Noname manuscript No.
(will be inserted by the editor)

The AAA ABox abduction solver

System description

Júlia Pukancová · Martin Homola

Received: date / Accepted: date

Abstract AAA is a sound and complete ABox abduction solver based on the Reiter's MHS algorithm and the Pellet reasoner. It supports DL expressivity up to $\mathcal{SROIQ}$ (i.e. OWL 2). It supports multiple observations, and allows to specify abducibles.

Keywords Description logics · abduction · implementation

1 Introduction

Abduction is a specific type of reasoning which explains why some observation does not follow from the information that we already have, i.e. from our knowledge base (KB). This problem of abduction was originally introduced by Peirce [19] in 1878. However studying abduction in the context of ontologies – and hence description logics (DL) – is relatively new [7]. Applications of abduction in this area include e.g. ontology debugging [26], system malfunction diagnosis [14], multimedia interpretation [20], and medical diagnosis [21].

A number of works studied ABox abduction and focused on algorithms for computing explanations [16, 12, 11, 6, 17, 8, 1, 3, 4, 22, 23, 18, 9].

Notably, multiple works adopted Reiter's Minimal Hitting Set (MHS) algorithm [25] exploiting a DL reasoner for satisfiability checking. This was proposed by Halland and Britz [12, 11], and later extended by Pukancová and Homola [22, 23] and Mrózek et al. [18] who have developed a black box approach and its proof-of-concept implementation dubbed the AAA solver[1] (*AAA* stands for *ABox Abduction Algorithm*) that we describe in this paper.

2 ABox abduction in description logics

Formally speaking, an *abduction problem* [7] is a pair $(\mathcal{K}, \mathcal{O})$ where $\mathcal{K}$ is an ontology (i.e. a DL knowledge base) and $\mathcal{O}$ is a set of axioms called observations. A *solution* of an abduction problem is another set of axioms $\mathcal{E}$ (also called *explanation*) s.t. $\mathcal{K} \cup \mathcal{E} \models \mathcal{O}$.

If $\mathcal{O}$ and $\mathcal{E}$ are assumed to be TBox axioms, we are talking about TBox abduction. On the other hand, if $\mathcal{O}$ and $\mathcal{E}$ are supposed to be ABox assertions, then we deal with ABox abduction [7].

Our work focuses entirely on ABox abduction, and we distinguish ABox assertions of several types. Concept assertions are of the form $\mathtt{a}\colon \mathtt{C}$, where $\mathtt{a}$ is an individual and $\mathtt{C}$ is a concept name or possibly a complex concept description. Role assertions are of the form $\mathtt{a}, \mathtt{b}\colon \mathtt{R}$, and negated role assertions of the form $\mathtt{a}, \mathtt{b}\colon \neg\mathtt{R}$, where $\mathtt{a}$, $\mathtt{b}$ are individuals and $\mathtt{R}$ is a role name. Reflexive role assertions of the forms $\mathtt{a}, \mathtt{a}\colon \mathtt{R}$ and $\mathtt{a}, \mathtt{a}\colon \neg\mathtt{R}$ are called loops.

Given the monotonicity of DLs, the number of possible explanations is generally too high and must be limited somehow to make the whole task meaningful.

This work was supported from Slovak National VEGA projects no. 1/1333/12 and 1/0778/18, and by the Comenius University grants no. UK/426/2015 and UK/266/2018.

J. Pukancová
Comenius University in Bratislava
Mlynská dolina, 84248 Bratislava, Slovakia
E-mail: pukancova@fmph.uniba.com

M. Homola
Comenius University in Bratislava
Mlynská dolina, 84248 Bratislava, Slovakia
E-mail: homola@fmph.uniba.com
ORCiD: 0000-0001-6384-9771

[1] `http://dai.fmph.uniba.sk/~pukancova/aaa/`

Therefore most works focus only on (syntactically) minimal explanations, i.e. such $\mathcal{E}$ that there is no other explanation $\mathcal{E}' \subsetneq \mathcal{E}$.

