

Extending the *SHOIQ(D)* Tableaux with DL-safe Rules: First Results

Vladimir Kolovski and Bijan Parsia and Evren Sirin
University of Maryland
College Park, MD 20740
kolovski@cs.umd.edu

Abstract

On the Semantic Web, there has been increasing demand for a rules-like expressivity that goes beyond OWL-DL. Efforts of combining rules languages and description logics usually produce undecidable formalisms [6], unless constrained in a specific way [4, 9, 12]. We look at one of the most expressive - but decidable - such formalisms proposed: DL-safe rules [12], and present a tableau-based algorithm for query answering in OWL-DL augmented with a DL-safe rules component. Our algorithm is an extension of the *SHOIQ* tableau [7] and is proved terminating, sound and complete. We also have a proof-of-concept implementation and preliminary empirical results. Knowing that we pay a high price in performance for this expressivity, we also explore possible optimizations.

1 Introduction

Even though OWL-DL [3] has been a great success as a language for ontology representation on the Semantic Web, there is already demand for even greater expressivity in the form of logic programming (LP) rules. Community interest in this area is growing, as evidenced by the recent (November 2005) creation of the Rule Interchange Format Working Group (RIF [2]). One approach to provide the needed additional expressivity is to couple in some way Description Logic with LP rule systems.

However, combining expressive Description Logics with rules is non-trivial since it can easily lead to undecidability if the actual integration is not restricted in some manner. There have been a number of proposals for combining Description Logics and LP rules while still retaining decidability [12, 4, 9]. The logic of DL-safe rules [12] is one of the most expressive, combining *SHIQ(D)*

and positive datalog rules. DL-safe rules allow classes and properties from the Description Logic component to occur freely in the head or the body of the rule, with the only restriction being that they can be applied only to explicitly named individuals. The same authors who coined the term also proposed the only practical reasoning algorithm for this formalism [11]. The algorithm is based on reducing the description logic knowledge base to a positive disjunctive datalog program, appending the DL-safe rules to this program and using optimized deductive database techniques for query answering.

As an alternative to a reduction to disjunctive datalog, in this paper we present a direct tableaux algorithm for DL-safe rules, as an extension to the *SHOIQ* tableau. Instead of converting the Description Logic knowledge base to a datalog program, we extend a highly optimized OWL-DL tableau reasoner to handle these rules. We believe that we will benefit from layering our algorithm on top of an efficient tableaux reasoner and our initial empirical results support this claim. Also, this approach allows our algorithm to cover the full expressivity of OWL-DL (*SHOIN*(\mathcal{D})) + DL-safe.

In the following section, we provide preliminaries and definitions of terms used later in this paper. Then, after presenting our algorithm, we discuss its performance in the preliminary tests, and the possible optimizations and future work.

2 Background

In this section we provide a brief overview of the terms and notation used later in the paper. We assume that the reader is familiar with the syntax and semantics of *SHOIN*(\mathcal{D}). Before introducing DL-safe rules, we need to describe the standard notation we will be using for rules. A *term* is either a constant (denoted by a, b, c) or a variable (denoted by x, y, z). An *atom* has the form $P(s_1, \dots, s_n)$, where P is a predicate symbol and s_i are terms. A literal is an atom or a negated atom. A *rule* has the form

$$H \leftarrow B_1, \dots, B_n$$

where H and B_i are atoms; H is called the rule *head*, and the set of all B_i is called the rule *body*. A *program* P is a finite set of rules. As for the semantics, we treat rules as logical implications, thus they can also be represented as:

$$H \vee \neg B_1 \vee \neg B_2 \vee \dots \vee \neg B_n$$

Definition 1 (*Semantics of a DL-safe Knowledge Base*)

We start with a *SHOIN*(\mathcal{D}) knowledge base, Σ , as defined above. Let V_P be a set of predicate symbols, such that $V_C \cup V_{IP} \subseteq V_P$. A DL-atom is an atom

of the form $A(s)$ where A belongs to V_C , or of the form $R(s, t)$ where R belongs to V_{IP} . A rule r is called *DL-safe* if each variable in r occurs in a non-DL-atom in the rule body. A program is *DL-safe* iff all of its rules are DL-safe.

Let Σ be a $\mathcal{SHOIN}(\mathbf{D})$ knowledge base and P a set of DL-safe rules. A **DL-safe Knowledge Base** is a pair $K = (\Sigma, P)$. Since $\mathcal{SHOIN}(\mathbf{D})$ is a subset of first order logic, we can give the semantics of K by $\pi(\Sigma) \cup P$ where $\pi(\Sigma)$ is a translation mapping $\mathcal{SHOIN}(\mathbf{D})$ to FOL.

