RuQAR : Reasoning with OWL 2 RL Using Forward Chaining Engines

Jaroslaw Bak

Institute of Control and Information Engineering,
Poznan University of Technology,
Piotrowo 3a, 60-965 Poznan, Poland
jaroslaw.bak@put.poznan.pl

Abstract. We present the Rule-based Query Answering and Reasoning framework (RuQAR). The tool supports ABox reasoning and query answering with OWL 2 RL ontologies executed by the forward chaining rule reasoners Jess and Drools. We describe RuQAR's main features, its architecture as well as implementation details.

1 Introduction

Ontologies, as a way of expressing knowledge, are becoming more and more popular in various fields, such as web technologies, database integration, medical systems etc. Ontologies can be expressed in the Web Ontology Language 2 (OWL 2). The language provides a set of profiles[1] which offer important advantages in different application scenarios. Among them, the OWL 2 RL profile is the most interesting one from our point of view. It enables an implementation of polynomial time reasoning algorithms in a standard rule engine. Nonetheless, a naive implementation of an OWL 2 RL reasoner is known to perform poorly with large ABoxes [3]. Additionally, description logic-based reasoners handle the TBox entailments better than the ABox ones. However, ABox reasoning can be performed more efficiently by a rule engine [7]. Nevertheless, the official list[2] of OWL 2 reasoners supporting OWL 2 RL is limited. Moreover, combining an OWL 2 RL reasoner with a currently used forward reasoning engine can be a tricky task, because existing reasoners usually provide their own format and reasoning algorithms. Furthermore, there is a lack of native and efficient rule sets that support OWL 2 RL reasoning in many popular rule engines, especially considering ABox reasoning. This motivated us to provide a tool which supports the application of an OWL 2 RL-based knowledge base in a forward chaining rule engine. Nevertheless, we do not limit ourselves to one particular engine or implementation. Instead, we aim at providing an easy-to-use framework for performing ABox reasoning with OWL 2 RL ontologies in any forward chaining rule engine such that it can be used in many rule-based applications.

[1] http://www.w3.org/TR/owl2-profiles/

[2] http://www.w3.org/2001/sw/wiki/OWL/Implementations

In this paper we provide a detailed description of the Rule-based Query Answering and Reasoning framework (RuQAR) which overcomes the aforementioned issues. The main goal of the tool is to support ABox reasoning as well as query answering within the OWL 2 RL profile. The remainder of this paper is organized as follows. Firstly, we describe main features of RuQAR. Then, we present its architecture as well as implementation details. Finally, we provide conclusions along with future development plans.

2 RuQAR Framework

2.1 Features

The RuQAR framework is aimed at providing easy-to-use functions that will support reasoning and query answering with ontologies within OWL 2 RL. These tasks should be performed by a forward reasoning rule engine. For this, an ontology needs to be transformed into rules that are readable by a chosen engine. According to this we have developed the following features for RuQAR:

1. The Abstract Syntax of Rules and Facts (ASRF) which is used to rise an abstraction level providing more universal representation of rules and facts. As a result the syntax enables easy translation into the language of any rule engine. An implementation of mappings between ASRF and the language of a chosen rule engine is required.
2. Transformation schema of an OWL 2 ontology into a set of rules and a set of facts expressed in ASRF. The transformation schema is presented in Figure 1. Firstly, an OWL 2 ontology is loaded into the HermiT[3] engine. Then, TBox reasoning is executed. We perform the following reasoning tasks: satisfiability checking, concept classification and subsumption as well as checking equivalence and disjointness between concepts. Finally, the resulting ontology is transformed into two sets: one of rules and one of facts. These sets reflect TBox and ABox separately. Both are expressed in the ASRF notation. By loading a translated and inferred ontology, produced by HermiT, into a rule engine we can derive more consequences during ABox reasoning than those supported by OWL 2 RL. However, it depends on an applied ontology (whether or not it uses constructs that are beyond OWL 2 RL). Nevertheless, RuQAR supports only rules that are presented in Table 1. Moreover, since we use the OWL-API [4] tool, HermiT can be exchanged through an another compatible reasoner (e.g. Pellet[4]).
3. Translation of the ASRF sets into Drools[5] and Jess[6] languages (we implemented appropriate mappings). As a result these sets can be used to perform ABox reasoning with the corresponding engines.

