CATS Solver: The Rise of Hybrid Abduction Algorithms

Jakub Kloc, Janka Boborová, Martin Homola and Júlia Pukancová

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

Abstract

The state-of-the-art complete algorithms to solve ABox abduction in DL include the original Reiter's algorithm for minimal hitting sets alongside its more recent updates: Wotawa's HST and Pill and Quaritch's RC-Tree. On the other hand, incomplete methods that quickly find some but not all solutions include Junker's QuickXplain and MergeXplain by Shchekotykhin et al. We present CATS, a new modular ABox abduction solver. It implements all the said algorithms together with the hybrid MHS-MXP, recently introduced by Homola et al., and two new analogous variants: HST-MXP and RCT-MXP, based on HST and RC-Tree, respectively. The user can choose any of the eight algorithms. The solver uses the JFact reasoner as a black box and thus allows any DL expressivity up to $\mathcal{SROIQ}$. The modular implementation served as a test bed for an evaluation and comparison of the implemented algorithms, which we conducted over the LUBM ontology. Out of the complete algorithms, the hybrid ones were proven to find explanations faster, and they were also more memory-efficient.

Keywords

Description logics, ABox abduction, abduction algorithms, solver, empirical evaluation

1. Introduction

Abduction [1, 2] is a fundamental way of reasoning, alongside induction and deduction. Its objective is to provide hypothetical explanations for a given observation that is not entailed by the available knowledge. As an *ampliative* form of reasoning, it generates genuinely new knowledge that cannot be derived through classical deductive reasoning. In the context of description logics (DL) and ontologies, it was long considered a non-standard reasoning problem [2]; however, recently it has been receiving more and more attention [3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]. DL-based abduction has been applied in various domains, such as ontology engineering [15], ontology debugging [16] and repair [17], medical diagnosis [18, 19, 20, 21, 22], system diagnosis [23], multimedia interpretation [24], and e-commerce [25].

Figure 1: CATS logo

There are various types of algorithms that can be utilized to solve an abduction problem. Our research focuses mainly on those that offer the following advantageous properties [26]: (1) Applicability to knowledge bases of *any expressivity*, or even to any logic (with entailment and model-theoretic semantics); (2) *Completeness*, meaning they guarantee finding all possible explanations; (3) *Black-box reasoning*, allowing supporting reasoning tasks required for computing explanations to be delegated to an external reasoner; (4) Requiring *no additional information* about the problem.

One such algorithm is the well-explored Reiter's minimal hitting set algorithm (MHS) [27], along with its hybrid version, MHS-MXP [28], which combines Reiter's algorithm with the efficient but incomplete divide-and-conquer MergeXplain (MXP) method [29]. Both were implemented in the MHS-MXP abduction solver [28], currently one of the few publicly available abduction solvers [30, 13], and the only one supporting multiple algorithms.

DL 2025: 38th International Workshop on Description Logics, September 3–6, 2025, Opole, Poland

kloc4@uniba.sk (J. Kloc); boborova@fmph.uniba.sk (J. Boborová); homola@fmph.uniba.sk (M. Homola); pukancova@fmph.uniba.sk (J. Pukancová)
https://dai.fmph.uniba.sk/~homola/ (M. Homola); https://dai.fmph.uniba.sk/~pukancova/ (J. Pukancová)
0009-0002-7487-9962 (J. Boborová); 0000-0001-6384-9771 (M. Homola); 0009-0001-3505-0716 (J. Pukancová)

However, the MHS-MXP solver has notable shortcomings. It suffers from memory management issues, with one of its evaluations resulting in out-of-memory errors in 57% of cases [31]. Additionally, it adopts a modified version of Reiter's MHS algorithm that preserves completeness at the cost of sacrificing some of the original optimizations, which, when combined, may lead to the omission of explanations [32].

To address this limitation, we explore alternatives to Reiter's algorithm that aim to resolve the incompleteness of MHS in a more efficient manner, namely, Wotawa's Hitting Set Tree (HST) algorithm [33] and RC-Tree (RCT) algorithm [34]. We apply them to solve the abduction problem, and propose new hybrid algorithms by combining them with the MergeXplain method. We then present an improved version of the solver that supports MHS, HST and RC-Tree, together with their hybrid versions MHS-MXP, HST-MXP and RCT-MXP, and the two incomplete algorithms MergeXplain and QuickXplain [35]. This new version, rebranded as CATS (Comenius Abduction Team Solver), is freely available[1].

Finally, we provide an empirical evaluation of the algorithms on a dataset derived from the LUBM ontology [36], focusing on the cumulative amount of computed explanations over time. The results show a significant advantage of the hybrid algorithms, particularly MHS-MXP and RCT-MXP. We also show that the hybrid algorithms are more efficient in memory usage.

2. Preliminaries

2.1. Description Logics

The vocabulary of $\mathcal{ALC}$ [37, 38] consists of three countable, mutually disjoint sets of non-logical symbols: individuals (denoted N_I), atomic concepts (denoted N_C) and roles (denoted N_R). $\mathcal{ALC}$ concepts are recursively built using the following grammar: $C, D ::= A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \forall r.C \mid \exists r.C$, where $A \in N_C$ and $r \in N_R$. The logical symbols used in the grammar, called constructors, are as follows: $\neg$ (complement), $\sqcap$ (concept intersection), $\sqcup$ (concept union), $\exists$ (existential restriction), $\forall$ (value restriction). The semantics is defined using an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, which consists of a non-empty domain $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$ such that: $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ for every $a \in N_I$; $C^{\mathcal{I}} \subset \Delta^{\mathcal{I}}$ for every concept C; $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ for every $r \in N_R$. A DL knowledge base (KB) $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consists of a TBox $\mathcal{T}$ (intensional knowledge) and an ABox $\mathcal{A}$ (extensional knowledge), whilst $\mathcal{T}$ contains general concept inclusion (GCI) axioms in form $C \sqsubseteq D$ for any concepts C, D, and $\mathcal{A}$ contains concept assertions $a\colon C$, and role assertions $a, b\colon R$ for $a, b \in N_I$, $R \in N_R$, and any concept C.

