ABox Abduction Solver Exploiting Multiple DL Reasoners

Drahomír Mrózek, Júlia Pukancová and Martin Homola

Comenius University in Bratislava
Mlynská dolina, 84248 Bratislava
`drahosmrozek@gmail.com,{pukancova,homola}@fmph.uniba.sk`

Abstract. We have implemented an ABox abduction solver based on Reiter's minimal hitting set algorithm. Our solver runs a DL reasoner as a black box, similarly to the existing AAA solver. The advantage of the current work is the use of OWL API, which allows to plug-in a number of different DL reasoners. In this paper we describe our implementation and present an evaluation results comparing AAA, which is tightly coupled with Pellet, and the current implementation running with three different reasoners: Pellet, JFact, and HermiT. The latter combination showed the most promising results.

Keywords: description logics · abduction · implementation · evaluation

1 Introduction

Abduction, originally introduced by Peirce [18], is non-standard reasoning problem whose goal is to provide explanations why some observation does not follow from a knowledge base. Abduction in DL naturally distinguishes between TBox and ABox abduction [5] the former working with observations and explanations on the TBox level, the latter assuming that observations and explanations are ABox assertions. We focus on ABox abduction which has interesting applications, e.g. in diagnostic reasoning [13,5,19] or in multimedia-interpretation [6,2]. A number of theoretical algorithms for ABox abduction were presented [14,10,9], and some were also implemented into abduction solvers [4,15,6,2,3].

In this paper, we extend the AAA solver [20] which focuses on expressive DLs. Compared to previous works, this solver is able to handle any DL expressivity up to $\mathcal{SROIQ}$, due to using a DL reasoner as a black box, and it is sound and complete. However, the main disadvantage of AAA is that it is tightly coupled with Pellet 2 [23] which is an outdated reasoner and it makes AAA less effective.

We describe a reimplementation of AAA, dubbed B, and report on our first experimental results, providing the following contributions: (a) B exploits OWL API [11] and thus it is able to modularly plugin different DL reasoners; (b) it includes more extensive pruning and an additional optimization technique called caching of inconsistent hitting-set candidates; (c) we report on first empirical results comparing AAA and B combined with Pellet 2, JFact [17], and HermiT [22,7] reasoners.

Table 1. Syntax and Semantics of $\mathcal{ALCHO}$

Constructor	Syntax	Semantics
complement	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
intersection	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
existential restriction	$\exists R.C$	$\{x \mid \exists y\, (x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
nominal	$\{a\}$	$\{a^{\mathcal{I}}\}$
Axiom	Syntax	Semantics
concept incl. (GCI)	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
role incl. (RIA)	$R \sqsubseteq S$	$R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
concept assertion	$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
role assertion	$R(a,b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$
neg. role assertion	$\neg R(a,b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \notin R^{\mathcal{I}}$

We plan to continue our experiments with B, especially testing it with additional DL reasoners. In the future it will serve as a base for next versions of AAA.

2 ABox Abduction in DL

We build on top of the $\mathcal{ALCHO}$ [1]. A vocabulary consists of countably infinite mutually disjoint sets of individuals $N_{\mathrm{I}} = \{a, b, \ldots\}$, roles $N_{\mathrm{R}} = \{P, R, \ldots\}$, and atomic concepts $N_{\mathrm{C}} = \{A, B, \ldots\}$. Concepts are recursively built using constructors $\neg$, $\sqcap$, $\exists$, $\{a\}$, as shown in Table 1. Additional concepts union $C \sqcup D := \neg(\neg C \sqcap \neg D)$ and value restriction $\forall R.C := \neg\exists R.\neg C$ are defined as syntactic sugar; and also $\neg\neg C := C$ by definition. A knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consists of a TBox $\mathcal{T}$, a finite set of GCI and RIA axioms, and an ABox $\mathcal{A}$, a finite set of assertions as given in Table 1.

An interpretation is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}} \neq \emptyset$ is a domain, and the interpretation function $\cdot^{\mathcal{I}}$ maps each individual $a \in N_{\mathrm{I}}$ to $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, each atomic concept $A \in N_{\mathrm{C}}$ to $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, each role $R \in N_{\mathrm{R}}$ to $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ in such a way that the constraints on the right-hand side of Table 1 are satisfied.

