ECAI 2008
M. Ghallab et al. (Eds.)
IOS Press, 2008

doi:10.3233/978-1-58603-891-5-731

Rule-based OWL Ontology Reasoning Using Dynamic ABOX Entailments

Georgios Meditskos and Nick Bassiliades[1]

Abstract. In the rule-based OWL reasoning paradigm, ontologies are mapped into an internal rule engine representation format and rules are applied, such as TBOX and ABOX OWL entailment rules, in order to deduce new knowledge. In this paper we briefly introduce the notion of dynamically generating ABOX entailment rules in order to enhance the ABOX reasoning performance of a rule engine. The proposed methodology is still based on entailments rules for reasoning, using generic TBOX entailments for handling OWL semantics about concepts and roles, and dynamic ABox entailments for handling ontology instances.

1 INTRODUCTION

OWL [16] is the W3C recommendation for creating and sharing ontologies on the Web. It provides the means for ontology definition and specifies formal semantics on how to derive new information. Several approaches have been followed for the development of reasoning engines able to handle OWL semantics, such as *Description Logic* algorithms [2], *theorem provers* [15] or *rule-engines* [8][12]. Each approach has advantages and disadvantages and the selection of the appropriate one is based on the domain or users' requirements [5][13].

In this work, we are focused on the rule-based OWL reasoning paradigm based on entailments and we describe a methodology that improves the time a rule engine needs in order to apply the OWL semantics over ontology individuals (ABOX). This is feasible by exploiting the schema information of OWL ontologies in order to generate *domain-dependent* ABOX rules.

2 BACKGROUND AND MOTIVATION

In the rule-based OWL reasoning paradigm, the *asserted* knowledge, that is the knowledge stemming directly from the ontology definition, is mapped into an internal rule engine representation format, and inference rules are applied in order to deduce new knowledge. The inference rules are based on OWL *entailments* [7], rules which describe the information that should be inferred based on existing knowledge. To exemplify, let *S* be the set of triples of an ontology, where *S* = {<A subClassOf B>, <B subClassOf C>}. By implementing the *rdfs9* entailment rule for subclass transitivity (Table 1), we get that *S* = {<A subClassOf B>, <B subClassOf C>, <A subClassOf C>} (the entailment rules can be found in [7]).

However, the high expressivity of OWL hampers the definition of a complete set of OWL entailments and rule languages can only handle a subset of OWL, known as *Description Logic Programs* (DLP) [4]. Despite this limitation, the combination of rules and ontologies is one of the hottest areas [1][6][10][14].

We are motivated by the fact that the majority of the ABOX entailment rules are based on generic TBOX information, such as the *rdfp4* entailment for role transitivity, which requires the property *p* to be transitive. Intuitively, the *rdfs9* entailment is a *specialized* form of the *rdfp4* entailment for the subClassOf property. However, the latter is more complicated than the former, requiring a double join in its body among a transitive property *p* and two instance *p* values. Our approach is based on such ABOX entailment *specializations*.

Table 1. Examples of entailment rules.

	if	*then*
rdfs9	c1 subClassOf c2, c2 subClassOf c3	c1 subClassOf c3
rdfp4	p type TransitiveProperty, x p y, y p z	x p z
rdfp1	p type FunctionalProperty, x p y, x p z	y sameAs z
rdfp12a	c1 equivalentClass c2	c1 subClassOf c2
rdfp14a	r hasValue y, r onProperty p, x p y	x type r

The TBOX entailments are either specialized, since they refer to built-in OWL constructs which are known in advance, such as the subclass transitivity (*rdfs9*), or they cannot be specialized before the termination of the TBOX inferencing procedure (*rdfp12a*). In contrast, ABOX entailments can be specialized, apart from some exceptions, provided that the TBOX inferencing is performed first. In that way, a *dynamic inference rule base* is generated, able to apply more efficiently ABOX semantics than a generic rule base, especially in large scale ABOX ontologies.

3 DYNAMIC ENTAILMENT GENERATION

The dynamic entailment methodology is based on the fact that most of the ABox entailments can be grounded into one or more simpler domain-dependent rules. More formally, an ABOX entailment rule is of the form

[1] Department of Informatics, Aristotle University of Thessaloniki, Greece, email: {gmeditsk, nbassili}@csd.auth.gr

$$T \wedge A_E(T) \wedge A_E \rightarrow A_I(T) \wedge A_I,$$

where T is the set of TBOX triple conjunctions, $A_E(T)$ is the set of individual triple conjunctions that use TBOX information, A_E is the set of individual triple conjunctions unrelated to TBOX, $A_I(T)$ is the conjunctive set of the inferred individual triples that use TBOX information, and A_I is the conjunctive set of the inferred individual triples unrelated to TBOX. The dynamic rule generation methodology performs the following rule transformation:

$$T \wedge A_E(T) \wedge A_E \rightarrow A_I(T) \wedge A_I \xrightarrow{t} \forall T : A_E(T) \wedge A_E \rightarrow A_I(T) \wedge A_I,$$

which generates T-dependent rules. To exemplify, consider the *rdfp1* entailment with T = {<p type FunctionalProperty>}, $A_E(T)$ = { <x p y>, <x p z>}, $A_E = \varnothing$, $A_I(T) = \varnothing$ and C = {<y sameAs z>}, which is transformed into:

$\forall p \mid$ <p type FunctionalProperty >:
rule: if <x p y> ∧ <x p z> **then** <y sameAs z>.

