Hybrid MHS-MXP ABox Abduction Solver: First Empirical Results

Martin Homola, Júlia Pukancová, Iveta Balintová and Janka Boborová

Comenius University in Bratislava, Mlynská dolina, 842 41 Bratislava, Slovakia

Abstract

MHS-MXP is a hybrid algorithm which adopts the divide-and-conquer strategy of fast but incomplete MergeXplain (MXP) and combines it with Minimal Hitting Set (MHS) to regain completeness. We describe a class of inputs on which MHS-MXP has an advantage. We provide an experimental implementation which enables us to perform first preliminary empirical evaluation on this class of inputs.

Keywords

ABox abduction, description logics, ontologies

1. Introduction

In the context of DL, *ABox abduction* [1] assumes a DL KB $\mathcal{K}$ and an extensional observation O (in form of an ABox assertion). Explanations (also extensional) are sets of ABox assertions $\mathcal{E}$ such that $\mathcal{K}$ together with $\mathcal{E}$ entails O. The MHS algorithm [2] is the classic method to find all explanations of an ABox abduction problem. MHS is mostly applied relying on an external DL reasoner for consistency checking as a *black box*.

MHS is complete. It searches through the space of all possible explanations, from the smallest (in terms of cardinality) towards the largest. 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. Notably, the worst-case complexity of the underlying problems is also hard. The MHS problem is NP-complete [3] and consistency checking of DL KBs repeatedly called by MHS depends on the particular DL, but for many DLs it may be exponential or worse.

Alternative strategies were explored. QuickXplain (QXP) [4], and more recently its extension MergeXplain [5] employ the *divide and conquer* strategy which allows to find one (QXP) or even multiple explanations (MXP) very efficiently. On the other hand, these methods are not complete, i.e., there is no warranty that all explanations will be found.

However, when MXP is run repeatedly, with slightly modified inputs, it divides the search space differently and it may return a different set of explanations than in the previous runs. Based on this key observation, we have proposed a combined algorithm, dubbed MHS-MXP [6],

that iterates runs of MXP and it uses MHS to steer the search space exploration in such a way that completeness is retained.

In the current report we study relevant properties of MHS-MXP that help not only to establish its correctness, but also hint the class of inputs on which it may have an advantage over MHS. We conjecture, that one such class of favourable inputs are those with a smaller number of shorter explanations. We provide a preliminary implementation and show first empirical results that supports the above conjecture.

The main advantage of our work compared to some other promising approaches in DL abduction [7, 8] lies in being a truly black box approach and thus it may be immediately paired with any DL for which a suitable reasoner is available (provided that a model can be extracted from it, see below). In fact, the approach is not limited to DLs and it can be applied in any case in which MHS is applicable. Different approaches to DL abduction, on the other hand, have other advantages, e.g. they are able to provide for richer abducibles than our approach.

2. Preliminaries

We assume familiarity with the basics of DL [9], including vocabulary consisting of individuals $N_I = \{a, b, ...\}$, roles $N_R = \{P, Q, ...\}$, and atomic concepts $N_C = \{A, B, ...\}$; complex concepts $C, D, ...$ built by constructors (e.g. $\neg$, $\sqcap$, $\sqcup$, $\exists$, $\forall$, in case of $\mathcal{ALC}$ [10]); a KB $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$ composed of a TBox $\mathcal{T}$ (with subsumption axioms of the form $C \sqsubseteq D$) and an ABox $\mathcal{A}$ (with concept assertions of the form $C(a)$ and (possibly negated [11]) role assertions of the form $R(a)$ and $\neg R(a)$). We also remind about the semantics that relies on models M of a KB $\mathcal{K}$, that satisfy all axioms or assertions ϕ in $\mathcal{K}$ ($M \vDash \phi$); and the reasoning tasks of checking the consistency of $\mathcal{K}$ (if it has a model) and entailment ($\mathcal{K} \vDash \phi$ if $M \vDash \phi$ for all its models M).

In *ABox abduction* [1], 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 $\mathcal{K} \cup \mathcal{E} \vDash O$. Explanations are drawn from some set of *abducibles* 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* $\mathcal{E} \subseteq$ Abd *such that* $\mathcal{K} \cup \mathcal{E} \vDash O$.

