 6), or
3. a tableau algorithm called on $\mathcal{K} \cup E_j \cup \neg E_i$ returns no model (line 7 to 16).

In case 1 not all the assertions of explanation E_j are in the intersection of found ABox encodings for $\mathcal{K} \cup E_i$. If an assertion $a : C$ is not an ABox encoding $\mathcal{B}_M$ for $\mathcal{K} \cup E_i$ it means that $a^M \notin C^M$ (definition 2.8). Therefore, if an assertion $a : C \in E_j$ is not in the intersection of all found ABox encodings for $\mathcal{K} \cup E_i$ it means there exists a model M for $\mathcal{K} \cup E_i$ for which $a^M \notin C^M$ holds. According to the basic reasoning problems of DL $\mathcal{K} \models a : C$ only hold if $a^M \in C^M$ for every model M of $\mathcal{K}$ (definition B.5). As case 1 indicates that there is at least one assertion $a : C$ in E_j and at least one model M for $\mathcal{K} \cup E_i$ such that $a^M \notin C^M$ holds, $\mathcal{K} \cup E_i \models a : C$ cannot be true, thus $\mathcal{K} \cup E_i \not\models E_j$, i.e. E_i is not semantically stronger than E_j.

In case 2 a model is found for the knowledge base $\mathcal{K} \cup E_i \cup \neg E_j$. This means that $\neg E_j$ is consistent with respect to the knowledge base $\mathcal{K} \cup E_i$ (definition 2.6). Therefore $\mathcal{K} \cup E_i \not\models E_j$, which means that E_i is not semantically stronger than E_j.

In case 3 no model could be found for the knowledge base $\mathcal{K} \cup E_j \cup \neg E_i$. Therefore $\mathcal{K} \cup E_j \cup \neg E_i$ cannot be consistent (definition B.5). When $\mathcal{K} \cup E_j \cup \neg E_i$ is not consistent then there is no possibility for $\neg E_i$ to be true when $\mathcal{K} \cup E_j$ is true, thus E_i must be true. From this the entailment $\mathcal{K} \cup E_j \models E_i$ follows and E_i can at most be semantically equal, but not semantically stronger than E_j.

As E_i cannot be semantically stronger than E_j in any of the cases that E_i is added to $\mathcal{E}^{\text{sem}}$ for every $\mathsf{E}_j \in \mathcal{E}^{\text{check}}$, every explanation $\mathsf{E}_i \in \mathcal{E}^{\text{sem}}$ is semantically minimal with respect to the set of explanations $\mathcal{E}^{\text{check}}$. As entailment of knowledge bases is transitive Baader et al. 2017 any explanation E_k for which $\mathcal{K} \cup \mathsf{E}_i \models \mathsf{E}_k$ is true, $\mathcal{K} \cup \mathsf{E}_i \models \mathsf{E}_j$ must be true for any E_j such that $\mathcal{K} \cup \mathsf{E}_k \models \mathsf{E}_j$ holds. $\mathcal{E}^{\text{check}}$ starts as a set of all explanations in $\mathcal{E}$. During the SMC algorithm the only explanations removed form $\mathcal{E}^{\text{check}}$ are the explanations E_k such that there exists no model for $\mathcal{K} \cup \mathsf{E}_k \cup \neg\mathsf{E}_j$ (7), i.e. if $\mathcal{K} \cup \mathsf{E}_k \models \mathsf{E}_j$. As we know that $\mathcal{K} \cup \mathsf{E}_i \models \mathsf{E}_j$ is true for any E_i such that $\mathcal{K} \cup \mathsf{E}_i \models \mathsf{E}_k$ is true, an explanation E_i cannot be semantically stronger then E_k when it is not semantically stronger than E_j. Thus every $\mathsf{E}_i \in \mathcal{E}^{\text{sem}}$ cannot be semantically stronger than any $\mathsf{E}_j \in \mathcal{E}$ if it is not semantically stronger than any $\mathsf{E}_k \in \mathcal{E}^{\text{check}}$. Therefore, every explanation $\mathsf{E}_i \in \mathcal{E}^{\text{sem}}$ is semantically minimal, assuming $\mathcal{E}$ are all the explanations to the abduction problem $\langle \mathcal{K}, \Phi \rangle$. □

Lemma 3.3 (Completeness)**.** *Given a knowledge base $\mathcal{K}$, an observation Φ, a finite set of explanations $\mathcal{E}$ produced by an algorithm A, and a finite set $\mathcal{B}$ consisting of elements $\mathcal{B}_i$, that represent the intersection of ABox encodings for $\mathcal{K} \cup \mathsf{E}_i$, the SMC algorithm initialized with $\mathcal{K}$, $\mathcal{E}$ and $\mathcal{B}$ as input produces a set $\mathcal{E}^{\text{sem}}$. Each ABox abduction explanation to the ABox abduction problem $\langle \mathcal{K}, \Phi \rangle$ that is semantically minimal and in the set of explanations $\mathcal{E}$ is in set $\mathcal{E}^{\text{sem}}$.*

Proof. Every explanation in $\mathsf{E}_i \in \mathcal{E}$ is added to $\mathcal{E}^{\text{sem}}$, unless there exists another explanation $\mathsf{E}_j \in \mathcal{E}^{\text{check}}$ such that a tableau algorithm called on $\mathcal{K} \cup \mathsf{E}_i \cup \neg\mathsf{E}_j$ returns no model (line 6), and:

1. the intersection of the explanation E_i and all the found ABox encodings for the explanation E_j are not equal to E_i (line 16), or
2. a tableau algorithm called on $\mathcal{K} \cup \mathsf{E}_j \cup \neg\mathsf{E}_i$ returns a model (line 10).

If there exists no model for $\mathcal{K} \cup \mathsf{E}_i \cup \neg\mathsf{E}_j$ then this is inconsistent (definition B.5). This means $\neg\mathsf{E}_j$ cannot be true when $\mathcal{K} \cup \mathsf{E}_i$ is true, therefore $\mathcal{K} \cup \mathsf{E}_i \models \mathsf{E}_j$ i.e. the explanation E_i cannot be semantically smaller than E_j. However E_i and E_j can still be semantically equal to E_j.

In case 1 not all the assertions of explanation E_i are in the intersection of found ABox encodings for $\mathcal{K} \cup \mathsf{E}_j$. In other words, there is at least one assertion $\mathsf{A} : \mathsf{C} \in \mathsf{E}_i$ and at least one model M

for $\mathcal{K} \cup \mathsf{E}_j$ such that $\mathfrak{a}^{\mathcal{M}} \notin \mathsf{C}^{\mathcal{M}}$ (definition 2.8). Consequently $\mathcal{K} \cup \mathsf{E}_i \models \mathfrak{a} : \mathsf{C}$ cannot be true. As the assertion $\mathfrak{a} : \mathsf{C}$ is part of the ABox E_i it can be concluded that $\mathcal{K} \cup \mathsf{E}_j \not\models \mathsf{E}_i$, and E_j is not semantically equal to E_i.