3 Tableaux Algorithm

In this section we present a tableaux-based algorithm for deciding the satisfiability of DL-safe knowledge bases. Our algorithm is inspired by the decidability proof for query answering in combined $\mathcal{SHOIN}(D) +$ DL-safe knowledge bases [12]. The main difference is that we push the DL-safe reasoning inside the tableau in order to get some goal directed behavior. Our approach extends the \mathcal{SHOIQ} tableau [7] thus we are able to handle the full expressivity of the combination of OWL-DL and DL-safe rules. We assume that the reader is familiar with the basics of the \mathcal{SHOIQ} tableau (detailed description is available in [7]).

The main reasoning service that our algorithm provides is query answering. More specifically, answering for a given query α and a DL-safe knowledge base $KB = (\Sigma, P)$ whether $KB \models \alpha$. This can be reduced to an unsatisfiability check of $KB \cup \{\neg\alpha\}$, so $KB \models \alpha$ iff $KB \cup \{\neg\alpha\}$ is unsatisfiable.

It is important to note that DL-safe rules do not change the TBox - they are applicable to individuals only. Thus, to check whether a DL-safe KB $K = (\Sigma, P)$ is satisfiable, we need to perform a consistency check of its ABox with respect to the TBox and the rules. Since ABoxes in general involve multiple individuals with arbitrary role relationships between them, the completion algorithm will work on a forest instead of a tree. Such a forest is a collection of trees whose root nodes correspond to the individuals present in the input Abox. The individuals in the input Abox are the explicitly asserted individuals, and are the *only* individuals to which the rules in P are applicable.

The algorithm works by constructing a model of the Abox w.r.t to the TBox and the DL-Safe rules program, and a \mathcal{SHOIQ} tableaux is a completion graph representing such a model. Each node x in the graph represents an individual, labeled with the set of concepts $\mathcal{L}(x)$ it has to satisfy, i.e, if $C \in \mathcal{L}(x), x \in C^I$. Each edge (x, y) in the graph is labeled with a role name R , and represents a pair occurring in the interpretation of the role, i.e., if $\mathcal{L}(x, y) = R, (x, y) \in R^I$. A formal description is given in the following definition.

Definition 2 (*$\mathcal{SHOIQ} +$ DL-safe Completion Forest*) (extended version of [8])
A completion forest \mathcal{F} for a \mathcal{SHOIQ} Abox \mathcal{A} with respect to a DL-safe knowledge base $K = (\Sigma, P)$ is a collection of trees whose distinguished root nodes may be

connected by edges in an arbitrary way. The completion forest can be described as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{L}, \mathcal{E}, P, \neq)$ where each individual $x \in \mathcal{V}$ can be labelled with a set of concept labels $\mathcal{L}(x)$ and each edge (x, y) can be labelled with a set of role names $\mathcal{E}(x, y)$. P is a program consisting of DL-safe rules R_i , each one of the form

$$H_i \vee \neg B_{i1} \vee \neg B_{i2} \vee \dots \vee \neg B_{in}$$

We also keep track of inequalities between nodes of the graph with a symmetric binary relation \neq between the nodes of \mathcal{G} . Given a *SHOIQ* Abox \mathcal{A} and a DL-safe program P , the algorithm initializes a completion forest \mathcal{F} that contains a root node x_0^i for each individual $a_i \in \mathcal{A}$, and an edge $\langle x_0^i, x_0^j \rangle$ if \mathcal{A} contains an assertion $(a_i, a_j) : R$ for a role R . The labels of these nodes and edges are initialized as follows:

$$\begin{aligned} \mathcal{L}(x_0^i) &= \{C \mid a_i : C \in \mathcal{A}\} \\ \mathcal{L}(\langle x_0^i, x_0^j \rangle) &= \{R \mid (a_i, a_j) : R \in \mathcal{A}\} \\ x_0^i \neq x_0^j &\text{ iff } a_i \neq a_j \in \mathcal{A} \end{aligned}$$

For the purpose of our paper, we assume standard blocking and clash definitions [8].

For a *SHOIQ* Abox \mathcal{A} , the algorithm starts with the completion forest \mathcal{F} . It applies the tableau completion rules and stops when a clash occurs. The Abox \mathcal{A} is consistent w.r.t. to the TBox, RBox and P iff the completion rules can be applied in a way that they yield a complete and clash-free completion forest, and inconsistent otherwise.

