Rule-Based Reasoning System for OWL 2 RL Ontologies

Jaroslaw Bak and Czeslaw Jedrzejek

Institute of Control and Information Engineering,
Poznan University of Technology,
M. Sklodowskiej-Curie Sqr. 5, 60-965 Poznan, Poland
{jaroslaw.bak,czeslaw.jedrzejek}@put.poznan.pl

Abstract. In this paper we present a method of transforming OWL 2 ontologies into a set of rules which can be used in a forward chaining rule engine. We use HermiT reasoner to perform the TBox reasoning and to produce classified form of an ontology. The ontology is automatically transformed into a set of Abstract Syntax of Rules and Facts. Then, it can be transformed into any forward chaining reasoning engine. We present an implementation of our method using two engines: Jess and Drools. We evaluate our approach by performing the ABox reasoning on the number of benchmark ontologies. Additionally, we compare obtained results with inferences provided by the HermiT reasoner. The evaluation shows that we can perform the ABox reasoning with considerably better performance than HermiT. We describe the details of our approach as well as future research and development.

1 Introduction

In the last decade, the use of ontologies in information systems has become more and more popular in various fields, such as web technologies, database integration, multi agent systems, natural language processing, etc. One of the most popular way to express an ontology is to use the Web Ontology Language (OWL) [12]. It is based on description logics (DLs) which are a family of knowledge representation languages.

In order to utilize all features that an ontology provides we need to apply a reasoning engine. However, we can use different engines with ontologies expressed in different OWL 2 Profiles [10] (as well as in different fragments of OWL 1.1[1], eg. Horn-$\mathcal{SHIQ}$). For instance, for an ontology within the OWL 2 EL profile we can use the HermiT[2] reasoner; but for an ontology within the OWL 2 QL profile, which expressive power is quite limited, we can use the REQUIEM[3] reasoner. As a result it is important to choose the right reasoner for a given ontology in order to obtain the best possible results in reasoning or query answering.

In this work we focus on ontology-based reasoning using a standard forward chaining rule engine. Thus, we mainly concentrate on the OWL 2 RL profile.

[1] http://www.w3.org/Submission/owl11-overview/
[2] http://www.hermit-reasoner.com/
[3] http://www.cs.ox.ac.uk/isg/tools/Requiem/

D. Hwang et al. (Eds.): ICCCI 2014, LNAI 8733, pp. 404–413, 2014.

However, the presented methodology can handle ontologies with expressivity beyond OWL 2 RL. This is enabled by employing HermiT to perform the TBox reasoning (with the terminological part of an ontology). As a result we can apply a rule-based reasoning engine to perform the ABox reasoning (with the assertional part of the ontology). It is in accordance with the idea behind OWL 2 RL - a requirement of scalable reasoning without the significant loss of the expressive power. In that case, a relatively lightweight ontology can be applied to perform inferences over a large number of instances.

In this paper we present a reasoning tool which is able to perform ontology-based reasoning using a standard forward chaining rule engine. The paper makes the following contributions:

- we present a transformation method of an OWL 2 ontology into a set of rules and a set of facts (if an ontology contains ABox),
- we propose Abstract Syntax of Rules and Facts (ASRF),
- we provide a reasoning schema compatible with our methodology,
- we describe an implementation of our approach using two forward chaining rule engines: Jess [4] and Drools[4],
- we evaluate our methodology by performing experiments using OWL 2 compatible ontologies and the number of reasoning engines.

The remainder of this paper is organized as follows. Firstly, we introduce the background and motivation of our work. Then, we describe our approach of an OWL 2 ontology transformation into two sets of rules and facts, respectively. Next, we provide our Abstract Syntax of Rules and Facts. Later, we present the implementation details as well as experiments. Finally, we describe the related work and we present the conclusions as well as future directions of our research.