In case 2 there exists a model for $\mathcal{K} \cup \mathsf{E}_j \cup \neg\mathsf{E}_i$ meaning that $\mathcal{K} \cup \mathsf{E}_j \cup \neg\mathsf{E}_i$ is consistent (definition 2.6). If $\mathcal{K} \cup \mathsf{E}_j \cup \neg\mathsf{E}_i$ is consistent, it is possible for the ABox $\neg\mathsf{E}_i$ to be true when $\mathcal{K} \cup \mathsf{E}_j$ holds, i.e. $\mathcal{K} \cup \mathsf{E}_j \not\models \mathsf{E}_i$. As E_i is not entailed by $\mathcal{K} \cup \mathsf{E}_j$ the explanations E_i and E_j cannot be semantically equal.

Lastly, the set $\mathcal{E}^{\text{check}}$ is a subset of $\mathcal{E}$, as the only explanations added to $\mathcal{E}^{\text{check}}$ come out of the set $\mathcal{E}$ (line 1). Consequently an explanation $\mathsf{E}_i \in \mathcal{E}$ is not added to the set $\mathcal{E}^{\text{sem}}$ only if there exists another $\mathsf{E}_j \in \mathcal{E}$ such that $\mathcal{K} \cup \mathsf{E}_i \models \mathsf{E}_j$ and $\mathcal{K} \cup \mathsf{E}_j \not\models \mathsf{E}_i$. According to definition 2.10 an explanation $\mathsf{E} \in \mathcal{E}$ is only semantically minimal when $\mathcal{K} \cup \mathsf{E} \not\models \mathsf{E}'$ or $\mathcal{K} \cup \mathsf{E}' \models \mathsf{E}$. In conclusion, every explanation $\mathsf{E}_i \in \mathcal{E}$ that is not added to $\mathcal{E}^{\text{sem}}$ cannot be semantically minimal, i.e. each ABox abduction explanation to the ABox abduction problem $\langle \mathcal{K}, \Phi \rangle$ that is semantically minimal and in the set of explanations $\mathcal{E}$ is in set $\mathcal{E}^{\text{sem}}$. □

By combining the two proofs it can be proven that the SMC algorithm returns all and only all semantically minimal explanations from a given set of explanations. To prove the correctness of the algorithm the termination of the algorithm has to be proven. As the algorithm iterates over two sets of explanations, and these sets must be finite, the algorithm must eventually terminate.

Theorem 3.4. *Given a knowledge base $\mathcal{K}$, an observation Φ, a finite set of explanations $\mathcal{E}$ produced by an algorithm A, and a finite set $\mathcal{B}$ consisting of elements $\mathcal{B}_i$, that represent the intersection of ABox encodings for $\mathcal{K} \cup \mathsf{E}_i$, the SMC algorithm initialized with $\mathcal{K}$, $\mathcal{E}$ and $\mathcal{B}$ as input: (1) returns a set $\mathcal{E}^{\text{sem}}$ contains all and only all semantically minimal ABox abduction explanations to the ABox abduction problem $\langle \mathcal{K}, \Phi \rangle$, within the constraints of algorithm A, and (2) eventually terminates.*

Proof. To prove that SMC finds all and only all semantically minimal ABox abduction explanations, within the limitations of algorithm A, it must be shown that all explanations in $\mathcal{E}^{\text{sem}}$

are semantically minimal with respect to the set $\mathcal{E}$, and that all explanations in set $\mathcal{E}$ that are semantically minimal are in set $\mathcal{E}^{\mathsf{sem}}$. This follows from lemma 3.2 and lemma 3.3, therefore we can conclude that the set $\mathcal{E}^{\mathsf{sem}}$ contains all and only all semantically minimal ABox explanations to the ABox abduction problem $\langle \mathcal{K}, \Phi \rangle$, within the constraints of algorithm A (1).

The SMC algorithm contains two loops, one that iterates over each element in the set $\mathcal{E}$ and one that iterates over each element in the set $\mathcal{E}^{\mathsf{check}}$. The input $\mathcal{E}$ is a finite set, and is not adjusted within the algorithm. The set $\mathcal{E}^{\mathsf{check}}$ is initiated as a copy of $\mathcal{E}$, therefore being a finite set on its own. During the algorithm the set $\mathcal{E}^{\mathsf{check}}$ is manipulated at one place, line 7. Here an element is removed form set $\mathcal{E}^{\mathsf{check}}$, thus making the size of the set $\mathfrak{n}-1$ which still is a finite set. As both loops iterate over finite sets, both loops must eventually terminate. There are no other constructs that can prevent termination in the SMC algorithm, therefore SMC eventually terminates (2). □

The semantic minimality checker uses the tableau algorithm to check the semantic minimality of explanations. While algorithm 8 is designed to minimize the amount of tableau algorithm calls, it is possible that the tableau algorithm is called twice for every explanation pair. As the tableau algorithm can be time intensive, the SMC algorithm is mainly based on the worst case complexity of the tableau algorithm. As this thesis uses the semantic minimality checker to select semantic minimal explanations from the explanations produced by AAA, and the explanations of AAA are in $\mathcal{ALCHO}$, the worst case complexity is determined by assuming that the explanations are in $\mathcal{ALCHO}$.

Theorem 3.5. *The worst case complexity of the SMC algorithm is ExpTime.*

Proof. The algorithm contains a loop that has as many steps as the input $\mathfrak{n}$. The loop itself contains a loop that has $\mathfrak{n}-1$ steps at most. Therefore, the worst case complexity of the SMC algorithm is at least $\mathsf{O}(\mathfrak{n}^2)$. There are two steps within the loops that have a higher complexity than $\mathsf{O}(\mathfrak{n}^2)$, the calls to the tableau algorithm. As the tableau algorithm is called at most twice every iteration, and the tableau algorithm for $\mathcal{ALCHO}$ is in ExpTime Baader et al. (2017), the worst case complexity of SMC is ExpTime. □

When the algorithm is used to select semantically minimal explanations that are in a higher expressivity than $\mathcal{ALCHO}$, the complexity of the SMC algorithm might be higher than ExpTime, depending on the needed tableau algorithm.

3.2.2 Implementation

The previous section proves that the SMC algorithm is sound and complete in ExpTime are provided. To test how the algorithm performs in practice an implementation of the algorithm was build. The application is written in Java and makes use of the Apache Jena framework.