This may still yield some undesirable explanations hence the desired explanations are often further constrained. The constraints that we assume in our work are that all explanations should be:

1. *consistent:* i.e. $\mathcal{K} \cup \mathcal{E}$ is consistent;
2. *relevant:* i.e. $\mathcal{E} \not\models \mathcal{O}$;
3. *explanatory:* i.e. $\mathcal{K} \not\models \mathcal{O}$.

3 The AAA solver

Building on ideas of Halland and Britz [11,12], we have developed an ABox abduction solver AAA [22,23] based on Reiter's MHS algorithm [25].

Let us first assume an observation $\mathcal{O} = \{O\}$ consisting of a single ABox assertion. Reiter showed that in order to find an explanation $\mathcal{E}$ s.t. $\mathcal{K} \cup \mathcal{E} \models \{O\}$ it suffices to find a hitting set of the set of all models of $\mathcal{K} \cup \{\neg O\}$. The MHS algorithm, which he also proposed, does this by constructing a so called HS-tree in which explanations are found as paths from the root to the leaves. It is often desired to find all (meaningful) explanations and shorter explanations are preferred, the HS-tree is therefore constructed using breadth-first search.

The solver takes an advantage of this and allows the search to be depth-bound, thus allowing to find all explanations up to a given cardinality. The rest of the HS-tree is cut off. Our algorithm is complete up to any given depth [23].

As we need to compute the models of $\mathcal{K} \cup \{\neg O\}$, auxiliary calls to a DL reasoner are used. The DL reasoner is called as a *black-box*, and it is called on-the-fly during the HS-tree construction. AAA also exploits optimization techniques suggested by Reiter such as model reuse and pruning.

AAA also handles observations $\mathcal{O} = \{O_1, \ldots, O_n\}$ consisting of a set of ABox assertions. For this case we have developed two approaches. The first one, so called splitting approach, computes the explanations for each $O_i \in \mathcal{O}$ separately, which are then combined. The other one, so called reduction approach, reduces the set of ABox assertions $\mathcal{O}$ into one ABox assertion O' in such a way, that all the explanations of $\mathcal{O}$ are preserved.

AAA thus supports observations in form of any set of ABox assertions (including complex concept assertions and negated role assertions). It supports explanations in form of sets consisting of atomic and negated atomic concept and role assertions. The supported DL expressivity for both the input ontology and the observations is up to $\mathcal{SROIQ}$ [13], i.e. OWL 2 [2].

Since version 0.9 the solver also allows to specify abducibles, i.e. individuals, concept names, and role names to which the search for explanations is limited. If the user knows beforehand in which part of the search space they are looking for explanations, this can greatly reduce the HS-tree size and thus the overall computation time.

4 Running AAA

Let us consider the following ontology (given in Manchester syntax) describing family relations, as our input knowledge base $\mathcal{K}$:

```
Prefix: : <http://example.org/>
Ontology: <http://example.org/ontology/>

ObjectProperty: hasChild

ObjectProperty: hasParent
    InverseOf: hasChild

Class: Man

Class: Woman
    DisjointWith: Man

Class: Parent
    EquivalentTo:
        hasChild some Man or Woman

Class: Child
    EquivalentTo:
        hasParent some Man or Woman

Class: Mother
    EquivalentTo:
        Parent and Woman

Class: Father
    EquivalentTo:
        Parent and Man

Class: Grandmother
    SubClassOf:
        Mother

Class: Grandfather
    SubClassOf:
        Father

Individual: tarzan
    Types: Man
```

The goal for AAA is to compute explanations for the observation “Jane is a mother”, i.e. for the observation $O = \{\texttt{jane} : \texttt{Mother}\}$. To run the solver, the input file (the ontology) and the output file must be stated (`-i` and `-out` switches), and the observation is given on the command line using the `-obs` switch. There are some additional optional switches which will be explained below. The example command line is as follows:

```
java -jar AAA.jar \
     -i family.owl \
     -obs "http://example.org/jane:
               http://example.org/Mother" \
     -l -d 5 \
     -out output.txt
```

