

# DL Abduction API v2 and GUI Interface

Extended Abstract

Jakub Kloc, Martin Homola and Júlia Pukancová

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

## Abstract

DL Abduction API, first presented last year, has been upgraded substantially. The API is now more tightly integrated with OWL API on which it relies for the representation of the passed values. The new version features a reworked set of implementation interfaces that are now more fine-grained and enable the integrators to cover the features relevant to their system more easily and modularly. In addition to the API library per se, we also present a GUI interface which currently integrates two DL abduction solvers (MHS-MXP and LETHE-Abduction) via the API.

## Keywords

abduction, description logics, ontology, software engineering

## 1. Introduction


Abduction [1] is a non-standard reasoning task wherein, given a formal language, the input consists of background knowledge and an observation that is not entailed from it. The task is to find possible explanations, i.e. formulas that together with the background knowledge entail the observation. Optionally, explanations may be limited to a set of formulas or symbols called the abducibles.


In description logics, the background knowledge is an ontology. One distinguishes between TBox and ABox abduction, wherein the observations and the explanations are TBox and ABox axioms, respectively, and possibly a mixed approach, dubbed KB abduction [2].


DL Abduction API [3], inspired by the popular OWL API [4], proposed a uniform interface for Java applications to integrate abduction reasoning services. The API was originally implemented in the MHS-MXP ABox abduction solver [5] but it offered a more generic interface that allowed for an application to pass the inputs to the abduction solvers (including the background ontology, the observations, and the abducibles) and to pass a plethora of additional control switches. Abduction is a hard task; therefore the API also offers an asynchronous execution mode under which the solver is run in the background and passes the explanations back to the integrating applications as soon as they are discovered.

In this report, we describe a new version of the API with three main improvements: (a) the API is now more tightly bound with the OWL API requiring an OWL API representation of all


---


 DL 2023: 36th International Workshop on Description Logics, September 2–4, 2023, Rhodes, Greece

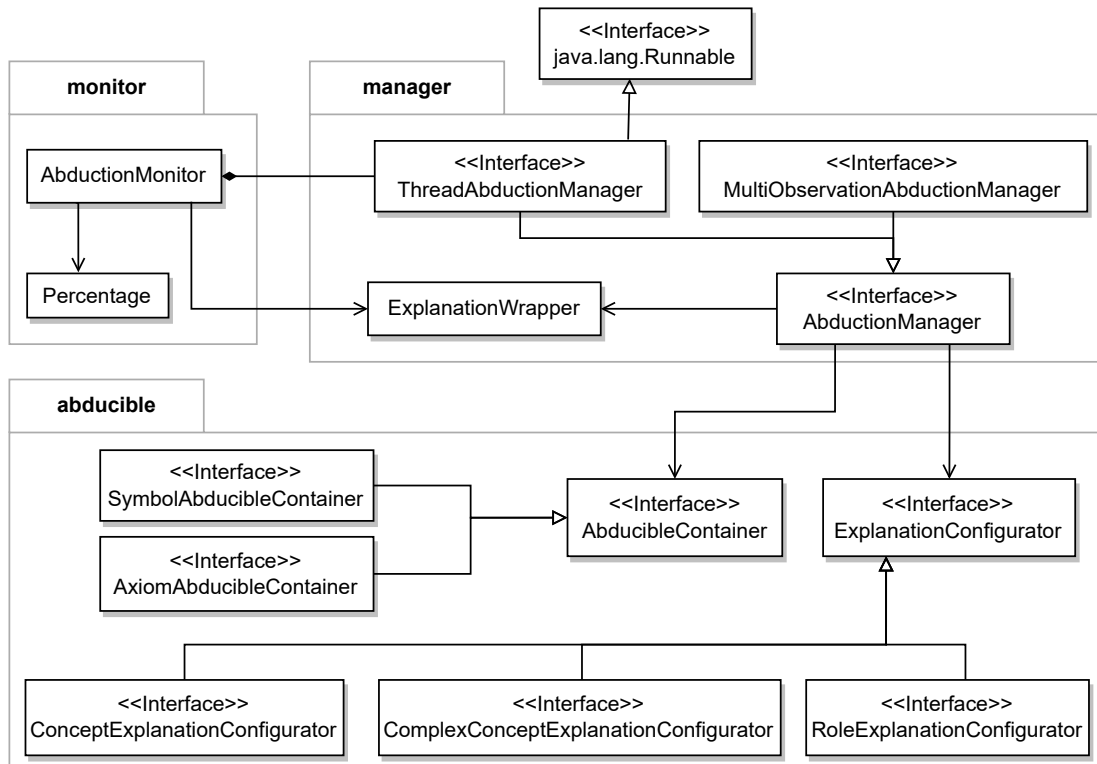
 kloc4@uniba.sk (J. Kloc); homola@fmph.uniba.sk (M. Homola); pukancova@fmph.uniba.sk (J. Pukancová)