We limit the explanations to atomic and negated atomic concept and role assertions; hence Abd $\subseteq \{A(a), \neg A(a) \mid A \in N_C, \ a \in N_I\} \cup \{R(a, b), \neg R(a, b) \mid R \in N_R, \ a, b \in N_I\}$. Note that we do not limit the observations, apart from allowing only one (possibly complex) ABox assertion.

According to Elsenbroich et al. [1] 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* ($\mathcal{E} \nvDash O$); and (c) *explanatory* ($\mathcal{K} \nvDash O$). Explanations that 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 explanations* $\mathcal{E}$ *and* $\mathcal{E}'$ *of* $\mathcal{P}$, $\mathcal{E}$ *is (syntactically)* smaller *than* $\mathcal{E}'$ *if* $\mathcal{E} \subseteq \mathcal{E}'$. *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* $\mathcal{E}$.

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

Input: Knowledge base $\mathcal{K}$, observation O, abducibles Abd
Output: $\mathcal{S}_{\mathcal{E}}$ all explanations of $\mathcal{P} = (\mathcal{K}, O)$ w.r.t. Abd

```
 1: M ← a model M of 𝒦 ∪ {¬O}            10:        prune n
 2: if M = null then                     11:     else if exists model M of 𝒦 ∪ {¬O} ∪ H(n) then
 3:     return "nothing to explain"      12:        label n by L(n) ← Abd(M)
 4: end if                               13:     else if H(n) is desired then
 5: T ← (V = {r}, E = ∅, L = {r ↦ Abd(M)})  14:        𝒮_ℰ ← 𝒮_ℰ ∪ {H(n)}
 6: for each σ ∈ L(r) create new σ-successor n_σ of r  15:     end if
 7: 𝒮_ℰ ← {}                              16:     for each σ ∈ L(n) create new σ-successor n_σ of n
 8: while exists next node n in T w.r.t. BFS do  17: end while
 9:     if n can be pruned then          18: return 𝒮_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.

3.1. Minimal Hitting Set

Adopting the well-known result of Reiter [2], computing all minimal explanations of $(\mathcal{K}, O)$ reduces to finding all minimal hitting sets of the set of models of $K \cup \{\neg O\}$ in the following sense. Also, if some of the models contain 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* $\{\text{Abd}(M) \mid M \vDash \mathcal{K} \cup \{\neg O\}\}$ *where* $\text{Abd}(M) = \{\phi \in \text{Abd} \mid M \nvDash \phi\}$.

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

The MHS algorithm (Algorithm 1) works by constructing an HS-tree $T = (V, E, L)$ in which each node is labelled by $\text{Abd}(M)$ for some model M of $\mathcal{K} \cup \{\neg O\}$ and whose edges are labelled by elements of the parent'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. Each label $L(n)$ can be found as $\text{Abd}(M)$ of any model of $\mathcal{K} \cup \{\neg O\} \cup H(n)$, 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 $\text{Abd}(M) = \emptyset$, then in accord with Observation 2 $H(n)$ cannot be extended to a hitting set.

Subset pruning eliminates non-minimal hitting sets: given a hitting set $H(n)$, nodes n' with $H(n) \subseteq H(n')$ are pruned. Equal-paths pruning prunes also nodes n' with $H(n) = H(n')$, even if $H(n)$ is not a hitting set. Once completed, a pruned HS-tree contains all minimal hitting sets [2]. MHS is sound and complete [2, 12].

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** $\neg\text{ISCONSISTENT}(\mathcal{B})$ **then**
2:     **return** "no explanation"
3: **else if** $\text{ISCONSISTENT}(\mathcal{B} \cup \mathcal{C})$ **then**
4:     **return** $\varnothing$
5: **end if**
6: $\langle \_, \Gamma \rangle \leftarrow \text{FINDCONFLICTS}(\mathcal{B}, \mathcal{C})$
7: **return** Γ

8: **function** $\text{FINDCONFLICTS}(\mathcal{B}, \mathcal{C})$
9:     **if** $\text{ISCONSISTENT}(\mathcal{B} \cup \mathcal{C})$ **then**
10:       **return** $\langle \mathcal{C}, \varnothing \rangle$
11:     **else if** $|\mathcal{C}| = 1$ **then**
12:       **return** $\langle \varnothing, \{\mathcal{C}\} \rangle$
13:     **end if**
14:     Split $\mathcal{C}$ into disjoint, non-empty sets $\mathcal{C}_1$ and $\mathcal{C}_2$
15:     $\langle \mathcal{C}_1', \Gamma_1 \rangle \leftarrow \text{FINDCONFLICTS}(\mathcal{B}, \mathcal{C}_1)$
16:     $\langle \mathcal{C}_2', \Gamma_2 \rangle \leftarrow \text{FINDCONFLICTS}(\mathcal{B}, \mathcal{C}_2)$
17:     $\Gamma \leftarrow \Gamma_1 \cup \Gamma_2$
18:     **while** $\neg\text{ISCONSISTENT}(\mathcal{C}_1' \cup \mathcal{C}_2' \cup \mathcal{B})$ **do**
19:       $X \leftarrow \text{GETCONFLICT}(\mathcal{B} \cup \mathcal{C}_2', \mathcal{C}_2', \mathcal{C}_1')$
20:       $\gamma \leftarrow X \cup \text{GETCONFLICT}(\mathcal{B} \cup X, X, \mathcal{C}_2')$
