CHARIOT - Extension of an existing Semantic Web Reasoner

Deliverable 4.5

WP4: Semantic Services & Data



Version	1.1
Due Date	30.09.2018
Delivery Date	30.09.2018
Nature	Report

List of Figures

Table of Acronyms

Acronym	Meaning
OWL	Web Ontology Language
QoS	Quality of Service
IoT	Internet of Things

Table of Contents

1. Introduction

WP4-specific introduction (see Fig 1). When we speak about *IoT entity*, we mean both simples and complex devices as well as virtual services.

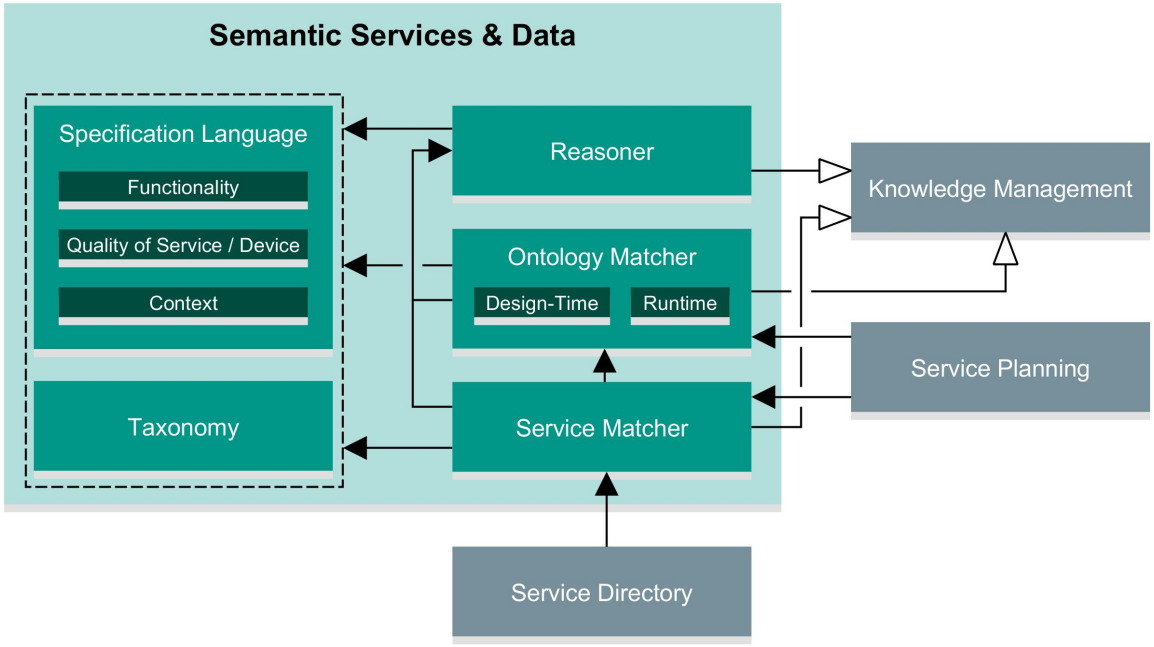

Figure 1: Overview of the Semantic Services & Data Work Package

2. Requirements

The requirements for an IoT reasoner can be specified as follows:

- The reasoner is lightweight and fast, thus, can be run online on a simple, restricted IoT device. The results then are allowed to be suboptimal.
- For offline purposes, the reasoner can use a more precise but slower algorithm.
- Due to the use of the JIAC V framework, an implementation in Java (or a Java interface) is preferred.
- The reasoner should be published under an open-source license, which allows to change the source code.
- The reasoner should work for the OWL language.
- SWRL support is needed, because it plays a crucial role in the service descriptions inside CHARIOT.

The requirements mentioned before are generic attributes, which should be fulfilled by a reasoner. Aside from this an IoT reasoner in the context of CHARIOT should be extended by implications that can be drawn by certain knowledge such as preconditions & effects of services/devices, current context and derived future context. A good overview over semantic reasoning is given in [5].

3. Background

In a very common speaking, a reasoner takes a set of axioms defined in an ontology and offers certain actions on these that can be taken out. To be specific, the actions provided by a reasoner are:

1. *Classification* [1]: Computes all subsumption relationships inferred in an ontology between concept, role, and attribute names in the ontology signature. It consists of three steps [2], which will be explained in the following.
    a. *Consistency*: Check, if there exists a model for the ontology. This means to check whether there exists a structure that satisfies all axioms.
    b. Satisfiability: For every class it is checked whether there exists a model for the ontology with an instance for the class.
    c. Subsumption
2. Retrieve instances, subclasses, or superclasses of a given, anonymous class expression
3. Answer conjunctive queries.