2 Background and Motivation

Rule-based approaches to ontology-based reasoning achieve significant gains in reasoning complexity [15]. However, the current specification of the OWL 2 RL Profile provides the number of predefined entailment rules as a starting point for practical implementation with rule-based systems. These rules, called the OWL 2 RL/RDF[5] rules are based on universally quantified first-order implications over RDF[6] triples which are represented as ternary predicate T with three elements: the subject, the predicate and the object. Moreover, OWL 2 RL/RDF rules follow the OWL 2 RDF-Based Semantics[7] which is the semantics of OWL 2 Full (which is known to be computationally undecidable with regard to consistency and entailment checking [2]). Nevertheless, if an OWL 2 RL ontology satisfies Theorem PR1 in [10] it follows OWL 2 Direct Semantics[8] which is the

[4] http://www.jboss.org/drools
[5] http://www.w3.org/2007/OWL/wiki/Profiles#OWL_2_RL
[6] http://www.w3.org/TR/rdf-primer/
[7] http://www.w3.org/TR/owl2-rdf-based-semantics/
[8] http://www.w3.org/TR/owl2-direct-semantics/

typical semantics for the OWL 2 RL Profile (description logic-based semantics). This characterization is one of the most confusing thing in the OWL 2 specification. As a result, an implementation of a reasoning engine which follows the OWL 2 Direct Semantics requires to satisfy preconditions in Theorem PR1. However, the assertional entailments obtained from a rule-based reasoning engine using OWL 2 RL/RDF rules over an OWL 2 RL ontology follow the Direct Semantics [14]. It means if we want to apply OWL 2 RL/RDF rules, we need to perform the TBox entailments using different reasoning engine. Though, we can raise the abstraction level (from triple-based rule representation) and instead represent the input OWL 2 RL ontology using an axiom-based data structure as shown in [11]. However, the main difference between both semantics is that the RDF-Based Semantics can be applied to arbitrary RDF graphs [7] which do not respect the various restrictions of the OWL 2 syntax. As a result one needs to decide which semantics is required in an application and then follow it during the implementation.

From the practical point of view, usually the terminological part of an ontology is rarely modified in contrast to the assertional part. In that case we can separate the TBox from the ABox. As a result we are able to apply different reasoning schemes and engines. Moreover, we need to perform the TBox reasoning only once (or every time it changes) using e.g. some DL reasoner and then we can perform the ABox reasoning using e.g. a rule-based engine each time when new individual assertions occur. Therefore, we can follow the OWL 2 RDF-Based Semantics in both reasoning engines. Such an approach is presented in [9], where the Pellet[9] engine is used with a rule-based system of Jena[10].

It is worth noticing the presented work is devoted to the development of the Rule-based Query Answering and Reasoning system (RuQAR). However, we present only the reasoning features while the query answering part remain to be carried out in the next release. Thus, the main idea of our work is to provide not only an OWL 2 RL reasoning framework but also a scalable query answering tool in which data is stored in a relational database.

3 OWL 2 RL Ontology Transformation

Application of a rule-based reasoning engine to an ontology-based reasoning requires a transformation method of an ontology into a set of rules. Since we mainly focus on the OWL 2 RL Profile, we split the reasoning process into two sub-processes: the TBox reasoning and the ABox reasoning. According to this we developed a methodology of transforming an OWL 2 ontology into a set of rules and a set of facts. In that case we can execute the TBox reasoning and the ABox reasoning separately. Moreover, as we want to perform a rule-based reasoning with different engines we propose Abstract Syntax of Rules and Facts (ASRF), thus enabling easy translation of an OWL 2 ontology into the language of a reasoning engine.

[9] `http://clarkparsia.com/pellet/`
[10] `http://jena.apache.org/`

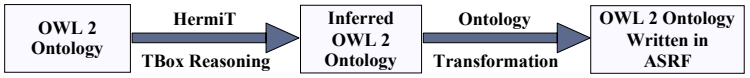

Fig. 1. OWL 2 ontology transformation schema

The transformation schema of an OWL 2 ontology into a set of rules and a set of facts expressed in ASRF is presented in Figure 1. Firstly, an OWL 2 ontology is loaded into the HermiT engine. We assume, that the ontology is consistent. Then, the TBox reasoning is executed. As a result we obtain a new classified version of the ontology (new TBox). Finally, the ontology is transformed into two sets: a set of rules and a set of facts (if it contains the assertional part). Both are expressed in the ASRF notation. In that way we separate the TBox part (set of rules) from the ABox part (set of facts). Thus, we can perform ABox reasoning with a forward chaining rule engine.