21:       $\mathcal{C}_1' \leftarrow \mathcal{C}_1' \backslash \{\sigma\}$ 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** $\text{GETCONFLICT}(\mathcal{B}, D, \mathcal{C})$
27:     **if** $D \neq \varnothing \wedge \neg\text{ISCONSISTENT}(\mathcal{B})$ **then**
28:       **return** $\varnothing$
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:     $D_2 \leftarrow \text{GETCONFLICT}(\mathcal{B} \cup \mathcal{C}_1, \mathcal{C}_1, \mathcal{C}_2)$
34:     $D_1 \leftarrow \text{GETCONFLICT}(\mathcal{B} \cup D_2, D_2, \mathcal{C}_1)$
35:     **return** $D_1 \cup D_2$
36: **end function**

Theorem 1. *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 [12].

3.2. MergeXplain

Both QXP [4] and MXP [5] were originally designed to find minimal inconsistent subsets (dubbed *conflicts*) of an over-constrained 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 such that 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.

Thus, if we just wanted to find one minimal explanation of an ABox abduction problem, adopting a result of Junker [4] we could use GETCONFLICT in the following way.

Theorem 2. *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 GETCONFLICT($\mathcal{K} \cup \{\neg O\}, \mathcal{K} \cup \{\neg O\},$ Abd) returns some minimal explanation $\delta \subseteq$ Abd of $\mathcal{P}$.*

MXP, captured in the FindConflicts function, finds as many conflicts as it is possible to reconstruct from one way in which $\mathscr{C}$ can be split. MXP relies on GetConflict to recover some of the conflicts that would be lost due to splitting. This ensures that it keeps the important property of QXP that at least one minimal is found in each run, if it exists.

This approach can be immediately adopted for ABox abduction: in order to find explanations for an abduction problem $\mathscr{P} = (\mathscr{K}, O)$ on Abd one needs to call MXP$(\mathscr{K} \cup \{\neg O\}, \text{Abd})$. This observation allows us to adopt the following result from Shchekotykhin et al. [5]:

Theorem 3. *Assume an ABox abduction problem $\mathscr{P} = (\mathscr{K}, O)$ and a set of abducibles* Abd. *If there is at least one explanation $\gamma \subseteq$ Abd of $\mathscr{P}$ then calling MXP$(\mathscr{K} \cup \{\neg O\}, \text{Abd})$ returns a nonempty set Γ of minimal explanations of $\mathscr{P}$.*

In fact, MXP is thorough in its decomposition of $\mathscr{C}$, which is broken to smaller and smaller subsets until they are consistent with $\mathscr{B}$ 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 3. *Given an ABox abduction problem $\mathscr{P} = (\mathscr{K}, O)$, a set of abducibles* Abd, *and any $\gamma \subseteq$ Abd s.t. $|\gamma| = 1$, if $\mathscr{K} \cup \gamma \vDash O$ then $\gamma \in$ MXP$(\mathscr{K} \cup \{\neg O\}, \text{Abd})$.*

Thus MXP is sound and it always finds at least one minimal explanation (Theorem 3), and it finds all explanations of size one (Observation 3). Still, MXP is not complete. Some explanations may be lost, especially in cases with multiple partially overlapping explanations.

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

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

In the while loop, GetConflict$(\mathscr{K} \cup \{\neg O\} \cup \{B(a), C(a)\}, \{B(a), C(a)\}, \{A(a)\})$ returns $X = \{A(a)\}$ while GetConflict$(\mathscr{K} \cup \{\neg O\} \cup \{A(a)\}, \{A(a)\}, \{B(a), C(a)\})$ 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 $\mathscr{C}_1'$ leaving it empty, and thus the other conflict is not found and $\Gamma = \{\{A(a), B(a)\}\}$ is returned.

Finally, not only is MXP able to find all explanations of size 1, but it also has the property that if no longer explanations are returned in a given run then in fact this is because there are none. So in such a case we are sure that we have found all explanations whatsoever in a single run and we do not have to search any further.

Theorem 4. *Given an ABox abduction problem $\mathscr{P} = (\mathscr{K}, O)$, a set of abducibles* Abd, *let $\Gamma = $ MXP$(\mathscr{K} \cup \{\neg O\}, \text{Abd})$. If there is no $\gamma \in \Gamma$ s.t. $|\gamma| > 1$, then for all minimal $\delta \subseteq$ Abd s.t. $\mathscr{K} \cup \delta \vDash O$ we have that $\delta \in \Gamma$.*

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

Input: knowledge base $\mathcal{K}$, observation O, set of abducibles Abd
Output: set $\mathcal{S}_{\mathscr{E}}$ of all explanations of $\mathscr{P} = (\mathcal{K}, O)$ of the class Abd

