TBox abduction in $\mathcal{ALC}$ using a DL tableau

Ken Halland[1 2], Katarina Britz[2] and Szymon Klarman[2]

[1] School of Computing, University of South Africa, Pretoria, South Africa
[2] Centre for Artificial Intelligence Research: UKZN and CSIR Meraka Institute, South Africa

hallakj@unisa.ac.za, abritz@csir.co.za and sklarman@csir.co.za

Abstract. The formal definition of abduction asks what needs to be added to a knowledge base to enable an observation to be entailed. TBox abduction in description logics (DLs) asks what TBox axioms need to be added to a DL knowledge base to allow a TBox axiom to be entailed. We describe a sound and complete algorithm, based on the standard DL tableau, that takes a TBox abduction problem in $\mathcal{ALC}$ and generates solutions in a restricted language. We then show how this algorithm can be enhanced to deal with a broader range of problems in $\mathcal{ALC}$.

Keywords: Description logics, abduction, tableau algorithms

1 Introduction

Abduction can be viewed as a form of non-standard reasoning where explanations are sought for certain observations in the context of some background knowledge. Stated differently, abductive reasoning is generally used to generate hypotheses about the possible or plausible causes of some phenomenon. Typical uses of abduction are in the fields of medical diagnosis, fault diagnosis and criminal investigation.

In formal logic, an abduction problem is normally specified in terms of an observation (in the form of one or more statements) which is not entailed by some background knowledge (in the form of a knowledge base), and asks what needs to be added to the background knowledge to entail the observation.

The biggest problem is how to narrow down the possibly infinite number of solutions to an abduction problem. Various criteria have been defined for this purpose, for example *consistency* – a solution should not introduce a contradiction with the background knowledge, *relevance* – a solution should be expressed in terms of the background knowledge, i.e. it should not independently entail the observation, and *minimality* – a solution should not hypothesize more than necessary.

Different forms of abduction have been defined formally in different logics. In their programmatic paper, Elsenbroich *et al.* [6] define and describe various forms of abduction in description logics (DLs). Among these is TBox abduction, which is useful for repairing missing subsumptions to debug a knowledge base.

In this paper, we describe an algorithm for performing a simplified form of TBox abduction, where the knowledge base and observation are in the DL $\mathcal{ALC}$, but the solutions are in a restricted language. The algorithm is sound and complete with respect to the restricted language and a minimality criterion. This extends previous work on ABox abduction [7].

Section 2 specifies some of the DL terminology used in this paper and gives an overview of the standard tableau algorithm for $\mathcal{ALC}$. Section 3 includes a definition of TBox abduction, and Section 4 describes an algorithm based on the standard algorithm to perform TBox abduction in $\mathcal{ALC}$. Section 5 explains how our abduction algorithm can be enhanced to deal with a broader range of abduction problems, and Section 6 provides an analysis of this algorithm in terms of its soundness, completeness and complexity. Finally, Section 7 discusses the prospects for future work.

2 Preliminaries

In this section, we firstly highlight some DL terminology relevant to our purposes. We then give a general description of the standard tableau algorithm for DLs and highlight aspects that are important for our current purposes. For details of DLs in general, for the definitions of the syntax and semantics of $\mathcal{ALC}$ in particular, and for a more detailed specification of the tableau algorithm, the reader is referred to the Description Logic Handbook [1].

A knowledge base $\mathcal{K}$ is *consistent* if it admits a model. A concept is *unsatisfiable* w.r.t. a knowledge base $\mathcal{K}$ if its interpretation is empty in all models of $\mathcal{K}$. An ABox assertion or TBox axiom α is *entailed* by a knowledge base $\mathcal{K}$ if α is true in all models of $\mathcal{K}$, in which case we write $\mathcal{K} \models \alpha$. In an abuse of notation, we often write $\mathcal{K} \models \Omega$ where Ω is a set of assertions and/or axioms. By this we mean that $\mathcal{K} \models \alpha$ for all $\alpha \in \Omega$.