An interpretation $\mathcal{I}$ satisfies an axiom ϕ ($\mathcal{I} \models \phi$) if and only if: if $\phi = C \sqsubseteq D$ then $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$; if $\phi = C(a)$ then $a^{\mathcal{I}} \in C^{\mathcal{I}}$; if $\phi = R(a, b)$ then $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$. $\mathcal{I}$ is a model of a KB $\mathcal{K}$ ($\mathcal{I} \models \mathcal{K}$) if and only if it satisfies all axioms in $\mathcal{K}$. $\mathcal{K}$ is consistent if and only if it has a model. $\mathcal{K}$ entails ϕ ($\mathcal{K} \models \phi$) if and only if ϕ is satisfied by every model of $\mathcal{K}$.

2.2. Abduction

In an ABox abduction problem [28, 2], we are looking for explanations for a given DL knowledge base $\mathcal{K}$ and an observations O (an ABox assertion), as defined below.

Definition 1 (ABox abduction problem)**.** *Let abducibles Abd be a finite set of ABox axioms, background knowledge $\mathcal{K}$ a DL knowledge base, and observation* O *an ABox axiom. An explanation of an ABox abduction problem $\mathcal{P} = (\mathcal{K}, \mathrm{O}')$ is a finite set of ABox axioms $\mathcal{E} \subseteq Abd$ such that $\mathcal{K} \cup \mathcal{E} \models \mathrm{O}$.*

The space of possible explanations is limited to axioms from the set of abducibles Abd. They are also typically limited to so-called *desired* explanations, i.e., (in this work) they must be consistent ($\mathcal{E} \cup \mathcal{K}$ is consistent); relevant ($O \not\models \mathcal{E}$); explanatory ($\mathcal{K} \not\models O$); and minimal (there is no other explanation $\mathcal{E}'$ s.t. $\mathcal{E}' \subseteq \mathcal{E}$).

The ABox abduction problem can be reduced to a consistency-checking problem [39]: $\mathcal{K} \cup \mathcal{E} \models O$ if and only if $\mathcal{K}' = \mathcal{K} \cup \mathcal{E} \cup \{\neg O\}$ *is inconsistent*, i.e., $\mathcal{K}'$ has no models. It has been established that

[1]https://github.com/Comenius-Abduction-Team/CATS-Abduction-Solver

finding all minimal explanations of some $\mathcal{P} = (\mathcal{K}, O)$ corresponds w.r.t. Abd to finding all minimal hitting sets of F, where F is the collection of negated models of $\mathcal{K} \cup \{\neg O\}$ w.r.t. Abd [11, 27].

Definition 2 (Minimal hitting set)**.** *Given a collection of sets F, a hitting set for F is a set H such that $H \cap S \neq \emptyset$ for every set $S \in F$. A hitting set H for F is minimal if there is no other hitting set H' for F such that $H' \subset H$.*

Definition 3 (Negated model w.r.t. Abd)**.** *Given a model $\mathcal{I}$, its negation w.r.t. Abd is a finite set of ABox axioms defined as $\{\neg\phi \in Abd \mid \mathcal{I} \models \phi\}$, where ϕ is an atomic (concept/role) assertion.*

2.2.1. Reiter's Minimal Hitting Set Algorithm

A classical algorithm for computing all minimal hitting sets, introduced by Reiter [27], was adapted for ABox abduction [11]. It builds a hitting set tree (HS-tree) in breadth-first order, where each node n is labelled by $\mathcal{L}(n)$ – a negated model of $\mathcal{K}' = \mathcal{K} \cup H(n) \cup \{\neg O\}$, where $H(n)$ is the set of edge-labels from the root to n. For each element $\sigma \in \mathcal{L}(n)$ there is a child n_σ of n where the edge (n, n_σ) is labelled by $\mathcal{L}(n, n_\sigma) = \sigma$. Each path $H(n)$ is thus a "candidate" hitting set, and its construction continues until it no longer can be extended, i.e., if $\mathcal{K} \cup H(n) \cup \{\neg O\}$ no longer has a model, then $H(n)$ is an explanation and the branch closes.

The algorithm uses three pruning conditions: the first two close nodes whose paths $H(n)$ are redundant w.r.t. other existing paths; the last one removes whole subtrees that are found to be unnecessary, because all hitting sets that could be found in them can also be found in another part of the tree. However, applying all the conditions can lead to loss of explanations [32]. One way to retain completeness is to omit the third condition at the cost of losing the optimization. With this applied, all explanations of size x are guaranteed to be found in the xth level of the tree. The algorithm though cannot detect when all explanations are found and it continues building the tree as long as new branches can be generated.

2.2.2. RC-Tree Algorithm

Several alternatives to Reiter's algorithm aim to address its incompleteness. One such method is the RC-Tree (RCT) algorithm [34].[2]

Instead of closing nodes with duplicate paths, RCT never creates such paths in the first place. Every node n is assigned an exclusion set $\theta(n)$, containing axioms that n cannot use to generate child edges. This includes axioms already used by n's ancestors and siblings, assuring that every path under n will be unique. Using this strategy, RCT tries to create a tree that is as small as possible, but still sufficient to find all explanations.

However, this means that removing a subtree with the third pruning condition would prevent some subsets of abducibles from ever being generated, even possible explanations. To prevent this, every pruning updates the θs in the tree – axioms that disappeared from the tree are removed from θs and they are used to generate new edges, no longer redundant. This way, unlike Reiter's algorithm, RCT can create branches in any level of the tree, not only the last one. This approach requires more memory (storing θs) and repeated tree traversal.

2.2.3. HST Algorithm

Wotawa's Hitting Set Tree (HST) algorithm [33], another variant of Reiter's method, also ensures that duplicate paths cannot be created. Instead of navigating through the search space based on the negated models, the algorithm systematically enumerates every possible combination of abducible axioms. However, at any given point, this only includes abducibles that have appeared in at least one negated model so far, since unused axioms cannot be part of any minimal hitting set (per Definition 2). To track this, every time a new axiom is encountered in a negated model, HST assigns it a unique integer index (the approach to generating all combinations uses integer intervals) . Unlike Reiter's and RC-Tree, HST

[2]Though these hitting set algorithms are not adapted for DL abduction, we use abduction-related terminology for simplicity.

uses negated models only for this purpose. Once all abducibles are numbered, the models are no longer needed to be stored nor obtained anew.