Our extension of the algorithm is in the form of a new completion rule, *RULE*-rule (for want of a better name). We reuse all of the original *SHOIQ* completion rules, and add the *RULE*-rule. The *RULE*-rule starts with generating all of the possible ground instantiations for a DL-safe rule (we will call them *bindings*). We need to be careful with the bindings because only named individuals **not** introduced by the tableaux expansion rules are considered. For every such binding, we check whether at least one of the ground literals holds true. If that is not the case, we have found a clash and the KB is inconsistent. Thus, for every literal in the rule we create a tableaux branch, in which we assert that the literal holds (in a manner explained in Figure 1) - this branch is then explored further. This is the most naive version of the algorithm, optimizations are discussed in the next sections.

Please note here that for brevity, we do not include all of the *SHOIQ* tableau completion rules (they are defined in [7]). We only present our new expansion rule which is to be applied with lowest priority.

→ *RULE rule:*
foreach DNF clause $R_i \in P$:
foreach ground substitution to R_i that produces a ground clause G :
foreach monadic ground literal $B(a) \in G$:
if $B \notin \mathcal{L}(a)$, then create a new branch where $\mathcal{L}(a) = \mathcal{L}(a) \cup \{B\}$
foreach dyadic ground literal $B(a, b) \in G$:
if B is of type $\neg A(a, b)$ and $\forall A. \neg\{b\} \notin \mathcal{L}(a)$, then
create a new branch where $\mathcal{L}(a) = \mathcal{L}(a) \cup \{\forall A. \neg\{b\}\}$
if B is of type $A(a, b)$ and $(a, b) \notin \mathcal{G}$ then
create a new branch with $\mathcal{G} = \mathcal{G} \cup \{(a, b)\}$ and $\mathcal{L}(a, b) = \{A\}$
if B is of type $A(a, b)$, $(a, b) \in \mathcal{G}$ and $A \notin \mathcal{L}(a, b)$ then
create a new branch where $\mathcal{L}(a, b) = \mathcal{L}(a, b) \cup \{A\}$

Figure 1: New Expansion Rule

4 Implementation and Evaluation

We implemented ¹ a proof-of-concept system of our algorithm by extending the DL reasoner Pellet [1]. The bottleneck of the algorithm is the non-determinism introduced by creating and exploring tableaux branches for every binding. Thus, as an easy optimization, we check whether, for a particular binding, the rule ground clause is trivially satisfied (one of the disjuncts occurs in the tableaux). If this is the case, we do not generate any new branches for that clause in the tableaux.

We conducted an experimental study to compare the performance of our approach with KAON2 [10]. Our test case of choice was a modified version of the LUBM benchmark [5], with one university and increasing ABox sizes. There are 46 defined classes and 30 object properties in the ontologies. We extended this ontology by adding a rules component consisting of 2 rules. The rules that we added are the following:

GraduateStudent(X):-
Person(X), takesCourse(X, Y), GraduateCourse(Y).

SpecialCourse(Z):-
FullProfessor(X), headOf(X, Y), teacherOf(X, Z).

We ran the experiments on an IBM ThinkPad T42 with Pentium Centrino 1.6GHz processor and 1.5GB of memory. The performance results (time in

¹Code available at: <http://svn.mindswap.org/pellet/branches/dlsafe>

Size of ontology	Pellet DL-Safe		KAON2
	Rule Branches ²	Consistency Check Time	Consistency Check Time
nothing			
LUBM, 1 rule, consistent, 17 individuals	22	110	661
LUBM, 1 rule, consistent, 94 individuals	95	521	621
LUBM, 1 rule, inconsistent, 94 individuals	45	611	651
LUBM, 2 rules, consistent, 17 individuals	39	250	701
LUBM, 2 rules, inconsistent, 94 individuals	153	35,912	691

Table 1: Evaluation Results for Consistency Checking

Size of ontology	Pellet DL-Safe without Pruning		Pellet DL-Safe with Pruning	
	Bindings Generated	Time	Bindings Generated	Time
LUBM, 2 rules,94 inds	804264	35,912	10281	5,048

Table 2: Evaluation Results for Pruning

milliseconds) for the consistency check of the ontologies are shown in table 1. *Rule Branch* stands for the number of additional branches introduced by the *RULE*-rule.

Please note that the first rule has 2 variables, and the second has 3. The additional variable in the second rule impacts the performance noticeably, since there are considerably more bindings possible for that rule. The bindings increase in an exponential manner when we increase the number of variables in the rule. For example, in the case of LUBM with two rules and 94 individuals, there were 804264 bindings to try. This observation leads to another optimization strategy - prune the bindings search space. To accomplish this, we embed the trivial satisfiability check in the generation of the bindings. This reduces the search space significantly, as it can be seen from Table 2.