After running the command, AAA computes the explanations and writes the output in the output file. Except some statistic information, such as computation times for the specific levels of the run, all the explanations are listed in the end of the file. An excerpt[2] from the output file:

```
----------------INPUT DETAILS----------------
Observation: jane: Mother
Depth limitation: 5
----------------TIME DETAILS-----------------
1 time 0.92 n 24 ta 22 r 2 p 0 ...
2 time 2.52 n 438 ta 202 r 59 p 177 ...
3 time 9.42 n 3796 ta 1376 r 464 p 1956 ...
4 time 53.94 n 22564 ta 7053 r 2796 p 12715 .
5 time 633.97 n 101362 ta 30536 r11892 p58934
----------------EXPLANATIONS-----------------
All explanations: {
{jane: Woman; tarzan, jane: hasParent},
{jane: Parent; jane: Woman},
{jane: Woman; jane, tarzan: hasChild},
{jane: Woman; jane, jane: hasChild},
{jane: Woman; jane, jane: hasParent},
{jane: Grandmother}}
```

We can see that the solver has found 6 explanations in this case.

4.1 Restricting MHS depth

The switch `-d` (already used in the example above) allows to limit the search to specific depth of the HS-tree, i.e. to search for explanations only up to a given cardinality.

The main reason why the user may want to use this restriction is because with greater depth, exploring each consecutive level of the HS-tree becomes exponentially[3] more time consuming (as apparent from the time statistics in the example above). Also, while our algorithm only returns minimal explanations, even among these, smaller explanations (in terms of cardinality) are often more preferred.

For example, from the 6 explanations in the listing above, only the explanations of the size 1 (containing only one assertion) may be desired. In such a case we modify the command as follows:

```
java -jar AAA.jar \
     -i family.owl \
     -obs "http://example.org/jane:
               http://example.org/Mother" \
     -l -d 1 \
     -out output.txt
```

In this case only one explanation is found:

```
----------------INPUT DETAILS----------------
Observation: jane: Mother
Depth limitation: 1
----------------TIME DETAILS-----------------
1 time 1.07 n 24 ta 22 r 2 p 0 ...
----------------EXPLANATIONS-----------------
All explanations: {
{jane: Grandmother}}
```

4.2 Avoiding loops

There are some unintuitive explanations amongst the six found in the first example run above. Depending on the domain and the application it may not be desirable to explain the observed phenomenon by reflexive relations of the form $\texttt{a}, \texttt{a} \colon \texttt{R}$ or $\texttt{a}, \texttt{a} \colon \neg\texttt{R}$. For instance, in our example we do not want to explain the observation that Jane is mother by the explanation that she is a child of herself.

The switch `-l` enables loops (i.e. reflexive role assertions) which are disallowed by default under the assumption that users would only rarely look for such explanations. The following example demonstrates the run of AAA without loops.

[2] All output file excerpts have been modified for readability: prefix part of IRIs were omitted and syntax of role assertions was rewritten to match this paper, some of the `TIME DETAILS` section have been cut off, and some outputs such as ontology statistics and other less relevant information has been removed. The shortcuts in `TIME DETAILS` are as follows: `time` – total time in seconds, `n` – number of nodes, `ta` – number of DL reasoner calls, `r` – reused models, `p` – pruned nodes.

[3] The MHS problem is NP-complete. Therefore for expressive DLs the combined worst-case complexity of the solver is “inherited” from the input ontology. For $\mathcal{ALC}$ ontologies it is ExpTime [5], for OWL 2 (i.e. $\mathcal{SROIQ}$) ontologies it is N2ExpTime) [15].

```
java -jar AAA.jar \
     -i family.owl \
     -obs "http://example.org/jane:
               http://example.org/Mother" \
     -d 5 \
     -out output.txt
```

AAA now disregards loops, hence out of the six original explanations of the observation $O = \{$`jane : Mother`$\}$, only four remain:

```
----------------INPUT DETAILS----------------
Avoid loops: true
Observation: jane: Mother
Depth limitation: 5
----------------TIME DETAILS-----------------
1 time 0.83 n 20 ta 16 r 4 p 0 ...
2 time 2.11 n 286 ta 139 r 40 p 107 ...
3 time 6.12 n 1920 ta 735 r 295 p 890 ...
4 time 19.67 n 8893 ta 3275 r 1167 p 4451 ...
5 time 87.62 n 31693 ta 11348 r 4264 p 16081
----------------EXPLANATIONS-----------------
All explanations: {
{jane: Grandmother},
{jane: Parent; jane: Woman},
{jane: Woman; tarzan, jane: hasParent},
{jane: Woman; jane, tarzan: hasChild}}
```

If loops are undesired they should always be omitted from the computation as this also reduces the search space and hence decreases the computation time. This can be also observed in our example runs above.