The tableau algorithm is a decision procedure for the consistency of a knowledge base. It tries to find (by a depth-first search) a model of the knowledge base by applying so-called *expansion rules*. The expansion rules only apply to assertions, so before the algorithm starts, the axioms in the knowledge base are converted to concept assertions by a process called *internalisation*. These assertions are added to the set of assertions on which the algorithm commences its work. The algorithm repeatedly attempts to apply the expansion rules to the assertions in this set until either a *clash* occurs (i.e. the set contains an assertion and its negation) in which case the algorithm backtracks to a splitting point, or until the set is *saturated* (i.e. no further expansion rules can be applied) in which case the algorithm terminates and reports that the knowledge base is consistent. A branch that ends in a clash is called a *closed branch* and a branch that ends in a saturated set is called an *open branch*. If all branches of the tableau are closed, the algorithm reports that the knowledge base is inconsistent.

Stated more formally: Given an axiom $\varphi = C \sqsubseteq D$ and an individual name a, let $\mu(\varphi, a)$ be the assertion $\neg C \sqcup D(a)$ and let $\neg\mu(\varphi, a)$ be the assertion $C \sqcap \neg D(a)$.

Then, given a knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, the root node of the search tree is labelled with the set of assertions $\Psi_0 = \mathcal{A} \cup \{\mu(\varphi, a) : \varphi \in \mathcal{T}$ and a used in $\mathcal{A}\}$. (If $\mathcal{A}$ is empty and there are thus no individual names, a dummy individual is introduced for the purpose of internalisation.) All assertions in Ψ_0 are converted to negation normal form. For each node labelled with Ψ_i, an expansion rule is chosen that can be applied to one or more assertions in Ψ_i. If the expansion rule is not a splitting rule, one or two simpler assertions are added to form an expanded set Ψ_{i+1} which is used to label the next node. If the expansion rule is a splitting rule (i.e. the $\sqcup$-rule), the algorithm forms two branches, removing the assertion to which the rule was applied (e.g. $C \sqcup D(a)$) and adding its component parts ($C(a)$ or $D(a)$) to form the sets Ψ_{i+1} and Ψ_{j+1} that label the nodes on each branch. One of the branches is chosen (say the one labelled with Ψ_{i+1}) and the process is repeated until either the current labelling set contains a clash, i.e. $\Psi_j \supseteq \{\psi_k, \neg\psi_k\}$, in which case the algorithm backtracks to node labelled with Ψ_{j+1}, or until no application of an expansion rule would expand the set, i.e. $\Psi_j = \{\psi_1, ..., \psi_{n'}\}$ is saturated, in which case the algorithm terminates.

If the axioms in the knowledge base form a so-called *cyclic TBox*, infinite branches can occur, but this can be addressed by a technique called *blocking* [1]. The algorithm also terminates when a branch is blocked, because the infinite branch represents an infinite model of $\mathcal{K}$ and is considered an open branch.

The standard algorithm described above performs *consistency checking* of a knowledge base. It can easily be adapted to perform the related reasoning task of *subsumption testing* [1], i.e. deciding whether an axiom φ is entailed by a knowledge base $\mathcal{K}$, as follows: The assertion $\neg\mu(\varphi, c)$ (where c is an individual name that does not appear in $\mathcal{K}$) is added to $\mathcal{K}$ and the algorithm described above is executed. If the algorithm reports that $\mathcal{K} \cup \{\neg\mu(\varphi, c)\}$ is consistent, we conclude that $\mathcal{K} \not\models \varphi$ (and *vice versa*).

3 TBox abduction

Attempts have been made to define abduction and implement reasoners that can make abductive inferences in many logics, including description logics [4, 5, 7–9]. *TBox abduction* (as opposed to general or so-called *knowledge base* abduction) asks what TBox axioms need to be added to a DL knowledge base to allow an observation (also in the form of a TBox axiom) to be inferred [6]:

Definition 1. *Let $\mathcal{L}$ and $\mathcal{L}'$ be DLs, $\mathcal{K}$ a knowledge base in $\mathcal{L}$ and $\varphi = C \sqsubseteq D$ a TBox axiom in $\mathcal{L}$ such that C and D are satisfiable concepts w.r.t. $\mathcal{K}$, $\mathcal{K}$ does not entail φ and $\mathcal{K} \cup \{\varphi\}$ is consistent. The pair $\langle \mathcal{K}, \varphi \rangle$ is called a TBox abduction problem. A set of TBox axioms Θ in $\mathcal{L}'$ is called an abductive solution for $\langle \mathcal{K}, \varphi \rangle$ if $\mathcal{K} \cup \Theta \models \varphi$.*

In general there are many solutions to any TBox abduction problem. We narrow down the solutions in three ways:

(i) *Consistency*: $\mathcal{K} \cup \Theta$ is consistent.