The transformation of an inferred TBox into a set of rules is performed in accordance with the OWL 2 RL/RDF rules. Generated rules contain two kinds of rules: equality rules and ontology-dependant rules (so-called the ABox rules). Equality rules are directly taken from Table 4 of the OWL 2 RL Profile. These rules are expressed in ASRF and they axiomatize the semantics of equality. They are ontology-independent in contrast to the ABox rules.

The transformation of an ontology, after classification performed by HermiT, into a set of ASRF rules is executed in the following way. For each supported rule (see Table 1 for the details) and the corresponding OWL 2 RL axiom we create a rule reflecting the expression in a given ontology. In other words, it means that rather than transforming the semantics of the OWL 2 RL language into rules we create rules according to this semantics and a given ontology. For instance, if we have an ObjectProperty *hasSibling* which is defined as a SymmetricObjectProperty we create a rule which reflects that when an instance of this property occurs, a symmetric instance should also occur (the following shortcuts are made: S for Subject, P for Predicate and O for Object):

$$\begin{array}{ll} If & (Triple\ \ (S\ \ ?x)\ (P\ \ \text{“}hasSibling\text{”})\ (O\ \ ?y)) \\ Then & (Triple\ \ (S\ \ ?y)\ (P\ \ \text{“}hasSibling\text{”})\ (O\ \ ?x)) \end{array} \tag{1}$$

Rule (1) follows the semantics of *prp-symp* rule from Table 5 in the OWL 2 RL Profile specification. For each OWL 2 RL axiom which occurs in a given ontology we generate semantically equivalent rule containing a direct reference to the ontology. The generated rule is an instantiated version of the corresponding OWL 2 RL/RDF rule for a particular TBox. Generated rules can be perceived as ontology instance related rules (instantiated rules or ABox rules). These rules are ontology-dependant because they express the semantics of a given ontology and are intended for reasoning with the facts. As a result we provide a set of rules which can be directly applied in a forward chaining engine after the translation from ASRF notation to the engine's language. Such an approach provides an execution of reasoning task directly with the assertional part. It has a positive influence on reasoning efficiency since the semantics of the TBox part

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

is directly represented by the generated ASRF rules. Moreover, an additional positive impact comes from the fact that the number of conditions in the body of each rule is smaller than in the corresponding OWL 2 RL/RDF rule.

Table 1 shows currently supported rules by our implementation. This set comes from the specification of OWL 2 [10]. 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" rule (e.g. *cls-nothing2* from Table 6 in the OWL 2 RL Profile) and each rule which does not have an impact on the ABox reasoning (e.g. all rules from Table 9 in the OWL 2 RL Profile). However, some rules remain to be implemented, e.g. *cls-maxqc3* from Table 6.

Presented 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 the 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 the OWL 2 RL Profile will not produce the same entailments as derived by an appropriate DL-based reasoner. In this case, the reasoning with rules generated by our methodology is sound but not complete. We observed such an issue in our evaluation with the LUBM benchmark where all results produced by our method were within entailments derived by HermiT. However, HermiT produced more results which is correct since the expressivity of LUBM is beyond OWL 2 RL.

4 Abstract Syntax of Rules and Facts

As we mentioned in previous section the TBox reasoning is performed with the HermiT engine. Then, an ontology is automatically transformed in Abstract

Syntax of Rules and Facts. The main purpose of developing such a syntax is to rise an abstraction level providing more universal representation of rules and facts (assertional part of a knowledge base). As a result the application of ASRF expressions requires mapping schema between the ASRF notion and the language of a selected reasoning engine.

We applied the Extended Backus-Naur Form (EBNF) [13] notation as a technique to express our Abstract Syntax of Rules and Facts. This context-free grammar is presented in Figure 2. Non-terminal symbols are inside brackets (< and >) while other symbols are the terminal ones.