We present our results compared to KAON2. While it is noticeable that we are lagging behind in terms of performance when the number of rules and ABox size grows, we need to stress that for the smaller cases, we perform better than KAON2. Also, our application covers the full expressivity of the logic underlying OWL-DL, unlike KAON2. We also noticed that KAON2 was performing slower if we added cardinality constraints (for example, a `FullProfessor` would have to teach at least 7 courses to attain a status of `OverachievingProfessor`).

5 Optimizations

We believe, that more advanced, yet practical, optimizations are feasible. For example, we do not need to generate all possible groundings of the rules. We can use a fix-point evaluation procedure as an optimization for our algorithm. By running that procedure first, we make sure that all of the obvious rules are fired - by obvious we mean those rules whose patterns we can match syntactically

against the individuals in the completion forest. After the fixpoint is reached, we can continue with our algorithm. However for a given grounding for a rule, if the rule was fired in the preprocessing phase, now we do not have to try each disjunct of its horn clause separately, thus decreasing the non determinism.

Another optimization would be to sort the rules after the fix point optimization and before running our algorithm. For a given rule r :

$$H \leftarrow B_1, \dots, B_n$$

we would like to try those groundings that satisfy the maximum amount of conditions B_i first. By doing this we will succeed in firing the 'almost obvious' applications of the rules first, and add more information in the concept labels of the tableau. In general, the more information we have in the concept labels, the higher the chances to reach a clash when grounding the horn clauses of the rules, yielding less non-determinism in the algorithm.

6 Related Work

The most relevant related work to ours is the only other practical reasoning algorithm that processes DL-Safe KBs . The algorithm is described in [11] (implemented in [10]) and is based on the $\mathcal{SHIQ}(D)$ logic and having restricted the DL-atoms in rules to concepts and simple roles. The algorithm is based on reducing the description logic knowledge base to a positive disjunctive datalog program which entails the same set of ground facts as the original knowledge base. As shown in Section 4, our algorithm compares reasonably well for small to midsized ontologies and for cases when cardinality constraints are used in the DL.

7 Conclusions and Future Work

This paper makes the following contributions:

- Provides a direct tableau decision procedure for the combination of OWL-DL and DL-safe rules.
- Provides preliminary empirical results of an implementation and discussion of possible optimizations.

As for our next steps, we will develop a more sophisticated algorithm that would handle more expressivity (SWRL) by integrating a First-Order Free Variable tableaux procedure with the \mathcal{SHOIQ} tableaux. DL-safe rules, being universally quantified formulae, will be introduced as free variable formulae in the tableaux. Then, we will look at suitable substitutions in the tableaux that would close as many branches as possible - this can be accomplished with unification.

References

- [1] Pellet - OWL DL Reasoner. <http://www.mindswap.org/2003/pellet>.
- [2] Rule interchange format working group, 2005. <http://www.w3.org/2005/rules/wg>.
- [3] M. Dean, D. Connolly, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. Web Ontology Language (OWL) Reference Version 1.0. W3C Working Draft 12 November 2002 <http://www.w3.org/TR/2002/WD-owl-ref-20021112/>.
- [4] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Al-log: integrating datalog and description logics. *Journal of Intelligent Information Systems*, 10:227–252, 1998.
- [5] J. Heflin, Z. Pan, and Y. Guo. The lehigh university benchmark LUBM. <http://swat.cse.lehigh.edu/projects/lubm/>, 2003.
- [6] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Groszof, and M. Dean. SWRL: A semantic web rule language combining OWL and RuleML, 2004. W3C Submission <http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/>.
- [7] I. Horrocks and U. Sattler. A tableaux decision procedure for shoiq. In *Proc. of IJCAI 2005*, pages 448 – 453.
- [8] I. Horrocks, U. Sattler, and S. Tobies. Reasoning with individuals for the description logic shiq. In D. MacAllester, editor, *Proc. of the CADE 2000*, number 1831, pages 482–496. Springer-Verlag, 2000.
- [9] A. Levy and M.-C. Rousset. CARIN: A representation language combining horn rules and description logics. *Artificial Intelligence*, 104(1-2):165–209, 1998.
- [10] B. Motik. KAON2 - ontology management for the semantic web. <http://kaon2.semanticweb.org/>, 2005.
- [11] B. Motik. *Reasoning in Description Logics using Resolution and Deductive Databases*. PhD thesis, Univesitt Karlsruhe, Germany, January 2006.
- [12] B. Motik, U. Sattler, and R. Studer. Query answering for owl-dl with rules. In *Proc. of ISWC 2004*, pages 549–563.