In [2] a reasoner further specified: “[...] it is sound, complete, and terminating: i.e., all entailments it finds do indeed hold, it finds all entailments that hold, and it always stops eventually. If a “inference engine” relies on an algorithm that is not sound or complete or terminating, we would not usually call it a reasoner”.

3.1 Basics

This section gives a quick overview over certain basics such as how a reasoner works internally.

Reasoner’s Internal Working

Two main approaches are existing in the field of reasoning: *consequence-driven reasoning* and *tableau-based reasoning*[1]. “Consequence-driven approaches have been shown to be very efficient for so-called Horn fragments of OWL [2]”. From a set of axioms in an ontology, deduction rules are applied to infer entailments (“consequences”) [4]. ELK uses this approach. “Tableau-based reasoner try to construct [...] a model using so-called completion rules [...], trying to extend it so that it satisfies all the axioms [2]”. Tableau-based approaches have been developed for more expressive DLs.

OWL

The OWL 2 Web Ontology Language is based on Description Logic, a family of formal knowledge representation languages. The primary exchange syntax for OWL 2 is *RDF/XML*, other possible syntaxes are, e.g., *Turtle*, an XML serialization, and a more "readable" syntax called the *Manchester Syntax*.

OWL Sublanguages

Full, *DL*, and *Lite* are three variants of OWL, each constituting different compromises between expressivity and computational complexity.

[1] https://en.wikipedia.org/w/index.php?title=Method_of_analytic_tableaux&oldid=856959697

OWL Full has provides maximum expressiveness, syntactic freedom, but without computational guarantees. The semantics of OWL Full is a mixture of RDFS and OWL DL (RDF-based semantics).
OWL DL is a restricted version of OWL Full. OWL DL provides very high expressiveness, computational completeness (all conclusions are guaranteed to be computable), and decidability (all computations can be finished in finite time). While OWL DL includes all OWL language constructors, they can be used only under certain restrictions. For example, OWL DL number restrictions may not be assigned to transitive properties.
OWL Lite is a subset of OWL DL designed for easy implementation. OWL Lite has limited applicability, because it is suitable only for classification hierarchies and simple constraints. Since the development of OWL Lite tools has proven almost as difficult as development of tools for OWL DL, OWL Lite is not widely used.[2]

OWL Profiles

EL, *QL*, and *RL* are three OWL profiles, each of which provides a different balance between expressive power and reasoning complexity, thereby providing more options for different implementation scenarios.
The *EL profile* was designed for handling ontologies with very large numbers of properties and/or classes, the *QL profile* is aimed at applications with a very large instance data volume and a priority for query answering, and the *RL profile* was designed for applications that require scalable reasoning with relatively high expressivity.[1]

SWRL

3.2 Existing Reasoners

Pellet

(Clark & Parsia)
was the first reasoner that supported all of OWL DL and has been extended to OWL 2. Pellet is implemented in **Java** and supports OWL 2 profiles including OWL 2 EL. From the webpage: “Pellet can be used with Jena or OWL API libraries. Pellet provides functionality to check consistency of ontologies, compute the classification hierarchy, explain inferences, and answer SPARQL queries.” The reasoner is open source (AGPL), but apparently no longer under active development. https://github.com/stardog-union/pellet

FaCT++

(Fast Classification of Terminologies, University of Manchester)
Implemented in **C++** (has an **JNI interface** to Java) at the University of Manchester, supports SWRL rules and shows exceptional performance on expressive ontologies in OWL DL. It is an open-source software and is distributed under the LGPL license and can be used as a back-end reasoner for **OWL API 4**. https://bitbucket.org/dtsarkov/factplusplus

HerMiT

(University of Oxford)

[2] https://stackoverflow.com/a/47088998

Supports OWL DL; can determine whether or not a given ontology is consistent and identify subsumption relationships between concepts, among other features. HermiT is based on a “hypertableau” calculus. http://www.hermit-reasoner.com/

ELK

From the webpage: “ELK is an ontology reasoner that aims to support the OWL 2 EL profile.” It is implemented in Java, uses the consequence-driven approach, features a currently only partially implemented OWL API interface and licensed under the Apache-2.0 license. The development status can be described as active. https://github.com/liveontologies/elk-reasoner