The ASRF syntax is a first-order logic-based notation. Each fact is an atom which consists of a set of terms. Each term is a variable (preceded by '?') or a constant. Furthermore, each term is one of the following types: *Subject*, *Predicate*, *Object* or *Argument*. Similarly, each atom is one of the following types: *Triple* or *Comparison* ($\leq, \neq$, etc.). Each rule consists of the body B (IF part) and the head H (THEN part) of a rule ($B \rightarrow H$). Both elements contain atoms. Variables are universally quantified. Moreover, we can use additional operators like 'or' statement to express disjunction (only in the body of a rule) which is in accordance with the OWL 2 RL Profile. Both the body and the head can contain constants and/or variables in their atoms. In contrast, it is not allowed in the facts representation. By allowing to use comparisons we support SWRL Built-ins that can be employed in order to compare values.

The default meaning of the head of each rule is to assert (infer) new triple (fact). In order for a rule to be applied, all the conditional elements that occur in the body must hold. For instance, rule (1) follows the ASRF syntax as well as example facts (2) and (3). Fact (3) is inferred by applying rule (1) to fact (2).

$$(Triple\ (S\ \ \text{“}Person1\text{”})(P\ \ \text{“}hasSibling\text{”})(O\ \ \text{“}Person2\text{”})) \tag{2}$$

$$(Triple\ (S\ \ \text{“}Person2\text{”})(P\ \ \text{“}hasSibling\text{”})(O\ \ \text{“}Person1\text{”})) \tag{3}$$

Our ASRF syntax is similar to the syntaxes of well-known rule languages like Jess or Clips[11]. However, it is less powerful and is limited to expressions available in the OWL 2 RL Profile. For instance, we can not infer about inconsistencies in a knowledge base.

5 Implementation and Experiments

RuQAR implements our method of transforming OWL 2 ontologies into a set of rules and a set of facts expressed in the ASRF syntax. The tool is developed in Java and allows to perform ABox reasoning with two state-of-the-art rule engines: Jess and Drools. RuQAR is implemented as a library which can be included in applications requiring efficient ABox reasoning. RuQAR uses the OWL API [6] to handle ontology files as well as to extract the logical axioms from the ontology. We use Drools in version 5.5 and Jess in version 7.1.

[11] `http://clipsrules.sourceforge.net/`

```
<Rule>            ::=  If <ConditionalAtom>+
                       Then <Atom>+

<ConditionalAtom> ::=  <Atom> |
                       <Logic-operator> <ConditionalAtom> |
                       <Comparison>

<Atom>            ::=  ( Triple ( Subject <Argument> )
                                ( Predicate <Constant> )
                                ( Object <Argument> ) ) |
                       ( Triple <Term>+ )

<Logic-operator>  ::=  AND | OR | NOT

<Term>            ::=  ( <TermType> <Argument> )

<TermType>        ::=  Subject | Predicate | Object

<Comparison>      ::=  ( <Argument> <Comparator> <Argument> )

<Argument>        ::=  <Constant> | <Variable>

<Comparator>      ::=  equal | not equal | greater than |
                       greater than or equal | less than |
                       less than or equal | different from

<Fact>            ::=  ( Triple ( Subject <Constant> )
                                ( Predicate <Constant> )
                                ( Object <Constant> ) )

<Constant>        ::=  A finite sequence of characters.

<Variable>        ::=  A finite sequence of characters
                       without white spaces preceded by '?' sign.
```

Fig. 2. Abstract Syntax of Rules and Facts in EBNF

We evaluated RuQAR using test ontologies taken from the KAON2 website[12]: Vicodi[13] - an ontology about European history, Semintec[14] - an ontology about financial domain and LUBM[15] - an ontology benchmark about organizational structures of universities. We used different datasets of each ontology (Semintec_0, Semintec_1, etc.) where the higher number means bigger ABox set.

