CEUR-WS.org/Vol-2663/paper-11.pdf

Merge, Explain, Iterate

Martin Homola, Júlia Pukancová, Júlia Gablíková, and Katarína Fabianová

Comenius University in Bratislava, Mlynská dolina, 84248 Bratislava, Slovakia
{homola|pukancova}@fmph.uniba.sk, {gablikova.julia|kfabianova11}@gmail.com

Abstract. ABox abduction is the reasoning problem to find explanations for a DL knowledge base (KB) and an observation given as an ABox assertion which does not follow from the KB. The explanations themselves are sets of ABox assertions such that the observation follows from the KB once an explanation set is included in it. One widely employed method to compute the explanations is the Minimal Hitting Set algorithm (MHS). MHS is complete, but it is also NP-complete and widely recognized as inefficient. There are also approximative methods such as MergeXplain (MXP) which are fast, but incomplete. This work describes a hybrid method called MHS-MXP which adopts the divide and conquer heuristics of MXP and applies it on MHS with the aim to make it more efficient at least on a favourable class of inputs of the ABox abduction problem. First experimental implementation is also available.

Keywords: Description logics · Abduction · MergeXplain · Minimal hitting set · Optimization

1 Introduction

Abduction was introduced by Peirce [7]. It is a reasoning problem to find explanations of an observation that is not entailed based on our current knowledge. In the context of DL, an *ABox abduction* problem [2] assumes a DL KB $\mathcal{K}$ and an extensional observation O (in from of an ABox assertion). Explanations (also extensional) are sets of ABox assertions $\mathcal{E}$ such that $\mathcal{K}$ together with $\mathcal{E}$ entail O.

If one wishes to find all explanations of an ABox abduction problem, the Reiter's MHS algorithm [10] is the classic method. MHS is complete. In a nutshell, it searches through the space of all possible explanations, starting from the smallest (in terms of cardinality) and continues towards the largest. The MHS algorithm (NP-complete) works on top of a DL solver (whose complexity depends on the particular DL). This approach is useful particularly in cases where there are smaller explanations. However in cases where there is (even a small number of) explanations of larger size this search strategy may be inefficient.

Alternative strategies were explored. Junker's QuickXplain (QXP) [5], and more recently its extension MergeXplain [11] proposed by Shchekotykhin et al.

Copyright © 2020 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

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

Concept	Syntax	Semantics
Atomic concept	A	$A^{\mathcal{I}}$
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 inclusion (GCI)	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
role inclusion (RIA)	$R \sqsubseteq S$	$R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
concept assertion	$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
role assertion	$R(a,b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$
negated role assertion	$\neg R(a,b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \notin R^{\mathcal{I}}$

employ *divide and conquer* strategy which allows to find one (QXP) or multiple explanations (MXP) very efficiently. On the other hand, these methods are approximative, i.e., they are not complete.

We base this work on the key observation that when MXP is run repeatedly, with a slightly modified inputs, it divides the search space differently and it may possibly return a different set of explanations than in the previous runs. We propose a combined algorithm, dubbed MHS-MXP, that iterates runs of MXP and it uses MHS to steer the search space exploration in such a way that completeness is retained. We also provide a preliminary experimental implementation.

Compared to pure MXP, this approach has the advantage that it is no longer approximative – it is complete. Such a hybrid algorithm may likely have an advantage when compared with MHS, at least on a certain class of inputs. Precise characterization of this class and further optimization of the algorithm is subject to our ongoing work.

