A prototypical OWL Full Reasoner based on First-Order Reasoning

Michael Schneider

FZI Research Center for Information Technology, Karlsruhe, Germany
schneid@fzi.de

Abstract. We report on our ongoing endeavour to create a prototypical reasoning system for the OWL 2 Full ontology language and several of its sublanguages, including RDFS and the OWL 2 RL/RDF Rules. Among the languages specified by the W3C OWL 2 standard, OWL 2 Full is the most expressive one, and several other languages, such as RIF and SPARQL 1.1, have dependencies on OWL 2 Full, but to date no reasoner has been implemented for this language. The basic idea underlying our system is to translate the semantics specification of OWL 2 Full and the RDF graphs representing the input ontologies into first-order logic formulae, and to apply first-order reasoners to this axiomatisation to perform one of the supported reasoning tasks: ontology consistency checking, entailment checking, or query answering. The paper explains the taken approach, summarizes the results of a recent evaluation of this approach, and gives an overview of the functionality, architecture and implementation of the reasoner.

Keywords: Semantic Web, Reasoning, OWL Full, FOL, ATP

1 OWL Full Reasoning based on First-Order Reasoning

This section provides the technical foundations and the approach underlying the reasoner, and reports on prior art and evaluation results for the approach.

OWL 2 Full. The ontology language OWL 2 Full has been specified by the World Wide Web Consortium (W3C) in 2009 as a semantic extension of RDFS [4] that provides full coverage of the language features of OWL 2 [10]. The syntax of OWL 2 Full is the *RDF Abstract Syntax* [5], i.e., every RDF graph is a valid OWL 2 Full ontology. The semantics of OWL 2 Full is the *OWL 2 RDF-Based Semantics* [6], which incorporates the semantics of RDFS and is specified like RDFS as a set of model-theoretic *"semantic conditions"*. The semantic conditions are given in the style of first-order logic (FOL) formulae and provide formal meaning for certain combinations of RDF triples that represent OWL 2 language constructs. The expressivity of the semantics of OWL 2 Full is roughly comparable with that of OWL 2 DL [10], but it also applies to ontologies beyond the syntactic restrictions that OWL 2 DL defines to retain computational decidability; in fact, this extended flexibility renders OWL 2 Full undecidable. However, due to design decisions originally made for RDF, certain kinds of OWL 2 DL entailments cannot directly be obtained in OWL 2 Full, but can still be received by means of a "mostly harmless" syntactic transformation of the input RDF graphs, which is called *"balancing"* and is described in Chapter 7 of [6].

Translation into First-Order Logic. Since the semantic conditions of OWL 2 Full have the form of FOL formulae, its semantics can directly be axiomatisized as a FOL theory. Further, every RDF graph can be written as a FOL axiom consisting of a conjunction of ternary atomic formulae, with existentially quantified variables for the blank nodes. The details of the translation can be found in [7]. As a concrete FOL syntax, we use the *TPTP language* [8], which is understood by the majority of existing FOL reasoners.

First-Order Reasoning. OWL 2 Full reasoning can be performed by feeding the TPTP formulae for the OWL 2 Full semantic conditions and the input RDF graphs into one or more FOL reasoning systems. *FOL theorem provers* will check whether an RDF graph is semantically inconsistent or whether an entailment holds between two RDF graphs. *FOL model finders* will check whether an RDF graph is consistent or whether an entailment does not hold. When used in parallel, the result is the first solution that one of the FOL reasoners provides, or "unknown", if none of the reasoners is able to find a solution within a given time limit. Some FOL theorem provers also support incremental *query answering* on FOL axiomsets and can be used to realize OWL 2 Full query answering.

Prior Art. An early application of our approach to a precursor of OWL has been reported by Fikes et al. [1], but the focus of this work was more on identifying technical problems in the language specifications rather than on practical reasoning. Hayes [3] provides a complete translation of OWL 1 Full into Common Logic, but does not report on any reasoning experiments. Hawke's reasoner *Surnia* [2] applies a theorem prover to an FOL axiomatisation of OWL 1 Full, but the implementation does not strictly follow the official semantic conditions. Comparable work has also been carried out for OWL DL reasoning, for example in [9]. To our best knowledge, the approach has not been applied to OWL 2 Full so far and no comprehensive evaluation of the approach has ever been carried out for RDF-based formalisms in general.

Evaluation Results. An experimental investigation of our approach has recently been reported in [7]. In summary, detection of inconsistency and entailment worked for basically the whole OWL 2 Full specification (datatype reasoning excluded). Even conclusions that are very "characteristic" for OWL 2 Full were always inferable, such as entailments from metamodeling or the reflective use of logical built-in vocabulary. For comparison, several state-of-the-art OWL reasoners were applied to the same test data and failed to a large extent. FOL reasoning was still possible after extending the test data by one Million generated RDF triples of semantically weak "bulk" data. Detecting non-entailments and consistent RDF graphs was, however, unsuccessful for OWL 2 Full, but was still successful for some language fragments, such as RDFS. A core issue was that reasoning with the complete OWL 2 Full axiomatisation is often difficult for existing FOL reasoners, but it was found that by reducing the axiom set to only axioms that are relevant for the result can greatly improve performance.

2 The OWL Full Reasoner

This section provides a concise overview of the functionality, architecture and implementation of the prototypical OWL Full reasoner that is currently developed following the investigation in [7].