An interpretation $\mathcal{I}$ satisfies an axiom φ (denoted $\mathcal{I} \models \varphi$) if the respective constraint in Table 1 is satisfied. It is a model of a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ (denoted $\mathcal{I} \models \mathcal{K}$) if $\mathcal{I} \models \varphi$ for all $\varphi \in \mathcal{T} \cup \mathcal{A}$. A knowledge base is consistent, if there is at least one interpretation $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{K}$. A knowledge base entails an axiom φ (denoted $\mathcal{K} \models \varphi$) if $\mathcal{I} \models \varphi$ for each $\mathcal{I} \models \mathcal{K}$.

We define $\neg\varphi := \neg C(a)$ for a concept assertion $\varphi = C(a)$. Thanks to presence of nominals in $\mathcal{ALCHO}$ [12,10] we are also able to define $\neg\varphi := \neg R(a,b) := \forall R.\neg\{b\}(a)$ for a role assertion $\varphi = R(a,b)$, and $\neg\varphi := R(a,b)$ for $\varphi = \neg R(a,b)$. In addition, $\neg\mathcal{A} = \{\neg\varphi \mid \varphi \in \mathcal{A}\}$ for any set of ABox assertions $\mathcal{A}$. The *ABox encoding* of an interpretation $\mathcal{I}$ is $M_{\mathcal{I}} = \{C(a) \mid \mathcal{I} \models C(a), C \in \{A, \neg A\}, A \in N_{\mathrm{C}}, a \in N_{\mathrm{I}}\} \cup \{R(a,b) \mid \mathcal{I} \models R(a,b),\ R \in N_{\mathrm{R}},\ a, b \in N_{\mathrm{I}}\} \cup \{\neg R(a,b) \mid \mathcal{I} \models \neg R(a,b),$

$R \in N_{\mathrm{R}}$, $a, b \in N_{\mathrm{I}}\}$. Note that ABox encodings obtained from models of finite knowledge base which has a finite signature are assumed to be finite. The are in no way homomorphic with the original models, as they ignore the anonymous part of the model. A negation $\neg M_{\mathcal{I}}$ of the ABox encoding of a model $\mathcal{I}$ of $\mathcal{K}$ is called an antimodel $\mathcal{K}$.

In *ABox abduction*, we are given a knowledge base $\mathcal{K}$ and an observation $\mathcal{O}$ consisting of ABox assertions, that is, some evidence we have observed. The task is to find an explanation $\mathcal{E}$, again, consisting of ABox assertions, such that $\mathcal{K} \cup \mathcal{E} \models \mathcal{O}$.

Definition 1 (ABox Abduction Problem [5]). *An ABox abduction problem is a pair $\mathcal{P} = (\mathcal{K}, \mathcal{O})$ such that $\mathcal{K}$ is a knowledge base in DL and $\mathcal{O}$ is a set of ABox assertions. A solution of $\mathcal{P}$ (also called explanation) is any finite set $\mathcal{E}$ of ABox assertions such that $\mathcal{K} \cup \mathcal{E} \models \mathcal{O}$.*

While Definition 1 establishes the basic reasoning mechanism of abduction, some of the explanations it permits are clearly undesired. The explanations should, at minimum, fulfil some basic sanity requirements.

Definition 2 ([5]). *Given an ABox abduction problem $\mathcal{P} = (\mathcal{K}, \mathcal{O})$ and its solution $\mathcal{E}$ we say that:*

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

An explanation should be consistent, as anything follows from inconsistency; and so, an explanation that makes $\mathcal{K}$ inconsistent does not really explain the observation. It should be relevant – it should not imply the observation directly without requiring the knowledge base $\mathcal{K}$ at all. And it should be explanatory, that is, we should not be able to explain the observation without it.

Hereafter, when we say explanation we always mean a consistent, relevant, and explanatory explanation, unless indicated otherwise. In addition, in order to avoid excess hypothesizing, minimality is required.

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

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

3 Our Approach

We have implemented an ABox abduction solver based on the approach of Pukancová and Homola [20] which implements the Reiter's minimal hitting set algorithm [21]. Likewise to the original AAA solver [20], the input observations can be any (also complex) concept and role assertions. The explanations are limited to sets of atomic and negated atomic concept and role assertions.

For a single observation O, a solution of an abduction problem $\mathcal{P} = (\mathcal{K}, O)$ according to Definition 1 can be obtained as any $\mathcal{E}$ s.t. $\mathcal{K} \cup \mathcal{E} \cup \{\neg O\}$ is inconsistent As showed by Reiter [21], we can compute the minimal explanations of $\mathcal{P}$ by finding all minimal hitting sets for all antimodels of $\mathcal{K} \cup \{\neg O\}$.