```
 1: Con ← {}                                                    ▷ Set of conflicts
 2: Mod ← {}                                                    ▷ Set of cached models
 3: if ¬isConsistent(𝒦 ∪ {¬O}) then
 4:     return "nothing to explain"
 5: else if Abd(M) = ∅ where Mod = {M} then
 6:     return 𝒮_ℰ = ∅
 7: end if
 8: T ← (V = {r}, E = ∅, L = ∅)                                ▷ Init. HS-Tree
 9: while there is next node n in T w.r.t. BFS do
10:     if γ ⊄ H(n) for all γ ∈ Con then                       ▷ Otherwise n is pruned
11:         ⟨_, Γ⟩ ← FindConflicts(𝒦 ∪ {¬O} ∪ H(n), Abd ∖ H(n))
12:         Con ← Con ∪ {H(n) ∪ γ ∣ γ ∈ Γ}
13:         if Γ ≠ ∅ and ∃γ ∈ Γ : |γ| > 1 then                 ▷ HS-tree is extended under n
14:             L(n) ← Abd(M) ∖ H(n) for some M ∈ Mod s.t. M ⊨ H(n)
15:             for each σ ∈ L(n) create new σ-successor n_σ of n
16:         end if
17:     end if
18: end while
19: return 𝒮_ℰ ← {γ ∈ Con ∣ γ is desired}
20: function isConsistent(𝒦)
21:     if there is M ⊨ 𝒦 then
22:         Mod ← Mod ∪ {M}
23:         return true
24:     else
25:         return false
26:     end if
27: end function
```

4. Combined MHS-MXP Algorithm

The idea to use MXP to find all explanations is based on the observation that running it multiple times in a row may result in a consecutive extension of the overall set of conflicts found so far. A naïve, and possibly to a large extent successful idea, would be to randomize the set splits MXP does in each recursive call. We would likely find different conflicts each time, however it would not be clear when to stop.

We will instead explore a hybrid approach, and we will show that by modifying MXP's inputs in its consecutive iterations, the search space exploration can be guided by the construction of an HS-tree from the obtained outputs, and thus completeness will be achieved.

The combined MHS-MXP algorithm, listed as Algorithm 3, therefore constructs the HS-tree T as usual, but in each node n, instead of simply retrieving one model of $\mathcal{K} \cup \{\neg O\} \cup H(n)$, it launches MXP by calling 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. The stored models are later used to construct the HS-tree and label its nodes. For this reason we also assume that FindConflicts uses this modified isConsistent function.

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 Γ are stored in Con. We then verify if all conflicts were already found or if the search needs to go on (line 13). From Theorem 3 we know that if no conflicts were returned in Γ, it means there are no conflicts whatsoever. Also from Observation 3 we know that all conflicts of size 1 are always found and returned in Γ. Finally, by Theorem 4 we have that if any larger conflicts remain, at least one is also present in Γ. Hence, if there is no $\gamma \in \Gamma$ with $|\gamma| > 1$ there are no other explanations to be found and the search can be terminated.

If however at least one such γ was returned in Γ 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, we first check if it can be pruned: it is 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 in line 13, that is, if no extension of $H(n)$ of size greater than 1 is found (by Theorem 4). Otherwise we extend the HS-tree below n.

To be able to do that, 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 our algorithm at this point such a model is already cached in Mod.

Observation 4. *For each node n of the HS-tree visited by the main loop of MHS-MXP($\mathcal{K}, O,$ Abd) either $H(n) \in$ Con or $\mathcal{K} \cup \{\neg O\} \cup H(n)$ is consistent and at least for one $M \in$ Mod, $M \vDash \mathcal{K} \cup \{\neg O\} \cup H(n)$.*

