Optimisation of Reasoning over Ontologies on Embedded Hardware for Avionics-Context Semantic-Aware to Generate Minimisation of Computation Time and Memory Footprint

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 explore ways to deploy such reasoning in embedded architectures focusing on optimisations of time/memory envelope and benchmarking. We first explore the related work on existing optimisations of some implementations to get a clear view on reasoning techniques, then we investigate the variability sources to get a clear view on what is going inside a reasoner. The results are (i) a better overview of reasoner methodologies, (ii) a notable categorisation of optimisation families and (iii) a clearer view of impacts concerning reasoning time determinism.

Keywords: Semantic Reasoning · Symbolic AI · Avionics · Embedded System · Ontology · Description Logic · 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 [13]. This has prompted an interest of a sheer number of researchers to extract prominent inventions using the potential of semantics, ontologies and reasoning over many domains of interest such as life science [6], mobiles [12] or avionics. In aeronautics, one of the envisioned applications can be the assistance to the aircrew in order to circumvent human workload limitations.

** Early Stage Ph.D.

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. We study the feasibility of implementing an ontology coupled to a reasoner in an embedded system.

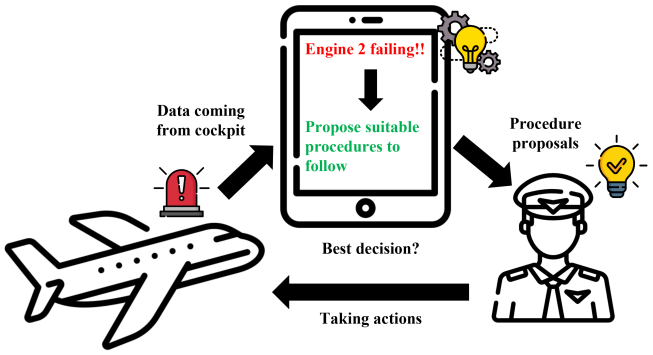

Fig. 1: Schema of a possible situation in our use case (icons from Flaticon.com)

Imagine in a cockpit a warning light shows up so then the pilot has to assess, as quickly as possible, the seriousness of the alarm in order to evaluate the criticality of the situation (see Fig. 1). 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 using explainability feature of reasoning [10]. Notwithstanding, for such case, even if we detain constraints on time and memory space we bound to have a minimal and deterministic reasoning time to help the pilot to understand the situation in time. [8] relates our work and gives a concrete view of time constraint: in the test campaign, 1 second is the maximum time response and some runs exceeds this limit and also show non-determinism of reasoning time for consistency or explanation service. This in not acceptable in the real-time context of avionics!

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.*

1.3 Constraints in On-Board Systems

Embedding ontologies and reasoner(s) is a challenge because these technologies were not initially thought for embedded devices. On the one hand, according to [12] and [7], the world of embedded systems such as mobiles or constrained devices imply many constraints concerning the deployment of a reasoner. In fact, 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 such as in [8].

Additionally in our case, another fully-fledged constraint is added: Time-constraints associated to Real-Time usage. In other words, we coerce time response but even more we need to bound it to meet the time directives.

2 State of the Art

In this section, we showcase the *sine qua non* of fundamentals to serenely tackle our problematic, 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 the profiles [3] OWL 2 EL, QL, and RL.

State of the Art Reasoners
The first step of our work was to gather well-known reasoners in the literature in order to have clues on which one(s) could guide us towards a concrete direction for our problematic. To this extent, we deeply studied 38 reasoners coming from [2] and *LiRoT* [7]. 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.