Evaluation schema for each ontology was the following. Firstly, we performed the TBox reasoning using HermiT. Then, the classified ontology was loaded into an engine and the ABox reasoning was executed. In each case we recorded the reasoning time and counted the resulting ABox size. We performed the ABox reasoning with the following engines: Jess, Drools and HermiT. We verified that the reasoners produced identical results (a similar empirical approach is applied in [3] and [11] in order to compare their OWL 2 RL reasoners with Pellet/RacerPro

[12] http://kaon2.semanticweb.org/
[13] http://www.vicodi.org
[14] http://www.cs.put.poznan.pl/alawrynowicz/semintec.htm
[15] http://swat.cse.lehigh.edu/downloads/index.html

and HermiT, respectively). However, HermiT provided more reasoning results in the LUBM case. It is correct, since only Vicodi is within the OWL 2 RL Profile. However, this is the main cause of extremely large differences of reasoning times in comparison to Jess and Drools. Nevertheless, all results inferred by Jess and Drools were among the results produced by HermiT. In each case we obtained better performance in ABox reasoning with Jess/Drools than with HermiT. For instance, for the Semintec_4.owl ontology, appropriate times for Jess, Drools and HermiT were the following (results were identical): over 3 seconds, over 5 seconds and over 16 seconds, respectively. As we can see from Figure 3 Jess performed better than Drools while HermiT was always on the third place. Obtained results confirm that our method increases the ABox reasoning in comparison to the DL-based reasoner.

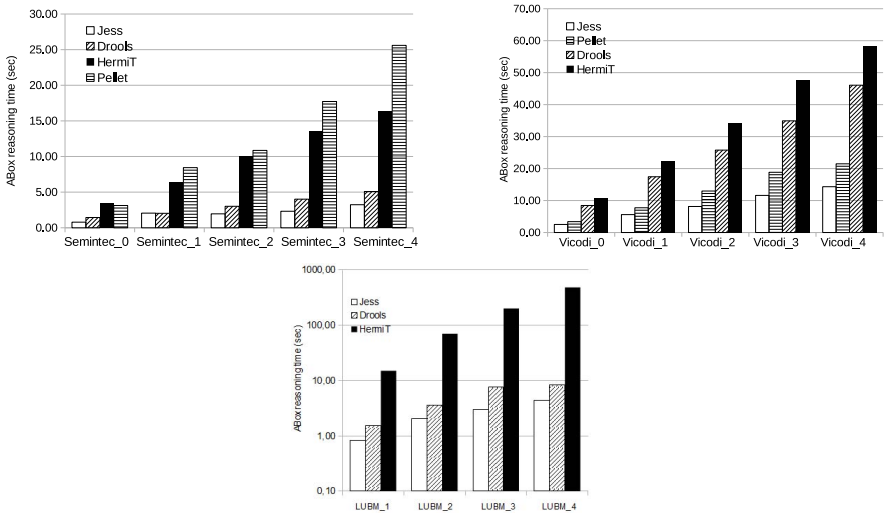

Fig. 3. The ABox reasoning times of the tested ontologies

6 Related Work

The most closely related work is an approach applied in DLEJena [9]. However, we do not restrict ourselves to one reasoning tool (Jena versus Jess and Drools). Furthermore, we apply slightly different transformation method - we do not use template rules to produce instantiated rules but Java-coded generation. Such an approach do not produce redundant instantiated rules as in [9]. Moreover, in this approach the entailment rules are generated at runtime while in our methodology ABox rules are generated before the reasoning process.

A pair of OWL 2 RL reasoners is presented in [11] where Drools and Jess are used to infer with rules directly representing the semantics of the OWL 2 RL Profile. Both aforementioned approaches follow the Direct Semantics which is the same semantics provided by RuQAR.

Another approach [5] provides OWL 2 RL reasoning and it is based on partial-indexing for optimising scalable rule-based materialisation using the set of template rules similar to DLEJena.

Scalable OWL 2 RL reasoner was presented in [8] where the inference engine is implemented inside the Oracle database system. This work introduces novel techniques for parallel processing as well as special optimisations of computing *owl:sameAs* relationships.

In [3] a method for storing asserted and inferred knowledge in a relational database is presented. Moreover, they also propose a novel database-driven forward chaining method which allows to perform scalable reasoning over OWL 2 RL ontologies with large ABoxes.

7 Conclusions and Future Work

In this paper we presented a transformation method of an OWL 2 ontology into one set of rules and one set of facts. We proposed Abstract Syntax of Rules and Facts in which both sets are expressed. Moreover, we described the reasoning schema, our implementation as well as performed experiments.