SMC is added as a function to the AAA implementation of Pukancová and Homola 2018. The function can be executed after all the explanations have been found to filter out all the semantically non-minimal explanations. Therefore, both the splitting method as well as the reduction method can be used to deal with multiple observations. To indicate when the alternative method is used to compute semantically minimal explanations the superscript *sem* is added to the name of the application, e.g. AAA_r^{sem} indicates that the AAA application ran with the reduction method and used the SMC to select the semantically minimal explanations. For clarity table 3.1 shows which methods and algorithms are used per implementation. Both Puckancova's implementation and the added function use the Pellet 2.0 reasoner for their tableau algorithm calls. This is the last open source version of Pellet and is already embedded in Pukancova's implementation.

Implementation	Splitting method	Algorithm *for generating explanations*	Algorithm *for selecting semantically minimal explanations*
AAA_s^{sem}	Splitting	MHS	SMC
AAA_s	Splitting	MHS	-
AAA_r^{sem}	Reduction	MHS	SMC
AAA_r	Reduction	MHS	-

Table 3.1: Characteristics of the tested implementations

First the implementation is tested on a set of random observations to check if the application returns all, and only all, semantically minimal explanations. To test performance the same three ontologies used by Pukancová and Homola (2017) and Mrózek et al. (2018) can be used:

LUBM, the Lehigh University Benchmark (Guo et al. 2005), Coffee, an ontology about coffee published by Carlos Mendes[1], and Family, a small ontology created by Pukancová (2018), which is suitable for testing and comparing outputs. LUBM is a standard benchmark ontology, which is used by several scholars to test reasoners (Du, Qi, et al. 2011; Del-Pinto and Schmidt 2018; Pukancová and Homola 2015). The Coffee ontology contains more axioms then LUBM, yet less concepts and roles (see table 3.2). As the ontologies are relatively compact, the semantically minimal explanations can be found manually.

Ontology	Num. concepts	Num. roles	Num. individuals	Num. axioms
LUBM	43	25	1	46
Coffee	41	6	2	291
Family	8	1	2	24

Table 3.2: Characteristics of the three ontologies

For each ontology five observations are chosen. The observations include both complex and simple concepts. Furthermore, at least one observation with multiple assertions and one observation containing a role assertion are tested. As the test ontologies do not contain any individuals, the chosen individuals are the same for each sample, the Coffee ontology is tested with the individual `drinkX`, the Family ontology is tested with the individual `Mary` and the LUBM ontology is tested with the individual `John`. To test role assertions the individuals `drinkY` and `UU` are used. Table 3.3 shows all the sample observations.

After the application is tested on its correctness, some additional tests should be done to compare the performance of the application. To compare the performance of the application the run time, but also the number of Tableau algorithm calls (TA calls) and number of used nodes are noted. For the number of nodes, all the explored nodes of the MHS tree are counted for the AAA implementations. As a TA can be very time expensive, the number of TA calls gives good insight in the risk of an implementation taking much time. The run time on its own gives an indication of the performance, but can vary significantly between different runs due to external factors, therefore is not a very trustworthy measurement. For the performance tests the first observation of each ontology in table 3.3 is used. All tests are performed a total of five

[1]The Coffee ontology is published on GitHub: `https://gist.github.com/cmendesce/56e1e16aee5a556a186f512eda8dabf3`

Ontology	Observation
Family	$Mary : Parent$
	$Mary : \exists hasChild.Parent$
	$Mary : \neg Male$
	$Mary : \forall hasChild.Person$
	$Mary : Grandfather \sqcup Grandmother$
Coffee	$drinkX : Base$
	$drinkX : \exists hasIngredient.SteamedMilk$
	$drinkX : \neg Drink$
	$(drinkX, drinkY) : isIngredientOf$
	$\{drinkX : ContainsChocolate, drinkX : ContainsMilk\}$
LUBM	$John : Employee$
	$John : Person \sqcap \exists worksFor.Organization$
	$John : Student$
	$John : \neg Chair$
	$(John, UU) : degreeFrom$

Table 3.3: Set of observations

times for each pair of observation and application. The results are then averaged to compare them to the other applications. The application is tested both with the test for semantically minimal explanations and without the semantically minimal check. All tests are conducted on a Dell latitude 7280 laptop running on a windows OS with a 7^{th} generation Intel i5 core.

4 | Results

The implementation AAA^{sem} of the Semantic Minimality Checker (SMC) relies on the explanations of the AAA application. This means that explanations that are generated by AAA can be removed from the set of explanations, yet no new explanations can be added to the set AAA is sound and complete for finding all explanations containing simple concepts, yet not complete for finding all explanations with complex concepts. Thus, AAA^{sem} is only complete for finding all explanations containing simple concepts. As semantic minimality is computed on the basis of comparison to other explanations, there can be semantically smaller explanations that contain complex concepts, which are not included.

This can be illustrated by the the example observation

$$E_1 = \{\mathsf{drinkX} : \mathsf{ContainsChocolate}, \mathsf{drinkX} : \mathsf{ContainsMilk}\}$$

. There are three explanations returned by AAA:

1. $\{\mathsf{drinkX} : \mathsf{Mocha}\}$,
2. $\{\mathsf{drinkX} : \mathsf{HotChocolate}\}$, and
3. $\{\mathsf{drinkX} : \mathsf{Cappucino}\}$.

All these explanations are returned as semantically minimal explanations, as they do not entail each other. If one looks in the coffee ontology two axioms can be found:

1. $\mathsf{ContainsChocolate} \equiv \exists \mathsf{hasBase.Chocolate} \sqcup \exists \mathsf{hasTopping.PowderedChocolate}$,

2. $ContainsMilk \equiv \exists hasBase.Milk \sqcup \exists hasTopping.MilkFoam$

Therefore, the following explanations would also be valid explanations according to definition 2.10:

1. $\{drinkX : \exists hasBase.Milk, drinkX : \exists hasBase.Chocolate\}$,
2. $\{drinkX : \exists hasTopping.MilkFoam, drinkX : \exists hasBase.Chocolate\}$,
3. $\{drinkX : \exists hasTopping.MilkFoam, drinkX : \exists hasTopping.PowderedChocolate\}$,
4. $\{drinkX : \exists hasBase.Milk, drinkX : \exists hasTopping.PowderedChocolate\}$

However, as these explanations contain a complex concept they are not returned by AAA and not used to test the semantic minimality of the previous explanations. If they would have been used to test the semantic minimality, not one of the explanations of AAA would be returned as semantically minimal.