(ii) *Relevance*: φ is not entailed by Θ.
(iii) *Minimality*: We distinguish three types:
  (a) *Syntactic minimality*: No proper subset of Θ is a solution.
  (b) *Semantic minimality*: There is no non-equivalent solution Θ' such that $\Theta \models \Theta'$.
  (c) *Strong semantic minimality*: There is no non-equivalent solution Θ' such that $\mathcal{K} \cup \Theta \models \mathcal{K} \cup \Theta'$.

Note that these notions of minimality are defined for the target language $\mathcal{L}'$.

The following proposition states some relationships between the three types of minimality:

Proposition 1. *(a) Every equivalence class of semantically minimal solutions has at least one syntactically minimal representative. (b) Every strongly semantically minimal solution is semantically minimal.*

Proof. (a) Let E be an equivalence class of semantically minimal solutions, and $\Theta \in E$. Let Θ' be a solution which is a syntactically minimal subset of Θ. Since $\Theta' \subseteq \Theta$, $\Theta \models \Theta'$. But since Θ is a semantically minimal solution such that $\Theta \models \Theta'$, Θ' must be equivalent to Θ (by definition of semantic minimality) and therefore an element of E. So E contains a syntactically minimal solution.
(b) Say Θ is a solution to an abduction problem, but is not semantically minimal. Then, by definition of semantic minimality, there is a non-equivalent solution Θ' such that $\Theta \models \Theta'$. Therefore $\mathcal{K} \cup \Theta \models \mathcal{K} \cup \Theta'$. In other words, Θ is not a strong semantically minimal solution. $\square$

Of the three types of minimality, semantic minimality is particularly useful for implementing the notion of not hypothesizing more than is necessary to entail an observation. This is built into the algorithm which we describe below, and provides the criterion for its completeness. Syntactic minimality is useful for discarding solutions that contain redundant axioms, and is also built into the algorithm. Strong semantic minimality is more useful for ranking solutions, since the definition induces a partial ordering on the set of semantically minimal solutions. We say that a solution Θ is *closer to strong semantic minimality* than a solution Θ' if $\mathcal{K} \cup \Theta' \models \mathcal{K} \cup \Theta$ and $\mathcal{K} \cup \Theta \not\models \mathcal{K} \cup \Theta'$.

4 Algorithm for TBox abduction

The basic idea is to use the standard tableau algorithm described in Section 2 to test whether an observation (in the form of an axiom) is entailed by a knowledge base. If the observation is not entailed (it should not be, by the definition of a TBox abduction problem), at least one branch of the tableau will not close. Any set of axioms that would close all such open branches (if added to the original knowledge base plus the negated observation), would form an abductive solution.

This can be illustrated by the following simple example: Say we test whether an observation φ is entailed by a knowledge base $\mathcal{K}$, and it is not entailed due to a single open branch containing the assertions $A_1(a)$ and $\neg A_2(a)$. Then, if we were to start the algorithm again to test whether φ is entailed by $\mathcal{K} \cup \{A_1 \sqsubseteq A_2\}$, this branch would be closed, since $A_1 \sqsubseteq A_2$ would be internalised as $\neg A_1 \sqcup A_2(a)$, which would create a branch with a clash $A_1(a)$ and $\neg A_1(a)$ and a branch with a clash $\neg A_2(a)$ and $A_2(a)$. $A_1 \sqsubseteq A_2$ is therefore an abductive solution for $\langle \mathcal{K}, \varphi \rangle$ since $\mathcal{K} \cup \{A_1 \sqsubseteq A_2\} \models \varphi$.

In the following discussion, we use the term *literal* to denote an atomic concept or its negation. The symbol L represents an arbitrary literal, and $\overline{L}$ it's complement. In other words, $\overline{L}$ is $\neg A$ if $L = A$, and A if $L = \neg A$.

Our algorithm for TBox abduction works as follows:

Firstly, it tests whether the observation is entailed by the knowledge base using the algorithm described in Section 2, but does not stop when an open branch is attained. It stores the current set of role and literal assertions, backtracks to the last splitting point and continues the search for the next open branch. Secondly, for each such set, it generates a set of axioms that (if added to the original knowledge base) would close the branch. Thirdly, using Reiter's minimal hitting set algorithm [10], it generates solutions from the axiom sets. (A solution contains one axiom to close each open branch.)