For the root r this trivially holds as it was only visited (as the first node in the main loop) if the consistency check in line 3 was positive, i.e. a suitable model was found and cached.

For any other node n, this holds due to FindConflicts was previously called in the parent node n' of n, and from Observation 3 we know that during that call all possible inconsistent extensions of $H(n')$ of size 1 were added to Con. So, if n was not pruned in 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.

Finally, by the time a complete HS-tree is constructed, all explanations are accumulated in Con. However, due to calls to FindConflicts where (nonempty) $H(n)$ was passed together with $\mathcal{K}$ as the consistent background theory, some of these conflicts in Con may be non-minimal and they have to be filtered out. At this point we also filter out any other undesired explanations. Then the remaining minimal and desired explanations are returned as $\mathcal{S}_{\mathcal{G}}$.

Theorem 5. *The MHS-MXP algorithm is sound and complete (i.e., it returns the set $\mathcal{S}_{\mathcal{G}}$ of all minimal desired explanations of $\mathcal{K}$ and O on Abd).*

5. Advantages and Limitations

First of all, it is immediately apparent from the properties of MXP alone, that our approach absolutely crushes MHS in cases when all explanations are of size one. By Observation 3 and Theorem 4 the search may immediately stop after one call to MXP in the root node of the HS-tree. Without this "look ahead" capability provided to the hybrid algorithm by MXP, pure MHS has no way of knowing it could stop and has to generate the HS-tree completely.

Let us now consider some cases when bigger explanations come into play.

Example 2. *Let $\mathscr{K} = \{A \sqcap B \sqsubseteq F, C \sqcap D(a), E(b)\}$, let $O = F(a)$ and let Abd = $\{A(a), B(a), C(a), D(a)\}$. There is exactly one explanation of $(\mathscr{K}, O)$, $\mathscr{E}_1 = \{A(a), B(a)\}$.*

If we run MHS-MXP, it first checks $\mathscr{K} \cup \{\neg F(a)\}$ for consistency and it obtains a model M thereof, say one with $\mathrm{Abd}(M) = \{A(a), C(a)\}$.

The call to FindConflicts in the root does not allow to terminate the search, since $\mathscr{E}_1$ was returned and $|\mathscr{E}_1| > 1$. Therefore n_1 and n_2 are added to the HS-tree with $H(n_1) = \{A(a)\}$ and $H(n_2) = \{C(a)\}$.

When FindConflicts is called in n_1, it returns one conflict $\{B(a)\}$ which together with $H(n_1)$ makes up for the explanation $\mathscr{E}_1$. What is more, this branch is consecutively cut off, as no greater conflicts were found. Notably, further exploration of branches extending $H(n_1)$ with $C(a)$ and $D(a)$ is avoided (in comparison with MHS).

Then FindConflicts is called in n_2 returning one conflict $\{A(a), B(a)\}$, corresponding to the non-minimal explanation $\{C(a), A(a), B(a)\}$. However, since there was a conflict extending $H(n_1)$ by a size greater than one, we may not terminate yet and must explore this branch in the HS-tree further, until only extensions of size one are returned by MXP in each path.

The case presented in Example 2 and similar cases with a small overall number of explanations can still be handled rather efficiently, compared to MHS. It can be observed that a significant part of the search space is cut off. However consider the following modification of the inputs.

Example 3. *Given the $\mathscr{K}$ and O as in Example 2, let Abd = $\{A(a), B(a), C(a), D(a), E(a), \neg E(a)\}$. The abduction problem $(\mathscr{K}, O)$ now has two explanations $\mathscr{E}_1$ and $\mathscr{E}_2 = \{E(a), \neg E(a)\}$, where the second one is undesired (inconsistent). When FindConflicts is called in the root node r it returns $\{\{A(a), B(a)\}, \{E(a), \neg E(a)\}\}$. W.l.o.g. we may assume that the same model M was used to label r and that $M \nvDash E(a)$. This time $\mathrm{Abd}(M) = \{A(a), C(a), E(a)\}$ and in addition to n_1 and n_2 as above also n_3 is generated with $H(n_3) = \{E(a)\}$.*