2.2.4. MHS-MXP Algorithm

Reiter's algorithm also inspired MHS-MXP [28], the only approach here specifically designed for ABox abduction. It leverages the MergeXplain (MXP) algorithm [29], which finds minimal inconsistent axiom sets using repeated calls to QuickXplain (QXP) [35]. QXP is much faster than Reiter's method but can find only one explanation. MXP improves this by finding multiple explanations but still cannot guarantee completeness.

MHS-MXP builds a HS-tree, but instead of consistency checks, it calls MXP using the current node's path. This allows MHS-MXP to: (1) find multiple explanations per node; (2) discover explanations of size x before reaching tree depth x; (3) and skip generating branches that cannot lead to valid explanations. Thanks to the third property, the algorithm can determine that all possible explanations have already been found and terminate even before the whole search space has been explored.

3. CATS solver

CATS (Comenius Abduction Team Solver) is a new version of the MHS-MXP solver, the original Java implementation of MHS-MXP [28] that also supported Reiter's MHS algorithm. It introduces significant changes to the code-base and functionality, most notably, a wider collection of abduction algorithms.

We implemented RC-Tree and HST, adapting them to the context of ABox abduction for the first time. Analogously to MHS-MXP being based on MHS, we also "hybridized" RCT and HST, introducing two brand new algorithms: RCT-MXP and HST-MXP. Additionally, we implemented the two fast, but incomplete methods, QuickXplain and MergeXplain, which were already present in the code as components of MHS-MXP. This brings the total number of available algorithms to 8.

The solver uses the OWL API to delegate consistency checks to an external reasoner. In the case of a successful consistency check, the algorithms require a representation of a model found by the reasoner. This functionality is currently implemented only by JFact [28].

3.1. Usage

An abduction problem and run parameters (such as the algorithm to be used, the abducibles, etc.) can be passed to the solver using either of the following: (1) a structured input file and a command-line application; (2) a graphical application; (3) a programmatical Java API [40]. The explanations found are saved into log files. Unlike the previous version, CATS log files track detailed statistics about the solver's run, such as memory used, counts of created nodes, branches, prunings, consistency checks, etc.

Abducibles can be set either directly or by specifying allowed individuals and concepts. If none are provided, all combinations of $C(a)$ and $\neg C(a)$ from the background knowledge are used; negations can be disabled to reduce their number. Solver performance depends heavily on abducible size, which increases computational load. Additionally, MHS-MXP has been shown to perform poorly with negations, as MXP identifies any set containing both $C(a)$ and $\neg C(a)$ as a possible explanation [28].

3.2. Implementation

The MHS-MXP solver implementation kept the code of the two original algorithms in the same class, differentiating between them with `if-else` blocks. This, however, made it hard to reasonably extend the code. We reworked this basic approach using the *composition-over-inheritance* [41] principle, creating a modular architecture where the solver's behaviour is composed of multiple objects realizing abstract interfaces.

The main interfaces are `ITreeBuilder`, which determines how the HS-tree is built, and `INodeProcessor`, which handles what should occur in each node – i.e., a consistency check, an MXP or QXP call. Consequently, any algorithm can be trivially hybridized by combining its tree builder with the MXP node processor. Simultaneously, the `RootOnlyTreeBuilder` class, used in the MXP and QXP implementation, can be used to run an operation just once. This provides the flexibility to abstract away from the HS-tree and implement any abduction algorithm with inputs and outputs matching the interfaces.

Apart from refactoring the code-base, we also identified multiple bugs. Our main focus was on the memory errors encountered in a past evaluation [31]. We discovered the culprit in the process of storing negated models, which created large numbers of redundant objects. A new approach is used in CATS, fixing the error and making the solver suitable for experiments on bigger ontologies than before.

The implementation includes various optimizations. Most notably, MHS and HST, alongside their hybrid versions, don't have to remember the whole HS-tree – they only store unprocessed nodes (those awaiting expansion and checks). In the case of MHS, this was already the case in the MHS-MXP solver and was made possible thanks to the omission of Reiter's third pruning condition. We implemented multiple other optimizations to prevent redundant objects and actions.

Some other changes in the new version include: more accurate time measuring; more reliable time-out mechanism; and automated tests, previously fully absent, to verify the algorithms' correctness and detect mistakes during development.

4. Evaluation

4.1. Methodology

We used the dataset of ABox abduction problems[3] from the previous evaluation [28]. The abduction problems for the evaluation were grouped as S_1–S_5, where each S_i contains problems with explanations of size i. All problems share the same background knowledge (the LUBM ontology [36]) and have the same two sets of abducibles: $\{C(a), \neg C(a) \mid a \in N_I, C \in N_C\}$ (with negations) and $\{C(a) \mid a \in N_I, C \in N_C\}$ (without negations), using the vocabulary from the background knowledge and the given observation. The former set of abducibles is considered "favourable" for the hybrids and the latter "unfavourable" (see Section 3.1). Later groups are not inherently harder, but larger explanations mean more of the search space must be explored and thus a deeper HS-tree must be built. Each problem has the same set of (desired) explanations, regardless of whether negations are allowed or not.

For each group, 5 problems were selected. Each algorithm ran twice with and twice without negations. The time limit was reduced from 4 to 2 hours, as in the previous evaluation, not much of interesting data was obtained from the runs after 2 hours.

The solver[4] was run on a server with the Ubuntu 24.04.1 operating system, using Java version 1.8.0_412. The server had 24 2.2 GHz processors and 64GB of RAM. Each run of the solver had 4GB of memory allocated to the JVM.

