Acceleration of Embedded Semantic Reasoning over Ontologies: Enabling Description Logics Reasoning on Embedded Hardware in Avionics-Context

Youssef Amari **

Thales, Toulouse, France and LAAS-CNRS, Toulouse, France
yamari@laas.fr

Abstract. Current advances in Artificial Intelligence (AI) technologies pave the way to consider new services that assist aircrew, possibly in embedded systems. Semantic reasoning using ontologies provide both opportunities and challenges for these services. Recent studies showcase that those reasoning approaches suffer from long and unpredictable execution times, and high memory consumption. Such limitations currently refrain the use of this approach in embedded systems. The objective of this work is to provide ways to deploy semantic reasoning by exploiting efficient optimisations on embedded architectures.

Keywords: Semantic Reasoning · Symbolic AI · Avionics · Embedded system · Ontology · Description logics · OWL · Reasoner · Optimisation.

1 Introduction/Motivation

Semantic reasoning dealing with ontologies was mostly initiated by the Semantic Web started in 2001 by the World-Wide Web Consortium (W3C). The goal of its founder, Tim Berners-Lee, was to ease the navigation on the Internet using semantic [18]. This has prompted an interest of a sheer number of researchers to tackle ups and downs of this technology. Their goal is to extracting the prominent inventions using the potential of semantics, ontologies and reasoning over many domains of interest such as life science [7], mobiles [16] or avionics.

In aeronautics, one of the envisioned applications can be the assistance to the aircrew in order to circumvent human workload limitations. A case in point, warning light shows up in the cockpit and the pilot has to assess, quickly as possible, the seriousness of the alarm in order to evaluate the criticality of the situation. Due to stringent real-time constraints, monitoring is consequently one of the most complex and critical features in the aviation domain. If perception and prediction steps can be performed by data-based approaches, the comprehension step of this situation must use aeronautic domain knowledge to give a structured meaning of the alarm. For such case, semantic reasoning appears as

** Early Stage Ph.D.

an appealing candidate to implement and manage a real-time situation monitoring. It is commonly employed for deductive knowledge. It is based on concepts such as first-order logic rules, ontologies, decisions trees, and reasoning. This technology detains desirable features such as its explainability, inter-operability between applications/humans and reasoning on conclusions [13].

1.1 Use Case: a Virtual Assistant

Thales is developing a virtual assistant exploiting symbolic AI technologies in order to solve tasks related to the assistance of the aircrew. This assistant will be deployed in embedded operational environment with limited computational power and time response constraints. The essence of our work is mirrored in our paper title: we study the feasibility of implementing knowledge base, such as an ontology, coupled to a reasoning methodology in embedded systems (see Fig. 1).

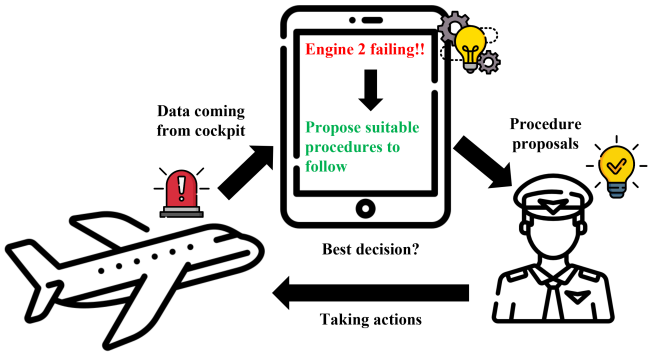

Fig. 1: Schema of a possible situation in our use case (icons from Flaticon.com)

1.2 Ontologies and Reasoners

We are dealing with knowledge bases in ontology format and these ontologies are formalised using the Web Ontology Language (OWL) based on Description Logics (DL). Typically an aircraft ontology can consist of the two regular boxes:

- a TBox which is the skeleton of the domain of aircraft, describing it using concepts and relations such as an aircraft detains two wings and wheels,
- an ABox which is the base of individuals related to our domain such as an Airbus A380 or Boeing 737 aircraft.