Now the search cannot be immediately cut off after MXP is called in any of the three nodes n_1, n_2, or n_3. E.g., in n_1 FindConflicts returns $\{\{B(a)\}, \{E(a), \neg E(a)\}\}$. Only branches where all but one element from each explanation is already present can be cut off safely.

The exercise we make in Example 3 shows that the larger the overall amount of explanations (including also various undesired explanations as $\mathscr{E}_2$) and also the greater their size, the less advantage MHS-MXP is likely to retain. It should also be noted that while adding complementary assertions (inducing inconsistent explanations) to abducibles does not make a difference for MHS, it does make a difference (towards the worse) for MHS-MXP. This is also true of assertions that are mutually inconsistent due to the background ontology (and therefore inducing another type of undesired explanations, called irrelevant).

The problem of finding all explanations of a given abduction problem is hard. MHS itself is NP-complete [3], and it still has to call DL reasoning (likely exponential or worse) in every node. While MHS-MXP provides an advantage on certain inputs we have no reason to suppose it is substantially better in the worst case. We were however interested to explore the improvement in the advantageous cases empirically on which we report in the following part of this paper.

6. Implementation and Evaluation

An implementation[1] of MHS-MXP was developed in Java using Maven for dependency management. Compared to the previous version of the implementation [6], the current version features: pure MHS mode, observations in form of a set of complex concept assertions, and equal-paths pruning (in addition to subset pruning).

An external DL reasoner is called in each node to query about the model. The model data is extracted through OWL API [13], using the getTypes method. This approach is preliminary and it has known issues[2], we are currently working on an alternate approach (cf. Conclusions).

Using the implementation, we ran[3] tests in order to understand if MHS-MXP may have a certain advantage at least on cases of inputs that we identified as favourable. We have used the LUBM ontology [14]. It does not contain disjunction, so we are able to test on it even given the limited model extraction method that we currently use.

LUBM does not feature atomic concepts that would allow explanations of size greater than 1. Hence in each input we have set the observation to $D(a)$ where a is an individual and D is a new auxiliary concept name. Then we have randomly generated a set of LUBM concept names $A_1, \ldots, A_n$, and added an equivalence $D \equiv A_1 \sqcap \cdots \sqcap A_n$ into the input ontology. If $A_1, \ldots, A_n$ are independent concepts with no subconcepts there is exactly one explanation $\{A_1(a), \ldots, A_n(a)\}$. If $A_1, \ldots, A_n$ are not independent the explanation will be shorter. If some of $A_1, \ldots, A_n$ have (independent) subconcepts there will be multiple partially overlapping explanations. We targeted explanations of size up to 5, hence we generated inputs for $n \in [1..5]$ – 50 inputs altogether. We have aggregated the inputs into five groups S1–S5 based on the size of the largest explanation. The inputs were generated randomly (however LUBM concepts with subconcepts were drawn twice as often to ensure higher number of explanations). The number of generated samples was consecutively reduced in order to obtain balanced groups S1–S5, each with 10 inputs.