4.2. Results and Interpretation

4.2.1. Number of Explanations over Time

Figures 2 and 3 show the average number of explanations found over time without and with negations, respectively. Each figure presents results for input groups S_1–S_5 and then all inputs combined. The X-axis shows time (seconds), and each curve (one per algorithm) tracks the cumulative number of explanations found (Y-axis), averaged over all runs in the group.

Each graph is scaled to highlight the most relevant data. The Y-axis adjusts to the graph's maximum value, while the X-axis ends at either the 7200-second limit or when all curves plateau. An early X-axis

[3]Evaluation inputs: https://github.com/Comenius-Abduction-Team/CATS-Abduction-Solver/tree/main/in/dl2025
[4]CATS version 2.0.0: https://github.com/Comenius-Abduction-Team/CATS-Abduction-Solver/releases/tag/v2.0.0

cutoff does not imply early termination – only that no further explanations were found. However, each curve ends at the maximal runtime encountered for that algorithm: if it is present for the whole 2 hours, the algorithm hit the time-out at least once (see Section 4.2.2 for more details). If it breaks earlier, no run of that algorithm lasted longer than the curve's endpoint.

Without Negations. In S_1, MXP and all hybrid algorithms found all explanations around the same time, slightly ahead of base algorithms (MHS, HST, RC-Tree), confirming MXP's effectiveness at identifying all size-1 explanations in the root.

From S_2 on, base algorithms begin to lag. HST-MXP is also significantly slower by S_3 and barely progressing in S_5 (39.8 explanations by 5106.5s). MHS-MXP and RCT-MXP overall perform similarly to each other: RCT-MXP is faster in S_4 and leads early in S_5, but plateaus around 3000–4000s, allowing MHS-MXP to pull ahead. This suggests RCT-MXP may scale less efficiently on deeper HS-trees due to its need to store and traverse the entire structure.

Among base algorithms, RC-Tree performed the best in S_2 and S_3, while HST was the slowest and least effective. None of the base algorithms found any explanations in S_4 or S_5. This highlights a limitation of previous evaluations: while they showed MHS could not guarantee all size-4 explanations before timeout, they did not reveal whether any size-4 explanations were found.

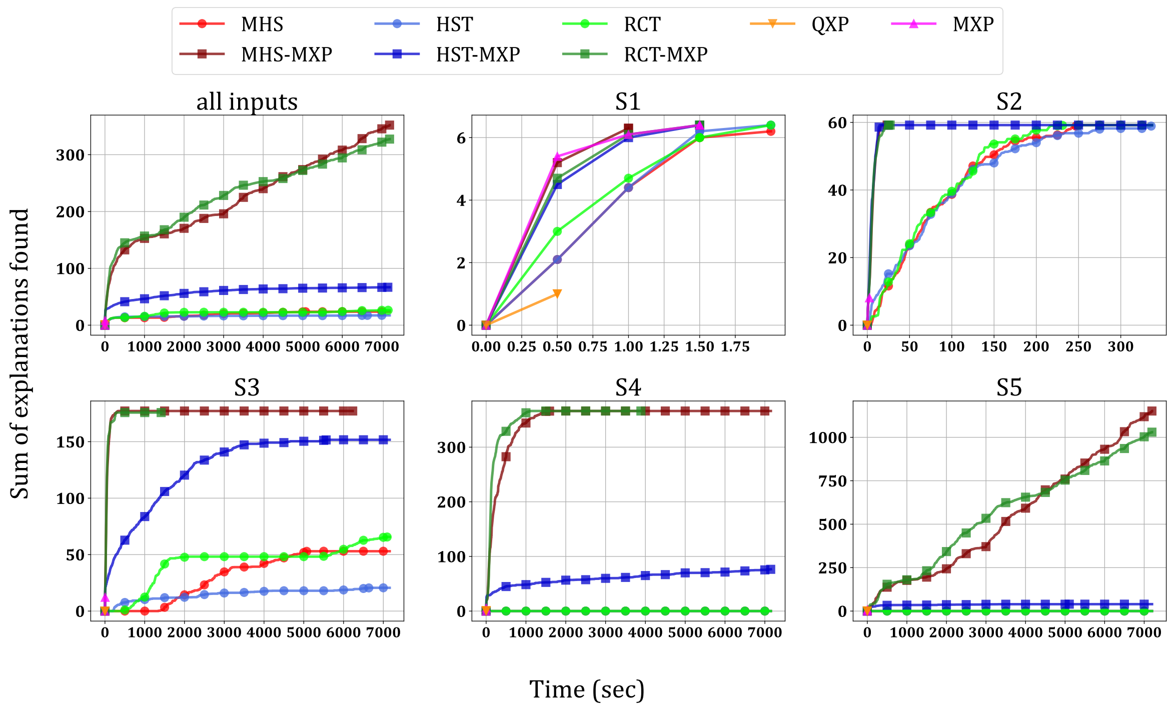

Figure 2: Average explanations found over time, cases without negations

With Negations. With negations allowed, base algorithms were generally faster than hybrids and MXP in S_1 and S_2, echoing previous results – except for HST, which lagged behind all complete algorithms in S_2. In S_3, it needed nearly two hours to match incomplete MXP's output. This reinforces HST as likely the least effective complete algorithm. Upon further inspection, we found that HST usually assigned number indices to all of the abducibles right at the beginning of its runs. As it generates combinations of numbered abducibles, this essentially turns HST into a brute-force method, generating all possible combinations of abducibles without using any strategy or heuristics.

Earlier evaluations portrayed MHS-MXP as consistently worse than MHS in cases with negations. This setting is indeed disadvantageous for hybrids, leading to low explanation counts and slow initial growth compared to cases without negations. However, from S_3 onward, hybrids proved faster and more productive than base algorithms, which found no explanations of size 4 or 5. Though neither guaranteed completeness, hybrids clearly performed better.

Notably, HST-MXP – generally the weakest hybrid – led among hybrids during a 1000–2000s window, and was the fastest in S_2 and S_4. While RCT-MXP found slightly more total explanations, HST-MXP may be more effective when prioritizing early results in a limited time. QXP failed to find any explanations, illustrating its low effectiveness with non-trivial abducibles.