2 Preliminaries

For simplicity we will introduce $\mathcal{ALCHO}$ [1]. However any other DL may be used due to the *black box* approach. A vocabulary consists of countably infinite mutually disjoint sets of individuals $N_{\mathrm{I}} = \{a, b, \ldots\}$, roles $N_{\mathrm{R}} = \{P, Q, \ldots\}$, and atomic concepts $N_{\mathrm{C}} = \{A, B, \ldots\}$. Concepts are recursively built using constructors $\neg$, $\sqcap$, $\exists$, as shown in Table 1. A KB $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$ consists of a finite set of GCI and RIA axioms (in TBox $\mathcal{T}$), and a finite set of assertions (in ABox $\mathcal{A}$) as given in Table 1. We also define $\neg\sigma := \neg C(a)$ if $\sigma = C(a)$; $\neg\sigma := C(a)$ if $\sigma = \neg C(a)$; $\neg\sigma := \neg R(a,b)$ if $\sigma = R(a,b)$; $\neg\sigma := R(a,b)$ if $\sigma = \neg R(a,b)$.

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 $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and each complex concept according to the right-hand side of Table 1.

An interpretation $\mathcal{I}$ satisfies an axiom σ (denoted $\mathcal{I} \models \sigma$) if the respective constraint in Table 1 is satisfied. It is a model of a KB $\mathcal{K}$ (denoted $\mathcal{I} \models \mathcal{K}$) if $\mathcal{I} \models \sigma$ for all $\sigma \in \mathcal{K}$. A KB $\mathcal{K}$ is consistent, if $\mathcal{I} \models \mathcal{K}$ for some interpretation $\mathcal{I}$. $\mathcal{K}$ entails an axiom σ (denoted $\mathcal{K} \models \sigma$) if $\mathcal{I} \models \sigma$ for each $\mathcal{I} \models \mathcal{K}$.

In *ABox abduction* [2], we are given a KB $\mathcal{K}$ and an observation O consisting of an ABox assertion. The task is to find an *explanation* $\mathcal{E}$, again, consisting of ABox assertions, such that $\mathcal{K} \cup \mathcal{E} \models O$. However, the set of all ABox expressions may be too broad to draw the explanations from (after all, it is infinite). Hence we typically consider some (finite and reasonably small) set of *abducibles* Abd, and require that $\mathcal{E} \subseteq \text{Abd}$.

Definition 1 (ABox Abduction Problem). *Let* Abd *be a finite set of ABox assertions. 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. An explanation of $\mathcal{P}$ (on* Abd*) is any finite set of ABox assertions $\mathcal{E} \subseteq \text{Abd}$ such that $\mathcal{K} \cup \mathcal{E} \models O$.*

In this work we limit the explanations to atomic and negated atomic concept and role assertions; hence $\text{Abd} \subseteq \{A(a), \neg A(a) \mid A \in N_{\mathrm{C}},\ a \in N_{\mathrm{I}}\} \cup \{R(a,b), \neg R(a,b) \mid R \in N_{\mathrm{R}},\ a,b \in N_{\mathrm{I}}\}$. Note that we do not limit the observations in any way, apart from allowing only one (possibly complex) ABox assertion.

While Definition 1 establishes the basic reasoning frame of abduction, some explanations may be unintuitive. According to Elsenbroich et al. [2] it is reasonable to require from each explanation $\mathcal{E}$ of $\mathcal{P} = (\mathcal{K}, O)$ to be: (a) *consistent* ($\mathcal{K} \cup \mathcal{E}$ is consistent); (b) *relevant* ($\mathcal{E} \not\models O$); and (c) *explanatory* ($\mathcal{K} \not\models O$). Explanations which satisfy these three conditions will be called *desired*. In addition, in order to avoid excess hypothesizing, minimality is required.

Definition 2 (Minimality). *Assume an ABox abduction problem $\mathcal{P} = (\mathcal{K}, O)$. Given two explanations $\mathcal{E}$ 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 an explanation $\mathcal{E}$ of $\mathcal{P}$ is (syntactically)* minimal *if there is no other explanation $\mathcal{E}'$ of $\mathcal{P}$ that is smaller than $\mathcal{E}$.*

3 Computing Explanations

We now review first the complete MHS algorithm and then the faster but approximative MXP algorithm. The hybrid approach that tries to combine "the best of both worlds" is then introduced in Section 4.

[1] Note that before we compare two explanations $\mathcal{E}$ 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.