We do this by searching through the candidate hitting sets breadth-first. For details see the report of Pukancová and Homola [20], on which we base our implementation. The algorithm SingleAbduction is listed below.

Algorithm 1 SingleAbduction($\mathcal{K}$, O, $D_{\max}$, $\mathcal{O}$)

```
Require: knowledge base $\mathcal{K}$, single observation $O$, maximum depth $D_{\max}$, minimal
    inconsistent candidates $\mathcal{S}_\perp$, set of observations $\mathcal{O}$
Ensure: set of all explanations $\mathcal{S}$, set of all minimal inconsistent candidates $\mathcal{S}_\perp$
 1: if $\mathcal{K} \cup \{O\}$ is inconsistent then
 2:     return $\emptyset$                    ▷ no consistent explanations as $\mathcal{K} \models \neg O$
 3: else if $\mathcal{K} \cup \{\neg O\}$ is inconsistent then
 4:     return "nothing to explain"
 5: end if
 6: $D \leftarrow 1$; $\mathcal{C} \leftarrow \{\emptyset\}$; $\mathcal{S} \leftarrow \emptyset$
 7: while $\mathcal{C} \neq \emptyset$ and $D \leq D_{\max}$ do
 8:     $\mathcal{C}_{\text{next}} \leftarrow \emptyset$                ▷ hitting set candidates for the next iteration
 9:     for all $c \in \mathcal{C}$ do
10:         if $s \not\subseteq c$ for all $s \in \mathcal{S}$
                and $c \not\models O$ for all $O \in \mathcal{O}$
                and memCons($\mathcal{K} \cup \mathcal{O}$, $c$, $\mathcal{S}_\perp$)
                and $\mathcal{K} \cup \{\neg O\} \cup c$ is inconsistent then
11:             $\mathcal{S} \leftarrow \mathcal{S} \cup \{c\}$                ▷ $c$ is a hitting set (explanation)
12:         else if $D \neq D_{\max}$ then
13:             for all $\phi \in$ redAntiModel($\mathcal{K}$, $O$, $\mathcal{O}$, $c$) do
14:                 $\mathcal{C}_{\text{next}} \leftarrow \mathcal{C}_{\text{next}} \cup \{c \cup \phi\}$
15:             end for
16:         end if
17:     end for
18:     $\mathcal{C} \leftarrow \mathcal{C}_{\text{next}}$
19:     $D \leftarrow D + 1$
20: end while
21: return $\mathcal{S}$, $\mathcal{S}_\perp$
```

The first three parameters are the input knowledge base $\mathcal{K}$, the observation O, and the maximum depth $D_{\max}$ which allows to limit the maximal length of explanations to search for. The remaining two parameters are auxiliary and they are important when SingleAbduction is called repeatedly to find explanations for multiple observations. For the single observation case the algorithm is called as SingleAbduction($\mathcal{K}, O, D_{\max}, \emptyset, \{O\}$).

The algorithm initializes the set of candidate explanations to $\{\emptyset\}$. Then we loop through the candidates c breadth-first, and process each candidate by either removing (pruning [20]) the candidate if it is not minimal, relevant, or consistent (first three if-conditions in line 10) or adding it to the set of explanations if $\mathcal{K} \cup \{\neg O\} \cup c$ is inconsistent, i.e. $\mathcal{K} \cup c \models O$ (the last if-condition in line 10). Finally, if $\mathcal{K} \cup \{\neg O\} \cup c$ was consistent (and $D_{\max}$ is not yet reached) then we obtain the respective antimodel and populate the candidates for the next level of search.

The algorithm features two additional optimizations. Firstly, we do not store only explanations (i.e., minimal hitting sets) found so far, but we also store all minimal candidates which are inconsistent (the set $\mathcal{S}_\perp$). This allows for additional pruning which is implemented in the memCons function.

```
1: function memCons( Knowledge base $\mathcal{K}$, hitting set candidate $c$, minimum inconsistent candidates $\mathcal{S}_\perp$)
2:     if $s \subseteq c$ for some $s \in \mathcal{S}_\perp$ then
3:         return false
4:     else if $\mathcal{K} \cup c$ is consistent then
5:         return true
6:     else
7:         $\mathcal{S}_\perp \leftarrow \mathcal{S}_\perp \cup \{c\}$
8:         return false
9:     end if
10: end function
```

Consecutively, we further extend pruning by reducing all antimodels by removing assertions ϕ if ϕ or $\neg\phi$ is present in $\mathcal{K} \cup \{\neg O\} \cup c$. It's straightforward to observe that all such candidates would be inconsistent, irrelevant, or not minimal. This is implemented in the redAntiModel function but also in the check in line 1 of SingleAbduction.