Furthermore, the depth of the MHS tree is restricted to a depth of three. This means that explanations produced by $AAA_r{}^{sem}$ can contain no more than three assertions. For explanations returned by $AAA_s{}^{sem}$ longer explanations might be found, still the application is only guaranteed to find all the semantically minimal explanations that contain at most three assertions. While this might be an issue for some ontologies where long explanations can occur, most ontologies do facilitate the option to form syntactically minimal explanations that contain more than three assertions (Pukancová 2018). The ontologies and sample observations that are used in this study do not have any minimal explanations that contain more than three assertions.

Given the restriction that only simple concept assertions in $\mathcal{ALCHO}$ can be produced, both $AAA_s{}^{sem}$ and $AAA_r{}^{sem}$ manage to select the correct explanations from the given set of explanations. All the returned explanations can be found in table 4.1. The implementation has been tested on each observation three times and the given result was consistent over the different runs.

Observation	returned explanations
Mary : Parent	{Mary : Mother}, {Mary : Father}
Mary : ∃hasChild.Parent	{Mary : Grandmother}, {Mary : Grandfather}
Mary : ¬Male	{Mary : ¬Person}, {Mary : Female}
Mary : ∀hasChild.Person	{Mary : Person}
Mary : Grandfather ⊔ Grandmother	-
drinkX : Base	{drinkX : Coffee}, {drinkX : Chocolate}, {drinkX : IceCream}, {drinkX : Milk}, {drinkX : Sugar}, {drinkX : Water}
drinkX : ∃hasIngredient.SteamedMilk	{drinkX : Macchiato}, {drinkX : FlatWhite}, {drinkX : PiccoloLatte}, {drinkX : Affogato}, {drinkX : CafeConLeche}, {drinkX : HotChocolate}, {drinkX : Cappuccino}, {drinkX : CafeLatte}, {drinkX : Mocha}
drinkX : ¬Drink	{drinkX : Base}, {drinkX : Topping}
(drinkX, drinkY) : isIngredientOf	{(drinkY, drinkX) : hasIngredient}
{drinkX : ContainsChocolate, drinkX : ContainsMilk}	{drinkX : Mocha}, {drinkX : HotChocolate} {drinkX : Cappuchino}
John : Employee	{John : Director}, {John : AdministrativeStaff}, {John : ResearchAssistant}, {John : Faculty}
John : Person ⊓ ∃worksFor.Organization	{John : Director} {John : ResearchAssistant} {John : Employee}
John : Student	{John : GraduateStudent} {John : UndergraduateStudent}
John : ¬Chair	{John : ¬Professor}
(John, UU) : degreeFrom	{(UU, John) : hasAlumnus}

Table 4.1: Found explanations per observation

One interesting observation during tests of the implementation was the different outcomes for the observation $\{\mathsf{drinkX} : \mathsf{ContainsChocolate}, \mathsf{drinkX} : \mathsf{ContainsMilk}\}$ and the observation $\{\mathsf{drinkX} : \mathsf{ContainsChocolate} \sqcap \mathsf{ContainsMilk}\}$. While both observations semantically have the same meaning, drinkX is both of type $\mathsf{ContainsChocolate}$ and of type $\mathsf{ContainsMilk}$, the definition of a relevant ABox Abduction explanation (definitions 2.10) causes a difference in the returned explanations. Relevance is defined as $\mathcal{E} \not\models \Phi$ for any explanation $\mathcal{E}$ and observation Φ. This means that there are 8 relevant explanations for the observation $\{\mathsf{drinkX} : \mathsf{ContainsChocolate} \sqcap \mathsf{ContainsMilk}\}$:

1. $\{\mathsf{drinkX} : \mathsf{Mocha}\}$,
2. $\{\mathsf{drinkX} : \mathsf{HotChocolate}\}$
3. $\{\mathsf{drinkX} : \mathsf{Cappuchino}\}$
4. $\{\mathsf{drinkX} : \mathsf{Macchiato}, \mathsf{drinkX} : \mathsf{ContainsChocolate}\}$,
5. $\{\mathsf{drinkX} : \mathsf{ContainsChocolate}, \mathsf{drinkX} : \mathsf{PiccoloLatte}\}$,
6. $\{\mathsf{drinkX} : \mathsf{Bonbon}, \mathsf{drinkX} : \mathsf{ContainsChocolate}\}$,
7. $\{\mathsf{drinkX} : \mathsf{Affogato}, \mathsf{drinkX} : \mathsf{ContainsChocolate}\}$,
8. $\{\mathsf{drinkX} : \mathsf{ContainsChocolate}, \mathsf{drinkX} : \mathsf{CafeConLeche}\}$,
9. $\{\mathsf{drinkX} : \mathsf{FlatWhite}, \mathsf{drinkX} : \mathsf{ContainsChocolate}\}$,

While explanation 4 to 8 might seem irrelevant as they partly copy the observation, $\mathcal{E} \not\models \Phi$ actually holds for every one of them. Now if the observation is split into a multiple observation problem, $\{\mathsf{drinkX} : \mathsf{ContainsChocolate}, \mathsf{drinkX} : \mathsf{ContainsMilk}\}$, then explanations 4 to 8 are actually irrelevant, e.g. $\{\mathsf{drinkX} : \mathsf{Macchiato}, \mathsf{drinkX} : \mathsf{ContainsChocolate}\} \not\models \Phi_1$. The semantic minimality checker does not eliminate any of the explanations as they have no entailment relation to each other. It depends on whether a user accepts explanations that are partly equal to the given observation, which input is preferable. Still users have to be aware that running two seemingly similar inputs $\mathsf{a} : \mathsf{C} \sqcap \mathsf{D}$ and $\{\mathsf{a} : \mathsf{C}, \mathsf{a} : \mathsf{D}\}$ can deliver different results.

4.1 Performance

To compare the performance of the implementation some extra runs have been done on each ontology. For the Family ontology the performance tests are done with the observation Mary : Parent, for the Coffee ontology the observation drinkX : Base was used, and the LUBM ontology was tested with the observation John : Employee. Each test has been repeated 5 times, the performance results are shown in table 4.2.