4.3 Introducing abducibles

Another meaningful restriction on the search space is to restrict the individuals, concepts, and roles that are included in the explanations only to a certain set, called *abducibles*. Especially in bigger ontologies, the user may be specifically interested only in explanations involving specific entities.

Abducibles can be specified using the switch `-abd` as an enumeration of symbols allowed in the explanations. In the example below, the abducibles are restricted to the individual `jane` and the concepts `Woman`, `Man`, `Parent`, and `Grandmother`:

```
java -jar AAA.jar \
     -i family.owl \
     -obs "http://example.org/jane:
               http://example.org/Mother" \
     -abd "http://example.org/jane,
           http://example.org/Woman,
           http://example.org/Man,
           http://example.org/Parent,
           http://example.org/Grandmother" \
     -l \
     -out output.txt
```

When AAA is run on this input, it returns the following output:

```
----------------INPUT DETAILS----------------
Observation: jane: Mother
Abducibles: jane, Woman, Man, Parent,
            Grandmother
----------------TIME DETAILS-----------------
1 time 0.64 n 5 ta 5 r 0 p 0 ...
2 time 0.72 n 17 ta 9 r 0 p 8 ...
----------------EXPLANATIONS-----------------
All explanations: {
{jane: Grandmother},
{jane: Parent; jane: Woman}}
```

As apparent from the example, providing a reasonable set of abducibles reduces the search space greatly, which is now limited to all possible (atomic and negated) ABox assertions that can be constructed from the provided sets of individuals, concept, and role names. This can significantly reduce the computation time – however, the user must have some prior assumptions as for where to look for the explanations.

4.4 Multiple observations

The AAA solver notably allows for observation-sets consisting of multiple assertions. For example, we may want to find explanations that explain both `jane: Mother` and `tarzan: Child`. We may do so, using the following command:

```
java -jar AAA.jar \
     -i family.owl \
     -obs "http://example.org/jane:
               http://example.org/Mother;
           http://example.org/tarzan:
               http://example.org/Child;" \
     -d 5 \
     -out output.txt
```

Note that in order to avoid nonsensical explanations with loops (8 in this case), we chose to disable loops. We obtain the following output:

```
----------------INPUT DETAILS----------------
Reduction: false
Avoid loops: true
Observation: jane: Mother; tarzan: Child
Depth limitation: 5
```

```
----------------TIME DETAILS-----------------
1 time 0.89 n 20 ta 16 r 3 p 1 ...
2 time 2.25 n 267 ta 124 r 37 p 106 ...
3 time 6.51 n 1635 ta 621 r 235 p 779 ...
4 time 20.40 n 6974 ta 2540 r 921 p 3513 ...
5 time 79.31 n 23352 ta 8436 r 2889 p 12027 .
1 time 0.0 n 20 ta 16 r 3 p 1 ...
2 time 0.36 n 250 ta 147 r 37 p 66 ...
3 time 2.31 n 1694 ta 730 r 244 p 720 ...
4 time 22.08 n 7261 ta 2919 r 1110 p 3232 ...
5 time 147.53 n 26242 ta 10395 r 3823 p 12024
----------------EXPLANATIONS-----------------
All explanations: {
{jane: Grandmother; tarzan, jane: hasParent},
{jane: Grandmother; jane, tarzan: hasChild},
{jane: Woman; jane, tarzan: hasChild},
{jane: Woman; tarzan, jane: hasParent}}
```

As mentioned above, AAA supports two different approaches to enable multiple observations. This run was obtained by the so called splitting approach (default) in which the algorithm is run separately for each element of the observation set, and then the results are combined. We can see in `TIME DETAILS` that the solver has indeed constructed two HS-trees.