To verify the second part of our conjecture, i.e. that MHS-MXP may perform better on inputs with smaller number of explanations, we also aggregated the same 50 inputs differently, into

---

groups C1–C5 accordingly. Basic characteristics of groups S1–S5 and C1–C5 are given in Table 1.

Table 1

Statistics for input groups: #: number of inputs; C_m, C_a, C_M: min, average, and max count of explanations; S_m, S_a, S_M: min, average, and max size of the largest explanation

Set	#	C_m	C_a	C_M	Set	#	C_m	C_a	C_M	S_m	S_a	S_M
S1	10	2	8	21	C1	9	2	5.8	10	1	1.11	2
S2	10	9	70.5	160	C2	11	16	52	100	1	2	3
S3	10	48	213.4	480	C3	17	112	260.3	480	2	3.39	4
S4	10	168	418.8	840	C4	4	504	640.5	840	4	4.25	5
S5	10	504	2628	6720	C5	9	1176	2864	6720	5	5	5

Our implementation supports atomic and negated atomic concept assertions as abducibles, where the latter may be suppressed by a switch. We have used this to obtain an unfavourable case for MHS-MXP (both atomic and negated atomic concepts allowed) and a favourable case (negated atomic concepts suppressed). However, all generated inputs only have explanations involving atomic concept assertions, hence each input has exactly the same number of explanations in either case (favourable and unfavourable) – only the search space in the unfavourable case (and the inherent difficulty for MHS-MXP to handle it) is larger.

Testing the inputs on the unfavourable case did not show significant difference between MHS and MHS-MXP. For the lack of space we omit details. The results on the favourable case are plotted in Figure 1. MHS runs are plotted on the left, and MHS-MXP runs are plotted on the right; results from groups S1–S5 are at the top, while groups C1–C5 are at the bottom.

To observe the efficiency of search space exploration we measured for each level of the HS-tree (x-axis) the time by which it is fully completed (y-axis). Group averages are plotted. Recall that when MHS completes level n it has already found all explanations of size up to n. In contrast MHS-MXP, after completing level n, has already found all explanations of size up to $n+1$ (cf. Observation 3).

MHS runs show similar times for S1–S5 up to level 2. S4–S5 reached time out in level 3, S2–S3 in level 4. Few S1 runs managed to finish levels 4 and 5. (Note that the results for levels 4 and 5 are "polluted" by the fact that only a small number of runs which finished before the time out are averaged here. However, levels 0–3 show the general trend.) In contrast, MHS-MXP was able to finish all runs within the same time out, even if sometimes required a few more levels than the maximal size of explanation for the given input. We can see a clear correlation between the maximal length of explanations and the required running time, in this respect each consecutive group between S1 and S5 proved to be harder.

The results on groups C1–C5 shows very similar trends, proving that MHS-MXP has greater advantage on inputs with smaller number of explanations.

7. Discussion

We have reported on an ABox abduction algorithm based on hybrid combination of MHS and MXP. The aim of this approach is to obtain a complete algorithm, that is more efficient than
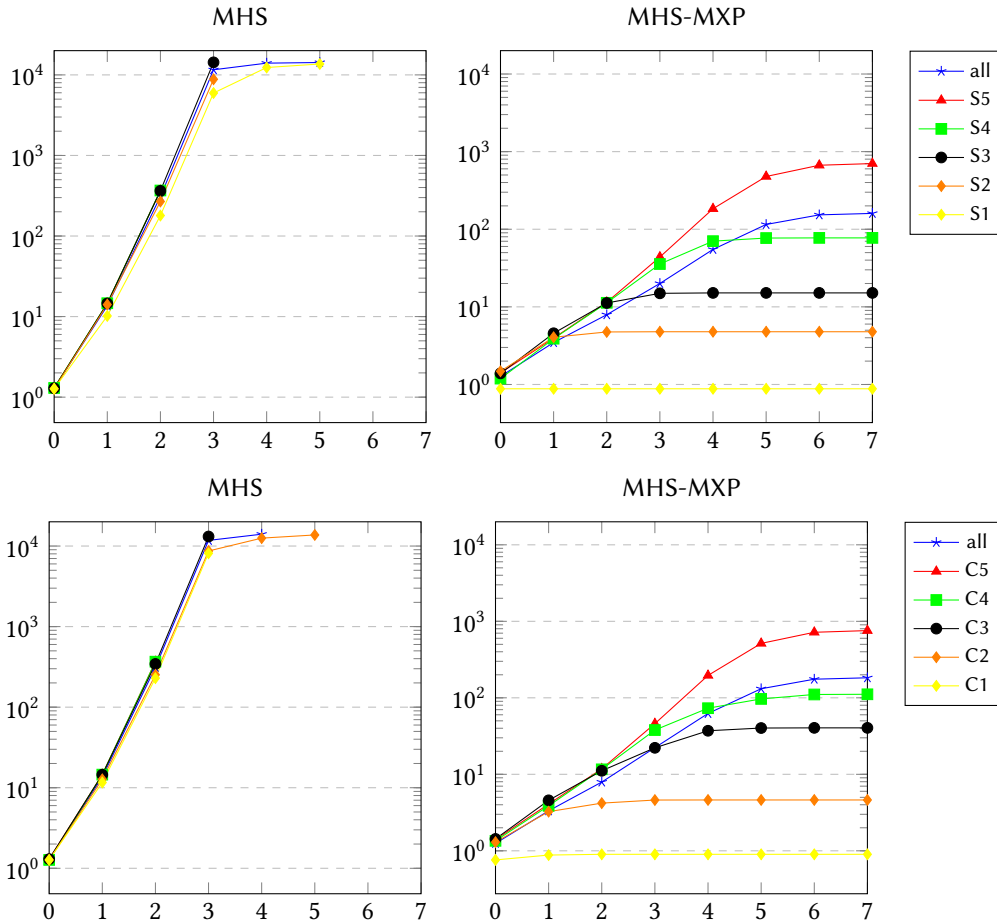
Figure 1: Average time (in the seconds) for computing individual levels from the certain group of inputs

MHS, at least on a class of favourable inputs that we have described. We have implemented the algorithm and our first preliminary evaluation on the LUBM ontology confirms this conjecture.