Algorithm	Runtime *seconds*	TA calls *average*	TA calls *maximum*	Nodes *number*	Explanations *average*
Family					
AAA_s^{sem}	3,20	29	29	91	4
AAA_s	3,14	22	22	91	4
AAA_r^{sem}	3,15	28	28	91	4
AAA_r	3,18	22	22	91	4
Coffee					
AAA_s^{sem}	98,43	3.360	3.361	7.381	13
AAA_s	94,89	3.298	3.300	7.217	13
AAA_r^{sem}	66,80	3.361	3.366	7.208	13
AAA_r	106,47	3.328	3.330	7.258	13
LUBM					
AAA_s^{sem}	124,88	2.376	2.378	16.298	15
AAA_s	126,02	2.487	2.489	16.298	15
AAA_r^{sem}	125,46	2.755	2.757	16.298	15
AAA_r	139,18	2.468	2.474	16.298	15

Table 4.2: Performance of the algorithms

Table 4.2 shows that all the implementations do a lot of TA calls, whether the semantic minimality check is done or not. While the use of the semantic minimality checker does add some more tableau algorithm checks in general, there are cases that it does less checks (for the LUBM ontology AAA_s^{sem} against AAA_s). This is mainly because the number of extra TA calls that the semantic minimality checker does is smaller than the variance in TA calls of the MHS algorithm, i.e. a bad run of the MHS algorithm affects the number of TA calls more than the semantic minimality checker adds. This effect is only visible in larger ontologies, for the ontology Family running AAA^{sem} clearly increases the number of TA calls in comparison to AAA.

The runtime of the algorithms is hardly influenced by the use of the semantic minimality option.

Again, the added time that the semantic minimality checker might need is subservient to the time variance of computing the MHS algorithm. While the number of nodes varies a bit for the Coffee ontology, there is no clear trend that shows an influence of the SMC on the number of nodes.

While running the SMC has a small effect on the number of TA calls, it is not clear from this table what portion can actually be attributed to the use of the SMC and what is due to variance of the MHS algorithm. Let us check the performance of the runs done to check the correctness of the implementation. The average number of TA calls that the SMC has done for each observation is displayed in table 4.3. Note that these averages only include the runs on AAA_r^{sem}, as the use of reduction or splitting only influences the MHS algorithm, it is not relevant to distinguish between them for the performance of SMC.

Observation	Explanations *all*	Explanations *sem. minimal*	TA calls *by SMC*
$\mathsf{Mary} : \mathsf{Parent}$	4	2	7
$\mathsf{Mary} : \exists \mathsf{hasChild.Parent}$	2	2	2
$\mathsf{Mary} : \neg \mathsf{Male}$	5	2	12
$\mathsf{Mary} : \forall \mathsf{hasChild.Person}$	8	1	20
$\mathsf{Mary} : \mathsf{Grandfather} \sqcup \mathsf{Grandmother}$	0	0	0
$\mathsf{drinkX} : \mathsf{Base}$	13	6	27
$\mathsf{drinkX} : \exists \mathsf{hasIngredient.SteamedMilk}$	11	8	15
$\mathsf{drinkX} : \neg \mathsf{Drink}$	17	2	39
$(\mathsf{drinkX}, \mathsf{drinkY}) : \mathsf{isIngredientOf}$	5	1	13
$\{\mathsf{drinkX} : \mathsf{ContainsChocolate}, \mathsf{drinkX} : \mathsf{ContainsMilk}\}$	3	3	3
$\mathsf{John} : \mathsf{Employee}$	15	4	35
$\mathsf{John} : \mathsf{Person} \sqcap \exists \mathsf{worksFor.Organization}$	16	3	42
$\mathsf{John} : \mathsf{Student}$	2	2	2
$\mathsf{John} : \neg \mathsf{Chair}$	4	1	8
$(\mathsf{John}, \mathsf{UU}) : \mathsf{degreeFrom}$	4	1	8

Table 4.3: Average performance of the SMC, calculated over 3 runs of AAA_r^{sem}

In table 4.3 it can be seen how many TA calls the SMC has done for each observation, on average. From comparing table 4.3 to table 4.2 it is clear that the number of TA calls that SMC does is mainly important for the smaller ontology Family. Let us look at the observation $\mathsf{Mary} : \mathsf{Parent}$, from table 4.2 we know that the MHS does roughly 22 TA calls. In table 4.3 it can be seen that the SMC does roughly 7 TA calls to find the semantically minimal explanations

which is about a third of the calls that the MHS algorithm did. So for this observation the SMC has a strong influence on the number of TA calls. However, for the observations $\mathsf{drinkX} : \mathsf{Base}$ and $\mathsf{John} : \mathsf{Employee}$ the number of TA calls that the SMC is relatively small in comparison to the number of calls the MHS algorithm does (27 versus 3328 for $\mathsf{drinkX} : \mathsf{Base}$ and 35 versus 2468 for $\mathsf{John} : \mathsf{Employee}$). This can be explained as the number of TA calls done by the MHS algorithm is mainly influenced by the size of the ontology, while the number of TA calls done by SMC is mainly based on the number of explanations that it has to check. As the number of explanations is not directly related to the size of an ontology, the influence that the SMC has on the performance becomes less significant when bigger ontologies are used.

Furthermore, table 4.3 shows that while the number of TA calls increases with the size of the explanation set, it does not grow exponentially. This shows that the optimisation techniques that are implemented in the SMC do have an influence on the performance. When no optimisation techniques would be used, at least one tableau call twould be needed for each combination of explanations. This means that for n explanations $n * (n-1)$ TA calls would be needed. In table 4.3 it can clearly be seen that for bigger explanation sets the number of TA calls is significantly lower than it would be without optimisation. Let us take the observation $\mathsf{John} : \mathsf{Person} \sqcap \exists \mathsf{worksFor}.\mathsf{Organization}$ as an example. The MHS returns 16 explanations, without any optimisation techniques 240 TA calls would be needed $(6 * 15)$, while the SMC only does 42 TA calls.

While the optimisation techniques reduce the number of TA calls significantly for big explanation sets, it has less influence on the smaller explanation sets. As the optimisation techniques are based on reusing information, this result is not surprising because one first has to find information in order to reuse it. Figure 4.1 shows the effect that the optimisation techniques have per explanation set size. The figure shows the ratio of the number of TA calls that are done by the SMC to the number of TA calls that could be expected without the optimisation techniques plotted against the input size. This ratio decreases rapidly when the explanation set size is bigger than 2. When the MHS algorithm returns more than 12 explanations the number of TA calls done by SMC is roughly 20% of the TA calls that would be needed without optimisation techniques.