Using the switch `-r` we can enable the so called reduction approach, which reduces the observation set into a single ABox assertion and runs the MHS algorithm only once, on the reduced input. The output for this approach is as follows:

```
----------------INPUT DETAILS----------------
Reduction: true
Avoid loops: true
Observation: jane: Mother; tarzan: Child
Depth limitation: 5
----------------TIME DETAILS-----------------
1 time 0.98 n 21 ta 16 r 3 p 2 ...
2 time 2.41 n 301 ta 129 r 55 p 117 ...
3 time 7.38 n 1921 ta 754 r 350 p 817 ...
4 time 24.65 n 9721 ta 3618 r 1558 p 4545 ...
5 time 125.87 n 37341 ta 13939 r 5260 p 18142
----------------EXPLANATIONS-----------------
All explanations: {
{jane: Woman; jane, tarzan: hasChild},
{jane: Woman; tarzan, jane: hasParent}}
{jane: Grandmother; tarzan, jane: hasParent},
{jane: Grandmother; jane, tarzan: hasChild},
```

As we can see, in this case the reduction approach is more efficient than splitting, but it is not always the case.

5 Implementation details

The AAA solver is implemented in Java (version 1.8). It has two main components, the implementation of the MHS algorithm and a DL reasoner (Pellet [28]) which is being called by the MHS component for consistency checks and for model retrieval. The Pellet reasoner calls are tightly integrated at the source-code level. Pellet is an open source OWL DL reasoner implemented in Java. We have used Pellet 2.3.1 which is the latest open-source version.

The advantage of using the DL reasoner as a black box, is that AAA consequently supports the same DL expressivity as is supported by the DL reasoner. Pellet features full OWL 2 support which means that any OWL 2 ontology may be used with our algorithm.

In the unrestricted case (when no abducibles are specified) the solver is able to process a single observation w.r.t. to the LUBM ontology [10] within seconds (up to depth 2), within minutes (up to depth 3) and within hours (up to depth 4) [24].

AAA is available for download at: `http://dai.fmph.uniba.sk/~pukancova/aaa/`.

6 Conclusions and future work

We have provided an overview of AAA, an ABox abduction solver for description logics. AAA is sound and complete up to $\mathcal{SROIQ}$ DL (i.e. OWL 2). It is implemented in Java, it is based on Reiter's Minimal Hitting Set algorithm, and it uses an external DL reasoner (Pellet) which is called as a black box.

AAA integrates Pellet directly at the source-code level; however experimental versions exploiting OWL API were also explored [18, 9]. We would like to explore this line also in the future and to provide a stable version based on OWL API that would enable the user to pick different DL reasoners. We also plan to implement different abduction strategies that could serve as an alternative to MHS, even some which are incomplete (but very fast) such as MergeXplain [9, 27].

Acknowledgements The authors wish to thank to Katarína Fabianová, Júlia Gablíková, and Drahomír Mrózek whose Master's projects were affiliated with the AAA solver.

References

1. Castano, S., Espinosa Peraldí, I.S., Ferrara, A., Karkaletsis, V., Kaya, A., Möller, R., Montanelli, S., Petasis, G., Wessel, M.: Multimedia interpretation for dynamic ontology evolution. J. Log. Comput. **19**(5), 859–897 (2009)