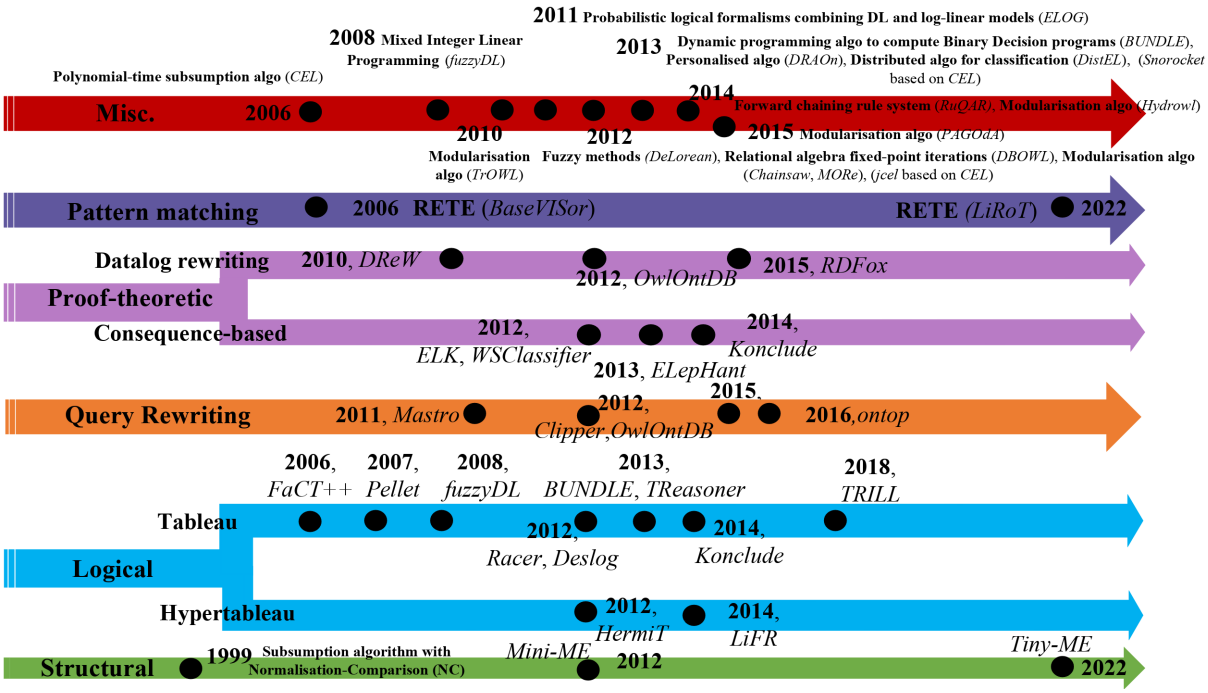

Fig. 2: Timeline regarding reasoning approaches appearance with their associated reasoners from 1999 to 2022

We have categorised the reasoners relatively to their 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. The same is for structural family. Moreover, to tackle performance issues concerning memory and time usage, we have thoroughly studied the existing and notable optimisations (see section 6) of the considered reasoners in order to evaluate how these improvements are suitable with our approach.

2.2 Is embedding reasoning conceivable?

Furthermore, we have extended our research to the reasoners dedicated to embedded systems to examine ups and downs in this area. There are few papers with the development of various approaches such as for Internet of Things (IoT).

Some fairly well-known reasoners are straightforwardly aimed at embedded architectures such as: *Pocket KRHyper* [12,20], *µOR* [5], *MiRE4OWL* [11], *ELepHant* [18], *COROR* [24], *Mini-ME* [16] and *Tiny-ME* [15], or *LiRoT* [7] used in IoT context called Semantic Web of Things (SWoT).

[23] depicts 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* [21] reasoner. [17] presents an OWL reasoner for embedded devices by constructing a global architecture using OWL 2 RL and CLIPS [1] rule engine. In [9], 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* [6] reasoner. [14] 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 [22] which role is to give the possibility to use Tableaux algorithm on mobile architectures, or $\text{RETE}_{\text{pool}}$ [25] that is dedicated to resource-constrained devices. It gathers RETE memory mechanisms in order to get a better memory footprint.

This is an interesting related work because it shows a plethora of implementations for a specific use case towards embedded or constraints hardware. But, the problem of implementing a reasoner on embedded hardware still exist with a continuous interest to the community. Plus, this encourages us even more to find a solution to our problematic.

3 Problem Statement and Contributions

Accordingly, the objective of our thesis is to propose an efficient reasoning over ontologies to be embedded in avionics-context by introducing optimisation to generate minimisation of reasoning time and memory consumption by tackling the next questions:

- How efficient are current reasoners to handle our problematic?
- Can we combine existing optimisations[2] from standard reasoners to achieve our goal?

