Merge, Explain, Iterate: A Combination of MHS and MXP in an ABox Abduction Solver*

Martin Homola, Júlia Pukancová, Janka Boborová, and Iveta Balintová

Comenius University in Bratislava, Mlynská dolina, 842 48 Bratislava, Slovakia
{homola,pukancova}@fmph.uniba.sk, boborova3@uniba.sk,
ivbalintova@gmail.com

Abstract. Minimal Hitting Set (MHS) is a well-known and complete method to compute all minimal explanations of an ABox abduction problem in Description Logics (DL). MHS is NP-complete and generally recognized as inefficient. We leverage on MergeXplain (MXP) which is fast but incomplete – by combining it with MHS in a hybrid algorithm MHS-MXP to regain completeness. In this paper, we describe: (a) the underlying theory to establish the completeness of MHS-MXP and show its relevant properties; (b) a class of inputs on which MHS-MXP has the greatest advantage; (c) an experimental implementation; (d) an empirical evaluation on both favourable and unfavourable inputs.

Keywords: Abduction · Description logics · Ontologies.

1 Introduction

ABox abduction [7] assumes a DL knowledge base (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 [15] is the classic method to find all minimal explanations of an ABox abduction problem. MHS systematically searches through all possible explanations, from the smallest (in terms of cardinality) towards the largest – thus it ensures *completeness*. It has a good chance to discover smaller explanations quite quickly, however if explanations of interest are larger, it is rather inefficient. Notably, the MHS problem itself is NP-complete [11] 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.

Alternatively QuickXplain (QXP) [10] and more recently its extension MergeXplain [17] employ a *divide and conquer* strategy to find one (QXP) or even multiple explanations (MXP) efficiently. But they are incomplete, i.e., there is no warranty that all explanations will be found. However, when MXP is run repeatedly, on slightly modified inputs, it divides the search space differently and it may return a different set of explanations. In fact, it is possible to regain completeness by using MHS on the background to track the search space exploration. We formally develop such a combined algorithm, that we call *MHS-MXP*. We study its relevant properties that allow us not only to establish its correctness

* Published on 23 Sep 2023 as: Homola, M., Pukancová, J., Boborová, J., Balintová, I. (2023). Merge, Explain, Iterate: A Combination of MHS and MXP in an ABox Abduction Solver. In: Gaggl, S., Martinez, M.V., Ortiz, M. (eds) Logics in Artificial Intelligence. JELIA 2023. LNCS, vol 14281. Springer, Cham. https://doi.org/10.1007/978-3-031-43619-2_24

but also to characterize inputs on which it may have an advantage over MHS: inputs with smaller explanations or with smaller number of explanations.

Our experimental implementation allows to switch between both algorithms. It integrates the JFact reasoner as a black box. MHS (and thus also MHS-MXP) requires not only to verify KB consistency but also to extract relevant information about the model. Not all DL reasoners can be used for this, but tableau reasoners such as JFact internally construct a sufficient part of the model – albeit it is not usual to output it. We were able to employ experimental features of OWL API to extract relevant model information from JFact. We then conducted an empirical evaluation on a favourable but also on an unfavourable class of inputs. MHS-MXP did not perform as well as MHS on the unfavourable class, however on the favourable class it outperformed MHS to a much larger extent.

Compared to other promising approaches in ABox abduction [5, 6, 4, 13], the main advantage of our work is that as a black-box approach and thus it may be paired with any DL reasoner (if it allows for model extraction). Tableau-based reasoners such as Pellet and JFact can handle DL expressivity up to $\mathcal{SROIQ}$ [9], i.e. up to OWL 2 [3]. Indeed, the other approaches may be more tractable, but they are limited in DL expressivity. Du et al. [5] rely on a translation to Prolog and is complete up to Horn-$\mathcal{SHIQ}$; Du et al. [6] focus on strong tractability for very large ABoxes with a limitation to first-order rewritable TBoxes. Both Del-Pinto and Schmidt [4] and Koopmann et al. [13] support DL expressivity up to $\mathcal{ALC}$. In theory, MHS-MXP is not limited to the DL setting. It can be applied in any case in which MHS is applicable.