Finally, as a post-processing step, the algorithm tests all solutions for consistency, relevance and semantic minimality.

We now give a more formal specification of the algorithm. The algorithm only produces solutions in a restricted language which we call $\mathcal{L}_{min}$. This language only allows atomic negation and limited existential and universal restriction (and no conjunctions or disjunctions). Furthermore, axioms of $\mathcal{L}_{min}$ may only be of the form $C \sqsubseteq D$ where C and D are concept descriptions of the following forms:

$$\begin{aligned} C &::= L \mid \exists R.\top \\ D &::= L \mid \forall R.\bot \\ L &::= A \mid \neg A \end{aligned}$$

The expressiveness of this target language is fairly modest since our goal was to provide a complete algorithm based on a standard DL tableau to generate all semantically minimal solutions. For example, if we were to allow concepts within the scope of quantifiers, we would either lose completeness or be forced to make the algorithm considerably more sophisticated.

Algorithm 1. (Restricted TBox abduction)

Given a knowledge base $\mathcal{K}$ and an axiom φ in $\mathcal{ALC}$, do the following:

1. Execute the standard tableau algorithm for testing whether $\mathcal{K} \models \varphi$. Whenever an open branch is attained (labelled by the set Ψ_i), store the set Ψ_i' consisting of the role and literal assertions from Ψ_i. Continue until the entire search tree has been traversed.
2. For each Ψ_i', generate a set of axioms Γ_i from which solutions will be built:
   (a) For each pair of concept assertions $L_1(a)$ and $L_2(a)$ in Ψ_i', add $L_1 \sqsubseteq \overline{L}_2$ to Γ_i.

   (b) For each combination of assertions $R(a, b)$ and $L(a)$ in Ψ'_i, add $\exists R.\top \sqsubseteq \overline{L}$ and $L \sqsubseteq \forall R.\bot$ to Γ_i.
   (c) For each assertion $R(a, b)$ in Ψ'_i, add $\exists R.\top \sqsubseteq \forall R.\bot$ to Γ_i.
3. Generate all solutions Θ obtainable by picking an axiom γ_i from each Γ_i such that $\Theta = \{\gamma_1, ..., \gamma_{m'}\}$ is syntactically minimal, i.e. there is no proper subset of Θ that would also contain a representative of each Γ_i. $\triangleleft$

Note that step 2(a) generates axioms of the form $L \sqsubseteq \overline{L}$, and steps 2(a) and (b) generate equivalent (contrapositive) pairs of axioms of the form $L_1 \sqsubseteq \overline{L}_2$ and $L_2 \sqsubseteq \overline{L}_1$, and $\exists R.\top \sqsubseteq \overline{L}$ and $L \sqsubseteq \forall R.\bot$. All the examples given below ignore axioms of the form $L \sqsubseteq \overline{L}$ and one of each pair of the abovementioned equivalent assertions for the sake of simplicity. This is also discussed in Section 6 as an optimisation step.

For the post-processing step, the following is done:
(i) *Consistency:* Use the standard tableau algorithm to test each solution Θ for consistency with $\mathcal{K}$. If $\mathcal{K} \cup \Theta$ is inconsistent, discard Θ.
(ii) *Relevance:* Use the standard tableau algorithm to test whether each solution Θ entails the observation φ. If $\Theta \models \varphi$, discard Θ.
(iii) *Semantic minimality:* Use the standard tableau algorithm to compare pairs of solutions Θ and Θ' for entailment. If $\Theta \models \Theta'$, discard Θ.

We now present a few examples that illustrate the solutions that these steps generate:

Example 1: Let $\mathcal{K} = \{A_1 \sqsubseteq A_2\}$ and $\varphi = A_1 \sqsubseteq A_3$. An obvious solution to $\langle \mathcal{K}, \varphi \rangle$ is $A_2 \sqsubseteq A_3$. The algorithm determines this as follows:

