Tableau-Based ABox Abduction for Description Logics: Preliminary Report

Júlia Pukancová and Martin Homola

Comenius University in Bratislava,
Mlynská dolina, 84248 Bratislava, Slovakia
{pukancova,homola}@fmph.uniba.sk

Abstract. ABox abduction is an important reasoning problem for description logics (DL) with applications in diagnosis, manufacturing control, multimedia interpretation, etc. Several abductive reasoners for DL were designed and implemented using a translation to first-order logic or logic programming. Algorithms for ABox abduction based directly on native DL reasoning techniques, such as tableau algorithms, were also proposed. Such approaches may potentially benefit from various optimization techniques implemented in tableau-based DL reasoners. In this paper we present our preliminary results with an implementation of an ABox abduction algorithm that is based on the Pellet reasoner.

Keywords: Abduction, description logics, implementation.

1 Introduction

Abduction [10] is a form of backward reasoning which is often used to derive explanations of some observation. Given a knowledge base $\mathcal{K}$ that models a certain problem, we are confronted with an observation O which is supposed to follow in situations captured by $\mathcal{K}$, but we are not able to explain O deductively. In abductive reasoning we ask the question – why is it that O does not follow from $\mathcal{K}$ – and we look for a hypothesis (or, explanation) H such that O follows from $\mathcal{K} \cup H$.

Considering abduction in the DL area [3], we need to further differentiate between TBox abduction, where we are interested in explaining why some intentional axiom (e.g., subsumption) does not follow from the given knowledge base which can be useful for instance in ontology engineering. On the other hand, in ABox abduction we have an observation in form of some data, that is, some recorded facts that are observed. We also typically look for an extensional explanation: some data that could explain the observation. Applications of ABox abduction can be found in diagnostics, such as diagnosing the condition of the patient from the observed symptoms [12] or diagnosing the behaviour of a manufacturing system [6]. But also in other areas, such as multimedia interpretation [11].

Several works were devoted to development of reasoning algorithms for ABox abduction. Klarman et al. [9] and Du et al. [2] designed and implemented such solvers using a translation to first-order logic or logic programming, respectively. Halland and Britz [5,4] proposed an ABox abduction algorithm based on a tableau reasoner. They conjectured that such an approach may possibly take advantage of optimization techniques already implemented in DL reasoners, however we are not aware of an implementation and experimental evaluation of their work.

In this paper, we describe an implementation of an ABox abduction reasoner similar to that proposed by Halland and Britz. The reasoner is based on Reiter's minimal hitting set algorithm [13] which is combined with a tableau reasoner, Pellet [14] in our case. Our implementation is based on a more tight integration of the minimal hitting set algorithm and the tableau reasoner as it was proposed by Reiter. So far it has a number of limitations which we plan to address in future research. Consecutively we plan to conduct an experimental evaluation and compare the performance with other approaches.

2 ABox Abduction in DL

In DL, the knowledge base $\mathcal{K}$ is typically split into the TBox $\mathcal{T}$, containing intensional knowledge in form of subsumption axioms, and the ABox $\mathcal{A}$ containing extensional knowledge in form of assertions. Subsumption axioms are of the form $C \sqsubseteq D$, meaning that C is more specific than D; concept assertions are of the form $a\colon C$, meaning that the individual a belongs to the concept C; while role assertions are of the form $a, b\colon R$, meaning that the individuals a and b are connected by the role R. Depending on the particular DL, concepts (and sometimes roles) may be atomic or constructed using different constructors (e.g., $\neg$, $\sqcap$, $\sqcup$, $\exists$, and $\forall$, for $\mathcal{ALC}$). Models of $\mathcal{K}$ are typically understood as interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ with a nonempty domain $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$ that satisfy all axioms in $\mathcal{K}$, however in this work we will homomorphically record them as enumerations of ABox assertions M such that for each individual a and atomic concept A either $\phi = a\colon A \in M$ or $\neg\phi = a\colon \neg A \in M$, and similarly for each pair of individuals a, b and each role R. A subsumption axiom or an ABox assertion ϕ is said to follow from $\mathcal{K}$ if it is satisfied in all models of $\mathcal{K}$, which is then denoted by $\mathcal{K} \models \phi$. For further details, refer for instance to the DL Handbook [1].

ABox abduction covers the case in which the observation is of the form of actual facts, that is ABox assertions. Formal definition was proposed by Elsenbroich et al. [3].