2 Preliminaries

We assume familiarity with the basics of DL [1, 2], including vocabulary consisting of individuals $N_\mathrm{I} = \{a, b, \ldots\}$, roles $N_\mathrm{R} = \{P, Q, R, \ldots\}$, and atomic concepts $N_\mathrm{C} = \{A, B, \ldots\}$; complex concepts $C, D, \ldots$ built by constructors (e.g. $\neg$, $\sqcap$, $\sqcup$, $\exists$, $\forall$, in case of $\mathcal{ALC}$ [16]); 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 [9]) role assertions of the form $R(a, b)$ and $\neg R(a, b)$). We also remind about the semantics that relies on models M of a KB $\mathcal{K}$, that satisfy all axioms or assertions ϕ in $\mathcal{K}$ ($M \models \phi$); and the reasoning tasks of checking the consistency of $\mathcal{K}$ (if it has a model) and entailment ($\mathcal{K} \models \phi$ if $M \models \phi$ for all its models M).

In *ABox abduction* [7], 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 $\mathcal{K} \cup \mathcal{E} \models O$. 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 $\mathcal{K} \cup \mathcal{E} \models O$.*

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

According to Elsenbroich et al. [7] 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 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 $\mathcal{E}$.*

3 Computing Explanations

We first review 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 [15], computing all minimal explanations of $(\mathcal{K}, O)$ reduces to finding all minimal hitting sets of the set of models of $\mathcal{K} \cup \{\neg O\}$ 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 correspond 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.

In a labelled tree $T = (V, E, L)$ with root $r \in V$, let $H(n)$ denote the union of edge-labels on the path from r to n, for any node $n \in V$. 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.

MHS (Algorithm 1) works by constructing an HS-tree. An HS-tree for $\mathcal{P} = (\mathcal{K}, O)$ is a labelled tree $T = (V, E, L)$ where (a) each node $n \in V$ is labelled by $L(n) = \text{Abd}(M)$ for a model M of $\mathcal{K} \cup \{\neg O\}$ s.t. $L(n) \cap H(n) = \emptyset$ or by $L(n) = \emptyset$ if such a model does not exist; (b) and for any $n \in V$ there is a σ-successor of n for every $\sigma \in L(n)$.

Each label $L(n)$ can be found as $\text{Abd}(M)$ of some 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.

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

$M \leftarrow$ a model M of $\mathcal{K} \cup \{\neg O\}$
if $M =$ `null` **then**
  return "nothing to explain"
end if
$T \leftarrow (V = \{r\}, E = \emptyset, L = \{r \mapsto \text{Abd}(M)\})$
for each $\sigma \in L(r)$ create new σ-successor n_σ of r
$\mathcal{S}_\mathcal{E} \leftarrow \{\}$
while exists next node n in T w.r.t. BFS **do**
  if n can be pruned **then**
    prune n
  else if exists model M of $\mathcal{K} \cup \{\neg O\} \cup H(n)$ **then**
    label n by $L(n) \leftarrow \text{Abd}(M)$
  else if $H(n)$ is desired **then**
    $\mathcal{S}_\mathcal{E} \leftarrow \mathcal{S}_\mathcal{E} \cup \{H(n)\}$
  end if
  for each $\sigma \in L(n)$ create new σ-successor n_σ of n
end while
return $\mathcal{S}_\mathcal{E}$

We apply first two of Reiter's pruning conditions: (1) *subset pruning* eliminates non-minimal hitting sets: given a hitting set $H(n)$, nodes n' with $H(n) \subseteq H(n')$ are pruned; (2) *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 [15]. MHS is sound and complete [15, 14].

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 [14].