To find solutions for multiple-observation abduction problem, we rely on the approach based on solving separate single-observation problems and combining the results [20]. The only minor improvement is in passing the set $\mathcal{S}_\perp$ between the SingleAbduction calls. The algorithm is called MultipleAbduction.

The main advantage of our approach compared to previous works [20,9] is that DL reasoner is not tightly integrated, but instead called using OWL API. This allows modular pairing with different reasoners. The reasoner is called from SingleAbduction and memCons whenever consistency or inconsistency checks are made. The function redAntiModel does not run reasoning again but merely extracts the model from last successful consistency check calling `getTypes` and `getObjectPropertyValues` from OWL API.

```
1: function redAntiModel( Knowledge base $\mathcal{K}$, observation $O$, observations $\mathcal{O}$,
   hitting set candidate $c$ )
2:     $\mathcal{AM} \leftarrow \emptyset$
3:     $\mathcal{M} \leftarrow$ ABox encoding of model of $\mathcal{K} \cup \{\neg O\} \cup c$
4:     for all axiom $\phi \in \mathcal{M}$ do
5:         if $\phi \notin \mathcal{K} \cup \mathcal{O} \cup c$ and $\neg\phi \notin \mathcal{K} \cup \mathcal{O} \cup c$ then
6:             $\mathcal{AM} \leftarrow \mathcal{AM} \cup \{\neg\phi\}$
7:         end if
8:     end for
9:     return $\mathcal{AM}$
10: end function
```

Algorithm 2 MultipleAbduction($\mathcal{K}$, $\mathcal{O}$, $D_{\max}$)

```
Require: knowledge base $\mathcal{K}$, observations $\mathcal{O}$, maximum depth $D_{\max}$
Ensure: set of all explanations $\mathcal{S}$, set of all minimal inconsistent candidates $\mathcal{S}_\perp$
1: $\Sigma \leftarrow \emptyset$; $\mathcal{S}_\perp \leftarrow \emptyset$
2: for all observations $O_i \in \mathcal{O}$ do
3:     $\mathcal{S}_i, \mathcal{S}_\perp \leftarrow$SingleAbduction($\mathcal{K}$, $O_i$, $D_{\max}$, $\mathcal{S}_\perp$, $\mathcal{O}$)
4:     if $\mathcal{S}_i = \emptyset$ then
5:         return $\emptyset$          ▷ $O$ has no explanation, hence also $\mathcal{O}$ has no explanations
6:     else if $\mathcal{S}_i \neq$ "nothing to explain" then
7:         $\Sigma \leftarrow \Sigma \cup \{\mathcal{S}_i\}$
8:     end if
9: end for
10: if $\Sigma = \{\}$ then
11:     return "nothing to explain"
12: end if
13: $\mathcal{S} \leftarrow \{\mathcal{E}_1 \cup \cdots \cup \mathcal{E}_m \mid \mathcal{E}_i \in \mathcal{S}_i, \mathcal{S}_i \in \Sigma, m = |\Sigma|\}$
14: $\mathcal{S} \leftarrow \{\mathcal{E} \in \mathcal{S} \mid \mathcal{E}$ is minimal, relevant, and memCons($\mathcal{K} \cup \mathcal{O}$, $\mathcal{E}$, $\mathcal{S}_\perp$) is true$\}$
15: return $\mathcal{S}$
```

4 Evaluation

Experimental evaluation was conducted with implementations of AAA and B, B paired with three different DL reasoners – Pellet, HermiT, and JFact. Two different experiments were conducted, one with a single observation and one with a multiple observation. The main goal was to compare the execution times. It is also interesting to trace the differences between the computation of the two implementations AAA and B, especially how is the search space pruned.

The source code of both implementations is available at `http://dai.fmph.uniba.sk/~pukancova/aaa/`.

4.1 Dataset and Methodology

We have chosen three ontologies for the evaluation: Family ontology (Our own small ontology of family relations)[2], Coffee ontology by Carlos Mendes[3], and LUBM (Lehigh University Benchmark [8]). The parameters of the ontologies are stated in Table 2.