– φ is internalised and negated as the assertion $A_1 \sqcap \neg A_3(c)$. The single axiom in $\mathcal{K}$ is internalised and applied to c to form the assertion $\neg A_1 \sqcup A_2(c)$. The initial set of assertions is therefore $\{A_1 \sqcap \neg A_3(c), \neg A_1 \sqcup A_2(c)\}$.
– The expansion rules of the tableau algorithm are applied and we end up with one closed and one open branch with the corresponding sets of literal assertions: $\{A_1(c), \neg A_3(c), \neg A_1(c)\}$ and $\{A_1(c), \neg A_3(c), A_2(c)\}$.
– From the set for the open branch, step 2(a) generates the following set of axioms from which solutions will be built: $\{A_1 \sqsubseteq A_3, A_1 \sqsubseteq \neg A_2, A_2 \sqsubseteq A_3\}$. Steps 2(b) and (c) are not applied since there are no role assertions involved.
– Step 3 generates the following solutions: $\{A_1 \sqsubseteq A_3\}, \{A_1 \sqsubseteq \neg A_2\}$ and $\{A_2 \sqsubseteq A_3\}$.
– The post-processing step rejects solutions that are not relevant (i.e. that entail the observation), namely $\{A_1 \sqsubseteq A_3\}$.

This leaves two solutions, namely $\{A_1 \sqsubseteq \neg A_2\}$ and $\{A_2 \sqsubseteq A_3\}$. Note that although $\Theta = \{A_1 \sqsubseteq \neg A_2\}$ makes A_1 unsatisfiable w.r.t. $\mathcal{K} \cup \Theta$, this does not violate the satisfiability requirement for the concepts in the observation in Definition 1. $\diamondsuit$

Example 2: Let $\mathcal{K} = \{A_1 \sqsubseteq \exists R.A_2\}$ and $\varphi = A_1 \sqsubseteq \exists R.A_3$. An obvious solution to $\langle \mathcal{K}, \varphi \rangle$ is $A_2 \sqsubseteq A_3$. The algorithm determines this as follows:

- The algorithm firstly internalises and negates φ and internalises the single axiom in $\mathcal{K}$ to form the initial set of assertions $\{A_1 \sqcap \forall R.\neg A_3(c), \neg A_1 \sqcup \exists R.A_2(c)\}$.
- This tableau requires blocking to avoid repeated application of the $\exists$−rule, and occurs when the $\exists$−rule is applied for the second time.
- The tableau has two open branches, with the following sets of role and literal assertions, respectively: $\{A_1(c), R(c, d_1), A_2(d_1), \neg A_3(d_1), \neg A_1(d_1)\}$ and $\{A_1(c), R(c, d_1), A_2(d_1), \neg A_3(d_1), R(d_1, d_2), A_2(d_2)\}$.
- Step 2(a) generates $A_2 \sqsubseteq A_3$, $A_2 \sqsubseteq A_1$ and $\neg A_1 \sqsubseteq A_3$, step 2(b) generates $\exists R.\top \sqsubseteq \neg A_1$ and step 2(c) generates $\exists R.\top \sqsubseteq \forall R.\bot$ to form the first set of axioms from which solutions will be built. Similarly, the second set is $\{A_2 \sqsubseteq A_3, \exists R.\top \sqsubseteq \neg A_1, \exists R.\top \sqsubseteq \neg A_2, \exists R.\top \sqsubseteq A_3, \exists R.\top \sqsubseteq \forall R.\bot\}$.
- Step 3 generates the following solutions from these sets: $\{A_2 \sqsubseteq A_3\}$, $\{A_2 \sqsubseteq A_1, \exists R.\top \sqsubseteq \neg A_2\}$, $\{A_2 \sqsubseteq A_1, \exists R.\top \sqsubseteq A_3\}$, $\{\neg A_1 \sqsubseteq A_3, \exists R.\top \sqsubseteq \neg A_2\}$, $\{\neg A_1 \sqsubseteq A_3, \exists R.\top \sqsubseteq A_3\}$, $\{\exists R.\top \sqsubseteq \neg A_1\}$ and $\{\exists R.\top \sqsubseteq \forall R.\bot\}$.

Note once again that $\{\exists R.\top \sqsubseteq \neg A_1\}$ makes A_1 unsatisfiable w.r.t. $\mathcal{K} \cup \Theta$. Furthermore, $\{\exists R.\top \sqsubseteq \forall R.\bot\}$ requires role R to be empty. In other words, no individuals may be related by R. This also makes A_1 unsatisfiable w.r.t. $\mathcal{K} \cup \Theta$. $\Diamond$

Of course, some abductive problems only have solutions that are not consistent or not relevant. An empty knowledge base, for example, cannot have any relevant solutions to any abductive problem.