 <https://dai.fmph.uniba.sk/~homola/> (M. Homola); <https://dai.fmph.uniba.sk/~pukancova/> (J. Pukancová)

 0000-0001-6384-9771 (M. Homola)

 © 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)



**Figure 1:** Simplified class diagram of the API, excluding the factory interface and exception classes

values passed as inputs and outputs; (b) it is now structured into a much finer set of interfaces, allowing for a more modular implementation of the API into solvers, which easily disregards irrelevant features; (c) it now also offers a GUI application that has been currently integrated with two DL abduction solvers MHS-MXP [5] and LETHE-Abduction [6].

## 2. Abduction API v2

The main class of the API<sup>1</sup> is the `AbductionManager`, which serves to set the input and run the solving process. Abducibles can be set using interfaces extending the `AbducibleContainer` interface, while those extending the `ExplanationConfigurator` interface can be used to configure which types of assertions should be allowed in the explanations. The `AbductionMonitor` class provides the base for an asynchronous mode, in which explanations are processed right as they are found, as opposed to waiting for the whole abduction to finish. A simplified class diagram of the API is shown in Figure 1. The API also contains the `AbductionFactory` interface and a hierarchy of exception classes. Those were omitted from the diagram for visual clarity.

The original interfaces contained a variety of methods that are not expected to be implemented by each solver. For instance, the original `AbducibleContainer` interface declared methods

<sup>1</sup>The API is available at <https://github.com/Comenius-Abduction-Team/DL-Abduction-API>

to manage abducibles both as vocabulary symbols and as axioms, and those that configure assertion types in explanations as well. A hypothetical solver which supports only symbol abducibles still would have to inherit all of these methods, but calling most of them would result in an exception being thrown.

We divided the interfaces into multiple smaller ones, each containing a group of logically related methods. This approach minimises the number of unsupported methods when working with a specific implementation of the API, e.g. the aforementioned solver would only implement the `SymbolAbducibleContainer` interface.

The original API used generic types for its input and output. We replaced them with OWL API types to ensure that every implemented solver can be worked with uniformly. We created a convenience class, the `ExplanationWrapper`, representing a single explanation. It may contain multiple OWL axioms and the explanation's string representation.

In the asynchronous mode, the solver runs in a standalone thread and sends explanations to other threads to be processed there. The threads must be synchronised using the `AbductionMonitor` instance provided by a manager implementing the `ThreadAbductionManager` interface. When a new explanation is found by the solver, it is stored in a set in the monitor. The solver thread notifies other threads waiting on the monitor and falls asleep. When the explanations are obtained from the monitor, the solver thread should be notified to continue running.

Apart from explanations, the solver can now also send a percentual value and a status message to the other thread, to inform the user about the abduction's current progress. Additionally, the monitor class provides more flexibility and control over the asynchronous mode, e.g., the time given to another thread to process explanations can be bound by a limit to prevent deadlocking.

The API's factory interface was enhanced, allowing every type of object available in the API to be initialised abstractly. The hierarchy of exception classes was tweaked as well. The main change is a new class, `NotSupportedException`, which allows to distinguish exceptions that are thrown because of unsupported functionality and not run-time errors.