Table 2. Parameters of the ontologies

Ontology	**Concepts**	**Roles**	**Individuals**	**Axioms**
Family ontology	8	1	2	24
Coffee ontology	41	6	2	291
LUBM	43	25	1	46

In all experiments, explanations for an observation are computed through both AAA and B, whilst B is run three times – once with Pellet, once with HermiT, and once with JFact. All experiments were done on a 6-core 3.2 GHz AMD Phenom^{TM} II X6 1090T Processor, 8 GB RAM, running Ubuntu 17.10, Linux 4.13.0, while the maximum Java heap size was set to 4GB. We have used the GNU `time` utility to measure the CPU time consumed by AAA while running in user mode, summed over all threads.

In the single observation experiment, the experiments are conducted iteratively for the maximal length of explanations from 1 to 5. In the multiple observation experiment, the iterations are only up to the maximal length of 3. For each experimental setting, the run is repeated for 10 times. From now on, all execution times are computed as the average values from 10 runs with the same experimental setting.

All experiments were executed while disallowing explanations with loops (i.e., reflexive role assertions), an optional feature of both solvers.

4.2 Single Observation Experiment

As mentioned above, the single observation experiment was conducted for all the three ontologies: Family, Coffee and LUBM. For each ontology, one single observation was chosen: for Family ontology $\mathsf{Mother(jane)}$, for Coffee ontology $\mathsf{Macchiato(a)}$, and for LUBM $\mathsf{Person(jack)}$.

The experiment was conducted iteratively for the maximal length of explanations from 1 to 5 with AAA and B paired with each reasoner.

The average execution times are plotted in Figure 1. The deviations computed for each set of 10 runs with the same experimental setting were quite low 2.504 % on average. The times for the four experiments – LUBM ontology for the maximal lengths 4 and 5 through implementation B with Pellet and JFact – actually do

[2] `http://dai.fmph.uniba.sk/~pukancova/aaa/ont/`

[3] `https://gist.githubcom/cmendesce/56e1e16aee5a556a186f512eda8dabf3`

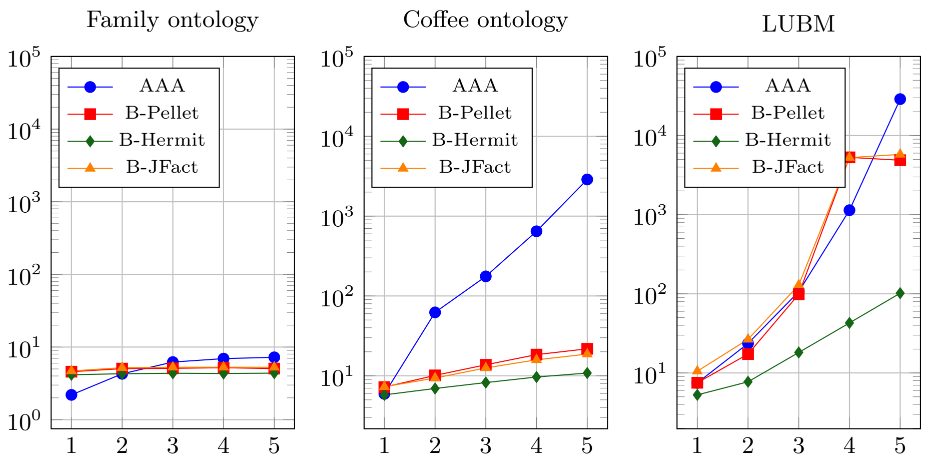
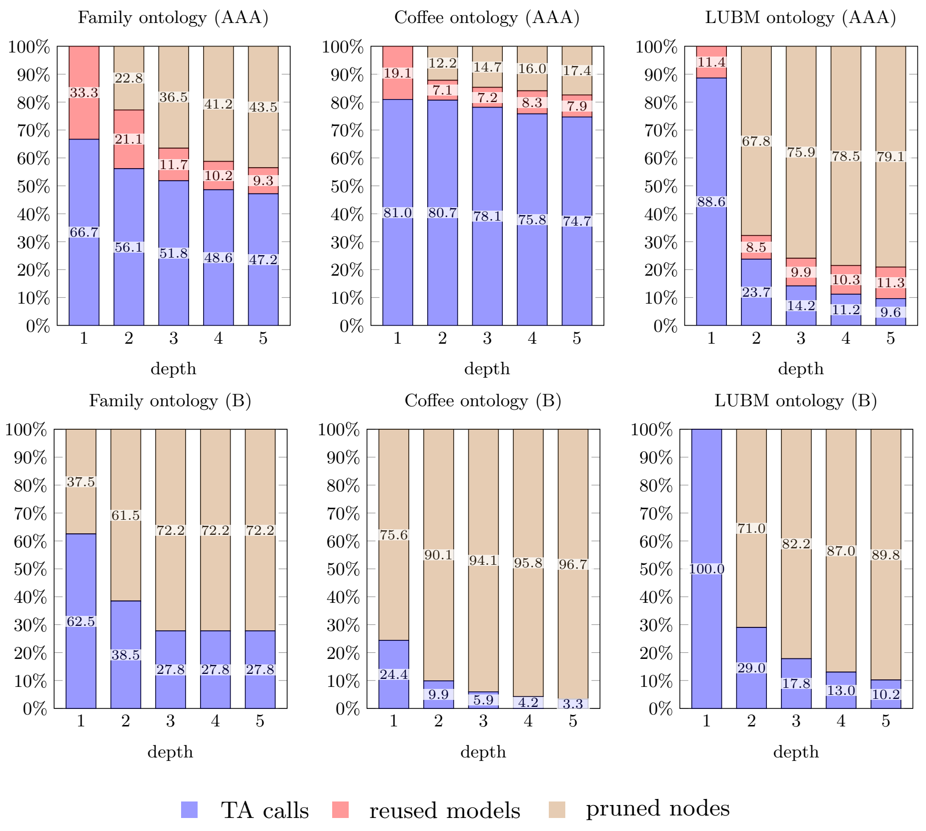