Mini-ME

From the webpage: “[...] a prototypical mobile reasoner for the *Semantic Web of Things*. It supports standard Semantic Web technologies through the OWL API and implements both standard reasoning tasks for knowledge base (KB) management (subsumption, classification, satisfiability) and non-standard inference services for semantic-based matchmaking and resource ranking (abduction, contraction and covering). Mini-ME is developed in **Java**, adopting Android as the current target computing platform, but running also on Java SE.” It currently supports **OWL API 3**. **The project is only available for academic review and evaluation with prohibiting any other usage**. http://sisinflab.poliba.it/swottools/minime/

Hydrowl

This reasoner is a “Hybrid Query Answering System for OWL 2 DL Ontologies”. From the webpage: “Hydrowl is based on novel hybrid techniques which attempt to combine many existing Semantic Web technologies in an effort to deliver scalable query answering over very expressive fragments of OWL 2 DL. Hence, it does not assume that expressivity is restricted to one of the known tractable profiles (e.g., OWL 2 QL), but it tries to tackle arbitrary OWL 2 DL ontologies.” Hydrowl is available under the GNU Affero General Public License. **However, the provided download links seem dead**. http://www.image.ece.ntua.gr/~gstoil/hydrowl/

3.3 Existing Work at the DAI-Lab / GT-ARC

According to *Nils Masuch / Johannes Fähndrich / Lars Borchert* the reasoner Pellet was used in the project AcRoSS. The reasoner itself was not modified, but, e.g., the support for the SWRL list construct was added. Another implementation uses the Pellet interfaces to get insight into why a certain model would be classified as unsolvable: For this the team developed methods to checks which concrete atom was responsible. https://gitlab.dai-labor.de/jiacv/servicematcher/tree/master/seMa2Reasoner

3.4 Existing Frameworks

Apache Jena

A free and open source Java framework for building Semantic Web and Linked Data applications. It consists of several APIs, including the three main parts RDF (RDF API & ARQ (SPARQL) API), OWL (Ontology API & Reasoning API) and triple stores (TDB & Fuseki). Jena's ontology and reasoning support is based on the RDF graph. Jena provides the usage of different built-in reasoners that cover all OWL sublanguages (see OWL Sublanguages). Furthermore, it is also possible to define own reasoners. (there is already an in-process interface to Pellet, see also Pellet). *https://jena.apache.org/*

3.5 Existing APIs

OWL API

An API for creating, manipulating and serialising OWL Ontologies in Java. The OWL API provides interfaces for working with reasoners such as FaCT++, HermiT, Pellet, Racer, JFact and Chainsaw. A great benefit of this API is the possibility to import multiple remote as well as local ontologies into a joint knowledge base (Jena is only capable of representing one ontology in a graph model). The current major version is OWL API 5. https://github.com/owlcs/owlapi

ONT-API

An API to connect Jena and OWL API; from the webpage: "ONT-API is a Jena-based OWL-API, thus, in addition to the OWL-API interfaces, there is also jena-way to interact with ontology graph directly." *https://github.com/avicomp/ont-api*

OWLlink

OWLlink[3] itself is an extensible implementation-neutral protocol for communication with OWL 2 reasoning systems. The protocol facilitates client applications to manage reasoners, to assert axioms and to access reasoning services via a set of standard queries. The OWLlink API is a reference implementation that implements the OWLlink protocol on top of the OWL API. It allows to turn OWL-API-aware reasoners into OWLlink servers and to access remote OWLlink servers from OWL API based applications. *https://sourceforge.net/projects/owllink-owlapi/*

4. Concept

Due to the requirements made earlier in this document, combined with the collected background knowledge about OWL and reasoners, the concept envisions to not just use and extend one single reasoner, since this seems to conflict with some of the requirements,

[3] http://www.owllink.org/

especially the IoT-related ones. Instead, a more sophisticated solution is preferred (see Fig 2).