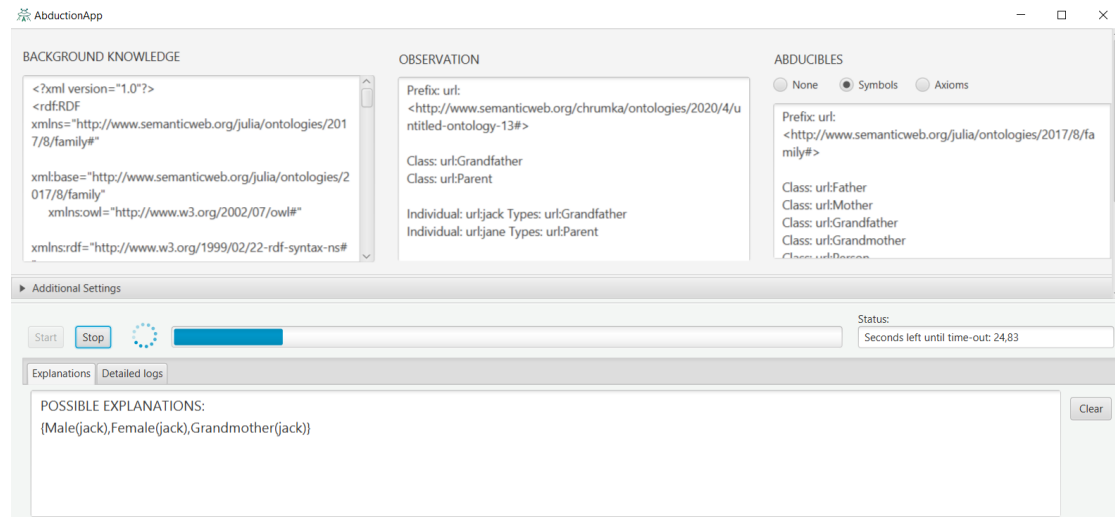
We developed two implementations of the API<sup>2</sup>, one based on an ABox solver using a combination of MHS and MXP algorithms (the MHS-MXP solver [5]), and the other on LETHE-Abduction [6], which uses forgetting to solve KB abduction. In the following example, we showcase how to solve a simple abduction problem and print the explanations:

```
owlOntology ontology = ...; //obtain the background knowledge
owlAxiom observation = ...; //obtain the observation
AbductionManager manager = new MhsMxpAbductionManager();
manager.setBackgroundKnowledge(ontology);
manager.setObservation(observation);
manager.solveAbduction();
manager.getExplanations().forEach(System.out::println);
```

`MhsMxpAbductionManager` is an object of the API encapsulating the MHS-MXP solver and for simplicity we just show the basic, synchronous execution mode (i.e., not the above-mentioned

---

<sup>2</sup>Available at <https://github.com/Comenius-Abduction-Team/MHS-MXP-algorithm> and <https://github.com/Comenius-Abduction-Team/LETHE-Abduction-API-implementation>, respectively



**Figure 2:** The GUI application while solving abduction

asynchronous mode).

### 3. GUI

The two implementations of the API can be used via a JavaFX GUI application<sup>3</sup> that we developed. Its interface is shown in Figure 2. The background knowledge, the observation and the abducibles are all entered in the GUI in the form of an OWL ontology, either in the RDF/XML or the Manchester syntax. After the abduction finishes, the explanations are printed as strings into a text area "console". The API's asynchronous mode is used to show potential new explanations right as they are found and to update a progress bar. Logs and other informative prints produced by the solvers are shown in a separate console.

### 4. Conclusions

We have described DL Abduction API v2, an improved and more mature version of the original API [3]. It allows for more efficient and more modular integration with DL abduction solvers and introduces a few additional features. Those include a GUI interface that is currently integrated with two solvers and provides a more user-convenient way to work with them.

### Acknowledgments

The authors are grateful to Patrick Koopmann for his valuable feedback on the API proposal. This research was sponsored by the Slovak Republic under the grant APVV-19-0220 (ORBIS) and

<sup>3</sup>Source code available at <https://github.com/Comenius-Abduction-Team/Abduction-App>

by the EU under the H2020 grant no. 952215 (TAILOR). J. Kloc was supported by an extraordinary scholarship awarded by Comenius University in Bratislava, Faculty of Mathematics, Physics and Informatics and by a DL student grant.

## 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] Z. Hlávková, M. Homola, P. Koopmann, J. Pukancová, An API for DL abduction solvers, in: *Proceedings of the 35th International Workshop on Description Logics (DL 2022)*, Haifa, Israel, Aug 7–10, 2022, volume 3263 of *CEUR-WS*, 2022.
- [4] M. Horridge, S. Bechhofer, The OWL API: A Java API for OWL ontologies, *Semantic Web 2* (2011) 11–21.
- [5] 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, Sep 20–22, 2023, *Proceedings*, volume 14281 of *LNCS*, Springer, 2023.
- [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.