Fig. 1. Depth vs. time in seconds for single observation



Family ontology (AAA)	1	2	3	4	5
TA calls	66.7	56.1	51.8	48.6	47.2
reused models	33.3	21.1	11.7	10.2	9.3
pruned nodes		22.8	36.5	41.2	43.5

Coffee ontology (AAA)	1	2	3	4	5
TA calls	81.0	80.7	78.1	75.8	74.7
reused models	19.1	7.1	7.2	8.3	7.9
pruned nodes		12.2	14.7	16.0	17.4

LUBM ontology (AAA)	1	2	3	4	5
TA calls	88.6	23.7	14.2	11.2	9.6
reused models	11.4	8.5	9.9	10.3	11.3
pruned nodes		67.8	75.9	78.5	79.1

Family ontology (B)	1	2	3	4	5
TA calls	62.5	38.5	27.8	27.8	27.8
pruned nodes	37.5	61.5	72.2	72.2	72.2

Coffee ontology (B)	1	2	3	4	5
TA calls	24.4	9.9	5.9	4.2	3.3
pruned nodes	75.6	90.1	94.1	95.8	96.7

LUBM ontology (B)	1	2	3	4	5
TA calls	100.0	29.0	17.8	13.0	10.2
pruned nodes		71.0	82.2	87.0	89.8

Fig. 2. Proportion of pruned nodes, reused models and TA calls for multiple observations

not correspond to the times when explanations were computed but the times when Java memory was exceeded.

The search space is pruned according to the implementation used. The proportion of non-pruned nodes (nodes for which tableau algorithm TA for consistency check is called) and pruned nodes is showed in Figure 2. Note that, AAA implements also model reuse, whilst B does not. The results for B are the same in cases when Java did not run out of memory for all three reasoners. For this reason there is always just one plot in case of B.

In this experiment, the implementation B using HermiT achieves the lowest times except in two cases – Family ontology with the maximal lengths 1 and 2. On the other hand, in 8 cases (from the overall 15 cases) AAA has the highest times. Let us remind, that B with Pellet and JFact reasoners ran out of memory in case of the LUBM ontology with the maximal lengths 4 and 5, so these times are not relevant.

4.3 Multiple Observation Experiment

The multiple observation experiment was conducted analogously. The only difference is that the observations are in the form of a set of ABox assertions, namely: for Family ontology $\{\mathsf{Father(jack)}, \mathsf{Mother(eva)}, \mathsf{Person(fred)}\}$, for Coffee ontology $\{\mathsf{Milk(a)}, \mathsf{Coffee(b)}, \mathsf{Pure(c)}\}$, and for LUBM $\{\mathsf{Person(jack)}, \mathsf{Employee(jack)}, \mathsf{Publication(a)}\}$.

For each ontology with the respective observation, explanations were computed iteratively with the maximal length from 1 to 3 through AAA, and trough B using Pellet, HermiT, and JFact. The average times are plotted in Figure 3. The average deviation was 7.06 %. The following experiments ran out of memory: Family ontology through B using Pellet and JFact with the maximal length 3, Coffee ontology through B using Pellet with the maximal length 3 and using JFact with the maximal lengths 2 and 3, LUBM through AAA with the maximal length 3, and through B using Pellet and JFact with the maximal lengths 1, 2, and 3. Note that, the only reasoner that did not run out of memory is HermiT.