3.2 MergeXplain

Both QXP [10] and MXP [17] 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 function GetConflict($\mathcal{B}, D, \mathcal{C}$), where the inputs $\mathcal{B}$ and $\mathcal{C}$ are as explained above, and D is an auxiliary control parameter. GetConflict 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 need to find one minimal explanation of an ABox abduction problem, adopting a result of Junker [10] we may 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 $\mathcal{P}$.*

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** Γ

8: **function** FindConflicts($\mathcal{B}, \mathcal{C}$)
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: $\mathcal{C}'_1 \leftarrow \mathcal{C}'_1 \backslash \{\sigma\}$ 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** 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**

The MXP algorithm is captured in the function FindConflicts($\mathcal{B}, \mathcal{C}$), where again $\mathcal{B}$ is the consistent background theory and $\mathcal{C}$ is the set of conflicts inconsistent with it. It returns a pair $\langle \mathcal{C}', \Gamma \rangle$, where Γ contains as many conflicts $\gamma \subseteq \mathcal{C}$ as it is possible to reconstruct from one way in which $\mathcal{C}$ can be split, and $\mathcal{C}' \subseteq \mathcal{C}$ is maximal set consistent with $\mathcal{B}$ that can be reconstructed from this 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 $\mathcal{P} = (\mathcal{K}, O)$ 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. [17]:

Theorem 3. *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* MXP($\mathcal{K} \cup \{\neg O\}$, Abd) *returns a nonempty set Γ of minimal 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 3. *Given an ABox abduction problem $\mathcal{P} = (\mathcal{K}, O)$, a set of abducibles* Abd, *and any $\gamma \subseteq$ Abd s.t. $|\gamma| = 1$, if $\mathcal{K} \cup \gamma \models O$ then $\gamma \in$ MXP($\mathcal{K} \cup \{\neg O\}$, 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 $\mathcal{K} = \{A \sqcap B \sqsubseteq D, A \sqcap C \sqsubseteq D\}$ and let $O = D(a)$. Let us ignore negated ABox expressions and start with $\text{Abd} = \{A(a), B(a), C(a)\}$. There are two minimal explanations of $\mathcal{P} = (\mathcal{K}, O)$: $\{A(a), B(a)\}$, and $\{A(a), C(a)\}$. Calling $\text{MXP}(\mathcal{K} \cup \{\neg O\}, \text{Abd})$, it passes the initial tests and calls FINDCONFLICTS$(\mathcal{K} \cup \{\neg O\}, \text{Abd})$.

FINDCONFLICTS needs to decide how to split $\mathcal{C} = \text{Abd}$ 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.

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

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

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 previously found models in the model cache