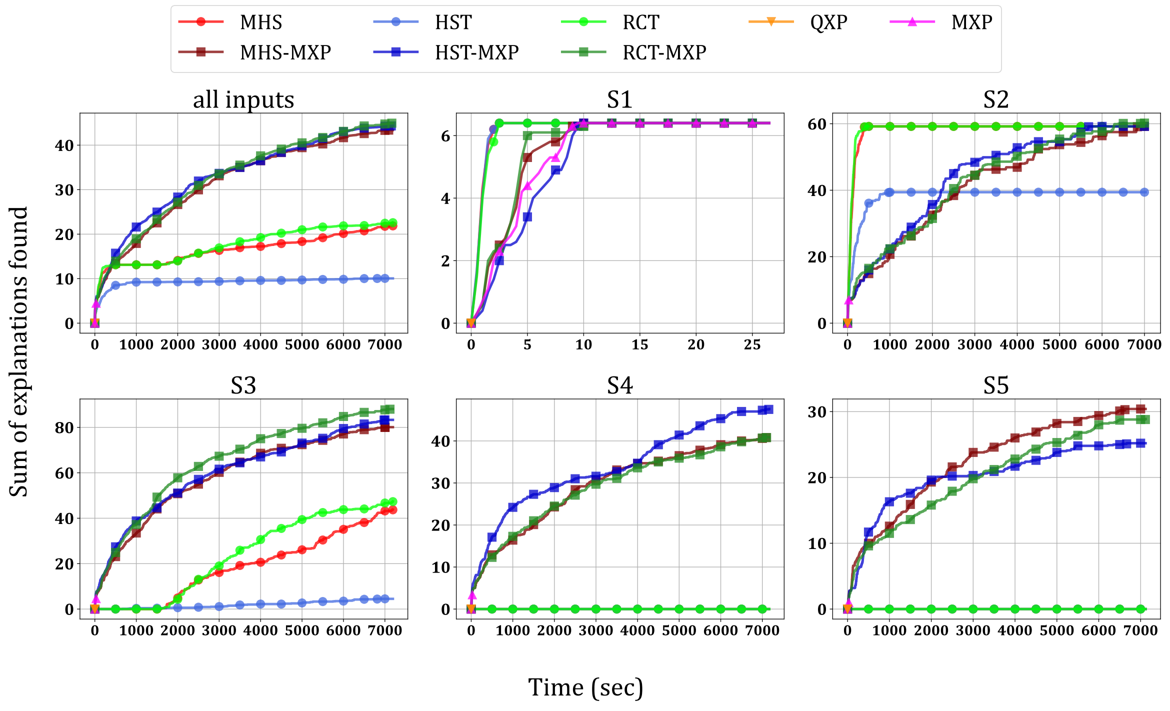

Figure 3: Average explanations found over time, cases with negations

4.2.2. Early Termination

In practice, it matters not only how many explanations an algorithm finds, but also whether it can terminate naturally instead of running unnecessarily long after all explanations are found.

QXP and MXP always terminated early, while all base algorithm runs and all hybrid runs with negations hit the time-out. Only hybrids without negations occasionally finished early (see Table1). Since HST-MXP usually timed out, we focused on MHS-MXP and RCT-MXP. Their average run times were similar, with RCT-MXP slightly faster in S_2 and S_5, and notably faster in S_3. As it also finished early in two more cases, it appears marginally more effective in this respect.

Table 1
Number of runs of hybrid algorithms that ended before time-out and average finish time (sec) of MHS-MXP and RCT-MXP runs that terminated before the time limit

	Number of runs ended before time-out			Avg time before time-out	
Input group	MHS-MXP	HST-MXP	RCT-MXP	MHS-MXP	RCT-MXP
S_1	10	10	10	0.68	0.75
S_2	10	0	10	9.76	9.04
S_3	10	1	10	1205.04	391.07
S_4	8	0	10	1487.67	1744.87
S_5	2	0	2	767.51	667.75

Note that MXP can always find all explanations of size 1. This means that in S_1 cases, the hybrids are able to find all explanations in their root and ideally should terminate almost immediately. However, negations in abducibles are in conflict with the hybrids' ability to terminate early, causing them to always hit time-out in S_1 with negations enabled. As it stands, MXP is the only algorithm that could find all explanations and then terminate in S_1 with negations.

4.2.3. Memory Usage

To plot memory usage over time, we recorded the average memory per HS-tree level. We applied linear interpolation between checkpoints for each run, then averaged the results across all runs – separately for cases with and without negations. Input groups were not distinguished, as their differences were minimal.

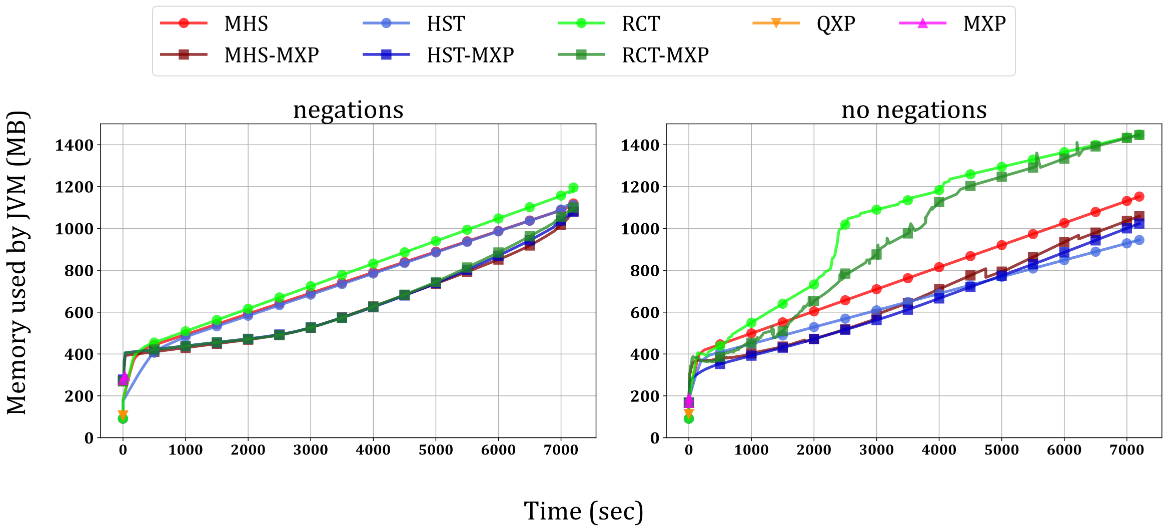

Figure 4: Average memory used by the Java Virtual Machine over time

The methodology is reflected in the graphs: cases with negations show smooth curves due to longer times needed to complete tree levels, resulting in fewer records. Most records appear early or at timeout, with the rest interpolated. This is especially true for the curves that are mostly straight (MHS, HST and HST-MXP with negations; all base algorithms without negations) – these algorithms had almost no records between 1000 seconds and the timeout.