Also in this experiment, the numbers of pruned nodes and non-pruned nodes were computed. The respective proportions are captured in Figure 4. In case of B, the proportions are the same with the three different reasoners (out-of-memory cases are ignored).

In this experiment, the times with HermiT were always the lowest. Also, while a number of experiments ran out of memory this was never the case with B combined with HermiT.

5 Conclusions

We have reimplemented the AAA ABox abduction solver [20]. The new implementation, called B, used OWL API [11] and thus can be run with different reasoner. In the evaluation we have compared AAA and B combined with Pellet 2 [23], JFact [17], and HermiT [22,7].

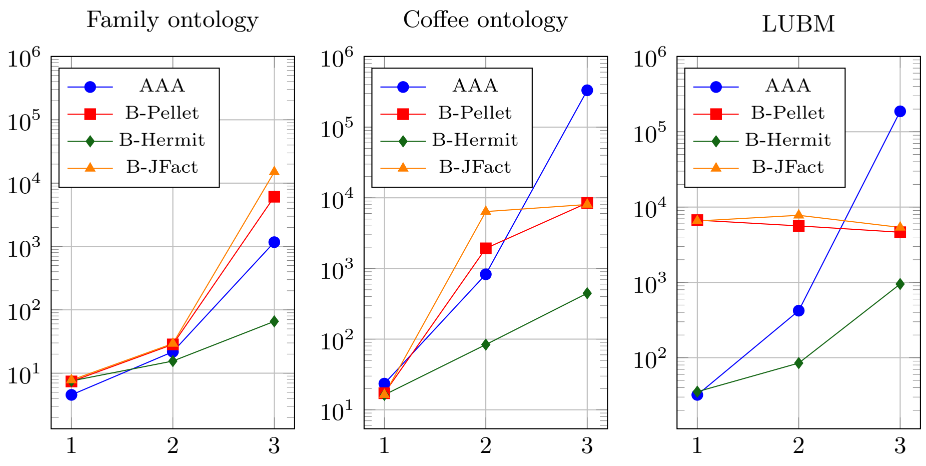


Fig. 3. Depth vs. time in seconds for multiple observation

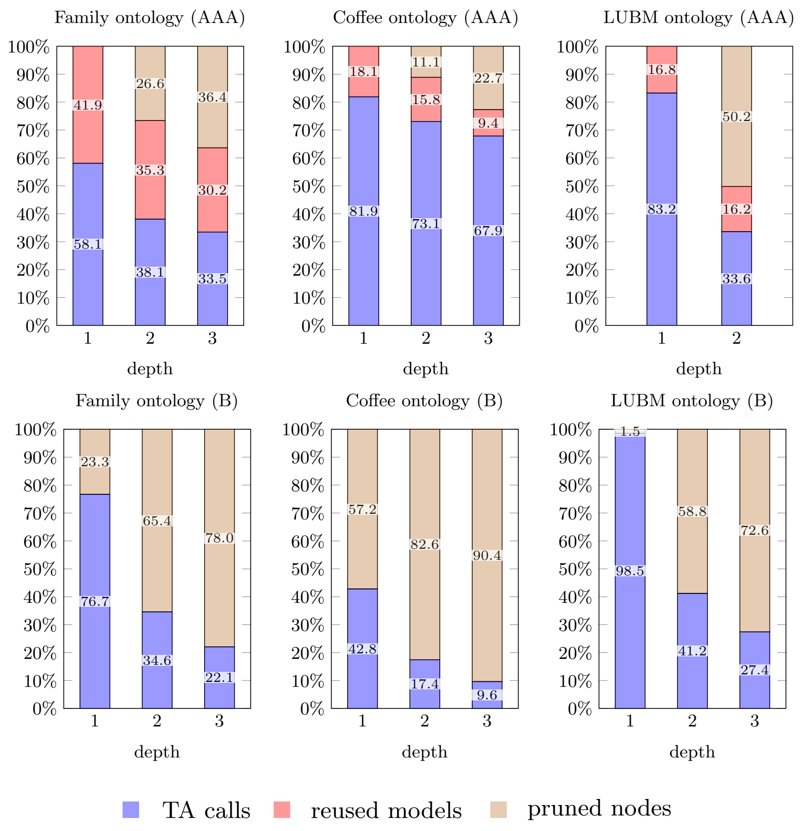


Fig. 4. Proportion of pruned nodes, reused models and TA calls for multiple observations

Not only that in the most cases HermiT achieved the best time, but it is also the only reasoner that did not exceed the memory. The combination of B and HermiT seems to be currently the best option for our abduction solver. Moreover, the evaluation has also showed that B, applying additional pruning techniques, processes the search space in a more optimal way.