According to [4], a semantic reasoner is a *a piece of software able to infer logical consequences from a set of asserted facts or axioms.* [...] *The inference rules are commonly specified by means of an ontology language, and often a description logic language.*

Embedding ontologies and reasoner(s) is a challenge because these technologies were not initially thought for embedded devices. On the one hand, according to [16] and [8], the world of embedded systems such as mobiles or constrained devices imply many constraints concerning the deployment of a reasoner, bearing in mind that numerous state of the art reasoners are too resource-intensive to be implemented on a such system. On that kind of systems the memory space, power, or running time are hugely limited. On the other hand, some studies show up that those technologies are known to suffer from long and unpredictable execution times, usually saturating memory capacities of usual computers. A case in point, a realistic example can be found in [10] in which a non-deterministic execution time is denoted and also it is declared that some reasoners are not adapted to scan large ontologies ($> 1,\ 000$ axioms).

2 State of the Art

In this section, we showcase the *sine qua non* of fundamentals, the existing work that seems well to tackle our research question and the different efforts made to perform embedded reasoning with a coarse grained focus on all embedded devices targeted in the literature.

2.1 Reasoning

Background: Description Logics, Web Ontology Language, and misc.
Description Logics (DL) are a decidable fragment of first-order logic (FOL). It is characterised by a syntax ("symbols") and a semantics ("formal meaning"). The Web Ontology Language (OWL) is an ontology standard language for the semantic Web and RDF is a protocol for web data. OWL has two versions, OWL 1 and 2, both consisting of fragments with a specific expressivity. A case in point, OWL 2 contains OWL 2 EL, QL, and RL.

State of the Art Reasoners
The first step of our work was to gather well-known reasoners. To this extent, we deeply studied 38 reasoners coming from [2] and *LiRoT* [8]. These reasoners have been considered due to their popularity and last update date (at least 2012). Moreover, only official reasoners have been considered because of their recognition and the associated solid community, hence ignoring prototypes implementations.

Importantly, a preliminary phase was to break down the inside of a reasoner by firstly dealing with the structure of its backbone which is the reasoning methodology. We have identified 5 main reasoning approaches among the considered list: *Tableau*, *Hypertableau*, *Consequence-based*, *Datalog rewriting*, or *RETE*. Furthermore, we have extracted the existing standard reasoning tasks such as satisfiability, classification, or consistency that are used to accompany reasoning process.

We have categorised the reasoners relatively to their reasoning methodology over time, as shown in Fig. 2. The purpose was to determine the active and inactive methodologies in order to stay update. In this figure, we observe that (*hyper*)*Tableau* algorithms were regularly used over time, this is the same observation for miscellaneous approaches whereas pattern matching approach, particularly RETE, was less implemented but the trend is apparently changing with the 2022 contribution. Moreover, to tackle performance issues concerning