2. 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)
3. Del-Pinto, W., Schmidt, R.A.: Forgetting-based abduction in $\mathcal{ALC}$. In: Proceedings of the Workshop on Second-Order Quantifier Elimination and Related Topics (SOQE 2017), Dresden, Germany, *CEUR-WS*, vol. 2013, pp. 27–35 (2017)
4. Del-Pinto, W., Schmidt, R.A.: Abox abduction via forgetting in ALC. In: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019, pp. 2768–2775. AAAI Press (2019)
5. Donini, F.M., Massacci, F.: Exptime tableaux for $\mathcal{ALC}$. Artificial Intelligence **124**(1), 87–138 (2000). DOI 10.1016/S0004-3702(00)00070-9
6. 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)
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. Espinosa Peraldí, I.S., Kaya, A., Möller, R.: Formalizing multimedia interpretation based on abduction over description logic ABoxes. In: Proceedings of the 22nd International Workshop on Description Logics (DL 2009), Oxford, UK, *CEUR-WS*, vol. 477 (2009)
9. Fabianová, K., Pukancová, J., Homola, M.: Comparing ABox abduction based on minimal hitting set and MergeXplain. In: Proceedings of the 32nd International Workshop on Description Logics, Oslo, Norway, June 18-21, 2019, *CEUR-WS*, vol. 2373 (2019)
10. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. Journal of Web Semantics **3**(2-3), 158–182 (2005)
11. Halland, K., Britz, K.: 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, pp. 51–58 (2012)
12. Halland, K., Britz, K.: Naïve ABox abduction in $\mathcal{ALC}$ using a DL tableau. In: Proceedings of the 2012 International Workshop on Description Logics, DL 2012, Rome, Italy, *CEUR-WS*, vol. 846 (2012)
13. 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)
14. Hubauer, T., Lamparter, S., Pirker, M.: Relaxed abduction: Robust information interpretation for incomplete models. In: Proceedings of the 24th International Workshop on Description Logics (DL 2011), Barcelona, Spain, July 13-16, 2011 (2011)
15. Kazakov, Y.: RIQ and SROIQ are harder than SHOIQ. In: G. Brewka, J. Lang (eds.) Principles of Knowledge Representation and Reasoning: Proceedings of the Eleventh International Conference, KR 2008, Sydney, Australia, September 16-19, 2008, pp. 274–284. AAAI Press (2008)
16. Klarman, S., Endriss, U., Schlobach, S.: ABox abduction in the description logic $\mathcal{ALC}$. Journal of Automated Reasoning **46**(1), 43–80 (2011)
17. Ma, Y., Gu, T., Xu, B., Chang, L.: An ABox abduction algorithm for the description logic $\mathcal{ALCI}$. In: Intelligent Information Processing VI – 7th IFIP TC 12 International Conference, IIP 2012, Guilin, China. Proceedings, *IFIP AICT*, vol. 385, pp. 125–130. Springer (2012)
18. Mrózek, D., Pukancová, J., Homola, M.: ABox abduction solver exploiting multiple DL reasoners. In: Proceedings of the 31st International Workshop on Description Logics, Tempe, Arizona, US, *CEUR-WS*, vol. 2211 (2018)
19. Peirce, C.S.: Deduction, induction, and hypothesis. Popular science monthly **13**, 470–482 (1878)
20. Petasis, G., Möller, R., Karkaletsis, V.: BOEMIE: Reasoning-based information extraction. In: Proceedings of the 1st Workshop on Natural Language Processing and Automated Reasoning co-located with 12th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2013), A Corunna, Spain, September 15th, 2013., pp. 60–75 (2013)
21. Pukancová, J., Homola, M.: Abductive reasoning with description logics: Use case in medical diagnosis. In: Proceedings of the 28th International Workshop on Description Logics (DL 2015), Athens, Greece, *CEUR-WS*, vol. 1350 (2015)
22. Pukancová, J., Homola, M.: Tableau-based ABox abduction for the $\mathcal{ALCHO}$ description logic. In: Proceedings of the 30th International Workshop on Description Logics, Montpellier, France, *CEUR-WS*, vol. 1879 (2017)
23. 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)
24. Pukancová, J.: Direct approach to ABox abduction in description logics. Ph.D. thesis, Comenius University in Bratislava (2018)
25. Reiter, R.: A theory of diagnosis from first principles. Artificial intelligence **32**(1), 57–95 (1987)
26. Schekotihin, K., Rodler, P., Schmid, W.: 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, *LNCS*, vol. 10833, pp. 340–359. Springer (2018)
27. 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)
28. Sirin, E., Parsia, B., Cuenca Grau, B., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. Journal of Web Semantics **5**(2), 51–53 (2007)