Definition 1 (ABox Abduction Problem [3]). *An ABox abduction problem is a pair $\mathcal{P} = (\mathcal{K}, O)$ such that $\mathcal{K}$ is a knowledge base in DL and O is an ABox assertion. A solution of $\mathcal{P}$ is any finite set H of ABox assertions such that and $\mathcal{K} \cup H \models O$.*

Abduction is a form of hypothetical reasoning. The solutions do not follow from the knowledge base in the deductive sense of the word, but they are rather hypothetical guesses of certain new knowledge (facts in the case of ABox abduction) that, if added to $\mathcal{K}$, allow to explain the observation O deductively. Therefore they are also called hypotheses.

Definition 1 is very general, it allows also certain trivial types of solutions. Further constraints are typically required to hold. The most basic constraints that are typically always required are the following three.

Definition 2 ([3]). *Given an ABox abduction problem $\mathcal{P} = (\mathcal{K}, O)$ and its solution H we say that:*

1. *H is* consistent *if $H \cup \mathcal{K} \not\models \bot$, i.e. H is consistent w.r.t. $\mathcal{K}$;*
2. *H is* relevant *if $H \not\models O$, i.e. H does not entail O;*
3. *H is* explanatory *if $\mathcal{K} \not\models O$, i.e. $\mathcal{K}$ does not entail O.*

Consistency is required, because if the knowledge base together with the hypothesis are inconsistent ($\mathcal{K} \cup H \models \bot$), anything follows. Such hypotheses would explain every observation and so they are not meaningful. Relevance is required, because the knowledge base $\mathcal{K}$ represents some background theory from which we are interested to derive the hypotheses. Therefore we should not be able to explain the observation without it. Finally, an abduction problem only needs explaining if the observation does not already follow from $\mathcal{K}$.

Even if all three of these constraints are applied the number of possible explanations is still very high. In fact it is even infinite, as deductive entailment in DL is monotonic: given an abduction problem $\mathcal{P} = (\mathcal{K}, O)$ and its solution H, we always have that also H' is a solution of $\mathcal{P}$ for any H' such that $H \subseteq H'$. Therefore further restrictions are required. These can be based on syntactic or semantic measures.

Definition 3 (Syntactic Minimality). *Assume an ABox abduction problem $\mathcal{P} = (\mathcal{K}, O)$. Given two solutions H and H' of $\mathcal{P}$, we say that H is (syntactically)* smaller *than H' if $H \subseteq H'$.*[1] *We further say that a solution H of $\mathcal{P}$ is* syntactically minimal *if there is no other solution H' of $\mathcal{P}$ that is smaller than H.*

Example 1. Consider the knowledge base $\mathcal{K}$ with two axioms:

$$\mathsf{Professor} \sqcup \mathsf{Scientist} \sqsubseteq \mathsf{Academician} \tag{1}$$

$$\mathsf{AssocProfessor} \sqsubseteq \mathsf{Professor} \tag{2}$$

Given the observation $O = \{\mathsf{jack}\colon \mathsf{Academician}\}$ we are able to find a number of abductive explanations of $\mathcal{P} = (\mathcal{K}, O)$, e.g., $H_1 = \{\mathsf{jack}\colon \mathsf{Professor}\}$, $H_2 = \{\mathsf{jack}\colon \mathsf{Scientist}\}$, $H_3 = \{\mathsf{jack}\colon \mathsf{Professor}, \mathsf{jack}\colon \mathsf{Scientist}\}$, $H_4 = \{\mathsf{jack}\colon \mathsf{AssocProfessor}\}$. We may observe that H_1 and H_2 are both smaller than H_3. Therefore H_3 is not a syntactically minimal explanation, while the other three are.

Indeed if H is a syntactically smaller solution of $\mathcal{P}$ than H', then the assertions of $H' \setminus H$ are useless; they are not required in order to explain $\mathcal{P}$. As abduction is a form of hypothetical reasoning, H' amounts to additional and unnecessary guessing, which is not desired. Even if two solutions are incomparable with respect to Definition 3 smaller solutions (with respect to set size) are often preferred. While syntactic measures allow to filter out a great number of undesired solutions, semantic measures are usually more preferred.