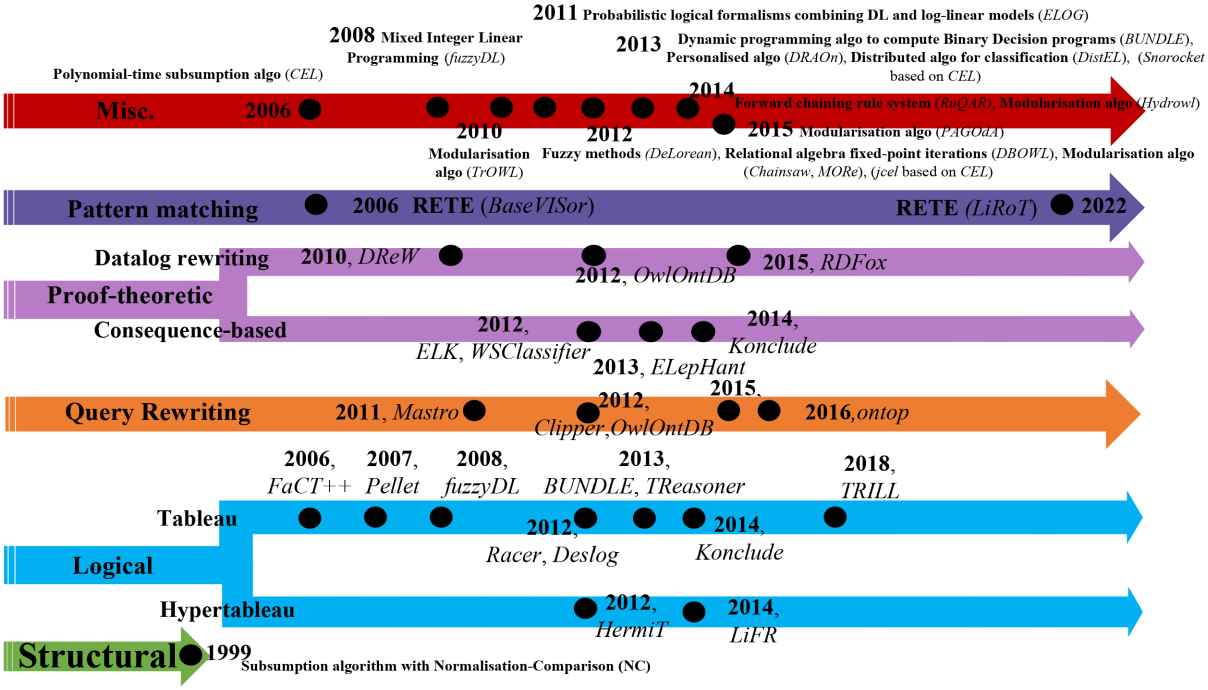

Fig. 2: Timeline regarding reasoning approaches appearance with their associated reasoners from 1999 to 2022.

memory and time usage, we have thoroughly studied the existing and notable optimisations of the considered reasoners in order to evaluate how these improvements are suitable with our approach. The results of this work are presented in section 6.

2.2 Is embedding reasoning conceivable?

Furthermore, we have extended our research to the reasoners dedicated to embedded systems. There are few papers tackling embedded reasoning with the development of various approaches.

Some fairly well-known reasoners are straightforwardly aimed at embedded architectures such as: *Pocket KRHyper*, [16,26], *μOR* [5], *MiRE4OWL* [15], *ELepHant* [24], *COROR* [30], *Mini-ME* [22] and *Tiny-ME* [21], or *LiRoT* [8].

[29] developed a mobile reasoner which implements an extension of Tableau algorithm. In fact, they trade inference result exactness for gain in efficiency, if the latter is needed. The authors describe their solution such as an improvement of *Pellet* [27] reasoner. [23] presents an OWL reasoner for embedded devices by constructing a global architecture. They chose to work with OWL 2 RL profile. A given input ontology is translated to the format of triple «.ntp» and a file containing rules from OWL 2 RL [3] is inserted into the CLIPS [1] rule engine. In [11], the authors describe a reasoner dedicated for Programmable Logic Controllers (PLC) which implements per se a consequence-base approach in order to classify ontology from the OWL 2 EL fragment. This profile was chosen because of its convenience with diagnostic models. The consequence-driven classification algorithm used is the same inside the *CEL* [7] reasoner. [12] proposes the PIRAmIDE infrastructure for reasoning on mobile environment. At its very heart, it uses an ontology reasoner called *Bossam* [14], that is a rule engine based on RETE algorithm. [17] exhibits an architecture that eases the implementation of ontology use and reasoning in mobile devices for supporting Context-Addressable Messaging. One notable optimisation is the employment of the manchester syntax to encode the ontology file rather than the RDF/XML one. This was done using DL-Lite ontology language. The mobile DL-Lite reasoner engine was developed by the authors. [19] introduces a solution using decision trees in a so-called Smart-Context that is defined as "*an autonomous intelligent context-aware pervasive system designed to enable context matching between inputs (information) and outputs (potential solutions) based on contextual information*". [9] studies the possibility to run well-known OWL reasoners in Android-based architectures. [6] deals with building context-aware applications for smartphone. It implements traditional OWL reasoners for inferences. [20] proposes an interesting approach that resembles our work direction. They developed the Delta-reasoner based on OWL 2 RL and SWRL[1] concerning context-aware applications. Also, some efforts have been made on optimising algorithms towards the embedding reasoning such as: m-Tableaux [28] which role is to give the possibility to use Tableaux algorithm on mobile architectures, or $\text{RETE}_{\text{pool}}$ [31] that is dedicated to resource-constrained devices. It gathers RETE memory mechanisms in order to get a better memory footprint.

3 Problem Statement and Contributions

Accordingly, the objective of our thesis is to tackle the problem of setting up ontologies and reasoning in embedded-context by tackling the next questions:

– How efficient are current reasoners to handle our problematic?

[1] Language for the Semantic Web used to symbolise rules and logic.

- Can we combine existing optimisations[2] from standard reasoners to achieve our goal?
- Can current reasoners for embedded systems provide more clues on optimisation techniques and hardware technologies choices?
- How to experiment the limits of reasoning runtime and guarantee a Worst-case Execution Time for embedding applications?

