Combining a DL Reasoner and a Rule Engine for Improving Entailment-Based OWL Reasoning

Georgios Meditskos and Nick Bassiliades

Department of Informatics
Aristotle University of Thessaloniki
{gmeditsk,nbassili}@csd.auth.gr

Abstract. We introduce the notion of the mixed DL and entailment-based (DLE) OWL reasoning, defining a framework inspired from the hybrid and homogeneous paradigms for integration of rules and ontologies. The idea is to combine the TBox inferencing capabilities of the DL algorithms and the scalability of the rule paradigm over large ABoxes. Towards this end, we define a framework that uses a DL reasoner to reason over the TBox of the ontology (hybrid-like) and a rule engine to apply a domain-specific version of ABox-related entailments (homogeneous-like) that are generated by TBox queries to the DL reasoner. The DLE framework enhances the entailment-based OWL reasoning paradigm in two directions. Firstly, it disengages the manipulation of the TBox semantics from any incomplete entailment-based approach, using the efficient DL algorithms. Secondly, it achieves faster application of the ABox-related entailments and efficient memory usage, comparing it to the conventional entailment-based approaches, due to the low complexity and the domain-specific nature of the entailments.

Keywords: Hybrid and Homogeneous Systems, Rule-based OWL Reasoning, Entailment Rules, Rule Engines, DL Reasoning.

1 Introduction

The Web Ontology Language (OWL) [29] is the W3C recommendation for creating and sharing ontologies on the Web and its theoretical background is based on the Description Logic (DL) [2] knowledge representation formalism, a subset of predicate logic. Existing sound and complete DL reasoners [12][44][45] implement tableaux algorithms [3]. However, although these systems perform well on complex TBox reasoning, they have a high ABox reasoning complexity on medium and large complexity TBoxes that constitutes a serious limitation regarding the efficient query answering capabilities needed in domains with large ABoxes [13][15][35].

Rules play an important role in the Semantic Web and, although there is not an unrestricted translation of DLs into the rule paradigm, they can be used in many directions, such as reasoning, querying, non-monotonicity, integrity constraints [4][11] [34]. Regarding rule-based OWL reasoning, the idea is to map OWL on a rule formalism that applies (a subset of) the OWL semantics in the KB of a rule engine. Practical examples of rule-based OWL reasoners are [21][23][27][28][33] that follow the entailment-based

OWL reasoning (EOR) approach [17], and [19] that reduces OWL into disjunctive Datalog [18].

We present our effort to combine the strong points of the DL and EOR paradigms; the former performs efficient TBox reasoning while the latter is characterized by ABox reasoning scalability (comparing it to the DL paradigm), and the simplicity of implementation. We define a mixed DL and rule-based (DLE) framework that embeds the TBox inferencing results of the DL paradigm, and it is able to handle larger ABoxes than the conventional EOR systems. The latter is achieved by substituting the TBox-related entailments with TBox queries to the DL reasoner, generating the ABox-related entailments. The DLE framework is inspired from the hybrid and homogeneous approaches for integration of rules and ontologies [1], using the DL reasoner to answer TBox queries and the rule engine for instance queries.

The rest of the paper is structured as follows. In section 2 we present related background and our motivation for the DLE framework. In section 3 we describe the classification of entailments on which our methodology is based. In section 4 we give the general reasoning principles of the DLE framework. In section 5 we present experimental results about the reasoning activity of DLE-based implementations. Finally, in sections 6 and 7, we present related work and we conclude, respectively.

2 Background and Motivation

The RDF and RDFS semantics can be captured using *entailments* [14] that are rules that denote the RDF triples [10] that should be inferred (rule head) based on existing ones (rule body). A triple has a *subject*, a *predicate* and an *object*, represented as $\langle s\ p\ o\rangle$, where s is an RDF URI reference or a blank node, p is an RDF URI reference (in [17] p is allowed to take blank nodes) and o is an RDF URI reference, a blank node or a literal. Examples of entailments can be found in Table 1.

Horst [16] defines the *pD** semantics as a weakened variant of OWL Full and in [17] they are extended to apply to a larger subset of the OWL vocabulary, using 23 entailments and 2 inconsistency rules. Many practical OWL reasoners are based on the implementation of entailments in a rule engine, such as [21][23][27][28][31][33]. These systems apply a larger set of entailments than the *pD** semantics define. For example, class intersection or the eRDFS [14] entailments (RDF entailment regimes [5]) are not considered in *pD** semantics.

In general, the approaches towards the combination of rules and ontologies are either *hybrid* or *homogeneous* [1]. In homogeneous approaches, the rule and ontology predicates are treated homogeneously, as a new single logic language. Practically, ontologies are mapped on a rule-based formalism that coexists in the KB with rule predicates [32][40][41]. The EOR is a type of a homogeneous approach, since any rule predicate of a rule program coexists with the entailment rules [1].