Definition 4 (Semantic Minimality). *Assume an ABox abduction problem $\mathcal{P} = (\mathcal{K}, O)$. Given two solutions H and H' of $\mathcal{P}$, we say that H is (semantically)* stronger *than H' (denoted by $H \preceq_{\mathcal{K}} H'$) if $\mathcal{K} \cup H \models H'$. A solution H of $\mathcal{P}$ is* semantically minimal *if for every H', $H' \preceq_{\mathcal{K}} H$.*

[1] Note that before we compare two solutions H and H' of $\mathcal{P}$ syntactically, we typically normalize the assertions w.r.t. (outermost) concept conjunction: as $a\colon C \sqcap D$ is equivalent to the pair of assertions $a\colon C$ and $a\colon D$, we replace the former form by the latter while possible.

Also, if H is a semantically stronger solution of $\mathcal{P}$ than H', then vice-versa H' is a (semantically) *weaker* solution of $\mathcal{P}$ than H.

Example 2. Reinspecting the explanations from Example 1 we may now observe that both H_1 and H_2 are weaker than H_3. However, the explanation $H_5 = \{\mathsf{jack}\colon \mathsf{Professor} \sqcup \mathsf{Scientist}\}$ is weaker then either of H_1 and H_2, and is semantically minimal.

In general, almost always only consistent, relevant, and explanatory solutions are considered. Syntactically minimal solutions are also typically preferred. However, semantic measures are considered the most important because they allow to compare a greater number of hypotheses. As we saw in Examples 1, 2 we were not able to compare H_1 and H_2 with H_3 purely syntactically.

While semantically weaker solutions are most typically preferred as this reduces the amount of guessing, this may also depend of the particular application. For example Petatis et al. [11] show an application of abduction in muti-media interpretation where semantically stronger solutions represent a higher level of abstraction and therefore they are more preferable.

3 ABox abduction algorithm

From Definition 1 we know that a set of assertions H is an explanation of $\mathcal{P} = (\mathcal{K}, O)$ if $\mathcal{K} \cup H \models O$. As entailment is reducible to consistency checking, we have that H is an explanation of $\mathcal{P}$ if $\mathcal{K} \cup H \cup \{\neg O\}$ is inconsistent. According to Reiter [13] one can find such explanations by considering all models of $\mathcal{K} \cup \{\neg O\}$ and constructing the set H by selecting and negating one assertion from each of these models. It follows that $\mathcal{K} \cup H \cup \{\neg O\}$ will have no models. For this reason Reiter uses the notion of a hitting set [7]:

Definition 5 (Hitting Set [13]). *Given a set of sets F, a hitting set H of F is any set such that $H \cap S \neq \{\}$ for every $S \in F$.*

A hitting set H of F is called *minimal* if there is no other hitting set H' of F such that $H' \subseteq H$. An interesting property of minimal hitting sets is that, given F the set of all negated models of $\mathcal{K} \cup \{\neg O\}$, called *conflict sets*,[2] the set of all minimal hitting sets of F corresponds to the set of all syntactically minimal explanations of $\mathcal{P}$. Therefore the task of finding these explanations reduces to the task of finding the minimal hitting sets of F.

To find all minimal hitting sets, Reiter constructs a structure called HS-tree.

Definition 6 (HS-tree [13]). *Suppose F is a collection of sets. An edge-labelled and node-labelled tree T is an HS-tree for F iff it is a smallest tree with the following properties:*

1. *Its root is labelled by ✓ if F is empty. Otherwise, its root is labelled by a set of F.*

[2] More precisely, since we are only interested in explanations composed of atomic (and negated atomic) concept assertions, that is, given a model M the respective conflict set is $\{a\colon \neg C \mid a\colon C \in M$ where C is either atomic or negated atomic concept$\}$.

2. *If n is a node of T, define $H(n)$ to be the set of edge labels on the path in T from the root node to n. If n is labelled by $\checkmark$, it has no successor nodes in T. If n is labelled by a set Σ of F, then for each $\sigma \in \Sigma$, n has a successor node n_σ joined to n by an edge labelled by σ. The label for n_σ is a set $S \in F$ such that $S \cap H(n_\sigma) = \{\}$ if such a set S exists. Otherwise, n_σ is labelled by $\checkmark$.*

HS-tree has the property, that the sets $H(n)$ where n is a leaf-node are all hitting sets for F, and the minimal hitting sets are all included. Since there are possibly hitting sets which are not minimal, the HS-tree is constructed breadth-first and some pruning is applied for the sake of optimization.

A branch of the tree can be pruned in the node n (such nodes are labelled by $\times$ instead of $\checkmark$) without losing any minimal hitting set: if there is n' such that $H(n') \subseteq H(n)$ and n' is labelled by $\checkmark$; or if there is n' such that $H(n') = H(n)$ and n' is labelled by some nonempty conflict set $S \in F$.