4 Research Methodology and Approach

Our objective is to explore acceleration methods for reasoning over ontologies. The chosen approach consists of 6 stages:

1. Establish a state of the art principally concerning the notable optimisations from standard reasoners, reasoners dedicated to embedded architectures and their implementation on hardware, and classify the various optimisations;
2. Find solutions to answer our problematic and implement on toy cases using benchmarks such as [25];
3. If deemed pertinent, improve existing and interesting reasoner approaches towards our goal;
4. Develop efficient mapping on (new) architectures such as a Graphics Processing Unit (GPU);
5. Implement the solution on the real case of virtual assistant for validation.

Our current work is focused on whether we can optimise reasoning approaches and how, and importantly understand the sources of reasoning runtime variability (non-determinism). In parallel, we are separating the wheat from the chaff in order to undertake the deemed necessary levers concerning our use case such as which reasoning tasks do we need or what is the targeted expressiveness? We are not reinventing the wheel, we mean that this is not a classical research question, as it arises in embedded systems. Although reasoners can perform well, the real challenge would be whether the reasoning execution is mastered, verifiable, and importantly bounded in memory space and reasoning runtime.

5 Evaluation Plan

During our evaluation process, we envisioned to mainly scrutinise the different causes of a low reasoning runtime and an high memory space usage. To do so, we will address the following tasks:

- Make tests with existing benchmarks, such as [25], to explore various sizes and expressiveness of ontologies against a set of reasoners and therefore get a global view of metrics evaluation, memory space and reasoning runtime, for comparison with our use case;

[2] Means the features principally geared towards better running time and/or memory footprint.

– Conduct reasoning experiments on different hardware architectures and compare each in turn. More specifically, we will deeply study the given reasoner over an ontology through a scalability process regarding its size. To do this, we will use a profiler to highlight for instance the number of processes used during the reasoning, or the appearing hotspots means the functions highly demanding in runtime. Also, a huge concern will be on the operations used at the low-level, their dependency, data location in memory, or the graph structure used by the reasoning approach.

We will also shed light on the following questions to help us get a grip on our problem: Which reasoning tasks (classification, satisfiability, consistency, etc.) are relevant for our use case? Which level of expressiveness is needed? What is the size of the ontology? Which reasoning methodologies the reasoner is using and which one could be pertinent? How to accelerate reasoning? By constraining the expressiveness, optimising algorithms, or both? When the ontology cannot be restricted, can we find efficient reasoners to deal with (very) highly expressive ontologies? Also, we need to analyse which types of hardware architecture could suit our needs.

6 Preliminary Results

Two preliminary results w.r.t our work are:

On the one hand, we have built a categorisation in Fig. 3 corresponding to the different optimisation families from studied existing reasoners, organised in a logical order from those concerning low-level to software ones. This figure

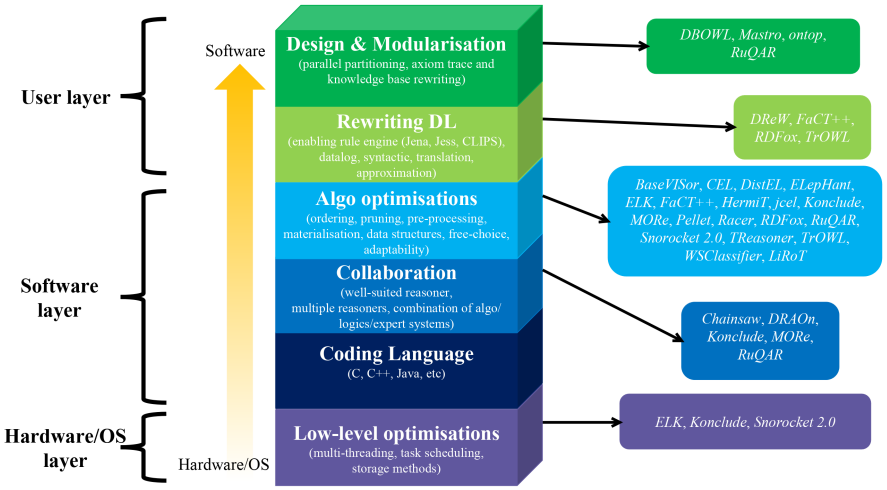

Fig. 3: Categorisation of different optimisation families from studied existing reasoners.

shows different families, each one concerning particular techniques of related optimisations. This optimisations tiling gives us more clues on which techniques could be useful for our use case.