[3] `http://www.hermit-reasoner.com/`
[4] `https://github.com/complexible/pellet`
[5] `http://www.drools.org/`
[6] `http://jessrules.com/`

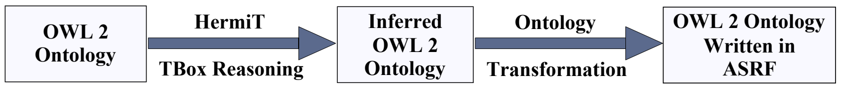

Fig. 1. OWL 2 ontology transformation schema

4. Support for the Semantic Web Rule Language (SWRL) [5] with its built-ins. Additionally, RuQAR checks if a rule is safe (whether each variable from the rule's head occurs in the rule's body).
5. Mapping method between an ontology and a relational database. Currently, we are developing an interface which is based on the R2RML specification.[7]
6. Query answering functions that allow for querying Jess and Drools engines with their native methods as well as our own methods. One of our method uses a relational database to store the ABox part of an ontology. Another one provides some optimizations regarding rules that are used while processing the query. It is based on a previously developed extended rules approach [1].
7. Reasoning and query answering management functions for Jess, Drools and HermiT. Moreover, additional functions are also available. For example, we can compare reasoning results (or query answers) computed by the engines.

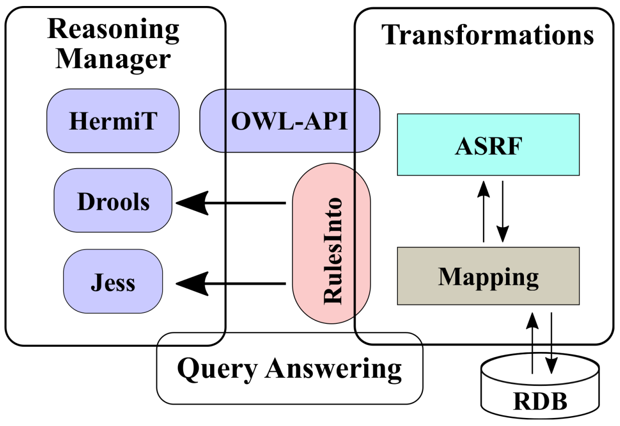

Fig. 2. The architecture of RuQAR

2.2 Architecture and Implementation Details

RuQAR is developed in Java as a library which can be included in applications requiring efficient ABox reasoning. The framework uses the following tools: Jess 7.1, Drools 5.5, HermiT 1.3.8 and OWL-API 3.4.10.

[7] `http://www.w3.org/TR/r2rml/`

The architecture of RuQAR is presented in Figure 2. The tool contains a set of modules that provide different functionalities. If a module overlaps another one in Figure 2 it means that they share some part of the RuQAR's Java code. The tool contains the following modules:

Transformations Module
This module uses OWL-API to handle ontology files as well as to extract the logical axioms from an ontology. The module creates ASRF rules and facts and is responsible for creating mappings between a relational database and an OWL ontology. It provides *RulesInto* interface which has to be implemented in order to translate the ASRF notion into the language of a selected reasoning engine. An implementation of this interface requires to define mapping between ASRF elements and the destination language. The current implementation provides mappings for Jess and Drools. Both engines use different languages as well as different approaches for rule and fact representations. Jess uses its own scripting language and represents rules and facts as internal Java objects. However, rules in Drools are represented as *when...then...* statements in which any Java class can occur (as a pattern). Moreover, facts can be represented as pure Java objects. As a result we developed an additional class *Triple* which can be applied in the body of each rule as well as to represent facts as *Triple* objects. Thus, we confirmed that the ASRF syntax can be applied in different rule reasoning engines.