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

Input: knowledge base $\mathcal{K}$, observation O, set of abducibles Abd
Output: 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(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 n can be pruned then
11:         prune n
12:     else
13:         ⟨_, Γ⟩ ← FindConflicts(K ∪ {¬O} ∪ H(n), Abd \ H(n))
14:         Con ← Con ∪ {H(n) ∪ γ | γ ∈ Γ}
15:         if ∃γ ∈ Γ : |γ| > 1 then                                  ▷ Extend HS-tree under n
16:             L(n) ← Abd(M) \ H(n) for some M ∈ Mod s.t. M ⊨ H(n)
17:             for each σ ∈ L(n) create new σ-successor n_σ of n
18:         end if
19:     end if
20: end while
21: return S_E ← {γ ∈ Con | γ is desired}
22: function isConsistent(K)
23:     if there is M ⊨ K then
24:         Mod ← Mod ∪ {M}
25:         return true
26:     else
27:         return false
28:     end if
29: end function
```

Mod. The stored models are later used to construct the HS-tree and label its nodes. Also FindConflicts will use this modified isConsistent function.

Then the main loop is initiated. For the root node r, pruning is never applied. Then 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 15). 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 Lemma 1 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 (line 10): n is pruned (1) either if there is a previously stored conflict $\gamma \in$ Con s.t. $\gamma \subseteq H(n)$, (2) or if there is another $n' \in V$ (that is not pruned) with $H(n') = H(n)$. This corresponds to Reiter's first two pruning conditions with condition (1) being modified to make use of conflicts cached in Con. If n is not pruned, 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 $\mathrm{Abd} \setminus H(n)$ as the set of conflicts.

If we are lucky, we might cut off this branch completely in line 15, that is, if no extension of $H(n)$ of size greater than 1 is found (by Lemma 1). 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.

Lemma 2. *For each node n of the HS-tree visited by the main loop of* MHS-MXP*$(\mathcal{K}, O, \mathrm{Abd})$ either $H(n) \in \mathrm{Con}$ or $\mathcal{K} \cup \{\neg O\} \cup H(n)$ is consistent and at least for one $M \in \mathrm{Mod}$, $M \models \mathcal{K} \cup \{\neg O\} \cup H(n)$.*

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 $\mathcal{S}_\mathcal{E}$.

Theorem 4. *The MHS-MXP 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*).*

This follows from the fact that the algorithm correctly reconstructs the HS-tree to a sufficient extent. The parts which are cut off in comparison to a complete HS-tree (line 15) can be omitted thanks to Observation 3 and Lemma 1.

5 Advantages and Limitations

Apparently MHS-MXP absolutely crushes MHS in cases when all explanations are of size one. By Observation 3 and Lemma 1, 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 $\mathcal{K} = \{A \sqcap B \sqsubseteq F, D \sqcap \neg C(a), E(b)\}$, let $O = F(a)$, and let $\mathrm{Abd} = \{A(a), B(a), C(a), D(a)\}$. There is exactly one explanation $\mathcal{E}_1 = \{A(a), B(a)\}$.

If we run MHS-MXP, it first checks $\mathcal{K} \cup \{\neg F(a)\}$ 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 $\mathcal{E}_1$ was returned and $|\mathcal{E}_1| > 1$. Therefore n_1 and n_2 are added to the HS-tree with $H(n_1) = \{A(a)\}$ and $H(n_2) = \{C(a)\}$.

Calling FindConflicts n_1 returns one conflict $\{B(a)\}$ which together with $H(n_1)$ makes up for the explanation $\mathcal{E}_1$. 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.

Cases similar to Example 2 with a small overall number of explanations can be handled rather efficiently, compared to MHS, as significant part of the search space is cut off. However consider the following modification of the inputs.

Example 3. Given $\mathcal{K}$ and O as in Example 2, let Abd $= \{A(a), B(a), C(a), D(a), E(a), \neg E(a)\}$. The abduction problem $(\mathcal{K}, O)$ has two explanations $\mathcal{E}_1 = \{A(a), B(a)\}$ and $\mathcal{E}_2 = \{E(a), \neg E(a)\}$, the second undesired (inconsistent). FindConflicts called in the root r now returns conflicts $\{\{A(a), B(a)\}, \{E(a), \neg E(a)\}\}$. W.l.o.g. we may assume that the same model M was used to label r and that $M \not\models E(a)$. This time 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.