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

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

- 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

We aim to explore optimisation methods for ontology reasoning with this approach:

1. Establish a state of the art on reasoners and those dedicated to embedded architectures to classify optimisations to see what exist and could help us towards an efficient solution;
2. Test Bench: generate an ontology corpus, using e.g. [19], and use a list of reasoners to test. The goal is find out the different ontology metrics and different processing steps that could impact the reasoners performances and their existing optimisations;
3. Based on the outcomes of the test bench step, we will propose new optimisation methods of reasoners to enhance performances thereof and then tackle embedding constraints.
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 and assess how understandable are the outputs for different users such as the consumer or the builder.

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. 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 [19], to explore various features of ontologies such as expressiveness or size of ABox against a set of reasoners and therefore get a global view of relations between ontology metrics, memory space and reasoning runtime. To avoid generic results, we will tailor them towards avionic-context;

– Conduct reasoning experiments on different hardware architectures and compare each in turn. More specifically, we will deeply study the given reasoner over ontologies through a scalability process regarding notable metrics such as its TBox size or expressiveness. 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 algorithmic 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 (e.g. classification) 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 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 shows different families, each one concerning

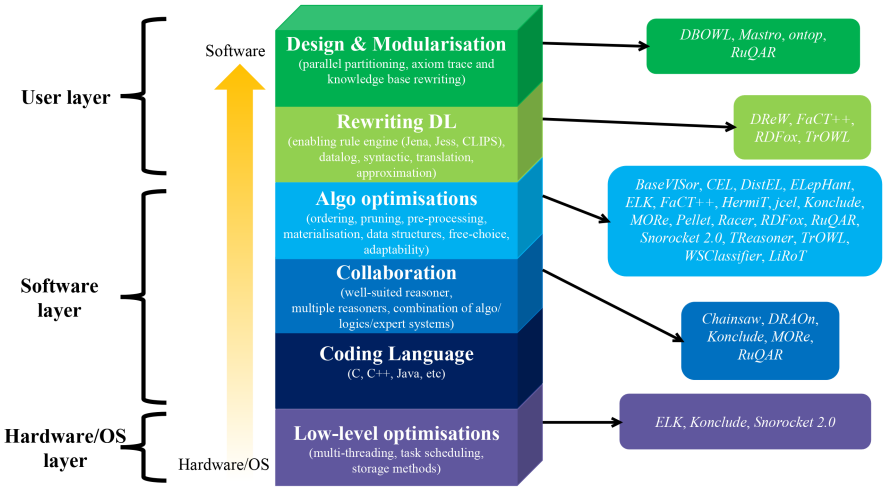

Fig. 3: Categorisation of different optimisation families from studied existing reasoners

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 that 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 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

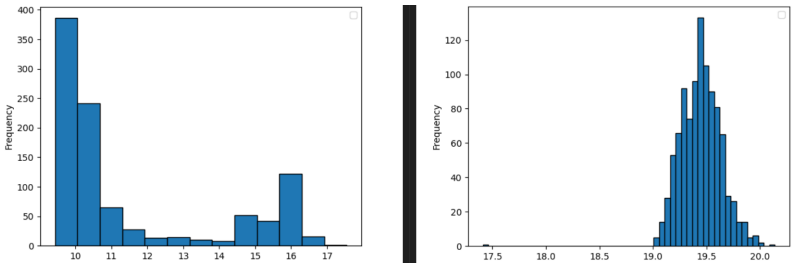

Fig. 4: Histogram regarding the distribution of *HermiT* runtime (seconds) with their frequency, 1000 times, against the same ontology consisting of 9, 227 axioms (on the left with noisy environment/on the right with isolation)

that our hypothesis 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 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:

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

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/
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. 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
7. 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
8. 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
9. Grimm, S., Watzke, M., Hubauer, T., Cescolini, F.: Embedded 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
10. 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
11. 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
12. Kleemann, T.: Towards Mobile Reasoning. (Jan 2006)
13. Matthews, B.: Semantic Web Technologies. E-learning **6** (Jan 2005)
14. 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
15. 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

16. Scioscia, F., et al.: Mini-ME Matchmaker and Reasoner for the Semantic Web of Things (Jan 2018)
17. 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
18. Sertkaya, B.: The ELepHant Reasoner System Description (Jan 2013)
19. 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
20. 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
21. 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
22. Steller, L., Krishnaswamy, S.: Efficient mobile reasoning for pervasive discovery (Mar 2009). https://doi.org/10.1145/1529282.1529562
23. 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
24. 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
25. 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