Both MHS and MHS-MXP rely on an external DL reasoner (as a black box) to compute models of the KB. However DL reasoners do not return models as a standard reasoning service. They do not even necessarily compute them, however, tableau-based reasoners compute a finite representation of a model which contains sufficient information for our approach. We currently employ a simplified workaround for model extraction which does not work in general. It allowed us to make experiments on LUBM, however we want to extend these to other (real world) ontologies. We are currently looking into the possibility to use OWLKnowledgeExplorerReasoner extension in OWL API for true model extension. The advantage of using an API would be that different (tableau-based) DL reasoners could then be plugged in. Another option would be to use a tight source-code integration of a DL reasoner such as in the case the MHS solver AAA [15] which tightly integrates Pellet [16].

Acknowledgments

References

[1] C. Elsenbroich, O. Kutz, U. Sattler, A case for abductive reasoning over ontologies, in: Proceedings of the OWLED*06 Workshop on OWL: Experiences and Directions, Athens, GA, US, volume 216 of *CEUR-WS*, 2006.

[2] R. Reiter, A theory of diagnosis from first principles, Artificial intelligence 32 (1987) 57–95.

[3] R. M. Karp, 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., 1972, pp. 85–103.

[4] U. Junker, 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, AAAI Press, 2004, pp. 167–172.

[5] K. M. Shchekotykhin, D. Jannach, T. Schmitz, 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.

[6] M. Homola, J. Pukancová, J. Gablíková, K. Fabianová, Merge, explain, iterate, in: S. Borgwardt, T. Meyer (Eds.), Proceedings of the 33rd International Workshop on Description Logics (DL 2020) co-located with the 17th International Conference on Principles of Knowledge Representation and Reasoning (KR 2020), Online Event [Rhodes, Greece], September 12th to 14th, 2020, volume 2663 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2020.

[7] W. Del-Pinto, R. A. Schmidt, Abox abduction via forgetting in ALC, in: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, Honolulu, Hawaii, USA, AAAI Press, 2019, pp. 2768–2775.

[8] P. Koopmann, W. Del-Pinto, S. Tourret, R. A. Schmidt, Signature-based abduction for expressive description logics, in: Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning, KR 2020, Rhodes, Greece, 2020, pp. 592–602.

[9] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, P. F. Patel-Schneider (Eds.), The Description Logic Handbook: Theory, Implementation, and Applications, Cambridge University Press, 2003.

[10] M. Schmidt-Schauß, G. Smolka, Attributive concept descriptions with complements, Artificial intelligence 48 (1991) 1–26.

[11] I. Horrocks, O. Kutz, U. Sattler, The even more irresistible $\mathcal{SROIQ}$, in: Proceedings,

Tenth International Conference on Principles of Knowledge Representation and Reasoning, Lake District of the United Kingdom, AAAI, 2006, pp. 57–67.

[12] J. Pukancová, M. Homola, ABox abduction for description logics: The case of multiple observations, in: Proceedings of the 31st International Workshop on Description Logics, Tempe, Arizona, US, volume 2211 of *CEUR-WS*, 2018.

[13] M. Horridge, S. Bechhofer, The OWL API: A java API for OWL ontologies, Semantic Web 2 (2011) 11–21.

[14] Y. Guo, Z. Pan, J. Heflin, LUBM: A benchmark for OWL knowledge base systems, Journal of Web Semantics 3 (2005) 158–182.

[15] J. Pukancová, M. Homola, The AAA abox abduction solver, Künstliche Intell. 34 (2020) 517–522.

[16] E. Sirin, B. Parsia, B. Cuenca Grau, A. Kalyanpur, Y. Katz, Pellet: A practical OWL-DL reasoner, Journal of Web Semantics 5 (2007) 51–53.