Towards CATS: A Modular ABox Abduction Solver Based on Black-Box Architecture (Extended Abstract)

Janka Boborová, Jakub Kloc, Martin Homola and Júlia Pukancová

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

Abstract
CATS, standing for *Comenius Abduction Team Solver*, is an ABox abduction solver for description logics that allows the users to choose among multiple abduction algorithms. The solver integrates the JFact reasoner as a black-box, thanks to which it handles expressivity up to $\mathcal{SROIQ}$ (OWL 2). It handles inputs and outputs via OWL API and it offers a command line, GUI, and API interface.

Keywords
Abduction, description logics, ontologies, solver

1. Introduction

Abduction [1] is a type of hypothetical inference aiming to explain an observation using the background knowledge of the problem domain. We specifically deal with ABox abduction [2]: Given a finite set of ABox assertions Abd (hypotheses), a knowledge base $\mathcal{K}$ and an ABox assertion $\mathcal{O}$ (observation), an ABox abduction problem is a pair $\mathcal{P} = (\mathcal{K}, \mathcal{O})$ and a solution of $\mathcal{P}$ (explanation) is a finite set of ABox assertions $\mathcal{E} \subseteq \mathrm{Abd}$ such that $\mathcal{K} \cup \mathcal{E} \models \mathcal{O}$. We also require explanations to have standard properties [2] such as minimality. In our work, $\mathcal{O}$ can include any concept or role assertion (possibly multiple ones), which may involve even complex concepts and negated roles. To prevent an infinite hypothesis space, Abd consists of only atomic (concept and role) assertions and their complements involving any named individuals from $\mathcal{O}$ or $\mathcal{K}$.

In the following example, we define an ABox abduction problem $\mathcal{P}_T = (\mathcal{K}_T, \mathcal{O}_T)$ to determine the potential causes of John's toothache.

$$\mathcal{K}_T = \{\mathsf{DentalTrauma} \sqsubseteq \mathsf{Toothache}, \mathsf{Cavity} \sqsubseteq \mathsf{SensitiveTeeth},$$
$$\mathsf{SensitiveTeeth} \sqcap \mathsf{DrankColdDrink} \sqsubseteq \mathsf{Toothache}\},$$
$$\mathcal{O}_T = \{\mathsf{Toothache}(\mathsf{john})\}.$$

All desired explanations of $\mathcal{P}_T$ are: $\mathcal{E}_1 = \{\mathsf{DentalTrauma}(\mathsf{john})\}$; $\mathcal{E}_2 = \{\mathsf{DrankColdDrink}(\mathsf{john}),$ $\mathsf{Cavity}(\mathsf{john})\}$; and $\mathcal{E}_3 = \{\mathsf{DrankColdDrink}(\mathsf{john}), \mathsf{SensitiveTeeth}(\mathsf{john})\}$.

There are some abduction solvers available [3, 4, 5, 6], however, they often offer only a single abduction algorithm and most of them are limited w.r.t. the expressivity of description logics (DLs) they are able to work with.

We introduce CATS, an ABox abduction solver for DLs with multiple alternate algorithms. The solver builds on our previous experience with ABox abduction, particularly the black-box architecture based on model extraction [7]. This allows us to plug-in an external DL reasoner from which a relevant part of a model can be extracted. The resulting solver can thus handle any expressivity the DL reasoner can.

The following abduction algorithms are currently implemented: (i) MHS: Reiter's classic algorithm [8] that gradually searches the hypothesis space by building a tree structure called the HS-tree. It is

known to be incomplete [9], though our version does not use the third pruning condition to regain completeness. (ii) HST: Wotava's variant of MHS [10]. It generates the HS-tree in a way that prevents duplicate paths from existing. This eliminates the need for Reiter's second pruning condition and thus avoids performing numerous subset checks while building the tree. (iii) MXP: A highly efficient but incomplete method by Shchekotykhin et al. [11], relying on the *divide and conquer* principle. Currently, it is the only incomplete method in CATS. (iv) MHS-MXP: A hybrid combination of MHS and MXP [7]. MXP can find multiple explanations in each node of the HS-tree and possibly prune the tree to reduce its size and terminate the algorithm faster. (v) HST-MXP: An analogous hybrid combination of HST and MXP.