In hybrid approaches, the rule and ontology predicates are separated and the ontology predicates can be used as constraints in rules. This is achieved by following a modular architecture, combining a DL reasoner for OWL reasoning and a rule engine for rule execution [6][7][9][24][39]. Therefore, while in homogeneous approaches OWL reasoning is performed only by rules, in the hybrid paradigm the OWL reasoning is performed only by the DL reasoner.

While the DL reasoners have poor ABox reasoning performance [13][15][35], the EOR paradigm has limited TBox reasoning completeness. For example, if *p* and *g* are both the inverse properties of *q*, then *p* and *g* should be inferred as equivalent properties. We observed that the [23][28][33] EOR systems do not deduce such a TBox relationship, in contrast, for example, to Pellet [44] which supports it. Notice that [23][28][33] treat the properties *p* and *g* as equivalent at the instance level through the implementation of the inverse entailment (*rdfp8ax*, Table 1). Thus, if $\langle x\ p\ y\rangle$ then $\langle y\ q\ x\rangle$, and therefore, $\langle x\ g\ y\rangle$. However, they do not infer that *p* and *g* are equivalent because they do not implement the corresponding entailment. In the EOR paradigm, some TBox entailments are either ignored, in order to speed up the TBox reasoning procedure, or they have not been considered during implementation. For both reasons, the result is an incomplete TBox reasoning procedure in the EOR paradigm.

The use of entailments makes the EOR paradigm also incomplete on ABox reasoning, for the same reasons we mentioned previously. Thus, the choice between an EOR system and a DL reasoner depends on the domain and the needs of the application (see also [4][38]). With DLE we tackle the TBox reasoning incompleteness of EOR. Our motivation can be summarized in the following observation: "*Why do we need to struggle to define the entailments for OWL TBox reasoning in the EOR paradigm, if we can make it effortless and more complete using the efficient DL algorithms*?".

In [37] the RDFS(FA) sublanguage of RDFS and the RDF Model Theory are discussed. In brief, RDFS(FA) eliminates the dual roles of RDFS, stratifying built-in RDF primitives in different layers. On the other hand, the RDF MT handles dual roles by treating classes and properties as objects. Practically, DL reasoners treat the OWL extension of the bottom two layers of RDFS(FA), while the entailments are defined in RDF MT. For example, RDFS(FA) distinguishes built-in from user-defined properties, while in RDFS there is no such restriction and thus it is more expressive (although it is argued that this expressivity is too confusing). The existing EOR systems treat OWL as an extension of RDF MT, allowing *anyone to say anything about anything*. This was the domain of interest in [30] and [31], where only a rule engine is considered for TBox and ABox entailments in the RDF MT. On the other hand, the DLE framework is an RDFS(FA)-oriented approach, since it uses a DL component to reason on OWL ontologies, *following though the EOR paradigm*.

Our DLE framework considers the EOR paradigm as a *rule program* (ABox entailments) over the KB of a DL reasoner after TBox reasoning, following the architecture of the hybrid paradigm. Therefore, any subsequent user-defined rule program would then coexist in the rule base of the rule engine with the ABox rule program, in the same manner as in the homogeneous paradigm. Thus, in contrast to the existing EOR implementations, the OWL reasoning in the DLE framework is performed both by a DL component and a rule engine. More specifically, it combines the TBox inference capabilities of the DL component to compute the subsumption hierarchy and the related semantics to the ontology roles, e.g. domain constraints, property types, etc., with the ability of a rule engine to process a large number of instances, applying domain-specific entailment rules that are generated based on the DL reasoner.

Apart from scalability issues, one of the attracting features of the EOR paradigm is that it can be easily implemented in any rule engine, e.g. Jess [22], or a Prolog engine [42], enabling the use of ontological information into rule programs, exploiting the research on efficient rule engines with different capabilities. The DLE framework is based on this practicality and actually enhances the EOR paradigm in two directions:

- It simplifies the development of an EOR system, disengaging the TBox reasoning procedure from any (incomplete) TBox entailment implementation that should take into account all the possible OWL TBox semantic derivations.
- The rule engine applies faster the domain-specific ABox entailments than the corresponding generic entailments of the traditional EOR systems, enhancing their scalability in terms of ABox reasoning time and memory utilization.