Example 3 shows that the larger the overall amount of explanations and the greater their size, the less advantage MHS-MXP is likely to retain. While adding complementary assertions to abducibles does not make a difference for MHS, it does for MHS-MXP (for the worse), as it generates more explanations (even if they are inconsistent and thus undesired). Similarly for mutually inconsistent abducibles (due to the background ontology) yielding irrelevant explanations.

Thus while MHS-MXP provides an advantage on certain inputs we have no reason to suppose it is substantially better in the worst case. It is difficult to estimate to which extent the problem of conflicting abducibles demonstrated in Example 3 would affect real world use, especially if users (knowledgable about the domain) would be able to specify abducibles suitable enough to contain all explanations they are interested in. There are no known real-world use cases to evaluate abductive reasoning with ontologies that would specify inputs with observations and respective sets of abducibles. Even works that conducted extensive empirical evaluations used artificially generated inputs [6, 4, 13].

To understand how MHS-MXP compares to MHS on unfavourable inputs with large amounts of conflicting abducibles, and jointly to which extent it is faster than MHS on favourable inputs without conflicting abducibles, we conducted an evaluation on which we report in the following.

6 Implementation

An implementation[1] of MHS-MXP was developed in Java. The *black box* implementation calls an external DL reasoner for consistency checks and extracts model information necessary to steer the HS-tree construction by both MHS and MHS-MXP. The latter is nontrivial as it is fairly nonstandard for any DL reasoner to make model data accessible to the user. In fact, some DL reasoners (e.g. consequence-based [12]) may not even construct any model-related structures, but tableau-based reasoners do construct a *completion graph* which is a finite representation of a model.

We are concerned with exploring all possible explanations that one can construct as (sets of) atomic and negated atomic concept and role assertions involving the named individuals from the ABox and from the input observation. Note that the corresponding part of the completion graph is always fully constructed by tableau-based DL reasoners and is not affected by blocking [1, 2, 9].

We rely on the `reasoner.knowledgeexploration` package[2] – an experimental package of OWL API – and its interface `OWLKnowledgeExplorerReasoner`. The interface is not commonly implemented by reasoners, however it is implemented by JFact[3], which we were hence able to use in our implementation.

It allows to read out information about the completion graph: nodes are accessed via the `getRoot(e)` method where `e` is the `OWLClassExpression` (nominal) corresponding to a given ABox individual. The `getObjectLabel()` method is then used to extract all atomic and negated atomic concepts to which the individual belongs (the latter is obtained as the complement of the former). The neighbouring nodes and the respective role assertions are then obtained using the `getObjectNeighbors()` method.

7 Evaluation

The experiments were executed on a virtual machine with 8 cores (16 threads) of Intel Xeon CPU E5-2695 v4, 2.10GHz, with 32GB RAM, running Ubuntu 20.04 and Oracle Java SE Runtime Environment v1.8.0_201. Execution times were measured using `ThreadMXBean` from the `java.lang.management` package. We measured user time – the actual time without system overhead. The maximum Java heap size to 4GB.

Using the implementation, we ran tests in order to understand how MHS-MXP compares to plain MHS (a) in the general case, and (b) in case of inputs that we identified as favourable. We have used the LUBM ontology [8] and the solver's abducibles settings to generate suitable inputs to verify both cases.

[1] The implementation and the evaluation datasets are available at https://github.com/boborova3/MHS-MXP-algorithm.

[2] http://owlcs.github.io/owlapi/apidocs_5/org/semanticweb/owlapi/reasoner/knowledgeexploration/package-summary.html

[3] https://github.com/owlcs/jfact

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	S_m	S_a	S_M
S1	10	1	7	20	1	1	1