Algorithm 1 MHS($\mathcal{K}$,O,Abd)

```
Require: knowledge base $\mathcal{K}$, observation $O$, set of abducibles Abd
Ensure: set $\mathcal{S}_\mathcal{E}$ of all explanations of $\mathcal{P} = (\mathcal{K}, O)$ of the class Abd
 1: $M \leftarrow$ a model $M$ of $\mathcal{K} \cup \{\neg O\}$
 2: if $M =$ null then
 3:     return "nothing to explain"
 4: end if
 5: $T \leftarrow (V = \{r\}, E = \emptyset, L = \{r \mapsto \text{Abd}(M)\})$        ▷ Initialize HS-tree
 6: for each $\sigma \in L(r)$ create new $\sigma$-successor $n_\sigma$ of $r$
 7: $\mathcal{S}_\mathcal{E} \leftarrow \{\}$
 8: while there is next node $n$ in $T$ w.r.t. BFS do
 9:     if $n$ can be pruned then
10:         prune $n$
11:     else if there is a model $M$ of $\mathcal{K} \cup \{\neg O\} \cup H(n)$ then
12:         label $n$ by $L(n) \leftarrow \text{Abd}(M)$
13:     else
14:         $\mathcal{S}_\mathcal{E} \leftarrow \mathcal{S}_\mathcal{E} \cup \{H(n)\}$
15:     end if
16:     for each $\sigma \in L(n)$ create new $\sigma$-succesor $n_\sigma$ of $n$
17: end while
18: return $\mathcal{S}_\mathcal{E}$
```

3.1 Minimal Hitting Set

Thanks to Reiter [10] we know that computing all minimal explanations of $(\mathcal{K}, O)$ reduces to finding all minimal hitting sets of the set of models of $K \cup \{\neg O\}$ (modulo negation). To find some explanation, we may simply collect in $\mathcal{E}$ assertions that invalidate each such model. Then $\mathcal{K} \cup \mathcal{E} \cup \{\neg O\}$ is inconsistent and hence $\mathcal{K} \cup \mathcal{E} \models O$. The inverse reduction is also easily found.[2] Hence disregarding the time needed to find the models of $\mathcal{K} \cup \{\neg O\}$, finding all explanations is NP-complete [6].

However we want to draw explanations only from the set of abducibles Abd, as given in Definition 1. It is easily observed, that to find such explanations it suffices to exclude non-abducibles from the search. In addition, if some of the models contains no abducibles then there are no explanations:

Observation 1. *The minimal explanations of* $(\mathcal{K}, O)$ *on* Abd *directly corresponds to the minimal hitting sets of* $\{\text{Abd}(M) \mid M \models \mathcal{K} \cup \{\neg O\}\}$ *where* $\text{Abd}(M) = \{\phi \in \text{Abd} \mid M \not\models \phi\}$.

Observation 2. *If* $\text{Abd}(M) = \emptyset$ *for some* $M \models \mathcal{K} \cup \{\neg O\}$*, then* $(\mathcal{K}, O)$ *has no explanations on* Abd*.*

[2] For $S = \{S_1, \dots, S_n\}$ let $\mathcal{K} = \{A_1 \sqcup \dots \sqcup A_k \sqsubseteq S_i \mid S_i \in S, S_i = \{A_1, \dots, A_k\}\} \cup \{S_1 \sqcap \dots \sqcap S_n \sqsubseteq S\}$. Then the minimal explanations of $(\mathcal{K}, S(a))$ exactly correspond to the to minimal hitting sets of S.

The MHS algorithm, also proposed by Reiter [10], works by constructing a HS-tree. As we are limited by abducibles, in our case a HS tree $T = (V, E, L)$ is a labelled tree in which each node is labelled by $\mathrm{Abd}(M)$ for some model M of $\mathcal{K} \cup \{\neg O\}$ and whose edges are labelled by elements of the parent node's label. If a node $n_1 \in V$ has a successor $n_2 \in V$ such that $L(\langle n_1, n_2\rangle) = \sigma$ then n_2 is a σ-successor of n_1.