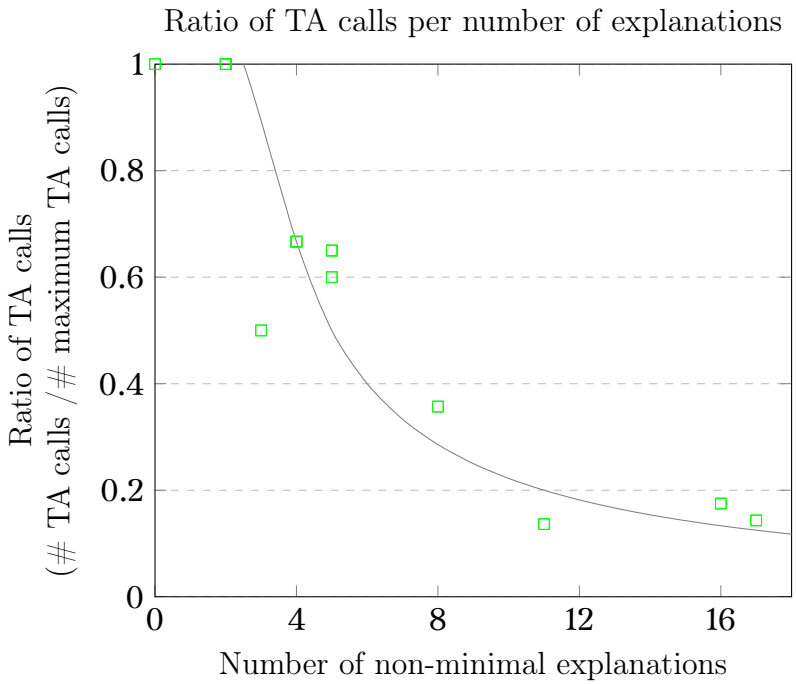


Figure 4.1: Ratio between actual TA calls and maximum TA calls per explanation set size.

5 | Conclusion & Discussion

This thesis builds on the foundations of ABox abduction to study the workings of semantically minimal explanations in depth. Previous research was strongly focused at finding explanations first and testing semantic minimality in a later stadium (Del-Pinto and Schmidt 2018), or even accepting semantically non-minimal explanations (Klarman et al. 2011; Pukancová and Homola 2015; Mrózek et al. 2018). The approaches studied in this paper give a deep insight in the formalisation of semantically minimal explanations. Although, this study did not succeed in developing an implementation that can recognize semantically minimal explanations at an early stage, the study gives a deep insight in the formalisation of semantically minimal explanations, and provides a working implementation for finding semantically minimal explanations in $\mathcal{ALCHO}$.

This thesis introduces two new algorithms to find semantically minimal ABox explanations to observations in the DL $\mathcal{ALCHO}$: SEMAR and the SMC algorithm. SEMAR is an adjusted tableau algorithm, introduced in section 3.1. This algorithm was aimed to optimize a standard tableau algorithm so it can identify semantically minimal explanations to an ABox abduction problem. The initial aim was to prove that SEMAR returns all and only all semantically minimal explanations to an ABox observation. SEMAR is only proven to be sound for ontologies that do not contain General Concept Inclusions (GCI's) that contain nominals, and single observations in negated normal form that do not contain any conjunctions. Completeness of SEMAR could not be formally proven, as many theoretical exceptions could occur due to the inclusion of nominals.

Furthermore, building an implementation of SEMAR was problematic because of the complex

optimization techniques added to implementations of the tableau algorithms for DL. Therefore, this thesis cannot answer how the tableau algorithm can be optimized to generate models that lead to all and only all semantically minimal solutions. It does provide a sound base that further research could build on to develop an implementation that is complete for finding all semantic minimal solutions using an adjusted tableau algorithm.

Section 3.1 suggests that further research is needed into the integration of lazy unfolding rules into SEMAR. The use of lazy unfolding rules is a common optimizing technique used by reasoners that are based on a tableau algorithm. Integrating them into an implementation of SEMAR might help solve the encountered memory issues.

Additionally, further research is suggested by section 3.1 into making SEMAR compatible for multi-assertion observations. When SEMAR is sound and complete for multi-assertion observation, then observations that contain conjunctions can be split into multiple assertions. Therefore, SEMAR would be able to process observations containing conjunctions as well.

Section 3.2 introduces the Semantic Minimality Checker (SMC) that is developed to select semantically minimal ABox abduction explanations from an existing set of explanations. This algorithm is proven to be correct and has a worst case complexity in ExpTime, when combined with AAA. The SMC algorithm is a sound and complete method that can complement the minimal hitting set such that all and only all semantically minimal solutions can be found. While it could be used to prune branches in a hitting set tree, section 3.2 explains that this is not advisable because it would not improve the performance of a minimal hitting set algorithm. To select the semantically minimal explanations from the explanations produced by a minimal hitting set algorithm the SMC algorithm could implement some optimization techniques that reduce the number of needed tableau algorithm calls, which has a bigger advantage for the performance than the pruned branches would give.

Lastly, an implementation of the second algorithm is tested on correctness and performance. The added number of TA calls of SMC is less or equal than what a simple semantic minimality algorithm would add. The performance advantages that SMC has over a simple semantic minimality algorithm increases significantly when the explanation set that has to be tested is

bigger than 2. Furthermore, the SMC algorithm has little influence on the computation time needed for the MHS algorithm, measured both in runtime and in number of TA calls. Especially on bigger ontologies the added computation time of SMC is relatively small in comparison to the computation time needed for the MHS algorithm.

While in theory the SMC algorithm can be used on any set of explanations, the implementation developed and tested for this thesis only uses the input of the AAA implementation. For future research it would be interesting to test the algorithm in combination with other ABox abduction implementations. Furthermore, testing the performance of this implementation against other implementations that return semantically minimal explanations would be valuable future research.

Lastly, this thesis introduces approaches to generate semantically minimal explanations. However, the results are still a set of possible explanations. For systems that ultimately need one explanation for the observed phenomena, further research is needed to select the best explanation out of the set of returned explanations. An interesting step in this direction is to develop a system that can do a targeted search for missing observations, based on what should be observable when an explanation is true.

Appendices

A | Concept operators

The following five operators can be used to construct complex concepts:

- $\sqcap$ A conjunction in the form of $A \sqcap B$ describes all elements that are in the extension of both A and B. The assertion $Mary : Employee \sqcap Manager$ can be read as "Mary is both an employee and a manager".

- $\sqcup$ A disjunction in the form of $A \sqcup B$ describes all elements that are in the extension of either A or B. The assertion $Mary : Employee \sqcup Employer$ can be read as "Mary is either employee or employer".

- $\neg$ A negation in the form of $\neg A$ describes all elements that are not in the extension of A. The assertion $Mary : \neg Employee$ can be read as "Mary is not an employee".

- $\exists$ An existential restriction in the form of $\exists r.A$ describes all elements that have at least one r-filler that is in the extension of A. The assertion $Mary : \exists worksFor.Employer$ can be read as "Mary works for at least one element that is an employer".