If $S \in F$ and $S' \in F$ with S a proper subset of S', then $F \setminus \{S'\}$ has the same minimal hitting sets as F. Therefore if there are nodes n and n' respectively labelled by S and S' of F such that $S' \subseteq S$, then we can prune each edge from node n labelled by $\alpha \in S \setminus S'$ including its subtree.

Now the pruned HS-tree has the following property.

Theorem 1 (Reiter [13]). *Let F be a collection of sets, and T a pruned HS-tree for F, as previously described. Then $\{H(n) \mid n$ is a node of T labelled by $\checkmark\}$ is the collection of minimal hitting sets for F.*

One possible approach how to use the minimal hitting set algorithm (MHS) is to start by generating all models of $\mathcal{K} \cup \{\neg O\}$ using the tableau algorithm for DL [1] (TA), create the collection of respective conflict sets F, and consecutively to construct the HS-tree.[3]

Once the HS-tree is constructed we are able to extract the explanations as the minimal hitting sets. However, since we have further requirements on the explanations as given in Definition 2 they still have to be checked whether they satisfy consistency and relevance. This can be done by calling TA and verifying the consistence of $\mathcal{K} \cup H$ and $H \cup \{\neg O\}$ respectively for each extracted hitting set H. This approach, which was also used by Halland and Britz [4], is illustrated in the following example.

Example 3. Find the explanation for the knowledge base $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$ and observation $O = a\colon C$. TBox $\mathcal{T}$ and ABox $\mathcal{A}$ are: $\mathcal{T} = \{E \sqsubseteq C, F \sqsubseteq D\}$, $\mathcal{A} = \{\}$.

1. Models for $\mathcal{K} \cup \{\neg O\}$ are in collection $MS = \{M_1, M_2, M_3\}$.
   $M_1 = \{a\colon \neg C, a\colon \neg D, a\colon \neg E, a\colon \neg F\}$
   $M_2 = \{a\colon \neg C, a\colon D, a\colon \neg E, a\colon F\}$
   $M_3 = \{a\colon \neg C, a\colon D, a\colon \neg E, a\colon \neg F\}$

[3] Note that this approach requires to enumerate the models (or, the conflict sets) which is only straight forwardly possible for logics with the finite model property [1]. Further extension would be required to handle also infinite models.

2. Collection of conflict sets for MS is $F = \{S_1, S_2, S_3\}$.
   $S_1 = \{a\colon C, a\colon D, a\colon E, a\colon F\}$
   $S_2 = \{a\colon C, a\colon \neg D, a\colon E, a\colon \neg F\}$
   $S_3 = \{a\colon C, a\colon \neg D, a\colon E, a\colon F\}$
3. Create HS-tree T according to Definition 6 (respective tree is shown in the Figure 1).
4. Extract the minimal hitting sets according to Theorem 1:
   $HS = \{\{a\colon C\}, \{a\colon D, a\colon \neg D\}, \{a\colon D, a\colon \neg F\}, \{a\colon E\}, \{a\colon F, a\colon \neg D\}, \{a\colon F, a\colon \neg F\}\}$
5. Filter from HS all inconsistent and irrelevant explanations according to Definition 2: $HS = \{\{a\colon E\}\}$.
6. The explanation for the observation O and the knowledge base $\mathcal{K}$ is $\{a\colon E\}$.

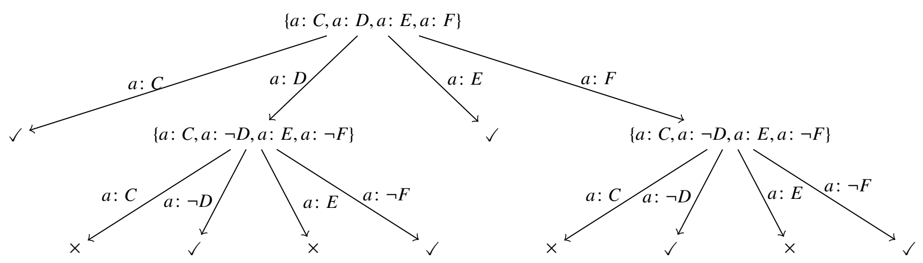

Fig. 1. HS-tree for the set F of Example 3

The computation of all models of a knowledge base is a costly pre-processing step. Instead, the set F can be computed step by step sequentially during the construction of HS-tree. We will adopt this approach in our implementation. Further details are described in the following section.