However, as we show in Theorem 2, if a problem in $\mathcal{ALC}$ has a solution in $\mathcal{L}_{min}$, Algorithm 1 will find it.

5 Enhancements to the algorithm

As it turns out, Algorithm 1 can be used to deal with a broader class of abductive problems than those permitted by Definition 1. Firstly, we can allow multiple TBox axioms in the observation. Secondly, we can enhance the algorithm to deal with concept assertions in the observation.

5.1 Observations involving multiple axioms

We can generalise the TBox abduction problem by allowing more than one axiom in the observation:

Definition 2. *Let $\mathcal{L}$ and $\mathcal{L}'$ be DLs, $\mathcal{K}$ a knowledge base in $\mathcal{L}$ and Φ a set of TBox axioms in $\mathcal{L}$ such that no axiom of Φ is entailed by $\mathcal{K}$ and $\mathcal{K} \cup \Phi$ is consistent. The pair $\langle \mathcal{K}, \Phi \rangle$ is called a generalised TBox abduction problem. A set of TBox axioms Θ in $\mathcal{L}'$ is called an abductive solution for $\langle \mathcal{K}, \Phi \rangle$ if $\mathcal{K} \cup \Theta \models \Phi$.*

Consistency and minimality are defined as for Definition 1, and relevance is defined as follows:

(ii) *Relevance*: No axiom of $\varPhi$ is entailed by $\varTheta$.

Note that since a set of axioms $\{C_1 \sqsubseteq D_1, ..., C_n \sqsubseteq D_n\}$ represents their implicit conjunction, the negation of such a set is equivalent to the assertion $(C_1 \sqcap \neg D_1) \sqcup ... \sqcup (C_n \sqcap \neg D_n)(c)$ for some individual name c not appearing in the knowledge base.

Example 3: Let $\mathcal{K} = \{A_1 \sqsubseteq A_2\}$ and $\varPhi = \{A_3 \sqsubseteq A_2, A_4 \sqsubseteq A_2\}$. An obvious solution to $\langle \mathcal{K}, \varPhi \rangle$ is $\{A_3 \sqsubseteq A_1, A_4 \sqsubseteq A_1\}$.

- The negation of $\varPhi$ and the internalised axiom in $\mathcal{K}$ will be used to form the initial assertion set $\{(A_3 \sqcap \neg A_2) \sqcup (A_4 \sqcap \neg A_2)(c), \neg A_1 \sqcup A_2(c)\}$.
- The tableau will have two open branches labelled with $\{A_3(c), \neg A_2(c), \neg A_1(c)\}$ and $\{A_4(c), \neg A_2(c), \neg A_1(c)\}$ respectively.
- From these sets, step 2(a) will generate two sets of axioms, namely $\{\neg A_1 \sqsubseteq A_2, A_3 \sqsubseteq A_1, A_3 \sqsubseteq A_2\}$ and $\{\neg A_1 \sqsubseteq A_2, A_4 \sqsubseteq A_1, A_4 \sqsubseteq A_2\}$.
- From these sets, step 3 will generate the solutions $\{\neg A_1 \sqsubseteq A_2\}, \{A_3 \sqsubseteq A_1, A_4 \sqsubseteq A_1\}, \{A_3 \sqsubseteq A_1, A_4 \sqsubseteq A_2\}, \{A_3 \sqsubseteq A_2, A_4 \sqsubseteq A_1\}$ and $\{A_3 \sqsubseteq A_2, A_4 \sqsubseteq A_2\}$.
- $\{A_3 \sqsubseteq A_1, A_4 \sqsubseteq A_2\}, \{A_3 \sqsubseteq A_2, A_4 \sqsubseteq A_1\}$ and $\{A_3 \sqsubseteq A_2, A_4 \sqsubseteq A_2\}$ are not relevant because they contain (and therefore entail) axioms in the observation.

This leaves two solutions, namely $\{\neg A_1 \sqsubseteq A_2\}$ and $\{A_3 \sqsubseteq A_1, A_4 \sqsubseteq A_1\}$. Note that, together with the original knowledge base, the first solution makes A_2 equivalent to $\top$. $\diamondsuit$

5.2 Observations involving concept assertions

We can generalise Definition 2 further so that the observation can also contain concept assertions. This is illustrated in the following examples:

Example 4: Let $\mathcal{K} = \{A_1(a), A_1 \sqsubseteq A_2\}$ and $\varPhi = \{A_3(a)\}$. Two obvious solutions to $\langle \mathcal{K}, \varPhi \rangle$ are $\{A_1 \sqsubseteq A_3\}$ and $\{A_2 \sqsubseteq A_3\}$.