On the other hand, we have conducted a first experiment by running $1,000$ times *HermiT* against the same ontology ($9,227$ axioms) on a classical laptop with a noisy environment[3]. We used Owlready2 package from Python language[4]. In Fig. 4, on the left, we observed a gloomy picture where runtime is non-immutable. Hence, we need to master this variability in order to obtain determinism concerning runtime. This finding spurred us to deepen the explanations of that and our first and logical hypothesis could be interferences caused by background services. Hence, we need to restrain interferences by measuring runtime of the reasoning process in an isolated work environment as it could be in a classical embedded device. Consequently, on another computer we isolated our reasoning from pollution of services/processes. We conducted the same experiment, and obtained the results showed in Fig. 4, on the right.

In this last experiment the distribution of running time is hugely more concentrated between 19 and 20 seconds whereas in the former one the data are more widely dispersed showing a high variability. It is clear that our hypothesis

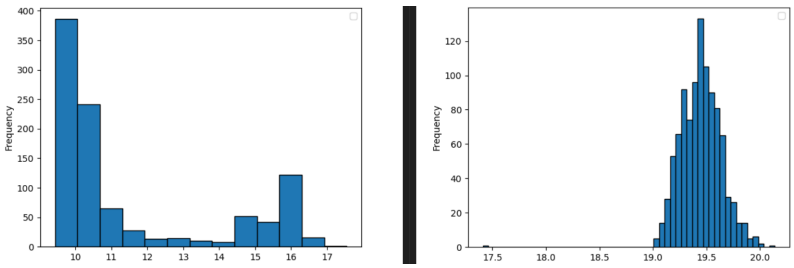

Fig. 4: Histogram regarding the distribution of *HermiT* runtime (seconds) with their frequency, 1000 times, against the same ontology consisting of 9, 227 axioms (on the left with noisy environment/on the right with isolation).

is answered but also our first listed problematic in section 3. We can conclude that the interferences substantially disturb the determinism of reasoning runtime in a work environment. So, reasoners are not as effective as they seem, concerning time predictability.

7 Conclusions/Lessons Learned

Our literature review has shown that all reasoners have various features and restrictions, related to their optimisations, for mastering their time and space

[3] Signifies services/process running in background.
[4] Reminder: Hermit is based on Java and hence a Java Virtual Machine is used.

footprint. We presented the first steps towards providing a clear evaluation phase in an embedded system context. At this step of our work, one question could be: can one dreams of an efficient reasoning in embedded systems? To the best of our knowledge, the answer should be positive.

Our future work include investigating how embedded architectures and hardware acceleration techniques can go in pair with optimisation of algorithms. Our ultimate goal is to implement a solution in the real case of virtual assistant.

Acknowledgments. I would like to express my gratitude to my supervisors M. Roy (LAAS-CNRS), F. De Grancey (Thales), and H. Waeselynck (LAAS-CNRS).

References

1. CLIPS: A Tool for Building Expert Systems, https://www.clipsrules.net/
2. List of Reasoners |, http://owl.cs.manchester.ac.uk/tools/list-of-reasoners/
3. OWL 2 Web Ontology Language Profiles (Second Edition), https://www.w3.org/TR/owl2-profiles/#OWL_2_RL
4. Semantic reasoner (Jan 2024), https://en.wikipedia.org/w/index.php?title=Semantic_reasoner&oldid=1193724481
5. Ali, S., Kiefer, S.: A Micro OWL DL Reasoner for Ambient Intelligent Devices. In: Advances in Grid and Pervasive Computing. Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01671-4_28
6. Anderson, C., Hübener, I., Xu, Y., David, K.: An Ontology-Based Reasoning Framework for Context-Aware Applications (Nov 2015). https://doi.org/10.1007/978-3-319-25591-0_34
7. Baader, F., Lutz, C., Suntisrivaraporn, B.: CEL — A Polynomial-Time Reasoner for Life Science Ontologies. In: Automated Reasoning. Berlin, Heidelberg (2006). https://doi.org/10.1007/11814771_25
8. Bento, A., Médini, L., Singh, K., Laforest, F.: Do Arduinos Dream of Efficient Reasoners? In: The Semantic Web. Cham (2022). https://doi.org/10.1007/978-3-031-06981-9_17
9. Bobed, C., Yus, R., Bobillo, F., Mena, E.: Semantic reasoning on mobile devices: Do Androids dream of efficient reasoners? Journal of Web Semantics (Dec 2015). https://doi.org/10.1016/j.websem.2015.09.002
10. De-Grancey, F., Audouy, A.: Towards the Deployment of Knowledge Based Systems in Safety-Critical Systems. In: The Semantic Web: ESWC 2023 Satellite Events. Cham (2023). https://doi.org/10.1007/978-3-031-43458-7_35
11. Grimm, S., Watzke, M., Hubauer, T., Cescolini, F.: Embedded $\mathcal{EL}$+ Reasoning on Programmable Logic Controllers. In: The Semantic Web – ISWC 2012. Berlin, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35173-0_5
12. Iglesias, J., Bernardos, A.M., Alvarez, A., Sacristan, M.: A Light Reasoning Infrastructure to Enable Context-aware Mobile Applications. In: 2010 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing (Dec 2010). https://doi.org/10.1109/EUC.2010.64
13. Ilkou, E., Koutraki, M.: Symbolic Vs Sub-symbolic AI Methods: Friends or Enemies? (2020), https://www.semanticscholar.org/paper/Symbolic-Vs-Sub-symbolic-AI-Methods%3A-Friends-or-Ilkou-Koutraki/9d80b0382d4576676610f502abb954e10b4e6037