3.1 Combining the Tableau Algorithm and the Minimal Hitting Set Algorithm

We will now describe the version of the abductive algorithm actually implemented in our work. As proposed by Reiter [13], to avoid computing the whole collection of conflict sets F we start by computing the first conflict set for the root node of the HS-tree as the negation of the first model obtained by calling TA on $\mathcal{K} \cup \{\neg O\}$. For any of the following nodes n we need to obtain a conflict set S such that $S \cap H(n) = \{\}$. This can be done by calling TA on $\mathcal{K} \cup \{\neg O\} \cup H(n)$.

Thus we construct the HS-tree even without pre-computing all elements of F beforehand and extract the explanations from the HS-tree as all minimal hitting sets.

The algorithm takes a DL ontology $\mathcal{K}$ as the input knowledge base and a single (even complex) concept assertion O as input observation. It starts by checking the consistency of $\mathcal{K} \cup \{\neg O\}$ by a call to TA. If $\mathcal{K} \cup \{\neg O\}$ is inconsistent then there is nothing to explain and the algorithm terminates without returning any explanations. In the other case it initializes the HS-tree with root n which is labelled by a conflict set S obtained by negating the model of $\mathcal{K} \cup \{\neg O\}$ returned by TA. The algorithm then creates a successor node n_s of n for each $s \in S$.

Consequently the main cycle is executed for each node n without label; the nodes are loaded w.r.t. breadth-first search:

1. create $\mathcal{K}'$ as the union of $\mathcal{K} \cup \{\neg O\}$ and the set of ABox assertions $H(n)$,
2. call TA for $\mathcal{K}'$ and get a new model M
3. create new conflict set S as the negation of M,
4. label n with S,
5. for every $s \in S$ create the n-successor n_s and label the edge with s.

If $\mathcal{K}'$ is inconsistent, i.e. it has no model, we will set $M = \{\}$ in step 2. Consequently the respective conflict set will be $S = \{\}$ and that implies that step 5 produces no successors. The label $\{\}$ is analogous to Reiter's label $\checkmark$, but we are not introducing a new symbol – the meaning of the empty set $\{\}$ is exactly the same as of Reiter's $\checkmark$.

In addition, it is possible to construct the set of minimal hitting sets HS on the fly during the run of our algorithm. The consecutively stored hitting sets will all be minimal thanks to breadth-first search. The following optimizations proposed by Reiter are also performed before calling TA for each node n in order to reduce the number of these calls:

- o_1. if there is an ABox assertion $a\colon C \in H(n)$ and also an ABox assertion $a\colon \neg C \in H(n)$ for some individual a and some concept C, then label node n by $\{\}$ and do not store $H(n)$ in HS, that is, close the path in n,
- o_2. otherwise if there is a hitting set $H \in HS$ such that $H \subseteq H(n)$, then label node n by $\{\}$ and do not store $H(n)$ in HS, that is, close the path in n,
- o_3. otherwise if there is a node n' such that $H(n') = H(n)$ and the label of n' is neither null nor $\{\}$, then label node n by $\{\}$ and do not store $H(n)$ in HS, that is, close the path in n,
- o_4. otherwise if there is a model M in the stored models MS $(M \in MS)$ and $H(n) \subseteq M$, then label the node n by negation of M, that is, reuse existing model.

If none of the previous conditions is satisfied, the algorithm calls TA. If the returned model M is $\{\}$ and if $H(n)$ is a relevant and consistent explanation according to Definition 2, then $H(n)$ is added to the set of stored hitting sets HS. If $M \neq \{\}$ then M is stored in MS. The hitting sets may also be output on the fly at the time when they are added into HS. It follows from the construction of the edge labels, that the output explanations will all be collections of atomic and negated atomic concept assertions. The run of the algorithm is demonstrated in Example 4.

Example 4. Let us find the explanations for the same knowledge base $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$ and observation $O = a\colon C$ as in Example 3, where $\mathcal{T} = \{E \sqsubseteq C, F \sqsubseteq D\}$, and $\mathcal{A} = \{\}$. Refer again to Figure 1 as the hitting tree constructed in this example will be the same.

1. Let TA compute a model for $\mathcal{K} \cup \neg O$: $M_1 = \{a\colon \neg C, a\colon \neg D, a\colon \neg E, a\colon \neg F\}$.
2. Create a conflict set for M_1: $S_1 = \{a\colon C, a\colon D, a\colon E, a\colon F\}$.
3. Create HS-tree T with root r labelled with S_1.

