Web Semantics: Science, Services and Agents on the World Wide Web 33 (2015) 30–49

Contents lists available at ScienceDirect

Web Semantics: Science, Services and Agents on the World Wide Web

journal homepage: www.elsevier.com/locate/websem

Optimising resolution-based rewriting algorithms for OWL ontologies☆

Despoina Trivela*, Giorgos Stoilos, Alexandros Chortaras, Giorgos Stamou

School of Electrical and Computer Engineering, National Technical University of Athens, Greece

ARTICLE INFO

Article history:
Received 30 September 2013
Received in revised form
1 December 2014
Accepted 17 February 2015
Available online 26 February 2015

Keywords:
Query answering
Query rewriting
Ontology
Description logics
Resolution

ABSTRACT

An important approach to query answering over OWL ontologies is via *rewriting* the input ontology (and query) into a new set of axioms that are expressed in logics for which scalable query answering algorithms exist. This approach has been studied for many important fragments of OWL like $\mathcal{SHIQ}$, Horn-$\mathcal{SHIQ}$, OWL 2 QL, and OWL 2 EL. An important family of rewriting algorithms is the family of *resolution-based* algorithms, mostly because of their ability to adapt to any ontology language (such algorithms have been proposed for all aforementioned logics) and the long years of research in resolution theorem-proving. However, this generality comes with performance prices and many approaches that implement algorithms that are tailor-made to a specific language are more efficient than the (usually) general-purposed resolution-based ones.

In the current paper we revisit and refine the resolution approaches in order to design efficient rewriting algorithms for many important fragments of OWL. First, we present an algorithm for the language DL-Lite$_{\mathcal{R},\sqcap}$ which is strongly related to OWL 2 QL. Our calculus is optimised in such a way that it avoids performing many unnecessary inferences, one of the main problems of typical resolution algorithms. Subsequently, we extend the algorithm to the language $\mathcal{ELHI}$ which is strongly related to OWL 2 EL. This is a difficult task as $\mathcal{ELHI}$ is a relatively expressive language, however, we show that the calculus for DL-Lite$_{\mathcal{R},\sqcap}$ requires small extensions. Finally, we have implemented all algorithms and have conducted an extensive experimental evaluation using many well-known large and complex OWL ontologies. On the one hand, this is the first evaluation of rewriting algorithms of this magnitude, while, on the other hand, our results show that our system is in many cases several orders of magnitude faster than the existing systems even though it uses an additional backwards subsumption checking step.

1. Introduction

Efficient management and querying of large amounts of (possibly distributed) data that are formally described using complex structures like ontologies is an important problem for many modern applications [1–3]. In such settings answers to user queries reflect both the stored data as well as the axioms that have been encoded in the ontology. However, query answering over OWL ontologies is a very challenging task mainly due to its very high computational complexity [4–6]. Even after intense implementation work and the design of modern sophisticated optimisations, direct (tableaux-based) approaches integrated in systems such as HermiT [7], Pellet [8], and Racer [9] are not yet able to cope with very large datasets. Moreover, in several important profiles of OWL 2 [10], like OWL 2 QL and OWL 2 EL, different methods for query answering have been investigated.

A prominent (indirect) approach to query answering over OWL ontologies is via rewriting the input into axioms expressed in formalisms for which efficient data management and retrieval systems are already available. More precisely, the input ontology $\mathcal{O}$ and query $\mathcal{Q}$ are transformed into a set of sentences $\mathcal{R}$, typically a datalog program (or in some cases even a union of conjunctive queries) called *rewriting*, such that for any dataset $\mathcal{D}$ the answers to $\mathcal{Q}$ w.r.t. $\mathcal{D}$ and $\mathcal{O}$ coincide with the answers to $\mathcal{Q}$ w.r.t. $\mathcal{D}$ and $\mathcal{R}$ discarding $\mathcal{O}$ [11–13]. Since $\mathcal{R}$ is a (disjunctive) datalog program query answering can be delegated to existing scalable (deductive) database systems.

Computing rewritings has been studied for various fragments of OWL. One of the first approaches supported the language

☆ This is a revised and extended version of the work presented in Chortaras (2011), and Trivela (2013).

* Corresponding author.
E-mail addresses: despoina@image.ntua.gr (D. Trivela), gstoil@image.ece.ntua.gr (G. Stoilos), achort@cs.ntua.gr (A. Chortaras), gstam@cs.ntua.gr (G. Stamou).

http://dx.doi.org/10.1016/j.websem.2015.02.001
1570-8268/

$\mathcal{SHIQ}$ [11], a large fragment of OWL, and the proposed techniques led to the development of KAON2 [14], one of the first practical systems for answering SPARQL queries over OWL ontologies. Recently, the technique has received considerable attention as it constitutes (perhaps) the standard approach to query answering over ontologies expressed in the languages DL-Lite [12,15], $\mathcal{ELHI}$ [13], and Horn-$\mathcal{SHIQ}$ [16]. DL-Lite and $\mathcal{ELHI}$ are particularly important as they are strongly related to the OWL 2 QL and OWL 2 EL profiles of OWL 2 [10]. Besides the theoretical works many prototype systems have been developed, prominent examples of which include Mastro [17], Presto [15], Quest [18], Rapid [19], Nyaya [20],[1] IQAROS [21], and Ontop [22] which support DL-Lite, Requiem [13], which supports $\mathcal{ELHI}$, and Clipper [16], which supports Horn-$\mathcal{SHIQ}$.