The HS-tree has the property that the node-label $L(n)$ and the union $H(n)$ of the edge-labels on the path from the root r to each node n are disjoint. For each node n such a label can be found as $\mathrm{Abd}(M)$ of any model of $\mathcal{K} \cup \{\neg O\} \cup H(n)$ which can be obtained by one call to an external DL reasoner. If no such model M exists then $H(n)$ corresponds to a hitting set. Note that if M exists but $\mathrm{Abd}(M) = \emptyset$, then in accord with Observation 2 $H(n)$ is not a hitting set.

In addition, *pruning* is applied during the HS-tree construction in order to eliminate non-minimal hitting sets. The most obvious case is when a node n already corresponds to a hitting set $H(n)$ and there is another node n' with $H(n) \subsetneq H(n')$. Any such n' can be pruned. Also if $H(n) = H(n')$, even if not yet a hitting set, one of the nodes is redundant and it can be pruned. Once completed, a pruned HS-tree (i.e., one on which all pruning conditions were applied) contains all minimal hitting sets [10]. In addition we also prune nodes which correspond to undesired explanations [8].

The resulting algorithm is given in Algorithm 1. This algorithm is sound and complete [10, 8, 9].

Theorem 3. *The MHS algorithm is sound and complete (i.e., it returns the set $\mathcal{S}_{\mathcal{E}}$ of all minimal desired explanations of $\mathcal{K}$ and O on* Abd.*)*

The fact that MHS explores the search space using breadth-first search (BFS) allows to limit the search for explanations by maximum size. The algorithm is still complete w.r.t. any given target size [9].

3.2 MergeXplain

Both QXP [5] and MXP [11] were originally designed to find minimal inconsistent subsets (dubbed *conflicts*) of an overconstrained knowledge base $\mathcal{K} = \mathcal{B} \cup \mathcal{C}$, where $\mathcal{B}$ is the consistent background theory and $\mathcal{C}$ is the "suspicious" part from which the conflicts are drawn. The algorithm is listed in Algorithm 2.

The essence of QXP is captured in the GetConflict function, which cleverly decomposes $\mathcal{C}$ by splitting it into smaller and smaller subsets in such a way that "on the way back" it is always able to reconstruct one minimal conflict, if it only exists. The auxiliary function isConsistent($\mathcal{K}$) encapsulates calls to an external reasoner; it returns true if $\mathcal{K}$ is consistent and false otherwise.

However, QXP only finds one conflict. Its extension MXP, captured in the FindConflicts function, finds as many conflicts as possible that can be reconstructed from a one way in which $\mathcal{C}$ can be split. During the reconstruction, MXP relies on GetConflict to recover some of the conflicts that would be lost due to splitting, which also ensures that it also keeps the important property that if at least one conflict exists, at least one is also found.

Algorithm 2 MXP($\mathcal{B}$,$\mathcal{C}$)