The algorithm will negate $\varPhi$ and internalise the single axiom in $\mathcal{K}$ to form the initial set of assertions $\{\neg A_3(a), A_1(a), \neg A_1 \sqcup A_2(a)\}$. The $\sqcup$-rule will be applied, creating one closed branch and one open branch with the labelling $\{\neg A_3(a), A_1(a), A_2(a)\}$. Step 2(a) will generate the set $\{A_1 \sqsubseteq A_3, A_2 \sqsubseteq A_3, A_1 \sqsubseteq \neg A_2\}$, and step 3 will generate solutions comprising one of each of the axioms in this set. The post-processing step will reject the solution $\{A_1 \sqsubseteq \neg A_2\}$ as it contradicts the knowledge base. $\diamondsuit$

Example 5: Let $\mathcal{K} = \{R(a, b)\}$ and $\varPhi = \{A(a)\}$. An obvious solution to $\langle \mathcal{K}, \varPhi \rangle$ is $\{\exists R.\top \sqsubseteq A\}$.

The algorithm will negate the observation and start with the set of assertions $\{\neg A(a), R(a, b)\}$. This set is already saturated, so steps 2(b) and (c) will generate the axioms $\exists R.\top \sqsubseteq A$ and $\exists R.\top \sqsubseteq \forall R.\bot$. Step 3 will therefore generate the solutions $\{\exists R.\top \sqsubseteq A\}$ and $\{\exists R.\top \sqsubseteq \forall R.\bot\}$. The latter solution will be rejected because it contradicts the knowledge base, leaving the former which is the one we expected. $\diamondsuit$

Now suppose the observation Φ consists of a set of concept assertions that involve different individual names, e.g. $\{C_1(a_1), ..., C_n(a_m)\}$. DL syntax does not allow us to express the negation of a set of assertions involving different individuals as a single assertion. We could start a separate tableau for each negated assertion and collect the axioms needed to close all open branches of all trees. Alternatively, we could store such a negated set as a special set of negated assertions $\{\neg C_1(a_1), ..., \neg C_n(a_m)\}$, where there are implicit disjunctions between the assertions. When the algorithm reaches a point where no other expansion rules can be applied, it picks one (negated assertion) and creates a branch with it. When the algorithm backtracks to this point, the next one is chosen. This works just like an application of the $\sqcup$-rule.

6 Analysis

In this section, we present theoretical results stating the soundness and completeness of Algorithm 1, analyse its complexity and discuss various optimisations.

Theorem 1. (Soundness) *All solutions generated by Algorithm 1 are TBox abduction solutions.*

Proof sketch: By the way solutions are constructed by steps 2(a), (b) and (c) of Algorithm 1, we show that any such solution Θ for an abduction problem $\langle \mathcal{K}, \varphi \rangle$ will close all the open branches of the tableau. In other words, the tableau for testing $\mathcal{K} \cup \Theta \models \varphi$ will close. $\square$

Theorem 2. (Completeness) *Algorithm 1 finds all semantically minimal solutions in $\mathcal{L}_{min}$ to a TBox abduction problem in $\mathcal{ALC}$ up to logical equivalence.*

Proof sketch: For any semantically minimal solution Θ to an abduction problem $\langle \mathcal{K}, \varphi \rangle$, we pick a minimal subset of Θ needed to close each open branch of the tableau for testing $\mathcal{K} \models \varphi$. By means of a lemma, we prove that each such set is equivalent to a singleton set. We then show that the algorithm will generate a solution equivalent to the union of these sets, which is also equivalent to Θ. $\square$

Remark: Theorem 1 does not state that Algorithm 1 only finds semantically minimal solutions. In fact, it generates some non-semantically minimal solutions, which explains the need for the post-processing step to discard them.

The complexity of the standard tableau algorithm for consistency checking with general TBoxes in $\mathcal{ALC}$ is NExpTime [3]. But this is because the algorithm only needs to find one open branch by making a series of non-deterministic choices at the splitting points, and the length of each branch is in $O(2^n)$ in the worst case (where n is the size of the input). Algorithm 1, however, generates all open branches, and this takes double exponential time in the worst case, i.e. $O(2^{2^n})$ different branches. This does not even take into account the time needed to work out the axioms used to build solutions.