On the other hand, B has exceeded the memory a couple more times than AAA, a problem that we would like to tune in the future. We would also like extend this evaluation with additional reasoners.

Acknowledgements. This work presents the results of the Master's thesis by Drahomír Mrózek [16]. It was supported by the Slovak national project VEGA 1/0778/18. Júlia Pukancová is also supported by an extraordinary scholarship awarded by Faculty of Mathematics, Physics, and Informatics, Comenius University in Bratislava, and by the Comenius University grant no. UK/266/2018. We would like to thank to anonymous reviewers from previous DL workshops for valuable suggestions.

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. Castano, S., Espinosa Peraldí, I.S., Ferrara, A., Karkaletsis, V., Kaya, A., Möller, R., Montanelli, S., Petasis, G., Wessel, M.: Multimedia interpretation for dynamic ontology evolution. J. Log. Comput. 19(5), 859–897 (2009)
3. Del-Pinto, W., Schmidt, R.A.: Forgetting-based abduction in $\mathcal{ALC}$. In: Proceedings of the Workshop on Second-Order Quantifier Elimination and Related Topics (SOQE 2017), Dresden, Germany. CEUR-WS, vol. 2013, pp. 27–35 (2017)
4. 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)
5. 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, GA, USA. CEUR-WS, vol. 216 (2006)
6. Espinosa Peraldí, I.S., Kaya, A., Möller, R.: Formalizing multimedia interpretation based on abduction over description logic ABoxes. In: Proceedings of the 22nd International Workshop on Description Logics (DL 2009), Oxford, UK. CEUR-WS, vol. 477 (2009)
7. Glimm, B., Horrocks, I., Motik, B., Stoilos, G., Wang, Z.: Hermit: An OWL 2 reasoner. Journal of Automated Reasoning 53(3), 245–269 (2014)
8. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. Journal of Web Semantics 3(2-3), 158–182 (2005)
9. 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. pp. 51–58 (2012)
10. 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. CEUR-WS, vol. 846 (2012)

11. Horridge, M., Bechhofer, S.: The OWL API: A java API for OWL ontologies. Semantic Web 2(1), 11–21 (2011)
12. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning, Lake District of the United Kingdom. pp. 57–67. AAAI (2006)
13. 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. pp. 47–56 (2011)
14. Klarman, S., Endriss, U., Schlobach, S.: ABox abduction in the description logic $\mathcal{ALC}$. Journal of Automated Reasoning 46(1), 43–80 (2011)
15. Ma, Y., Gu, T., Xu, B., Chang, L.: An ABox abduction algorithm for the description logic $\mathcal{ALCI}$. In: Intelligent Information Processing VI – 7th IFIP TC 12 International Conference, IIP 2012, Guilin, China. Proceedings. IFIP AICT, vol. 385, pp. 125–130. Springer (2012)
16. Mrózek, D.: Abduction Solver Exploiting Multiple DL Reasoners. Master's thesis, Comenius University in Bratislava (2018)
17. Palmisano, I.: Jfact dl reasoner. `http://jfact.sourceforge.net/`
18. Peirce, C.S.: Deduction, induction, and hypothesis. Popular science monthly 13, 470–482 (1878)
19. 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. CEUR-WS, vol. 1350 (2015)
20. Pukancová, J., Homola, M.: Tableau-based ABox abduction for the $\mathcal{ALCHO}$ description logic. In: Proceedings of the 30th International Workshop on Description Logics, Montpellier, France. CEUR-WS, vol. 1879 (2017)
21. Reiter, R.: A theory of diagnosis from first principles. Artificial intelligence 32(1), 57–95 (1987)
22. Shearer, R., Motik, B., Horrocks, I.: Hermit: A highly-efficient OWL reasoner. In: Proceedings of the Fifth OWLED Workshop on OWL: Experiences and Directions, Karlsruhe, Germany. CEUR-WS, vol. 432 (2008)
23. Sirin, E., Parsia, B., Cuenca Grau, B., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. Journal of Web Semantics 5(2), 51–53 (2007)