CATS[1] is a freely available software written in Java, using the OWL API [12] to work with DL objects.

2. Black-Box Integration with a DL Reasoner

To solve ABox abduction in a complete way, MHS and the algorithms derived from it require to run repeated knowledge base consistency checks. Additionally, after each successful consistency check, the algorithms need to obtain the ABox assertions that are satisfied in a model of the knowledge base to navigate the search for explanations in the hypothesis space [8]. This latter functionality is referred to as *model extraction*. In a truly black-box setting, we wish to delegate these tasks to an external DL reasoner.

While consistency checking is a standard task for any DL reasoner, model extraction is not. Not all DL reasoners need to construct a model (or its sufficient representation) to check knowledge base consistency. But for instance, tableau-based DL reasoners [13, 14, 15] build a finite representation of a model (dubbed completion graph). This representation is sufficient [16] to extract the facts required in the basic ABox abduction setting, namely the set of all atomic concept and role assertions that are satisfied in the model.

CATS currently uses a modified version of the JFact reasoner[2] [7]. To our best knowledge, this is the only DL reasoner to date that supports the `OWLKnowledgeExplorerReasoner` interface[3] of the OWL API that enables to read the completion graph. Possibly, other tableau-based DL reasoners could implement this OWL API interface in the future. This would grant CATS the potential to experiment with different reasoners in a modular way, to compare their capabilities, and to give the user the option to choose between them.

3. CATS Solver

CATS supports any OWL 2 [17] ontology as the background knowledge of an abduction problem. It currently implements the abduction algorithms listed above. The option to choose from different algorithms can be utilised in multiple ways. Firstly, CATS may be used to empirically compare their performance. Secondly, the user can choose an algorithm that is most suitable for their needs, as each algorithm has an advantage over the others in specific situations. For example, when the user wants to obtain any set of explanations *quickly*, it is best to use MXP. When all explanations are required, different algorithms are necessary. MHS-MXP can be very efficient if we limit the hypothesis space to atomic assertions only. If negations are included, it is better to use MHS or HST.

The solver also provides various options for configuring the abduction problem, such as the maximal length of the explanations found, the maximal duration of solving, or limiting what symbols or axioms the explanations can include.

By default, CATS is a command-line application that uses structured text files as its inputs. Let us show an example of CATS's input using the problem $\mathcal{P}_T$ from the introduction:

---

[1]https://github.com/Comenius-Abduction-Team/CATS-Abduction-Solver
[2]https://github.com/boborova3/jfact
[3]https://owlcs.github.io/owlapi/apidocs_4/org/semanticweb/owlapi/reasoner/knowledgeexploration/OWLKnowledgeExplorerReasoner.html

```
-f: files/toothache_ontology.owl
-o: Prefix: pref: <http://www.semanticweb.org/janbo/ontologies/2024/4/
    untitled-ontology-10#> Class: pref:Toothache
    Individual: pref:john Types: pref:Toothache
-alg: MHS-MXP
```

$\mathcal{K}_T$ was processed into `toothache_ontology.owl`. For readability, $\mathcal{O}_T$ is split into three lines, but in a proper input file, it would have to be in one. To compute explanations, MHS-MXP will be used. This example can be run using command `java -jar cats.jar <relative path to the input file>`. Subsequently, we obtain an output:

```
1; 1; 0,05; {{DentalTrauma(john)}}
2; 2; 0,05; {{Cavity(john),DrankColdDrink(john)},
            {SensitiveTeeth(john),DrankColdDrink(john)}}
0,08
```

The last output line contains the total running time. Other lines are in the form: `<length n>;<number of explanations>;<tree level completion time>; {<found explanations of the length n>}`. As we can see, with MHS-MXP, we found all explanations $\mathcal{E}_1$–$\mathcal{E}_3$ in 0.08 seconds. Because this is a small example, we cannot demonstrate the differences between the algorithms in time efficiency. We would get the same result using MHS, HST and HST-MXP. The only difference is in MXP, which could not find $\mathcal{E}_3$.