Input: background theory $\mathcal{B}$, set of possibly faulty constraints $\mathcal{C}$
Output: a set of minimal conflicts Γ

```
1: if $\neg$isConsistent($\mathcal{B}$) then
2:     return "no explanation"
3: else if isConsistent($\mathcal{B} \cup \mathcal{C}$) then
4:     return $\emptyset$
5: end if
6: $\langle \_, \Gamma \rangle \leftarrow$ FindConflicts($\mathcal{B}, \mathcal{C}$)
7: return $\Gamma$

8: function FindConflicts($\mathcal{B}, \mathcal{C}$) returns a tuple $\langle \mathcal{C}', \Gamma \rangle$
9:     if isConsistent($\mathcal{B} \cup \mathcal{C}$) then
10:        return $\langle \mathcal{C}, \emptyset \rangle$
11:    else if $|\mathcal{C}| = 1$ then
12:        return $\langle \emptyset, \{\mathcal{C}\} \rangle$
13:    end if
14:    Split $\mathcal{C}$ into disjoint, non-empty sets $\mathcal{C}_1$ and $\mathcal{C}_2$
15:    $\langle \mathcal{C}'_1, \Gamma_1 \rangle \leftarrow$ FindConflicts($\mathcal{B}, \mathcal{C}_1$)
16:    $\langle \mathcal{C}'_2, \Gamma_2 \rangle \leftarrow$ FindConflicts($\mathcal{B}, \mathcal{C}_2$)
17:    $\Gamma \leftarrow \Gamma_1 \cup \Gamma_2$
18:    while $\neg$isConsistent($\mathcal{C}'_1 \cup \mathcal{C}'_2 \cup \mathcal{B}$) do
19:        $X \leftarrow$ GetConflict($\mathcal{B} \cup \mathcal{C}'_2, \mathcal{C}'_2, \mathcal{C}'_1$)
20:        $\gamma \leftarrow X \cup$ GetConflict($\mathcal{B} \cup X, X, \mathcal{C}'_2$)
21:        $\mathcal{C}'_1 \leftarrow \mathcal{C}'_1 \setminus \{\sigma\}$ where $\sigma \in X$
22:        $\Gamma \leftarrow \Gamma \cup \{\gamma\}$
23:    end while
24:    return $\langle \mathcal{C}'_1 \cup \mathcal{C}'_2, \Gamma \rangle$
25: end function

26: function GetConflict($\mathcal{B}, D, \mathcal{C}$)
27:    if $D \neq \emptyset \wedge \neg$isConsistent($\mathcal{B}$) then
28:        return $\emptyset$
29:    else if $|\mathcal{C}| = 1$ then
30:        return $\mathcal{C}$
31:    end if
32:    Split $\mathcal{C}$ into disjoint, non-empty sets $\mathcal{C}_1$ and $\mathcal{C}_2$
33:    $D_2 \leftarrow$ GetConflict($\mathcal{B} \cup \mathcal{C}_1, \mathcal{C}_1, \mathcal{C}_2$)
34:    $D_1 \leftarrow$ GetConflict($\mathcal{B} \cup D_2, D_2, \mathcal{C}_1$)
35:    return $D_1 \cup D_2$
36: end function
```

This can be immediately adopted for ABox abduction: in order to find explanations for an abduction problem $\mathcal{P} = (\mathcal{K}, O)$ on Abd one needs to call MXP($\mathcal{K} \cup \{\neg O\}$, Abd). This observation allows us to adopt the following result from Shchekotykhin et al. [11]:

Theorem 4. *Assume an ABox abduction problem $\mathcal{P} = (\mathcal{K}, O)$ and a set of abducibles* Abd*. If there is at least one explanation $\gamma \subseteq$ Abd of $\mathcal{P}$ then calling* $\mathrm{MXP}(\mathcal{K} \cup \{\neg O\}, \mathrm{Abd})$ *returns a nonempty set Γ of explanations of $\mathcal{P}$.*

In fact, MXP is thorough in its decomposition of $\mathcal{C}$, which is broken to smaller and smaller subsets until they are consistent with $\mathcal{B}$ or until only sets of size 1 remain. This directly implies that all conflicts of size 1 will always be found and returned by a single run of MXP. This observation will prove to be useful for our hybrid algorithm.

Observation 5. *Given an ABox abduction problem $\mathcal{P} = (\mathcal{K}, O)$, set of abducibles* Abd*, and any $\gamma \subseteq$ Abd s.t. $|\gamma| = 1$, if $\mathcal{K} \cup \gamma \models O$ then $\gamma \in \mathrm{MXP}(\mathcal{K} \cup \{\neg O\}, \mathrm{Abd})$.*

Thus MXP is sound, and if an explanation exists, it always finds at least one (and it finds all of size one). However MXP is still approximative, i.e., it is not complete. Some explanations may be lost, especially in case of abduction problems with multiple partially overlapping explanations.