4. Create r-successors n_C, n_D, n_E, and n_F with edges labelled with $a\colon C$, $a\colon D$, $a\colon E$, and $a\colon F$.
5. For n_C, the next node following via breadth-first search, none of the pruning techniques can be applied nor any of the models can be reused. Thus compute a model for n_C, i.e. for $\mathcal{K} \cup \neg O \cup \{a\colon C\}$. The model is $\{\}$.
6. Label n_C by $\{\}$, i.e. create no successors.
7. Check $H(n_C)$ for relevance and consistency – it is not relevant, so we do not store it in HS.
8. For n_D, the next node following via breadth-first search, none of the pruning techniques can be applied nor any of the models can be reused. Thus compute a model for n_D, i.e. for $\mathcal{K} \cup \neg O \cup \{a\colon D\}$. The model is $M_2 = \{a\colon \neg C, a\colon D, a\colon \neg E, a\colon F\}$.
9. Create a conflict set for M_2: $S_2 = \{a\colon C, a\colon \neg D, a\colon E, a\colon \neg F\}$.
10. Create successors n_{DC}, $n_{D\neg D}$, n_{DE}, and $n_{D\neg F}$ of n_D with edges labelled with $a\colon C$, $a\colon \neg D$, $a\colon E$, and $a\colon \neg F$.
11. For n_E, the next node following via breadth-first search, none of the pruning techniques can be applied nor any of the models can be reused. Thus compute a model for n_E, i.e. for $\mathcal{K} \cup \neg O \cup \{a\colon E\}$. The model is $\{\}$.
12. Label n_E by $\{\}$, i.e. create no successors.
13. Check $H(n_E)$ for relevance and consistency – both are satisfied, so we store it in HS. We now have $HS = \{\{a\colon E\}\}$.
14. For n_F, the next node following via breadth-first search, none of the pruning techniques can be applied, but the model M_2 can be reused.
15. For n_F, create a conflict set, successors and labelling analogously as for n_D.
16. Continue with the following nodes w.r.t. breadth-first search, while in some nodes apply pruning techniques (e.g. $H(n_{D\neg D})$ contains a clash, $H(n_{DE}) \subseteq H(n_E)$, etc.). Note that for those nodes for which TA is called the model is always $\{\}$.
17. When the HS-tree is completed (in fact, we will end up with the same tree as shown in Figure 1), HS will contain only one hitting set. – $\{a\colon E\}$, i.e. the only explanation.

Observe that it was sufficient for our algorithm to compute models M_1 and M_2 to complete the whole process. We never computed M_3 from Example 3. Hence by tighter integration of MHS and the TA calls we save some of the work the approach of Halland and Britz [4] does already during pre-processing. Of course a thorough experimental evaluation would be needed to understand which of the approaches is more effective in practice.

The detailed pseudo code of our algorithm is included in the Algorithm 3.1. We will now comment on the pseudo code in details:

Line 1: Variable *inputKB* is used to store the initial ontology *Ont* and the negation of observation O.

Lines 2-3: $\mathcal{K}$ is the working instance of the knowledge base for which we will be checking the consistency and subsequently obtaining a model via TA.

Lines 8-9: HS is initialized as the empty set and the minimal hitting sets will be stored in it subsequently. MS is initialized by adding the first model that was computed and the other models will be stored in it subsequently.

Algorithm 1 ABox Abductive Algorithm

```
Require: Ontology $Ont$ and observation $O$
Ensure: $HS$: Set of the explanations
 1: $inputKB \leftarrow Ont \cup \neg O$
 2: $\mathcal{K} \leftarrow inputKB$
 3: $M \leftarrow$ call TA with input $\mathcal{K}$
 4: if $M = \{\}$ then
 5:     print "The observation follows from the ontology."
 6:     return $\{\}$
 7: end if
 8: $HS \leftarrow \{\}$
 9: $MS \leftarrow \{M\}$
10: create new hitting tree $T$ with root $r$
11: label $r$ in $T$ by conflict set from $M$
12: create successors of $r$ in $T$
13: $n \leftarrow$ successor of $r$ in $T$ w.r.t. breadth-first search
14: while $n \neq$ null do
15:     if (clash in $H(n)$)
              or ($S \in HS$ and $S \subseteq H(n)$)
              or ($n' \in T$ and $H(n) = H(n')$ and $L \neq$ null and $L \neq \{\}$ and $L$ is label of $n'$) then
16:         $M \leftarrow \{\}$
17:     else if $N \in MS$ and $H(n) \subseteq N$ then
18:         $M \leftarrow N$
19:     else
20:         $\mathcal{K} \leftarrow inputKB \cup H(n)$
21:         $M \leftarrow$ call TA with input $\mathcal{K}$
22:         if $M = \{\}$ then
23:             if $H(n)$ is relevant and consistent explanation then
24:                 $HS \leftarrow HS \cup H(n)$
25:             end if
26:         else
27:             $MS \leftarrow MS \cup M$
28:         end if
29:     end if
30:     label $n$ in $T$ by conflict set from $M$
31:     create successors of $n$ in $T$
32:     $n \leftarrow$ next node in $T$ w.r.t. $n$ and breadth-first search
33: end while
34: return $HS$
```