S2	10	8	69.5	159	2	2	2
S3	10	47	212.4	479	3	3	3
S4	10	251	417.8	839	4	4	4
S5	10	503	2627	6719	5	5	5

Set	#	C_m	C_a	C_M	S_m	S_a	S_M
C1	9	1	4.8	9	1	1.11	2
C2	11	14	51	99	1	2	3
C3	13	111	212.92	299	2	3.15	4
C4	8	359	524.75	839	3	4	5
C5	9	1175	2863	6719	5	5	5

In order to generate inputs with explanations of size up to n, we used a fresh individual a and composed observations in the form $A_1 \sqcap \cdots \sqcap A_n(a)$ where $A_1, \ldots, A_n$ were randomly drawn from LUBM concept names. If all $A_1, \ldots, A_n$ have mutually independent proper subconcepts then there is at least one explanation of size n. If some of them have more subconcepts then the input will have more explanations. If some A_i, A_j have shared subconcepts then (some of) the explanations will be shorter.

Targeting explanations of size up to 5, 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.

Our implementation supports atomic and negated atomic concept assertions as abducibles, where the latter may be suppressed by a switch. In accordance with our observations from Section 5, we have used this feature to obtain an unfavourable case for MHS-MXP (both atomic and negated atomic concepts allowed) and a favourable case (negated atomic concepts suppressed). Notably, 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.

Each individual input was run five times and the results were averaged. The timeout was set to 4 hours (=14,440 seconds).

The results for the unfavourable and favourable case are shown in Figures 1 and 2, respectively. For each case the charts analogously plot the average time per group (y-axis) in which all explanations of a given size (x-axis) are guaranteed to be found. Input groups S1–S5 are shown on the left (a), and input groups C1–C5 in the right (b). Note that for MHS this equates to the time by which it fully explores the HS-tree down to depth x, however, by Observation 3, for MHS-MXP this is the time by which it fully explores the HS-tree down to depth

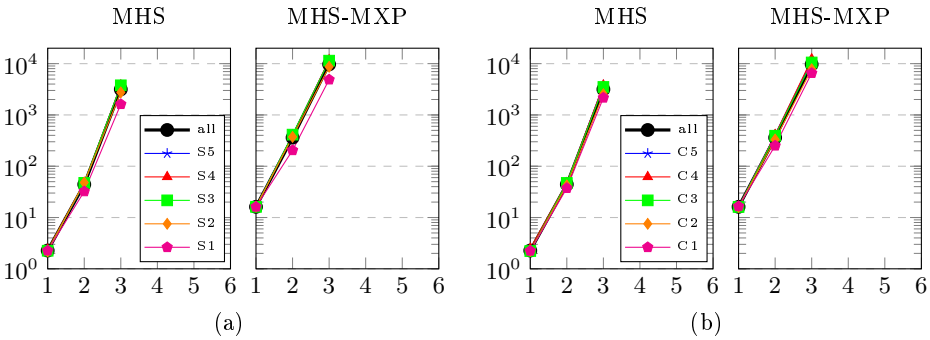

Fig. 1. Unfavourable inputs: Average time in seconds (y-axis) for fully exploring the search space up to the particular explanation size (x-axis) for input groups (a) S1–S5, (b) C1–C5

$x-1$. Also note that once the respective group terminated it has fully explored the search space up to any size, hence the line is constantly extended towards the right; in contrast, if more than one third of inputs in the group reached the timeout before reaching the x-value, the y-value was omitted from the plot.

Looking first at the unfavourable inputs (Figure 1) we observe that both algorithms exhibit steep exponential growth. Both managed to compute all explanations of size 3 within the timeout, but MHS reached this point faster. Notably, there is little difference among the groups S1–S5 and C1–C5. This case is strongly unfavourable for MSH-MXP, attacking its main weakness by swamping the search space with the highest possible amount of mutually inconsistent abducibles. The observed result is consistent with this setting.

In the favourable case (Figure 2) the advantage of MHS-MXP over MHS is way more significant; e.g. while MHS again did only fully explore the search space up to the size 3 within the 4 hour timeout, MHS-MXP managed to reach this point with the average time of 26 seconds. Also our conjecture of MHS-MXP having the greatest advantage on inputs with smaller count and/or smaller maximal size of explanations verified very clearly. We observe a clear correlation in the increase of computation time and the greatest size of an explanation (groups S) and likewise for the count of explanations (groups C). All groups except for S5 and C5 terminated within the timeout, while in S5 and C5 approx. half of the inputs reached the timeout. (Note that even in such cases, MHS-MXP found all explanations, the search continued and the timeout was reached due to the presence of irrelevant explanations of size greater than 5).

8 Conclusions

We have designed and implemented a hybrid combination of MHS and MXP algorithms and formally proved its correctness. One of the main disadvantages of MHS is that it always needs to tediously inspect each candidate solution in the

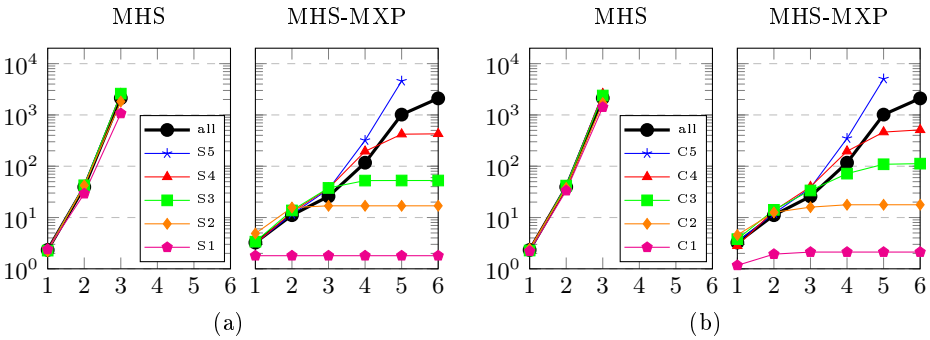