- $\forall$ An value restriction in the form of $\forall r.A$ describes all elements for which every r-filler is in the extension of A. The assertion $Mary : \forall worksAt.Department$ can be read as "Mary only works at elements that are a department". Note that every element that does not have an r-filler is automatically in the extension of $\forall r.X$, regardless of the concept X, because it still holds that every r-filler (which is none) is in the extension of X.

B | Additional DL knowledge base definitions

In section 2.2 of this thesis the basic theory behind description logics is explained. For the sake of readability not all formal definitions are presented in the DL section. For readers that are not familiar with DLs the work of Baader et al. (2017) provides a proper overview of the theory behind DLs. This appendix provides some additional definitions that might be helpful for readers not familiar with Baader's work.

Definition B.1 (ABox assertions)**.** *Let* $\mathbf{I}$ *be a set of individual names disjoint from the set of concept names* $\mathbf{C}$ *and the set of role names* $\mathbf{R}$*. For individual names* $\mathtt{a}, \mathtt{b} \in \mathbf{I}$*, a possibly compound concept* C*, and a role name* $\mathtt{r} \in \mathbf{R}$*, an expression of the form:*

- $\mathtt{a} : \mathsf{C}$ *is called a concept assertion, and*
- $(\mathtt{a}, \mathtt{b}) : \mathtt{r}$ *is called a role assertion.*

An ABox assertion can either be a concept assertion, or a role assertion. A finite set of ABox assertions is called an ABox.

Definition B.2 (TBox axioms)**.** *For possibly compound concepts* C *and* D*, an expression of the form:*

- $\mathsf{C} \sqsubseteq \mathsf{D}$ *is called a general concept inclusion (GCI), and*
- $\mathsf{C} \equiv \mathsf{D}$ *is called an equivalence axiom.*

An equivalence axiom $\mathsf{C} \equiv \mathsf{D}$ *is an abbreviation for* $\mathsf{C} \sqsubseteq \mathsf{D}, \mathsf{D} \sqsubseteq \mathsf{C}$. *An TBox axiom can either be a general concept inclusion, or an equivalence axiom. A finite set of TBox axioms is called a TBox.*

Definition B.3 (Nominal)**.** *For an individual name* $\mathfrak{b} \in \mathbf{I}$, $\{\mathfrak{b}\}$ *is called a nominal. For any DL* $\mathcal{L}$*, the DL* $\mathcal{LO}$ *is obtained from* $\mathcal{L}$ *by allowing nominals as additional concepts. For an interpretation* $\mathcal{I}$*, its mapping* $\cdot^{\mathcal{I}}$ *is extended as follows:*

$$(\{\mathfrak{a}\})^{\mathcal{I}} = \{\mathfrak{a}^{\mathcal{I}}\}$$

Definition B.4 (Role inclusion axiom)**.** *For roles* $\mathfrak{r}, \mathfrak{s} \in \mathbf{R}$, $\mathfrak{r} \sqsubseteq \mathfrak{s}$ *is called a role inclusion axiom (RIA). For any DL* $\mathcal{L}$*, the DL* $\mathcal{LH}$ *is obtained from* $\mathcal{L}$ *by allowing role inclusion axioms in the TBoxes. For an interpretation* $\mathcal{I}$ *to be a model of a* $\mathcal{LH}$ *TBox it has to satisfy all concept and role inclusion axioms in the TBox. A role inclusion axiom* $\mathfrak{r} \sqsubseteq \mathfrak{s}$ *is satisfied by an interpretation* $\mathcal{I}$ *iff*

$$\mathfrak{r}^{\mathcal{I}} \subseteq \mathfrak{s}^{\mathcal{I}}.$$

Definition B.5 (Basic DL reasoning problems)**.** *Let* $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ *be a knowledge base, C and D possibly compound concepts, and* $\mathfrak{b}$ *an individual name. We say that:*

1. C *is satisfiable with respect to* $\mathcal{T}$ *if there exists a model* M *of* $\mathcal{T}$ *and some* $\mathfrak{d} \in \Delta^{\mathsf{M}}$ *with* $\mathfrak{d} \in \mathsf{C}^{\mathsf{M}}$*;*
2. C *is subsumed by* D *with respect to* $\mathcal{T}$*, written* $\mathcal{T} \vDash \mathsf{C} \sqsubseteq \mathsf{D}$*, if* $\mathsf{C}^{\mathsf{M}} \subseteq \mathsf{D}^{\mathsf{M}}$ *for every model* M *of* $\mathcal{T}$*;*
3. C *and* D *are equivalent with respect to* $\mathcal{T}$*, written* $\mathcal{T} \vDash \mathsf{C} \equiv \mathsf{D}$*, if* $\mathsf{C}^{\mathsf{M}} = \mathsf{D}^{\mathsf{M}}$ *for every model* M *of* $\mathcal{T}$*;*
4. $\mathcal{K}$ *is consistent if there exists a model of* $\mathcal{K}$*;*
5. $\mathfrak{b}$ *is an instance of* C *with respect to* $\mathcal{K}$*, written* $\mathcal{K} \vDash \mathfrak{b} : \mathsf{C}$*, if* $\mathfrak{b}^{\mathsf{M}} \in \mathsf{C}^{\mathsf{M}}$ *for every model* M *of* $\mathcal{K}$*.*

Bibliography

Aliseda-Llera, A. (1997). "Seeking Explanations: Abduction in Logic, Philosophy of Science and Artificial Intelligence". PhD thesis. Amsterdam: Institute for Logic, Language and Computation (cit. on pp. 5, 9, 11, 12, 17).

Baader, F., Horrocks, I., Lutz, C., and Sattler, U. (2017). *Introduction to Description Logic*. Cambridge University Press (cit. on pp. 18, 22, 25, 78, 80, 97).

Barnes, E. (1995). "Inference to the loveliest explanation". *Synthese* 103.2, pp. 251–277. (Cit. on p. 10).

Campos, D. G. (2011). "On the distinction between Peirce's abduction and Lipton's inference to the best explanation". *Synthese* 180.3, pp. 419–442. (Cit. on pp. 8, 11).

Del-Pinto, W. and Schmidt, R. A. (2018). "ABox abduction via forgetting in $\mathcal{ALC}$ (long version)". *arXiv preprint*. arXiv: `1811.05420 [cs.AI]` (cit. on pp. 7, 30, 82, 92).

Du, J., Qi, G., Shen, Y.-D., and Pan, J. Z. (2011). "Towards practical ABox abduction in large OWL DL ontologies." In: *Proceedings of the 25th AAAI Conference on Artificial Intelligence*. (San Francisco, California, US, Aug. 7–11, 2011) (cit. on pp. 28, 82).