Without negations, faster level completions yield more real data, producing more uneven curves. RC-Tree and RCT-MXP, which revisit levels frequently, have the jaggedest curves.

Without negations, algorithms fall into three groups: RCT(-MXP) uses the most memory and MHS(-MXP) slightly more than HST(-MXP). Interestingly, hybrids generally start less memory demanding than their base versions, but the gap narrows over time. With negations, RC-Tree and RCT-MXP are still the most memory-intensive, though the differences are smaller. Once again, base algorithms use more memory than hybrids.

RCT(-MXP)'s high memory use is expected due to storing the full HS-tree and more data per node. HST(-MXP)'s lower usage than MHS(-MXP) is surprising, possibly tied to pruning behaviour (never duplicates a path) or model storage, though causes are unclear without analysing the deeper metrics.

Although we mostly focus on the complete algorithms, it is practical to explore properties of MXP in the S_1 cases, where it can always find all explanations. Table 2 shows average memory usage of complete algorithms in S_1. As MXP does not need to construct a tree, it is obviously less memory-heavy than the others. The exception is the hybrids without negations. In those cases, hybrids can terminate right after the root and essentially are the same as MXP.

Table 2
Average total memory usage in the S_1 group (in MB)

Negations	MHS	HST	RCT	MHS-MXP	HST-MXP	RCT-MXP	**MXP**
no	767.31	667.74	953.83	152.75	153.78	151.92	152.44
yes	750.16	797.84	787.22	625.55	550.46	625.74	284.47

5. Conclusions

We have described CATS Solver, a new ABox abduction solver for DL implementing eight algorithms. CATS is based on the JFact reasoner as a black box, and it supports any DL expressivity up to $\mathcal{SROIQ}$/OWL 2. The modular implementation of the solver enables the user to freely choose an algorithm best suited for a given application. For instance, if just one of a few explanations needs to be found, the incomplete but fast methods QuickXplain and MergeXplain may be used. If one wishes to find as many explanations as possible, one of the five complete algorithms may be used. The solver offers a command-line and graphical user interface, together with an API Java library.

The modular implementation of the algorithms also enables us to compare them. To this end, we have conducted an empirical evaluation of all eight algorithms on a dataset derived from the LUBM ontology. The results showed that out of the classic algorithms, including Reiter's MHS, Wotawa's HST and Pill and Quaritch's RC-Tree performed the best most of the time, followed by MHS, with HST performing constantly worse than the two. As a more interesting result, all three hybrid algorithms perform significantly better than any of the three classic versions. The difference in performance of the hybrids is much more fine-grained with HST-MXP being the slowest most of the time (but also the fastest in a small number of cases), while MHS-MXP and RCT-MXP are much harder to distinguish between.

Importantly, contrary to the previous evaluation [28], the new evaluation methodology focusing on the progress of explanations being found in time (and possibly a number of optimizations that were implemented) helped us to clearly demonstrate that the hybrid algorithms are always performing better, even on less favourable use cases.

In the future, we plan [42] to conduct a more detailed evaluation on real-world ontologies. The performance of the solver may also possibly be improved by integration with other reasoners than JFact.

Acknowledgments

The authors would like to express their gratitude to Ján Kľuka for his valuable insights and helpful discussions. This research was funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I05-03-V02-00064.

Declaration on Generative AI

During the preparation of this work, the authors used Grammarly and gpt-4o in order to: Grammar and spelling check, Improve writing style. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] C. S. Peirce, Illustrations of the logic of science VI: Deduction, induction, and hypothesis, Popular Science Monthly 13 (1878) 470–482.

[2] 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*, CEUR-WS.org, 2006. URL: https://ceur-ws.org/Vol-216/submission_25.pdf.

[3] S. Colucci, T. D. Noia, E. D. Sciascio, F. M. Donini, M. Mongiello, Concept abduction and contraction in description logics, in: Proceedings of the 2003 International Workshop on Description Logics (DL2003), Rome, Italy September 5-7, 2003, volume 81 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2003. URL: https://ceur-ws.org/Vol-81/donini.pdf.

[4] M. Bienvenu, Complexity of abduction in the $\mathcal{EL}$ family of lightweight description logics, in: Principles of Knowledge Representation and Reasoning: Proceedings of the Eleventh International Conference, KR 2008, Sydney, Australia, September 16-19, 2008, AAAI Press, 2008, pp. 220–230. URL: http://www.aaai.org/Library/KR/2008/kr08-022.php.

[5] T. D. Noia, E. D. Sciascio, F. M. Donini, A tableaux-based calculus for abduction in expressive description logics: Preliminary results, in: Proceedings of the 22nd International Workshop on Description Logics (DL 2009), Oxford, UK, July 27-30, 2009, volume 477 of *CEUR-WS*, CEUR-WS.org, 2009. URL: https://ceur-ws.org/Vol-477/paper_52.pdf.

[6] S. Klarman, U. Endriss, S. Schlobach, ABox abduction in the description logic $\mathcal{ALC}$, Journal of Automated Reasoning 46 (2011) 43–80. URL: https://doi.org/10.1007/s10817-010-9168-z. doi:10.1007/S10817-010-9168-Z.