Moreover, the *rdfp14a* entailment, with T = {<r hasValue z>, <r onProperty p>}, $A_E(T)$ = { <x p z>}, $A_I(T) = \varnothing$, $A_I(T)$ = {<x type r>} and $C = \varnothing$, is transformed into:

$\forall r \mid$ <r hasValue z> ∧ <r onProperty p>:
rule: if <x p z> **then** <x type r>.

4 EXPERIMENTAL RESULTS

We used the CLIPS [3] production rule engine in order to apply thirteen entailments over the LUBM [11] university ontology. Five extensional datasets D_i were generated, each one of approximately 12,000 triples. Table 2 depicts the time needed to apply the dynamic and the generic rules over different dataset sizes. The dynamic approach generates about 300 rules and, despite the great number of rules, the ABOX reasoning procedure terminates considerably faster than the generic approach, where only 13 rules are applied.

Table 2. Dynamic and Generic ABOX reasoning times.

	Dynamic (sec)	Generic (sec)
12,000	36.750	86.063
24,000	61.078	167.000
36,000	84.797	255.859
48,000	10.7406	393.109
60,000	129.719	512.312

5 RELATED WORK

To the best of our knowledge, the existing rule-based reasoners that use entailments follow the generic methodology, that is both the TBOX and the ABOX entailments are generic and ontology-independent. SweetProlog [9], Jena [12] and OWLIM [8] are some example systems that are based on general purpose rule engines, e.g. Prolog, or on rule engines built from scratch, such as the TRREE engine of OWLIM. Notice that the default Jena rule engine for OWL reasoning is a hybrid implementation, using forward chaining rules in order to generate backward chaining rules.

6 CONCLUSIONS

In this paper we presented a methodology of performing rule-based OWL reasoning based on generic TBOX and on dynamic ABox entailment rules. In that way, we are able to use the TBOX rules as the basis for generating domain-dependent ABOX inferencing rules. The main characteristic of these rules is that they join less conditional elements in their body, achieving better activation times in rule engines, than their corresponding generic entailments.

Currently we are working on combining a rule engine with a DL reasoner in order to dynamically generate ABOX inferencing rules based on the inferencing capabilities of the DL paradigm.

ACKNOWLEDGEMENTS

This work was partially supported by a PENED program (EPAN M.8.3.1, No. 03EΔ73), jointly funded by the European Union and the Greek Government (General Secretariat of Research and Technology/GSRT).

REFERENCES

[1] G. Antoniou, C.V. Damasio, B. Grosof, I. Horrocks, M. Kifer, J. Maluszynski, P.F. Patel-Schneider, Combining Rules and Ontologies. A Survey, *Reasoning on the Web with Rules and Semantics*, REWERSE Deliverables, 2005.

[2] F. Baader, U. Sattler, An Overview of Tableau Algorithms for Description Logics, *Studia Logica*, vol. 69, pp. 5-40, 2001

[3] CLIPS, http://www.ghg.net/clips

[4] B. Grosof, I. Horrocks, R. Volz, S. Decker, Description logic programs: Combining logic programs with description logics, *WWW* 2003, pp. 48–57. ACM, 2003.

[5] P. Hitzler, J. Angele, B. Motik, R. Studer, Bridging the Paradigm Gap with Rules for OWL. *In Proc. of the W3C Workshop on Rule Languages for Interoperability*, Washington, USA, 2005

[6] I. Horrocks, P.F. Patel-Schneider, A Proposal for an OWL Rules Language, *13th Int. WWW Conf.*, ACM, New York (2004)

[7] H.J. Horst, Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary, *Journal of Web Semantics*, vol. 3, pp. 79-115, 2005

[8] A. Kiryakov, D. Ognyanov, D. Manov, OWLIM - a Pragmatic Semantic Repository for OWL, *Proc. Workshop Scalable Semantic Web Knowledge Base Systems*, USA, 2005

[9] L. Laera, V. Tamma, T.B. Capon, G. Semeraro, SweetProlog: A System to Integrate Ontologies and Rules, Rules and Rule Markup Languages for the Semantic Web, 2004.

[10] A.Y. Levy, M.-C. Rousset, Combining Horn rules and description logics in CARIN, *Artificial Intelligence*, 104(1-2), 165–209 (1998).

[11] Y. Guo, Z. Pan, J. Heflin, LUBM: A Benchmark for OWL Knowledge Base Systems, Journal of Web Semantics, 3(2), pp. 158-182, 2005

[12] B. McBride, Jena, Implementing the RDF Model and Syntax Specification, 2nd International Workshop on the Semantic Web, Hong Kong, China, 2001

[13] B. Motik, I. Horrocks, R. Rosati, U. Sattler, Can OWL and Logic Live Together Happily Ever After?, Proc. 5th ISWC, Athens, USA, 2006

[14] R. Rosati, On the decidability and complexity of integrating ontologies and rules, *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 3(1), pp. 61-73, July 2005.

[15] D. Tsarkov, A. Riazanov, S. Bechhofer, I. Horrocks, Using Vampire to reason with OWL, *International Semantic Web Conference*, pp. 471-485, 2004.

[16] Web Ontology Language - OWL, http://www.w3.org/2004/OWL/