Functionality. The reasoner supports the languages *OWL 2 Full*, *RDFS* and the *OWL 2 RL/RDF Rules* [10]. Custom extensions to the supported languages, such as additional support for rules or Boolean property expressions, are easily possible by extending the corresponding FOL axiomatisations. Supported reasoning tasks are *ontology consistency checking* and *entailment checking*, both regarding positive and negative results, and incremental *query answering* based on SPARQL-style basic graph patterns. The reasoner can be accessed either programmatically by its *native API*, or via a *command line tool*, or as a simple *web service* using a web browser. We further plan implementing the reasoning interfaces of the RDF frameworks *Jena* (`jena.sourceforge.net`) and *Sesame* (`openrdf.org`). The reasoner operates either in *"raw mode"* to precisely conform to the semantics specification of the respective language, or in *"balanced mode"* with balancing being applied to the input RDF graphs to better cope with practical reasoning scenarios (see the discussion on balancing in Sec. 1). All FOL reasoners that understand the *TPTP language* can be used as reasoning backends. All applied FOL reasoners are executed in *parallel*, which allows, for example, to use specialized reasoners for theorem proving and model-finding to conjointly decide a consistency or entailment checking problem. In the future, there will be support for building *hybrid systems* with other Semantic Web reasoners, which will, for example, allow to combine the performance and scalability strengths of "light-weight" RDF entailment-rule reasoners with the sophisticated logic-based reasoning capabilities of FOL reasoners.

Architecture and Reasoning Process. The reasoner consists of a collection of orchestrated components. The *TPTP Parser* is used to deserialize the axiomatisation of the OWL 2 Full semantic conditions into a *TPTP Model* object. The *RDF Parser* is used to deserialize the input RDF graphs into *RDF Model* objects. If the "balanced mode" is enabled, balancing of the RDF input is performed by the *Balancing Component*. The *Relevant Semantic Conditions Selector* and the *Relevant Input Triples Selector* are applied to the OWL 2 Full axiomatisation and the RDF input to remove some of the axioms and RDF triples that are not relevant for the computation of the result. These two components work according to heuristic approaches, which will be improved over time. The input RDF models are then converted into TPTP models using the *RDF-to-TPTP Converter*. All TPTP models are now given to the *FOL Reasoner Execution Unit*, which is responsible for performing reasoning. The execution unit combines all the TPTP models into a single TPTP model and applies all registered FOL reasoners to it in parallel. FOL reasoners are accessed via their implementation of the *FOL Reasoner Interface*, which has methods for inconsistency and consistency detection, entailment and non-entailment detection, and incremental query answering; a concrete FOL reasoner may only support part of this functionality. The execution unit will return a single result that is based on the outputs of all FOL reasoners. From this result, the *Result Renderer* produces the final outcome, which is either an error message, or one of "true", "false" or "unknown" in the case of consistency or entailment checking, or an incrementally created sequence of answers in the case of query answering.

Implementation Details. The reasoning system is written in the *Java 6* programming language (`java.com`). Internally, we make use of the *TPTP parser library* (`www.freewebs.com/andrei_ch/TPTP_2007.01.30.tgz`) to handle FOL formulae and the *Jena RDF framework* is used for handling RDF graphs. Different FOL reasoners can be used as backend reasoning engines. We have successfully experimented with *Vampire 0.6* (`vprover.org`) for theorem proving, *Paradox 4.0* (`www.cse.chalmers.se/~koen/code/`) for model-finding, and *E 1.3* (`eprover.org`) for query answering. FOL reasoners are accessed as external processes. A facility to remotely access FOL reasoners is planned for the future.

Acknowledgements. This work has been supported by the projects *SEALS* (European Commission, EU-IST-2009-238975) and *THESEUS* (German Federal Ministry of Economics and Technology, FK OIMQ07019).

References

1. Fikes, R., McGuinness, D., Waldinger, R.: A First-Order Logic Semantics for Semantic Web Markup Languages. Tech. Rep. KSL-02-01, Knowledge Systems Laboratory, Stanford University (January 2002)
2. Hawke, S.: Surnia. Homepage (2003), `http://www.w3.org/2003/08/surnia`
3. Hayes, P.: Translating Semantic Web Languages into Common Logic. Tech. rep., IHMC (2005), access: `http://www.ihmc.us/users/phayes/CL/SW2SCL.html`
4. Hayes, P. (ed.): RDF Semantics. W3C Recommendation (10 February 2004), access: `http://www.w3.org/TR/rdf-mt/`
5. Klyne, G., Carroll, J.J. (eds.): Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation (10 February 2004), access: `http://www.w3.org/TR/rdf-concepts/`
6. Schneider, M. (ed.): OWL 2 Web Ontology Language: RDF-Based Semantics. W3C Recommendation (27 October 2009), access: `http://www.w3.org/TR/owl2-rdf-based-semantics/`
7. Schneider, M., Sutcliffe, G.: Reasoning in the OWL 2 Full Ontology Language using First-Order Automated Theorem Proving. In: Proc. CADE 23. LNAI, vol. 6803, pp. 446–460 (2011), extended version at `http://arxiv.org/abs/1108.0155`
8. Sutcliffe, G.: The TPTP Problem Library and Associated Infrastructure. The FOF and CNF Parts, v3.5.0. J. Automated Reasoning 43(4), 337–362 (2009), current version at `http://tptp.org/TPTP/TR/TPTPTR.shtml`
9. Tsarkov, D., Riazanov, A., Bechhofer, S., Horrocks, I.: Using Vampire to Reason with OWL. In: Proc. ISWC 2004. LNCS, vol. 3298, pp. 471–485 (2004)
10. W3C OWL Working Group (ed.): OWL 2 Web Ontology Language: Document Overview. W3C Recommendation (27 October 2009), access: `http://www.w3.org/TR/owl2-overview/`