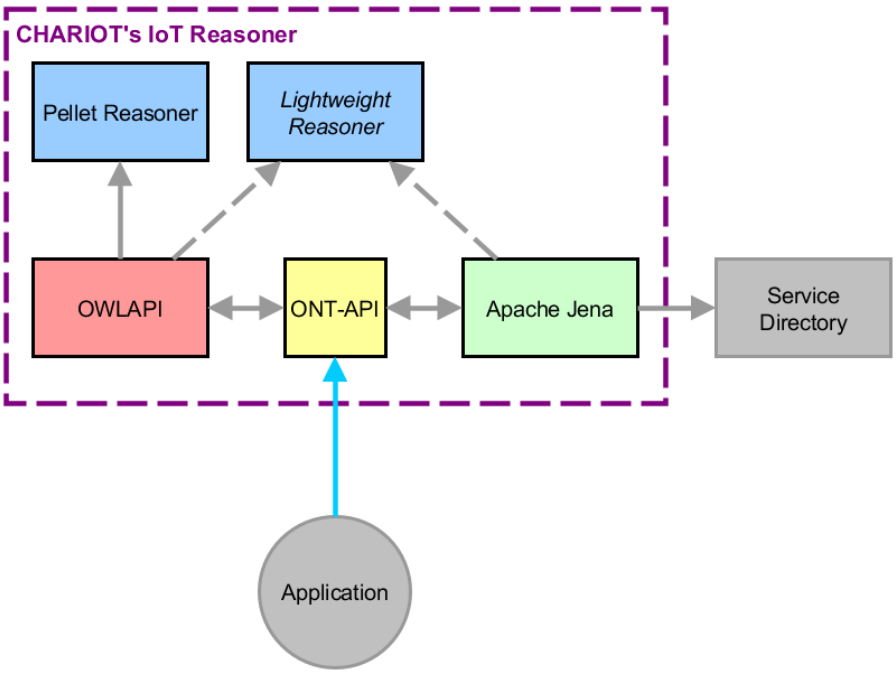

Figure 2: Overview of the CHARIOT Reasoner

In the image the reasoner for CHARIOT can be seen as the purple box that consists of several submodules. An application will use the reasoner through an adequate API. Because of the strength of the OWLAPI it will be used in combination with the extendable Pellet reasoner for the “precise” mode. The Apache Jena Framework will be used to have a direct interface to the service directory. It is reasonable to be able to retrieve additional information about provided services and devices throughout the reasoning process. The ONT-API is needed to be able to use bot the OWLAPI and the Jena Framework in a joint model.

The Pellet reasoner will be used and extended as the main, precise but slow reasoner as it is open-source and Java-based. The yet unnamed “Lightweight Reasoner”, unfortunately, has to be a modified version of the built-in Jena reasoners, or a self-developed OWL QL reasoner (the former option is preferred). Dependent on how this “fast” mode reasoner is implemented it also has to be determined if the reasoner is used by the OWLAPI or the Jena Framework.

References

Web Links

- http://owl.cs.manchester.ac.uk/tools/list-of-reasoners/: a list of existing reasoners. Each reasoner has a brief description. Unfortunately, this list is a little outdated. However, this list can be used for a first overview.
- https://www.w3.org/2001/sw/wiki/OWL/Implementations: Another list of existing reasoners along with brief descriptions (which are more structured). Even editors, development environments and APIs are listed.

- https://jena.apache.org/documentation/inference/: an extensive webpage about the inference support of Jena. It serves good as an entry point into reasoning with the Jena framework.
- https://www.w3.org/TR/owl2-overview/: a non-normative high-level overview of the OWL 2 Web Ontology Language.

Literature

[1] Lembo D., Santarelli V., Savo D.F. (2013). Graph-Based Ontology Classification in OWL 2 QL. In: Cimiano P., Corcho O., Presutti V., Hollink L., Rudolph S. (eds) The Semantic Web: Semantics and Big Data. ESWC 2013. Lecture Notes in Computer Science, vol 7882. Springer, Berlin, Heidelberg

[2] Uli Sattler, Robert Stevens, Phillip Lord (2014). How does a reasoner work?. *Ontogenesis*. http://ontogenesis.knowledgeblog.org/1486

[3] Dentler, K., Cornet, R., Ten Teije, A., & De Keizer, N. (2011). Comparison of reasoners for large ontologies in the OWL 2 EL profile. *Semantic Web*, *2*(2), 71-87.

[4] Polleres, Axel, et al., eds. Reasoning Web. Semantic Technologies for the Web of Data: 7th International Summer School 2011. Galway, Ireland, August 23-27, 2011, Tutorial Lectures. Vol. 6848. Springer Science & Business Media, 2011.

[5] K. Dentler, R. Cornet, A. Ten Teije, and N. De Keizer. Comparison of reasoners for large ontologies in the OWL 2 EL profile. Semant. Web, vol. 2, no. 2, pp. 71–87, 2011.

Interesting Links

http://vowl.visualdataweb.org/webvowl.html