Fig. 2. Favourable inputs: Average time in seconds (y-axis) for fully exploring the search space up to the particular explanation size (x-axis) for input groups (a) S1–S5, (b) C1–C5

whole search space – even in cases when there is just small number of solutions, and even after all of them have been already found. In such cases the advantage of the combination with MXP shows to be the most promising.

The empirical evaluation supports this conjecture. While on favourable inputs MHS-MXP significantly outperformed MHS, we have also found unfavourable cases on which MHS-MXP performs somewhat worse than MHS. Improving such cases is part of our ongoing work: for instance, we did not yet modify the inner working of the MXP called in each HS-tree node. This offers space for optimization, e.g. to exploit the cached conflicts Con from the previous runs when MXP splits the set of conflicts $\mathcal{C}$. We also want to characterize the performance on a wider scale of input classes to better understand the trade-off.

While there are currently approaches in ABox abduction [5, 6, 4, 13] which are more tractable, they are also limited in supported DL expressivity, in our solver we were able to achieve black-box integration with JFact which supports DL expressivity up to $\mathcal{SROIQ}$ (i.e. OWL 2).

Even for tableau-based reasoners, model extraction is not a standard. So far, we were able to plug-in JFact exploiting the `OWLKnowledgeExplorerReasoner` extension of OWL API. In the future, we would like to integrate more tableau reasoners into our solver and allow for modular switching.

In the future, we would also like to look into a possible upgrade of the MHS part of the algorithm by some of its known more effective versions, e.g. by Wotawa [18].

While in this work we have studies and applied the MHS-MXP algorithm on the problem of ABox abduction in DL, it is worth noting that it is also applicable in other cases, more precisely in any case in which MHS is applicable. It can be applied in other languages (e.g. in propositional abduction, in which models can be extracted from a SAT solver) and even on other problems (e.g. computing justifications and maximally consistent subsets).

Acknowledgments. We would like to express our thanks to anonymous reviewers for their valuable feedback on this and also on the previous version of this report. This research was sponsored by the Slovak Republic under the grant APVV-19-0220 (ORBIS) and by the EU under the H2020 grant no. 952215 (TAILOR) and under Horizon Europe grant no. 101079338 (TERAIS).

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. Baader, F., Horrocks, I., Lutz, C., Sattler, U.: An Introduction to Description Logic. Cambridge University Press (2017)
3. Cuenca Grau, B., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P., Sattler, U.: OWL 2: The next step for OWL. J. Web Semant. **6**(4), 309–322 (2008)
4. Del-Pinto, W., Schmidt, R.A.: Abox abduction via forgetting in ALC. In: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, Honolulu, Hawaii, USA, pp. 2768–2775, AAAI Press (2019)
5. 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)
6. Du, J., Wang, K., Shen, Y.: A tractable approach to ABox abduction over description logic ontologies. In: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27-31, 2014, Québec City, Québec, Canada., pp. 1034–1040 (2014)
7. 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)
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. 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)
10. 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)
11. 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)
12. Kazakov, Y., Krötzsch, M., Simančík, F.: The incredible ELK. Journal of Automated Reasoning **53**(1), 1–61 (2014)
13. Koopmann, P., Del-Pinto, W., Tourret, S., Schmidt, R.A.: 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, pp. 592–602 (2020)
14. 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)

15. Reiter, R.: A theory of diagnosis from first principles. Artificial intelligence **32**(1), 57–95 (1987)
16. Schmidt-Schauß, M., Smolka, G.: Attributive concept descriptions with complements. Artificial intelligence **48**(1), 1–26 (1991)
17. 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)
18. Wotawa, F.: A variant of reiter's hitting-set algorithm. Inf. Process. Lett. **79**(1), 45–51 (2001)