Nevertheless, we are able to tighten the upper bound to ExpSpace, known to be subsumed by 2ExpTime, by augmenting the algorithm. First, we note that the size of the vocabulary used in the input is not larger than n. The solution language $\mathcal{L}_{min}$ permits at most polynomially many axioms in n over this vocabulary. Then we can do the following:

- Whenever a new open branch is attained, we can generate and store all the axioms in $\mathcal{L}_{min}$ that can close it rather than the set of role and literal assertions. Then we can dump the branch.
- Since there are at most $O(2^n)$ sets of axioms that can be generated from a polynomially sized set, we only need exponential space to store them.
- Although in general Reiter's minimal hitting set algorithm requires space exponential in the number of sets [11], once again, the restricted language limits the number of solutions to at most $O(2^n)$ (the power set of the set of all axioms).

Hence, altogether we consume exponential space.

Further, the post-processing phase is reducible to a finite sequence of standard entailment problems, of ExpTime-complete complexity in $\mathcal{ALC}$ or possibly lower in the restricted language [1].

There are some obvious optimisations that would improve the efficiency of our algorithm. Firstly, one could store only the sets of axioms for each open branch rather than the sets of role and literal assertions (as stated above). Secondly, one could discard equivalent, irrelevant and inconsistent axioms as they are generated (in steps 2(a) and (b)). This would save them from being processed by step 3.

Finally, a distributed version of Reiter's hitting set algorithm could be performed, where candidate solutions are built as the sets of axioms are populated. This could involve elimination of duplicate axioms, and whole sets of axioms that are supersets of others.

7 Conclusion and future work

In this paper, we have described how to implement a restricted form of TBox abduction in $\mathcal{ALC}$ using a modified DL tableau.

It would be possible to generalise the algorithm to produce solutions in a less restricted language, but not one that could generate all possible solutions in $\mathcal{ALC}$. Future work could look at getting the algorithm to generate more solutions that are also meaningful.

Our algorithm does not implement many of the optimisations (e.g. back-jumping and caching) commonly used in DL tableau algorithms. Incorporating these into our algorithm, as well as those mentioned at the end of Section 6, would improve it.

This work also promises to be transferable to other more expressive DLs.

8 Acknowledgements

This work is based on research supported in part by the National Research Foundation of South Africa (Grant No. 85482).

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.: *The Description Logic Handbook*, Cambridge University Press (2003)
2. Baader, F., Horrocks, I., Sattler, U.: Chapter 3: Description Logics. In: van Harmelen, F., Lifschitz, V., Porter, B., editors: *Handbook of Knowledge Representation*, Elsevier (2007)
3. Baader, F., Sattler, U.: Overview of Tableau Algorithms for Description Logics, in *Studia Logica*, vol 69 (2000)
4. Di Noia, T., Di Sciascio, E., Donini, F.M.: Computing information minimal match explanations for logic-based matchmaking, in *Proc. of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, vol 02, IEEE Computer Society (2009)
5. Du, J. Qi, G., Shen, Y-D., Pan, J.Z.: Towards practical ABox abduction in large OWL DL ontologies, in *Proc. of the 25th AAAI Conference* (2011)
6. Elsenbroich, C., Kutz, O., Sattler, U.: A case for abductive reasoning over ontologies, in *Proc. of the OWLED'06 Workshop*, vol 216 (2006)
7. Halland, K., Britz, K.: ABox abduction in $\mathcal{ALC}$ using a DL tableau, in *Proc. of SAICSIT 2012*, (2012)
8. Klarman, S., Endriss, U., Schlobach, S.: ABox abduction in the description logic $\mathcal{ALC}$, *Journal of Automated Reasoning*, vol 46:1 (2011)
9. Lambrix, P., Wei-Kleiner, F., Dragisic, Z. Ivanova, V.: Repairing missing is-a structure in ontologies is an abductive reasoning problem, in *Proc. of WoDOOM13*, (2013)
10. Reiter, R.: A theory of diagnosis from first principles, *Artificial Intelligence*, vol 32 (1987)
11. Wotawa, F.: A variant of Reiter's hitting-set algorithm, *Information Processing Letters*, vol 79 (2001)