The current version of RuQAR is able to perform the ABox reasoning with considerably better performance than HermiT. Nevertheless, RuQAR is not an OWL 2 RL conformant[16] implementation since it cannot handle arbitrary RDF graph. However, presented approach results in better reasoning performance, since some inferences are omitted (unsupported OWL 2 RL/RDF rules).

In the next release we are planning to handle relational database as well as optimized query processing (currently we can only use query methods available in forward chaining engines: Jess and Drools). Moreover, we plan to optimize reasoning process by applying and extending methods described in [3] and in [1]. Due to the ASRF syntax, applied optimizations will be usable in Jess and Drools, and in other forward chaining rule engines.

To the best of our knowledge presented work is the first implementation of the OWL 2 RL reasoning in Drools and Jess (except the work presented in [11] that implements directly the semantics of OWL 2 RL) which can be applied in any application requiring efficient ABox reasoning.

We also plan to perform tests with the latest versions of Jess and Drools, 8.0 and 6.0, respectively. It will be useful to check if the reasoning efficiency is increased in the newer versions. As a result in Drools we will be able to compare two different algorithms: ReteOO (Drools 5.5) and PHREAK (Drools 6.0).

Acknowledgments. This work has been funded by the Polish National Science Centre (decision no. DEC-2011/03/N/ST6/01602) and DS 04/45/DS-PB/0105 grant.

[16] `http://www.w3.org/TR/owl2-conformance/`

References

1. Bak, J., Brzykcy, G., Jedrzejek, C.: Extended rules in knowledge-based data access. In: Palmirani, M. (ed.) RuleML 2011. LNCS, vol. 7018, pp. 112–127. Springer, Heidelberg (2011)
2. Boris, M.: On the properties of metamodeling in owl. Journal of Logic and Computation 17(4), 617 (2007)
3. Faruqui, R.U., MacCaull, W.: OwlOntDB: A scalable reasoning system for OWL 2 RL ontologies with large ABoxes. In: Weber, J., Perseil, I. (eds.) FHIES 2012. LNCS, vol. 7789, pp. 105–123. Springer, Heidelberg (2013)
4. Hill, E.F.: Jess in Action: Java Rule-Based Systems. Manning Publications Co., Greenwich (2003)
5. Hogan, A., Pan, J.Z., Polleres, A., Decker, S.: SAOR: Template rule optimisations for distributed reasoning over 1 billion linked data triples. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 337–353. Springer, Heidelberg (2010)
6. Matthew Horridge and Sean Bechhofer. The owl api: A java api for working with owl 2 ontologies. In: OWLED (2009)
7. Horrocks, I., Patel-Schneider, P.F.: Knowledge representation and reasoning on the semantic web: Owl. pp. 365–398 (2010)
8. Kolovski, V., Wu, Z., Eadon, G.: Optimizing enterprise-scale OWL 2 RL reasoning in a relational database system. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 436–452. Springer, Heidelberg (2010)
9. Meditskos, G., Bassiliades, N.: Dlejena: A practical forward-chaining owl 2 rl reasoner combining jena and pellet. J. Web Sem. 8(1), 89–94 (2010)
10. Motik, B., Grau, B.C., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: Owl 2 web ontology language profiles. In: W3C Recommendation, 2nd edn. (2012)
11. O'Connor, M.J., Das, A.: A pair of owl 2 rl reasoners. In: Klinov, P., Horridge, M. (eds.) OWLED. CEUR Workshop Proceedings, vol. 849. CEUR-WS.org (2012)
12. Patel-Schneider, P.F., Horrocks, I.: Owl 1.1 web ontology language (2006), `http://www.w3.org/Submission/owl11-overview/`
13. Pattis, R.E.: Ebnf: A notation to describe syntax (1980)
14. Polleres, A., Hogan, A., Delbru, R., Umbrich, J.: RDFS and OWL reasoning for linked data. In: Rudolph, S., Gottlob, G., Horrocks, I., van Harmelen, F. (eds.) Reasoning Weg 2013. LNCS, vol. 8067, pp. 91–149. Springer, Heidelberg (2013)
15. Volz, R.: Web ontology reasoning with logic databases. PhD thesis (2004)