Line 12: The main cycle is running while there are still nodes from the HS-tree T that are not explored w.r.t. breadth-first search.

Lines 15-16: If one of the path-trimming condition o_{1-3} is satisfied, then M is set to empty set.

Lines 17-18: If it is possible to reuse some model N already stored in MS according to the optimization o_4, then set M to N.

Lines 20-21: If none of previous conditions was satisfied, it is necessary to call TA in order to check the consistency of the updated $\mathcal{K}$.

Lines 22-24: If TA returned an empty model, and if the path from root to node n is a relevant and consistent explanation, then algorithm found a minimal hitting set and stores it in HS.

Line 27: If TA found another nonempty model, then algorithm stores it in MS.

Line 30-32: The node n is labelled, its successors are created and n is set to next node from T.

4 Implementation

Our algorithm is implemented in pure Java. The run of the algorithm combines calls to the tableau algorithm and the implementation of the minimal hitting set algorithm.

Calling the tableau algorithm is executed by checking the consistency of the respective knowledge base through the Pellet reasoner [14]. Pellet is an open source OWL DL reasoner implemented in Java. We have used Pellet 2.3.1 which is the latest open-source version. It features full OWL 2 support which means that theoretically OWL 2 ontology may be used with our algorithm, however additional care needs to be taken for logics without the finite model property [1].

The input OWL ontology is loaded in Pellet from a file in RDF/XML syntax. Consequently our algorithm is able to add ABox assertions as needed to the knowledge base at any time, and it may also start reasoning at any time. One of the main tasks of Pellet is to verify the consistency of the knowledge base, but not to enumerate or return models. Pellet works on top of the ABox and tries to complete it using the tableau rules. Once done, it verifies the consistency by checking if the completed ABox contains a clash or not. Our algorithm then gets the model as the extraction of atomic concept assertions from the Pellet's ABox. To actually complete the model, if for some individual a and concept A the assertion $a\colon A$ is not present in the ABox we add $a\colon \neg A$ to the model.

Our own implementation includes the minimal hitting set algorithm and its combination with the tableau reasoning using Pellet. All of the optimizations used in Algorithm 3.1 were implemented. Our entire algorithm is based on the construction of the HS-tree, thus our implementation works with its own tree structure, node and edge labelling, collection of conflict sets and models, and so on. For checking the relevance and consistency of the hitting sets it is again necessary to call Pellet, as well as for knowledge base consistency checking.

As we can see in Algorithm 3.1 the algorithm is storing the minimal hitting sets on the fly. It stores only the minimal hitting sets, hence when the main cycle finishes then the answer of the algorithm is the stored collection of minimal hitting sets HS, i.e. the explanations.

Our implementation is available for download at: `http://dai.fmph.uniba.sk/~pukancova/aaa/`.

5 Related Work

Most relevant to our work is the one of Halland and Britz [5,4] who also combine a tableau-based DL reasoner with the Reiter's MHS algorithm. To the best of our knowledge, the actual implementation of the work of Halland and Britz was not described in the literature. It is the aim of our work to implement and evaluate this approach. As noted above, one actual difference in our work is that we avoid to compute all models by TA during pre-processing and instead we call TA on demand during the run of MHS. Another difference is that for now we only support single concept assertion as the observation.

In comparison to the translation-based approaches [2,9] we exploit the tableau-based reasoning directly in DL. Tableau-based reasoning may be more effective in case of DL and we want to exploit various tableau optimization techniques as well.

Yet another interesting approach to abduction was presented by Petatis et al. [11]. Their abduction algorithm is described by Kaya et al. [8]. The abductive reasoning service is tailored to a specific application where only a part of the KB is used abductively represented by conjunctive DL-safe rules. The abduction is then executed by a backward-chaining procedure. Our work is different in that we want to derive a general purpose abductive reasoner for DL in which the whole DL knowledge base is reasoned with abductively.