14. Jang, M., Sohn, J.C.: Bossam: An Extended Rule Engine for OWL Inferencing (Nov 2004). https://doi.org/10.1007/978-3-540-30504-0_10
15. Kim, T., Park, I., Hyun, S.J., Lee, D.: MiRE4OWL: Mobile Rule Engine for OWL. In: 2010 IEEE 34th Annual Computer Software and Applications Conference Workshops (Jul 2010). https://doi.org/10.1109/COMPSACW.2010.62
16. Kleemann, T.: Towards Mobile Reasoning. (Jan 2006)
17. Koziuk, M., Domaszewicz, J., Schoeneich, R.O., Jablonowski, M., Boetzel, P.: Mobile Context-Addressable Messaging with DL-Lite Domain Model. In: Smart Sensing and Context. Berlin, Heidelberg (2008). https://doi.org/10.1007/978-3-540-88793-5_13
18. Matthews, B.: Semantic Web Technologies. E-learning **6** (Jan 2005)
19. Moore, P., Hu, B., Wan, J.: Smart-Context: A Context Ontology for Pervasive Mobile Computing. Comput. J. (Feb 2010). https://doi.org/10.1093/comjnl/bxm104
20. Motik, B., Horrocks, I., Kim, S.M.: Delta-reasoner: a semantic web reasoner for an intelligent mobile platform. In: Proceedings of the 21st International Conference on World Wide Web. New York, NY, USA (Apr 2012). https://doi.org/10.1145/2187980.2187988
21. Ruta, M., et al.: A multiplatform reasoning engine for the Semantic Web of Everything. Journal of Web Semantics (Jul 2022). https://doi.org/10.1016/j.websem.2022.100709
22. Scioscia, F., et al.: Mini-ME Matchmaker and Reasoner for the Semantic Web of Things (Jan 2018)
23. Seitz, C., Schönfelder, R.: Rule-Based OWL Reasoning for Specific Embedded Devices. In: The Semantic Web – ISWC 2011. Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25093-4_16
24. Sertkaya, B.: The ELepHant Reasoner System Description (Jan 2013)
25. Singh, G., Bhatia, S., Mutharaju, R.: OWL2Bench: A Benchmark for OWL 2 Reasoners. In: The Semantic Web – ISWC 2020 (2020). https://doi.org/10.1007/978-3-030-62466-8_6
26. Sinner, A., Kleemann, T.: KRHyper – In Your Pocket. In: Automated Deduction – CADE-20. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2005). https://doi.org/10.1007/11532231_33
27. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. Journal of Web Semantics (Jun 2007). https://doi.org/10.1016/j.websem.2007.03.004
28. Steller, L., Krishnaswamy, S.: Efficient mobile reasoning for pervasive discovery (Mar 2009). https://doi.org/10.1145/1529282.1529562
29. Steller, L.A., Krishnaswamy, S., Gaber, M.M.: A Weighted Approach to Partial Matching for Mobile Reasoning. In: The Semantic Web - ISWC 2009. Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04930-9_39
30. Tai, W., Keeney, J., O'Sullivan, D.: Resource-Constrained Reasoning Using a Reasoner Composition Approach. Semantic Web (Jan 2015). https://doi.org/10.3233/SW-140142
31. Van Woensel, W., Abidi, S.S.R.: Optimizing Semantic Reasoning on Memory-Constrained Platforms Using the RETE Algorithm. In: The Semantic Web. Cham (2018). https://doi.org/10.1007/978-3-319-93417-4_44