Example 1. Let $\mathcal{K} = \{A \sqcap B \sqsubseteq D, A \sqcap C \sqsubseteq D\}$ and let $O = D(a)$. Let us ignore negated ABox expressions and start with $\mathrm{Abd} = \{A(a), B(a), C(a)\}$. There are two minimal explanations of $\mathcal{P} = (\mathcal{K}, O)$: $\{A(a), B(a)\}$, and $\{A(a), C(a)\}$. Calling $\mathrm{MXP}(\mathcal{K} \cup \{\neg O\}, \mathrm{Abd})$, it passes the initial tests and calls FindConflicts$(\mathcal{K} \cup \{\neg O\}, \mathrm{Abd})$.

FindConflicts needs to decide how to split $\mathcal{C} = \mathrm{Abd}$ into $\mathcal{C}_1$ and $\mathcal{C}_2$. Let us assume the split was $\mathcal{C}_1 = \{A(a)\}$ and $\mathcal{C}_2 = \{B(a), C(a)\}$. Since both $\mathcal{C}_1$ and $\mathcal{C}_2$ are now conflict-free w.r.t. $\mathcal{K} \cup \{\neg O\}$, the two consecutive recursive calls return $\langle \mathcal{C}'_1, \emptyset \rangle$ and $\langle \mathcal{C}'_2, \emptyset \rangle$ where $\mathcal{C}'_1 = \{A(a)\}$ and $\mathcal{C}'_2 = \{B(a), C(a)\}$.

In the while loop, GetConflict$(\mathcal{K} \cup \{\neg O\} \cup \{B(a), C(a)\}, \{B(a), C(a)\}, \{A(a)\})$ returns $X = \{A(a)\}$ while GetConflict$(\mathcal{K} \cup \{\neg O\} \cup \{A(a)\}, \{A(a)\}, \{B(a), C(a)\})$ returns $B(a)$, and hence the first conflict $\gamma = \{A(a), B(a)\}$ is found and added into Γ.

However, consecutively $A(a)$ is removed from $\mathcal{C}'_1$ leaving it empty, and thus the other conflict is not found and $\Gamma = \{\{A(a), B(a)\}\}$ is returned.

4 Combined MHS-MXP Algorithm

The hybrid algorithm is based on the observation that running MXP multiple times, each time on a slightly modified input, results in consecutive extension of the overall conflicts that are found. We will show, that the construction of a HS-tree may serve to guide this iterations in such a way, that completeness is achieved.

The combined MHS-MXP algorithm, listed as Algorithm 4, therefore constructs the HS-tree as usual, but in each node n, instead of simply retrieving one model of $\mathcal{K} \cup \{\neg O\} \cup H(n)$, it calls FindConflicts.

It starts by checking the consistency of $\mathcal{K} \cup \{\neg O\}$. We use a modified isConsistent function which stores all models in the model cache Mod. This is