6 Conclusions

In this preliminary report we have described our implementation of an abductive reasoning algorithm for DL that combines Reiter's minimal hitting set algorithm with a tableau-based DL reasoner, Pellet in this case. Our implementation so far has the following limitations:

1. observations are only in the form of a single concept assertion;
2. explanations are limited to a set of atomic and negated atomic concept assertions, that is the explanation H_5 from Example 2 will not be computed by our algorithm;
3. the computed explanations are syntactically minimal, however our algorithm is not yet able to take also semantic minimality into the account;
4. while some basic optimization techniques are already implemented (especially HS-tree pruning), many of them are still missing (e.g., some form of tableau caching or incremental tableau reasoning).

In the future we would like to address the limitations above, and we would like to conduct an empirical evaluation of the implementation by trial runs on selected ontologies, similarly as described by Du et al. [2].

We would also like to implement a version similar to the approach described by Halland and Britz [4] where the models are computed in an exhaustive run of the tableau reasoner in the pre-processing step and only then the minimal hitting set algorithm is executed. We would like to compare the performance of both approaches as well as some of the approaches based on translation to first-order logic or logic programming [2,9].

Acknowledgments. The research reported in this paper was supported by VEGA projects 1/1333/12 and 1/0867/16. Júlia Pukancová is also supported by an extraordinary scholarship awarded by Faculty of Mathematics, Physics, and Informatics, Comenius University in Bratislava.

References

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
2. Du, J., Qi, G., Shen, Y., Pan, J.Z.: Towards practical ABox abduction in large description logic ontologies. Int. J. Semantic Web Inf. Syst. 8(2), 1–33 (2012)
3. Elsenbroich, C., Kutz, O., Sattler, U.: A case for abductive reasoning over ontologies. In: Proceedings of the OWLED*06 Workshop on OWL: Experiences and Directions, Athens, Georgia, USA, November 10-11, 2006 (2006)
4. Halland, K., Britz, K.: Abox abduction in $\mathcal{ALC}$ using a DL tableau. In: 2012 South African Institute of Computer Scientists and Information Technologists Conference, SAICSIT '12, Pretoria, South Africa, October 1-3, 2012. pp. 51–58 (2012)
5. Halland, K., Britz, K.: Naïve ABox abduction in $\mathcal{ALC}$ using a DL tableau. In: Proceedings of the 2012 International Workshop on Description Logics, DL-2012, Rome, Italy, June 7-10, 2012. Sun SITE Central Europe (CEUR) (2012)
6. Hubauer, T., Legat, C., Seitz, C.: Empowering adaptive manufacturing with interactive diagnostics: A multi-agent approach. In: Advances on Practical Applications of Agents and Multiagent Systems - 9th International Conference on Practical Applications of Agents and Multiagent Systems, PAAMS 2011, Salamanca, Spain, 6-8 April 2011. pp. 47–56 (2011)
7. Karp, R.M.: Reducibility among combinatorial problems. In: Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York. pp. 85–103 (1972)
8. Kaya, A., Melzer, S., Möller, R., Espinosa, S., Wessel, M.: Towards a foundation for knowledge management: Multimedia interpretation as abduction. In: Proceedings of the 2007 International Workshop on Description Logics (DL 2007), Brixen-Bressanone, near Bozen-Bolzano, Italy, 8-10 June, 2007 (2007)
9. Klarman, S., Endriss, U., Schlobach, S.: ABox abduction in the description logic $\mathcal{ALC}$. Journal of Automated Reasoning 46(1), 43–80 (2011)
10. Peirce, C.S.: Deduction, induction, and hypothesis. Popular science monthly 13, 470–482 (1878)
11. Petasis, G., Möller, R., Karkaletsis, V.: BOEMIE: Reasoning-based information extraction. In: Proceedings of the 1st Workshop on Natural Language Processing and Automated Reasoning co-located with 12th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2013), A Corunna, Spain, September 15th, 2013. pp. 60–75 (2013)
12. Pukancová, J., Homola, M.: Abductive reasoning with description logics: Use case in medical diagnosis. In: Proceedings of the 28th International Workshop on Description Logics (DL 2015), Athens,Greece, June 7-10, 2015. (2015)
13. Reiter, R.: A theory of diagnosis from first principles. Artif. Intell. 32(1), 57–95 (1987)
14. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. Web Semantics: science, services and agents on the World Wide Web 5(2), 51–53 (2007)