Some approaches for computing rewritings have exploited the resolution-based calculi [23]. In this setting, the input is first transformed into a set of clauses which is then saturated using resolution to derive new clauses. The latter can either contain function symbols or be function-free, while the output rewriting consists of all the derived function-free clauses. Using resolution has at least two benefits. First, such calculi are worst-case optimal and allow for a large number of existing optimisations developed in the field of theorem-proving. Second, since there exist many resolution-based decision procedures for expressive fragments of first-order logic [24,25] it is (relatively) easier to design a resolution-based rewriting algorithm for an expressive fragment of OWL compared to designing a custom made one. For example, to the best of our knowledge, none of the tailor made systems for DL-Lite can currently support more expressive fragments of OWL, while a resolution-based algorithm for all aforementioned fragments exists.

However, the efficiency of resolution-based approaches has also been criticised [26]. Even with all the existing optimisations the saturation produces many clauses unnecessarily. More precisely, it can produce several clauses that contain function symbols and which are not subsequently used to derive other function-free clauses. Since these are neither part of the output rewriting nor do they contribute to the derivation of members of the rewriting their generation is superfluous with respect to query answering. Moreover, exhaustive application of the resolution rule is likely to create long derivations of clauses that are eventually redundant (subsumed) and the standard optimisations of resolution are not enough to provide a scalable approach. Consequently, the first generation systems (e.g., Requiem) have already been surpassed [27].

Motivated by the desire to design efficient rewriting algorithms that can also support expressive fragments of OWL we present novel resolution-based rewriting algorithms. We start from DL-Lite and we show how a rewriting can be computed by greatly restricting the standard (binary) resolution calculus initially used in [28]. Roughly speaking, our calculus generates intermediate clauses that contain function symbols only when it is known that these will contribute to the generation of other function-free clauses. This is implemented by a new resolution inference rule, called *shrinking*, which packages many inference steps into one macro-step and employs certain restrictions over the resolvents.

Subsequently, we extend our approach to the ontology language $\mathcal{ELHI}$ by investigating whether a rewriting algorithm that is again based on the shrinking rule can be defined. This is technically a very challenging task as the structure of $\mathcal{ELHI}$ axioms implies many complex interactions between the clauses (note that, in contrast to DL-Lite, checking concept subsumption in $\mathcal{ELHI}$ is in ExpTime). However, we show that a rewriting can be computed by an algorithm that contains an (arguably small) extension of the shrinking rule of DL-Lite, called *n-shrinking*, plus a new resolution rule, called *function* rule, which captures a very specific type of interaction between roles (binary predicates of the form $R(x, y)$) and their inverses (i.e., $R(y, x)$). Moreover, this new rule is strongly related to the extension of shrinking to *n*-shrinking. More precisely, if the new rule is never applied, then *n*-shrinking reduces precisely to the shrinking rule of DL-Lite. Hence, our algorithm has very good "pay as you go" characteristics. That is, if the ontology is expressed in $\mathcal{ELH}$ (i.e., does not allow for inverse roles), then it is guaranteed that the new rule is never applied and *n*-shrinking can be simplified to shrinking, while the more inverse roles are used in axioms the more the interaction between these two rules, which can create a bottleneck. However, realistic ontologies usually contain few inverse roles, hence we expect that the algorithm would usually behave well in practice. Experimental evaluation and analysis verify our remarks.

Next, we discuss some implementation and optimisation issues which led us to the design and implementation of Rapid, a practical resolution-based system for computing rewritings. More precisely, we discuss how one can present the rewriting in a compact form reducing its size as well as some further optimisation for pruning redundant clauses.

Finally, we conducted an extensive experimental evaluation using a new test suite that includes several real-world large-scale DL-Lite and $\mathcal{ELHI}$ ontologies hence greatly extending all existing benchmarks. Regarding the experiments, our comparison against several state-of-the-art systems has provided many encouraging results. More precisely, our results show that existing systems cannot always handle large-scale and complex ontologies as in several cases they fail to terminate after running for more than 3 h. In contrast Rapid is in the vast majority of cases able to compute a rewriting within a few seconds. Hence, to the best of our knowledge, Rapid is currently the only system that can handle ontologies of this complexity and size. Yet, there are still many difficult cases that no system can handle.

2. Preliminaries

In this section we introduce the ontology languages $\mathcal{ELHI}$ and DL-Lite which are strongly related to OWL 2 EL and OWL 2 QL respectively; we briefly recall some basic notions from first-order logic and resolution theorem-proving; we provide the definition of conjunctive queries and of query rewriting; and we present an overview of the query rewriting algorithm implemented in the Requiem system since our calculi can be seen as a refinement of this algorithm.

2.1. OWL ontologies and description logics

We focus on OWL (2) ontologies interpreted under the direct semantics which are related to Description Logics (DL) [29]. DLs provide the theoretical underpinning for many fragments of OWL and there is a close connection between the functional syntax of OWL and DLs [30,31]. For brevity we will adopt the DL notation and terminology; hence, we will call classes and object properties as atomic concepts and roles, respectively.

Let *CN*, *RN*, and *IN* be countable pairwise disjoint sets of atomic concepts, atomic roles, and individuals. $\mathcal{ELHI}$-*concepts* and $\mathcal{ELHI}$-*roles* are defined using the syntax in the left-hand side of the upper two parts of Table 1, while on the right-hand side the corresponding OWL functional syntax is given. An $\mathcal{ELHI}$-*ontology* $\mathcal{O}$ is a finite set of $\mathcal{ELHI}$-*axioms* of the form depicted in the lower part of Table 1. The Description Logic DL-Lite$_{R,\sqcap}$ (for simplicity DL-Lite in the following) is obtained from $\mathcal{ELHI}$ by disallowing concepts of the form $\exists R.C$ in the left-hand side of axioms. We call

[1] Nyaya actually supports linear Datalog$^{\pm}$.