Algorithm 3 MHS-MXP($\mathcal{K}$,O,Abd)

```
Require: knowledge base $\mathcal{K}$, observation $O$, set of abducibles Abd
Ensure: set $\mathcal{S}_\mathcal{E}$ of all explanations of $\mathcal{P} = (\mathcal{K}, O)$ of the class Abd
 1: Con ← {}                                                    ▷ Set of conflicts
 2: Mod ← {}                                              ▷ Set of cached models
 3: if ¬isConsistent($\mathcal{K} \cup \{\neg O\}$) then
 4:     return "nothing to explain"
 5: else if Abd($M$) = ∅ where Mod = {$M$} then
 6:     return $\mathcal{S}_\mathcal{E} = \emptyset$
 7: end if
 8: $T \leftarrow (V = \{r\}, E = \emptyset, L = \emptyset)$                        ▷ Init. HS-Tree
 9: while there is next node $n$ in $T$ w.r.t. BFS do
10:     if $\gamma \not\subseteq H(n)$ for all $\gamma \in$ Con then        ▷ Otherwise $n$ is pruned
11:         $\langle \_, \mathit{\Gamma} \rangle \leftarrow$ FindConflicts($\mathcal{K} \cup \{\neg O\} \cup H(n)$, Abd $\setminus H(n)$)
12:         Con ← Con $\cup \{H(n) \cup \gamma \mid \gamma \in \mathit{\Gamma}\}$
13:         if $\mathit{\Gamma} \neq \emptyset$ and $\left|\text{Abd} \setminus \bigcup\{\gamma \in \mathit{\Gamma} \mid |\gamma| = 1\}\right| > 1$ then
                                                    ▷ HS-tree is extended under $n$
14:             $L(n) \leftarrow$ Abd($M$) $\setminus H(n)$ for some $M \in$ Mod s.t. $M \models H(n)$
15:             for each $\sigma \in L(n)$ create new $\sigma$-successor $n_\sigma$ of $n$
16:         end if
17:     end if
18: end while
19: return $\mathcal{S}_\mathcal{E} \leftarrow \{\gamma \in$ Con $\mid \gamma$ is desired$\}$
20: function isConsistent($\mathcal{K}$)
21:     if there is $M \models \mathcal{K}$ then
22:         Mod ← Mod $\cup \{M\}$
23:         return true
24:     else
25:         return false
26:     end if
27: end function
```

because it will be important to remember the models, even if they are found inside the calls to FindConflicts.

Then the main loop is initiated. For the root node r, FindConflicts is simply called passing $\mathcal{K} \cup \{\neg O\}$ as the background theory and Abd as the set of conflicts (as $H(n) = \emptyset$ at this point). The obtained conflicts $\mathit{\Gamma}$ are stored in Con. We then verify if all conflicts were already found or the search needs to continue (line 13). From Theorem 4 we know that if no conflicts were found in $\mathit{\Gamma}$, it means there are no conflicts whatsoever. Also from Observation 5 we know that all conflicts of size 1 were already found in $\mathit{\Gamma}$, which means that any minimal conflicts may only possibly remain if at least two abducibles are not present (as singletons) in $\mathit{\Gamma}$. If neither of these is true then the HS-tree is extended under r using the model M that was previously found and stored in Mod.

When consecutively any other node $n \neq r$ is visited by the main loop, the node is immediately pruned, if $H(n)$ contains any of the conflicts already stored in Con. If not, we now want to use MXP with the goal to explore as much as possible of that part of the space of explanations that extends $H(n)$. Therefore we call FindConflicts passing $\mathcal{K} \cup \{\neg O\} \cup H(n)$ as the background theory and

Abd $\setminus H(n)$ as the set of conflicts. If we are lucky, we might cut off this branch completely (due to Theorem 4 and Observation 5, line 13).

Now in order to extend the HS-tree under n we need a model of $\mathcal{K} \cup \{\neg O\} \cup H(n)$. However, we do not need to run another consistency check here, as by design of out algorithm at this point such model is already cached in Mod.

Observation 6. *For each node n of the HS-tree visited by the main loop of* MHS-MXP*($\mathcal{K}, O, \mathrm{Abd}$) either $H(n) \in \Gamma$ or $\mathcal{K} \cup \{\neg O\} \cup H(n)$ is consistent and at least for one $M \in \mathrm{Mod}$, $M \models \mathcal{K} \cup \{\neg O\} \cup H(n)$.*

This holds due to FindConflicts was previously called in the parent node n' of n, and from Observation 5 we know that during that call all possible inconsistent extensions of $H(n')$ of size 1 were added to Γ. Hence if n was not pruned on line 10, $H(n) = H(n') \cup \{\sigma\}$ must be consistent with $\mathcal{K} \cup \{\neg O\}$. Moreover, since FindConflicts did not prove $\{\sigma\}$ being a conflict for $\mathcal{K} \cup \{\neg O\} \cup H(n')$, at some point it must have checked the consistency of $\mathcal{K} \cup \{\neg O\} \cup H(n')$ together with $\{\sigma\}$ or some superset thereof with a positive result, and at that point the respective model was added to Mod.