CATS can be also used via a graphical interface in the DL Abduction App[4] [18], which can be seen in Figure 1. Another way of utilising the solver is by integrating it directly into Java code through the DL Abduction API[5] [18]. This library allows developers to work with the solver through abstract interfaces, without the need to know how it functions internally.
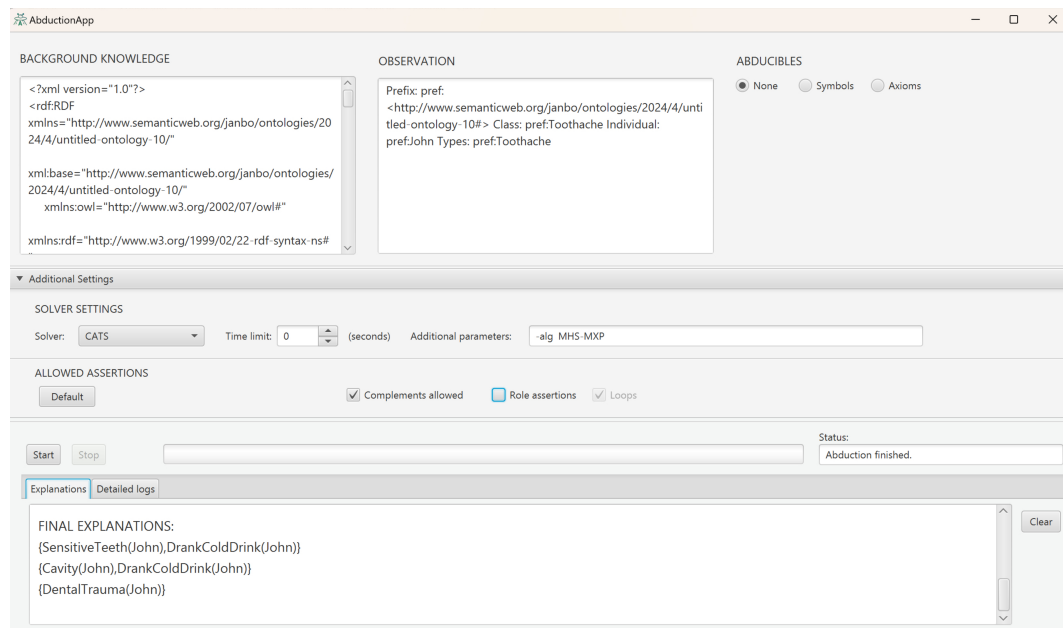

Figure 1: Using CATS via the DL Abduction App.

4. Outlook

Our main future goal is to extend CATS with more well-known abduction algorithms, such as HS-DAG [9] (and its improved version, RC-Tree [19]) and combine them with MXP. We will also add the option to use other efficient incomplete algorithms, such as QuickXplain [20], that is already implemented in the solver, but currently only used as a part of MXP.

In the future, we would like to conduct a comparative evaluation of the algorithms, taking advantage of the fact that they all share the same code-base and modular architecture. However, further considerations are needed in regard to what test cases and methodology to use. Some works [21, 6, 7] already performed evaluations, but to our knowledge, there is no standard proven procedure yet.

Acknowledgments

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*, 2006.

[3] 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., 2014, pp. 1034–1040.

[4] 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.

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

[6] 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.

[7] 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 *LNCS*, Springer, 2023, pp. 338–352.

[8] R. Reiter, A theory of diagnosis from first principles, Artif. Intell. 32 (1987) 57–95.

[9] 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.

[10] F. Wotawa, A variant of Reiter's hitting-set algorithm, Inf. Process. Lett. 79 (2001) 45–51.

[11] K. M. Shchekotykhin, D. Jannach, T. Schmitz, MergeXplain: Fast computation of multiple conflicts for diagnosis, in: Proceedings of the 24th International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, AAAI Press, 2015.

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

[13] I. Palmisano, JFact DL reasoner, http://jfact.sourceforge.net/, .

[14] B. Glimm, I. Horrocks, B. Motik, G. Stoilos, Z. Wang, Hermit: An OWL 2 reasoner, Journal of Automated Reasoning 53 (2014) 245–269.

[15] 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.

[16] 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.

[17] B. Cuenca Grau, I. Horrocks, B. Motik, B. Parsia, P. Patel-Schneider, U. Sattler, OWL 2: The next step for OWL, J. Web Semant. 6 (2008) 309–322.

[18] 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), Rhodes, Greece, September 2–4, 2023, volume 3515 of *CEUR-WS*, 2023.

[19] 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.

[20] 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.

[21] 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.