[7] J. Du, G. Qi, Y. Shen, J. Z. Pan, Towards practical ABox abduction in large OWL DL ontologies, in: Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, US, August 7-11, 2011, AAAI Press, 2011, pp. 1160–1165. URL: https://doi.org/10.1609/aaai.v25i1.8070. doi:10.1609/AAAI.V25I1.8070.

[8] K. Halland, K. Britz, Abox abduction in $\mathcal{ALC}$ using a DL tableau, in: 2012 South African Institute of Computer Scientists and Information Technologists Conference, SAICSIT '12, Pretoria, South Africa, ACM, 2012, pp. 51–58. URL: https://doi.org/10.1145/2389836.2389843. doi:10.1145/2389836.2389843.

[9] J. Du, G. Qi, Y. Shen, J. Z. Pan, Towards practical ABox abduction in large description logic ontologies, Int. J. Semantic Web Inf. Syst. 8 (2012) 1–33. URL: https://doi.org/10.4018/jswis.2012040101. doi:10.4018/JSWIS.2012040101.

[10] J. Du, K. Wang, Y. Shen, 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., AAAI Press, 2014, pp. 1034–1040. URL: https://doi.org/10.1609/aaai.v28i1.8852. doi:10.1609/AAAI.V28I1.8852.

[11] J. Pukancová, M. Homola, Tableau-based ABox abduction for the $\mathcal{ALCHO}$ description logic, in: A. Artale, B. Glimm, R. Kontchakov (Eds.), Proceedings of the 30th International Workshop on Description Logics, Montpellier, France, July 18-21, 2017, volume 1879 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2017. URL: http://ceur-ws.org/Vol-1879/paper11.pdf.

[12] W. Del-Pinto, R. A. Schmidt, Forgetting-based abduction in $\mathcal{ALC}$, in: Proceedings of the Workshop on Second-Order Quantifier Elimination and Related Topics (SOQE 2017), Dresden, Germany, volume 2013 of *CEUR-WS*, CEUR-WS.org, 2017, pp. 27–35. URL: https://ceur-ws.org/Vol-2013/paper13.pdf.

[13] 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. URL: https://doi.org/10.24963/kr.2020/59. doi:10.24963/KR.2020/59.

[14] P. Koopmann, Signature-based abduction with fresh individuals and complex concepts for description logics, in: Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021, ijcai.org, 2021, pp. 1929–1935. URL: https://doi.org/10.24963/ijcai.2021/266. doi:10.24963/IJCAI.2021/266.

[15] P. Lambrix, Completing and debugging ontologies: State-of-the-art and challenges in repairing ontologies, ACM J. Data Inf. Qual. 15 (2023) 41:1–41:38. URL: https://doi.org/10.1145/3597304. doi:10.1145/3597304.

[16] K. Schekotihin, P. Rodler, W. Schmid, OntoDebug: Interactive ontology debugging plug-in for Protégé, in: Foundations of Information and Knowledge Systems - 10th International Symposium, FoIKS 2018, Budapest, Hungary, May 14-18, 2018, Proceedings, volume 10833 of *LNCS*, Springer, 2018, pp. 340–359. URL: https://doi.org/10.1007/978-3-319-90050-6_19. doi:10.1007/978-3-319-90050-6_19.

[17] F. Wei-Kleiner, Z. Dragisic, P. Lambrix, Abduction framework for repairing incomplete $\mathcal{EL}$ ontologies: Complexity results and algorithms, in: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada., AAAI Press, 2014, pp. 1120–1127. URL: https://doi.org/10.1609/aaai.v28i1.8858. doi:10.1609/AAAI.V28I1.8858.

[18] S. M. Rashid, J. P. McCusker, D. M. Gruen, O. Seneviratne, D. L. McGuinness, A concise ontology to support research on complex, multimodal clinical reasoning, in: The Semantic Web - 20th International Conference, ESWC 2023, Hersonissos, Crete, Greece, May 28 - June 1, 2023, Proceedings, volume 13870 of *Lecture Notes in Computer Science*, Springer, 2023, pp. 390–407. URL: https://doi.org/10.1007/978-3-031-33455-9_23. doi:10.1007/978-3-031-33455-9_23.

[19] C. Martini, Abductive reasoning in clinical diagnostics, in: L. Magnani (Ed.), Handbook of Abductive Cognition, Springer, 2023, pp. 467–479. doi:10.1007/978-3-031-10135-9_13.

[20] M. Obeid, Z. Obeid, A. Moubaiddin, N. Obeid, Using description logic and ABox abduction to capture medical diagnosis, in: Advances and Trends in Artificial Intelligence. From Theory to Practice - 32nd International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2019, Graz, Austria, July 9-11, 2019, Proceedings, volume 11606 of *LNCS*, Springer, 2019, pp. 376–388. URL: https://doi.org/10.1007/978-3-030-22999-3_33. doi:10.1007/978-3-030-22999-3_33.

[21] J. Pukancová, M. Homola, Abductive reasoning with description logics: Use case in medical diagnosis, in: Proceedings of the 28th International Workshop on Description Logics (DL 2015), Athens, Greece, volume 1350 of *CEUR-WS*, CEUR-WS.org, 2015. URL: https://ceur-ws.org/Vol-1350/paper-60.pdf.

[22] D. Al-Darras, B. Al-Shboul, N. Obeid, Towards using ontology-based systems for explainable medical diagnosis in nutrition domain, in: 12th International Conference on Information Technology, ICIT 2025, Amman, Jordan, May 27-30, 2025, IEEE, 2025, pp. 528–533. URL: https://doi.org/10.1109/ICIT64950.2025.11049209. doi:10.1109/ICIT64950.2025.11049209.

[23] T. Hubauer, C. Legat, C. Seitz, Empowering adaptive manufacturing with interactive diagnostics: A multi-agent approach, in: Advances on Practical Applications of Agents and Multiagent Systems – 9th International Conference on Practical Applications of Agents and Multiagent Systems, PAAMS 2011, Salamanca, Spain, volume 88 of *Advances in Intelligent and Soft Computing*, Springer, 2011, pp. 47–56. URL: https://doi.org/10.1007/978-3-642-19875-5_6. doi:10.1007/978-3-642-19875-5_6.