5 Implementation

We provide a preliminary implementation in Java. It is based on previous implementations of MHS and MXP [3] which are merged into the hybrid algorithm.

We rely on OWL API [4] to call an external DL reasoners as a black box; Pellet, HermiT or JFact are supported. As OWL API does not have a dedicated method to extract models from the DL reasoner we extract models by iterating through abducibles and checking if they are consistent with a current state of the reasoner.

The current version has several limitations. Only single atomic ABox observation of the form $A(a)$ is supported. The set of abducibles can not be specified yet, instead, all possible ABox assertions of form $A(a)$ are included in the set, for any individual a occurring in the observation $\mathcal{O}$ and any atomic concept A occurring in $\mathcal{K}$. Further extension of the implementation and its optimization is ongoing work. The implementation is available online.[3]

6 Conclusions

MHS-MXP is a new hybrid ABox abduction algorithm for DL. The algorithm builds on the divide and conquer strategy employed by the incomplete MXP and it retains completeness by repetitive runs of MXP while tracking the search space exploration using the HS-tree construction.

The combined MHS-MXP search strategy may likely have an advantage on a certain class of inputs, especially those featuring (even a smaller number of)

[3] `http://dai.fmph.uniba.sk/~pukancova/mhs-mxp/`

explanations of greater size. Exploring such search space with MHS, which relies on breadth-first search, is rather inefficient.

In the future we would like to characterize the inputs for which MHS-MXP is suitable and also to test this characterization also an empirical evaluation. We would also like to further develop our implementation and explore various possible optimization techniques. For instance, more aggressive model caching can be tried: one does not have to call MXP in every node if there is already a suitable model in cache that can be used to label the node. This however invalidates Observation 6 and additional consistency checks will be required. In addition, conflicts that are cached in Con may be used to save consistency checks inside the MXP calls. It would be interesting to implement and empirically test such possible optimization techniques.

Acknowledgments. This work was supported by the Slovak Research and Development Agency under the Contract no. APVV-19-0220 (ORBIS), by the Slovak VEGA agency under Contract no. 1/0778/18 (KATO), and by the EU H2020 programme under Contract no. 952215 (TAILOR).

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. 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, US. CEUR-WS, vol. 216 (2006)
3. Fabianová, K., Pukancová, J., Homola, M.: Comparing ABox abduction based on minimal hitting set and MergeXplain. In: Proceedings of the 32nd International Workshop on Description Logics, Oslo, Norway, June 18-21, 2019. CEUR-WS, vol. 2373 (2019)
4. Horridge, M., Bechhofer, S.: The OWL API: A java API for OWL ontologies. Semantic Web **2**(1), 11–21 (2011)
5. Junker, U.: QuickXplain: Preferred explanations and relaxations for over-constrained problems. In: Proceedings of the Nineteenth National Conference on Artificial Intelligence, Sixteenth Conference on Innovative Applications of Artificial Intelligence, San Jose, California, US. pp. 167–172. AAAI Press (2004)
6. Karp, R.M.: Reducibility among combinatorial problems. In: Proceedings of a symposium on the Complexity of Computer Computations, March 20–22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York. pp. 85–103 (1972)
7. Peirce, C.S.: Illustrations of the logic of science VI: Deduction, induction, and hypothesis. Popular Science Monthly **13**, 470–482 (1878)
8. 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)
9. Pukancová, J., Homola, M.: ABox abduction for description logics: The case of multiple observations. In: Proceedings of the 31st International Workshop on Description Logics, Tempe, Arizona, US. CEUR-WS, vol. 2211 (2018)

10. Reiter, R.: A theory of diagnosis from first principles. Artificial intelligence **32**(1), 57–95 (1987)
11. Shchekotykhin, K.M., Jannach, D., Schmitz, T.: MergeXplain: Fast computation of multiple conflicts for diagnosis. In: Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina. AAAI Press (2015)