Du, J., Wan, H., and Ma, H. (2017). "Practical TBox abduction based on justification patterns." In: *Proceedings of the 31st AAAI Conference on Artificial Intelligence*. (San Francisco, California, US, Feb. 4–9, 2017). Ed. by S. Singh and S. Markovitch (cit. on p. 27).

Du, J., Wang, K., and Shen, Y.-D. (2015). "Towards tractable and practical ABox abduction over inconsistent description logic ontologies." In: *Proceedings of the 29th AAAI Conference on Artificial Intelligence*. (Austin, Texas, US, Jan. 25–29, 2015) (cit. on p. 28).

Du, J., Wang, K., Shen, Y.-D., et al. (2014). "A tractable approach to ABox abduction over description logic ontologies." In: *Proceedings of the 28th AAAI Conference on Artificial In-*

telligence, AAAI 2014. (Québec City, Québec, Canada, July 27–31, 2014) (cit. on pp. 7, 28, 29).

Elsenbroich, C., Kutz, O., and Sattler, U. (2006). "A case for abductive reasoning over ontologies". In: *Proceedings of the 3rd OWLED 2006 Workshop on OWL: Experiences and Directions*. (Athens, Georgia, US, Nov. 10–11, 2006). Ed. by B. C. Grau, P. Hitzler, C. Shankey, and E. Wallace. CEUR Workshop Proceedings 216 (cit. on pp. 6, 12, 26, 28).

Guo, Y., Pan, Z., and Heflin, J. (2005). "Lubm: a benchmark for owl knowledge base systems". *Journal of Web Semantics* 3.2, pp. 158–182. (Cit. on p. 82).

Halland, K. and Britz, K. (2012). "ABox abduction in $\mathcal{ALC}$ using a DL tableau". In: *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference*. SAICSIT '12. Pretoria, South Africa: ACM, pp. 51–58. (Cit. on pp. 29, 30, 36).

Harman, G. H. (1965). "The inference to the best explanation". *The Philosophical Review* 74.1, pp. 88–95 (cit. on p. 10).

Klarman, S., Endriss, U., and Schlobach, S. (2011). "ABox abduction in the description logic $\mathcal{ALC}$". *Journal of Automated Reasoning* 46.1, pp. 43–80. (Cit. on pp. 28, 29, 92).

Koopmann, P. and Schmidt, R. A. (2015). "LETHE: saturation-based reasoning for non-standard reasoning tasks." In: *Proceedings of the 4th International Workshop on OWL Reasoner Evaluation*. (Athens, Greece, June 6, 2015). Ed. by M. Dumontier, B. Glimm, R. Gonçalves, M. Horridge, E. Jiménez-Ruiz, N. Matentzoglu, B. Parsia, G. Stamou, and G. Stoilos. CEUR Workshop Proceedings 1387. Aachen, pp. 23–30 (cit. on p. 30).

Lipton, P. (2003). *Inference to the Best Explanation*. Routledge (cit. on pp. 5, 10).

Lucas, P. (1997). "Symbolic diagnosis and its formalisation". *The Knowledge Engineering Review* 12.2, pp. 109–146 (cit. on pp. 12, 17).

Ma, Y., Gu, T., Xu, B., and Chang, L. (2012). "An ABox abduction algorithm for the description logic $\mathcal{ALCI}$". In: *Intelligent Information Processing VI*. Ed. by Z. Shi, D. Leake, and S. Vadera. Vol. 385. Berlin, Heidelberg: Springer, pp. 125–130 (cit. on p. 29).

Mayer, M. C. and Pirri, F. (1993). "First-order abduction via tableau and sequent calculi". *Logic Journal of the IGPL* 1.1, pp. 99–117. URL: https://doi.org/10.1093/jigpal/1.1.99 (cit. on p. 12).

Mill, J. S. (2011). *A System of Logic, Ratiocinative and Inductive. Being a Connected View of the Principles of Evidence, and the Methods of Scientific Investigation*. Vol. 2. Cambridge Library Collection - Philosophy. Cambridge University Press. URL: https://doi.org/10.1017/CBO9781139149846 (cit. on p. 8).

Mrózek, D., Pukancová, J., and Homola, M. (2018). "ABox abduction solver exploiting multiple DL reasoners". In: Ortiz and Schneider (2018) (cit. on pp. 7, 29, 30, 39, 40, 81, 92).

Ortiz, M. and Schneider, T., eds. (2018). *Proceedings of the 31st International Workshop on Description Logics (DL)*. (Tempe, Arizona, US, Oct. 27–29, 2018). CEUR Workshop Proceedings 2211. Aachen (cit. on p. 101).

Pearl, J. (2009). *Causality. Models, Reasoning and Inference*. Cambridge University Press. URL: https://doi.org/10.1017/CBO9780511803161 (cit. on pp. 11, 15).

Pearl, J. and Mackenzie, D. (2018). *The Book of Why. The New Science of Cause and Effect*. New York: Basic Books (cit. on p. 5).

Peirce, C. S. (1878). "Deduction, induction, and hypothesis". *Popular Science Monthly* 13, pp. 470–482 (cit. on pp. 5, 10, 12).

Pukancová, J. (2018). "Direct Approach to ABox Abduction in Description Logics". PhD thesis. Bratislava: Faculty of Mathematics, Physics and Informatics, Comenius University in Bratislava (cit. on pp. 71, 76, 82, 85).

Pukancová, J. and Homola, M. (2015). "Abductive reasoning with description logics: use case in medical diagnosis." In: *Proceedings of the 28th International Workshop on Description Logics (DL)*. (Athens, Greece, June 7–10, 2015). Ed. by D. Calvanese and B. Konev. CEUR Workshop Proceedings 1350 (cit. on pp. 7, 72, 82, 92).

– (2017). "Tableau-based ABox abduction for the $\mathcal{ALCHO}$ description logic." In: *Proceedings of the 30st International Workshop on Description Logics (DL)*. (Montpellier, France, July 18, 2017–July 21, 2018). Ed. by A. Artale, B. Glimm, and R. Kontchakov. CEUR Workshop Proceedings 1879 (cit. on pp. 7, 29, 36, 38, 39, 81).

– (2018). "ABox abduction for description logics: the case of multiple observations." In: Ortiz and Schneider (2018) (cit. on pp. 29, 38–40, 42, 43, 81).

Reiter, R. (1987). "A theory of diagnosis from first principles". *Artificial intelligence* 32.1, pp. 57–95. (Cit. on pp. 12, 29, 31–35, 38).

Zhao, Y. and Schmidt, R. (2018). "On concept forgetting in description logics with qualified number restrictions". In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, pp. 1984–1990. (Cit. on p. 27).

Index