[24] O. Gries, R. Möller, A. Nafissi, M. Rosenfeld, K. Sokolski, M. Wessel, A probabilistic abduction engine for media interpretation based on ontologies, in: Web Reasoning and Rule Systems - Fourth International Conference, RR 2010, Bressanone/Brixen, Italy, September 22-24, 2010. Proceedings, volume 6333 of *Lecture Notes in Computer Science*, Springer, 2010, pp. 182–194. URL: https://doi.org/10.1007/978-3-642-15918-3_15. doi:10.1007/978-3-642-15918-3_15.

[25] S. Colucci, T. D. Noia, E. D. Sciascio, F. M. Donini, M. Mongiello, Concept abduction and contraction for semantic-based discovery of matches and negotiation spaces in an e-marketplace, Electronic Commerce Research and Applications 4 (2005) 345–361. URL: https://doi.org/10.1016/j.elerap.2005.06.004. doi:10.1016/J.ELERAP.2005.06.004.

[26] P. Rodler, How should I compute my candidates? A taxonomy and classification of diagnosis computation algorithms, in: ECAI 2023 - 26th European Conference on Artificial Intelligence, September 30 - October 4, 2023, Kraków, Poland - Including 12th Conference on Prestigious Applications of Intelligent Systems (PAIS 2023), volume 372 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2023, pp. 1986–1993. URL: https://doi.org/10.3233/FAIA230490. doi:10.3233/FAIA230490.

[27] R. Reiter, A theory of diagnosis from first principles, Artificial intelligence 32 (1987) 57–95. URL: https://doi.org/10.1016/0004-3702(87)90062-2. doi:10.1016/0004-3702(87)90062-2.

[28] M. Homola, J. Pukancová, J. Boborová, I. Balintová, Merge, explain, iterate: A combination of MHS and MXP in an ABox abduction solver, in: Logics in Artificial Intelligence - 18th European Conference, JELIA 2023, Dresden, Germany, September 20-22, 2023, Proceedings, volume 14281 of *Lecture Notes in Computer Science*, Springer, 2023, pp. 338–352. URL: https://doi.org/10.1007/978-3-031-43619-2_24. doi:10.1007/978-3-031-43619-2_24.

[29] 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, July 25-31, 2015, AAAI Press, 2015, pp. 3221–3228. URL: http://ijcai.org/Abstract/15/454.

[30] J. Pukancová, M. Homola, The AAA ABox abduction solver, Künstliche Intell. 34 (2020) 517–522. URL: https://doi.org/10.1007/s13218-020-00685-4. doi:10.1007/S13218-020-00685-4.

[31] J. Boborová, Optimization of the MHS-MXP algorithm, Master's thesis, Comenius University in Bratislava, 2023.

[32] R. Greiner, B. A. Smith, R. W. Wilkerson, A correction to the algorithm in Reiter's theory of diagnosis, Artif. Intell. 41 (1989) 79–88. URL: https://doi.org/10.1016/0004-3702(89)90079-9. doi:10.1016/0004-3702(89)90079-9.

[33] F. Wotawa, A variant of Reiter's hitting-set algorithm, Inf. Process. Lett. 79 (2001) 45–51. URL: https://doi.org/10.1016/S0020-0190(00)00166-6. doi:10.1016/S0020-0190(00)00166-6.

[34] I. Pill, T. Quaritsch, RC-Tree: A variant avoiding all the redundancy in Reiter's minimal hitting set algorithm, in: 2015 IEEE International Symposium on Software Reliability Engineering Workshops, ISSRE Workshops, Gaithersburg, MD, USA, November 2-5, 2015, IEEE Computer Society, 2015, pp. 78–84. URL: https://doi.org/10.1109/ISSREW.2015.7392050. doi:10.1109/ISSREW.2015.7392050.

[35] 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, July 25-29, 2004, San Jose, California, USA, AAAI Press / The MIT Press, 2004, pp. 167–172. URL: http://www.aaai.org/Library/AAAI/2004/aaai04-027.php.

[36] Y. Guo, Z. Pan, J. Heflin, LUBM: A benchmark for OWL knowledge base systems, Journal of Web Semantics 3 (2005) 158–182. URL: https://doi.org/10.1016/j.websem.2005.06.005. doi:10.1016/J.WEBSEM.2005.06.005.

[37] M. Schmidt-Schauß, G. Smolka, Attributive concept descriptions with complements, Artificial intelligence 48 (1991) 1–26. URL: https://doi.org/10.1016/0004-3702(91)90078-X. doi:10.1016/0004-3702(91)90078-X.

[38] 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.

[39] S. Rudolph, Foundations of description logics, in: Reasoning Web. Semantic Technologies for the Web of Data - 7th International Summer School 2011, Galway, Ireland, August 23-27, 2011, Tutorial Lectures, volume 6848 of *Lecture Notes in Computer Science*, Springer, 2011, pp. 76–136. URL: https://doi.org/10.1007/978-3-642-23032-5_2. doi:10.1007/978-3-642-23032-5_2.

[40] J. Kloc, M. Homola, J. Pukancová, DL abduction API v2 and GUI interface (extended abstract), in: Proceedings of the 36th International Workshop on Description Logics (DL 2023) co-located with the 20th International Conference on Principles of Knowledge Representation and Reasoning and the 21st International Workshop on Non-Monotonic Reasoning (KR 2023 and NMR 2023)., Rhodes, Greece, September 2-4, 2023, volume 3515 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023. URL: https://ceur-ws.org/Vol-3515/abstract-15.pdf.

[41] E. Gamma, R. Helm, R. Johnson, J. M. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Professional, 1994.

[42] J. Boborová, J. Kloc, M. Homola, J. Pukancová, On the way to diverse datasets for evaluating ABox abduction algorithms (extended abstract), in: Proceedings of the 38th International Workshop on

Description Logics (DL 2025), Opole, Poland, September 3–6, 2025, 2025. To appear.