Reasoning Manager Module
The module provides a set of functions to handle reasoning engines. We can use this module to perform ABox and TBox reasoning separately using HermiT. We can manage Drools and Jess when performing ABox reasoning or query answering. We can save results as a new OWL ontology. Additionally, we can obtain the profile information about a given ontology. Whether it is within OWL 2 RL or not. If it is not, the violations are provided (mainly due to the OWL-API functions).

Query Answering Module
This module supports query answering with Drools and Jess. We can use native methods of those engines or our (optimized) methods that include a relational database access. Application of our method requires rules to be modified. However, this module provides such a modification which can be applied automatically with the ASRF rules. As a result we provide an optimized set of rules which can be translated into Drools or Jess rules out of the box.

RuQAR implements our method of transforming OWL 2 ontologies into a set of rules and a set of facts expressed in the ASRF syntax. Table 1 shows currently supported rules by our implementation. This set comes from the specification of OWL 2 [8]. However, this set is smaller than the original one. We decided to use the simplest subset of OWL 2 RL/RDF rules which is easily implementable in any reasoning engine. Moreover, we excluded each rule which is a "constraint"

Table 1. Currently supported OWL 2 RL entailment rules.

OWL 2 RL Specification Table	**Supported Rules**
Table 4. The Semantics of Equality	eq-sym, eq-trans, eq-rep-p eq-rep-s, eq-rep-o
Table 5. The Semantics of Axioms about Properties	prp-dom, prp-rng, prp-fp, prp-ifp, prp-symp, prp-trp, prp-eqp1, prp-spo1, prp-eqp2, prp-inv1, prp-inv2
Table 6. The Semantics of Classes	cls-int1, cls-int2, cls-uni, cls-svf1, cls-svf2, cls-avf, cls-hv1, cls-hv2, cls-maxc2
Table 7. The Semantic of Class Axioms	cax-sco, cax-eqc1, cax-eqc2

rule (e.g. *cls-nothing2* from Table 6 in the OWL 2 RL profile) and each rule which does not have an impact on ABox reasoning (e.g. all rules from Table 9 in the OWL 2 RL profile).

In ontology-to-ASRF transformation we translate each logical ontology axiom into its equivalent rule. As a result the transformation materializes the semantics of a given ontology in a set of Datalog-like rules (we consider it as a non-naive translation). Since we are focused on ABox reasoning, each rule should be perceived as an implementation of the axioms from a given ontology. According to this, we obtain a significant performance gain [2].

The transformation method may produce more entailments during reasoning than those represented by OWL 2 RL/RDF rules. It is caused by the fact that we apply TBox reasoning with a DL-based reasoner. However, it depends on the expressivity of a given ontology. Nevertheless, the application of our method to ontology beyond OWL 2 RL will not produce the same consequences as derived by an appropriate DL-based reasoner (in our case the set of inferences will be smaller). As a result, the reasoning with rules generated by our methodology is sound but not complete.

Since RuQAR generates rules and facts in the native language of a rule engine, the tool can be easily integrated into an existing environment. For example, assume that we have an application X which uses Drools to perform reasoning with some data. If we create an ontology that describes the data, and then we translate the data into RDF triples, we can employ RuQAR to generate rules (and facts) that can be easily and directly applied with X.

3 Conclusions and Future Work

In this paper we presented the RuQAR tool which is a framework that provides many useful functionalities to: (i) perform ABox reasoning and query answering with OWL 2 RL ontologies executed by forward chaining rule reasoners, (ii) translate an OWL ontology into rules, (iii) use SWRL rules together with ontologies, (iv) manage reasoning engines and to (v) store OWL individuals in a relational database.

Moreover, presented work is the first implementation of OWL 2 RL reasoning in Drools and Jess (except the work presented in [9] that implements directly the semantics of OWL 2 RL) which can be applied in any application requiring efficient ABox reasoning. Transformation provided by RuQAR should be considered as a generic one. Rules generated for different rule engines are the same – they differ only in the language of a reasoning engine. The next step of development is to provide rules that are optimized according to a chosen rule engine.

Currently, we are implementing R2RML mappings which will be useful to measure the performance of query answering algorithms. In this case we are going to use the NPD benchmark [6] which is specifically designed for Ontology Based Data Access systems (since RuQAR provides relational data access it can be perceived as a OBDA/RBDA[8]-like system). We also plan to perform tests with the latest versions of Jess and Drools, 8.0 and 6.2, respectively. It will be useful to check if reasoning efficiency as well as query answering performance have been increased in newer versions. As a result, in Drools case, we will be able to compare two different algorithms: ReteOO (Drools 5.5) and PHREAK (Drools 6.2).

More information about the RuQAR framework can be found at its web page: `http://etacar.put.poznan.pl/jaroslaw.bak/RuQAR.php`.

Acknowledgments. The work presented in this paper was supported by UMO-2011/03/N/ST6/01602 grant and 04/45/DSMK/0124 project.

References

1. Jaroslaw Bak, Grażyna Brzykcy, and Czeslaw Jedrzejek. Extended rules in knowledge-based data access. In *Proceedings of the 5th international conference on Rule-based modeling and computing on the semantic web*, RuleML'11, pages 112–127, Berlin, Heidelberg, 2011. Springer-Verlag.
2. Jaroslaw Bak, Maciej Nowak, and Czeslaw Jedrzejek. RuQAR: Reasoning framework for OWL 2 RL ontologies. In Valentina Presutti, Eva Blomqvist, Raphaël Troncy, Harald Sack, Ioannis Papadakis, and Anna Tordai, editors, *The Semantic Web: ESWC 2014 Satellite Events - ESWC 2014 Satellite Events, Anissaras, Crete, Greece, May 25-29, 2014, Revised Selected Papers*, volume 8798 of *Lecture Notes in Computer Science*, pages 195–198. Springer, 2014.

[8] `http://wiki.ruleml.org/index.php/Rule-Based_Data_Access`

3. Aidan Hogan and Stefan Decker. On the ostensibly silent W in OWL 2 RL. In Axel Polleres and Terrance Swift, editors, *Web Reasoning and Rule Systems*, volume 5837 of *Lecture Notes in Computer Science*, pages 118–134. Springer Berlin Heidelberg, 2009.
4. Matthew Horridge and Sean Bechhofer. The OWL API: A Java API for working with OWL 2 ontologies. In *OWLED*, 2009.
5. Ian Horrocks, Peter F. Patel-schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. SWRL: A semantic web rule language combining OWL and RuleML. 2004. Accessed: 04/04/2013.
6. Davide Lanti, Martin Rezk, Guohui Xiao, and Diego Calvanese. The NPD Benchmark: Reality Check for OBDA Systems. In *Proc. of the 18th Int. Conf. on Extending Database Technology (EDBT 2015)*, 2015. To appear.
7. Georgios Meditskos and Nick Bassiliades. Combining a DL reasoner and a rule engine for improving entailment-based OWL reasoning. In *Proceedings of the 7th International Conference on The Semantic Web*, ISWC '08, pages 277–292, Berlin, Heidelberg, 2008. Springer-Verlag.
8. Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, and Carsten Lutz. OWL 2 Web Ontology Language Profiles (Second Edition). W3C Recommendation, 2012.
9. Martin J. O'Connor and Amar Das. A pair of OWL 2 RL reasoners. In Pavel Klinov and Matthew Horridge, editors, *OWLED*, volume 849 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2012.