3 Entailment Classification

The DLE framework is based on the classification of entailments into *terminological*, *hybrid* and *exceptional*. In this section we present the necessary and sufficient conditions for performing such a classification. Notice that, since we are based on a DL reasoner, the DLE framework is not compatible with RDF ontologies and handles only the OWL vocabulary [29]. Thus, we do not capture relationships such as that `owl:Class` is equivalent or subclass to `rdfs:Class`, `owl:Thing` is equivalent or subclass to `rdfs:Resource` or `owl:ObjectProperty` is equivalent or subclass to `rdfs:Property`. Furthermore, `owl:ObjectProperty` and `owl:DatatypePrope-rty` are disjoint sets [29]. Since we approach the entailment-based reasoning from the OWL perspective, we substitute any reference to `rdfs:Resource` and `rdfs:Class` in entailment rules with `owl:Thing` and `owl:Class`, respectively (Table 1).

We present a definition of an entailment rule that we follow in the rest of the paper.

Definition 1. *An entailment rule for an RDF graph G is of the form*

$$\langle s_1\, p_1\, o_1\rangle \langle s_2\, p_2\, o_2\rangle \ldots \langle s_n\, p_n\, o_n\rangle \rightarrow \langle s'_1\, p'_1\, o'_1\rangle \langle s'_2\, p'_2\, o'_2\rangle \ldots \langle s'_m\, p'_m\, o'_m\rangle,$$

where $n \geq 1$, $m \geq 1$, s_i, s'_i, p_i and p'_i are RDF URI references or blank nodes, and o_i and o'_i are RDF URI references, blank nodes or literals. The $\langle s_n\, p_n\, o_n\rangle$ triples denote the condition of the entailment and the $\langle s'_m\, p'_m\, o'_m\rangle$ triples the conclusion. The condition of the rule denotes the RDF triples that should exist in G, and the conclusion the RDF triples that should be added in G.

If $n = 0$, then all the conclusion triples should always exist in G (axiomatic triples [14][17]). If $m = 0$, then the entailment denotes that the triple pattern of the body should be viewed as inconsistent (inconsistency entailment).

3.1 Terminological and Assertional Triples

We present a classification of triples into *terminological* (*T-triples*) and *assertional* (*A-triples*), according to the OWL DL vocabulary V [29] of their components (prefixes have been omitted).

Definition 2. *A triple $t = \langle s\ p\ o \rangle$ is a terminological triple, denoted as $t_T = \langle s\ p\ o \rangle_T$, iff*

- $p \in$ {`domain`, `range`, `subClassOf`, `subPropertyOf`, `inverseOf`, `equivalentProperty`, `equivalentClass`, `intersectionOf`, `unionOf`, `complementOf`, `onProperty`, `hasValue`, `someValuesFrom`, `allValuesFrom`, `maxCardinality`, `minCardinality`, `cardinality`, `disjointWith`}, *or*
- $p =$ `type` $\wedge\ o \in$ {`ObjectProperty`, `DatatypeProperty`, `FunctionalProperty`, `InverseFunctionalProperty`, `SymmetricProperty`, `TransitiveProperty`, `Class`, `Restriction`}.

A T-triple denotes information about the TBox of the ontology (class and property axioms), such as subclass relationships, class equivalence, property types, etc. The A-triples are defined as the complement of the terminological.

Definition 3. *A triple $t = \langle s\ p\ o \rangle$ is an assertional triple, denoted as $t_A = \langle s\ p\ o \rangle_A$, iff it is not a terminological, that is $t_A \Leftrightarrow \neg t_T$.*

Intuitively, an A-triple denotes information about the ABox of the ontology, such as instance class membership or instance equality/inequality (`sameAs` /`differentFrom`) (we consider the `oneOf` construct as simple $i : C$ assertions). Thus, $\langle p$ `domain` $c \rangle_T$ and $\langle p$ `type` `FunctionalProperty`$\rangle_T$, whereas $\langle x$ `sameAs` $y \rangle_A$. Therefore, a triple component can either be bound to a term of the OWL vocabulary or not be bound. In the former case, we refer to the component as *constant*, whereas in the latter as *variable* (blank node). For example, the triple $\langle p$ `domain` $c \rangle$ has a *variable* subject and object, and a *constant* predicate. We use the notation $var(c)$ to denote that the c component of the triple is a *variable*.

3.2 Terminological, Hybrid and Exceptional Entailments

Based on the triple classification of section 3.1, we define the *terminological*, *hybrid* and *exceptional* entailments.

Definition 4. *An entailment is considered as a terminological (T-entailment), if and only if it contains only T-triples in its conclusion.*

Definition 5. *An entailment is considered as a hybrid (H-entailment), if and only if it contains both T- and A-triples in its condition and only A-triples in its conclusion.*

Definition 6. *An entailment is considered as an exceptional (E-entailment), if and only if it contains only A-triples in its condition and conclusion.*

Table 1 depicts some indicative examples of entailment classification, as well as some eRDFS entailments needed for OWL TBox reasoning [14] (denoted as *extX*).

4 Reasoning on the DLE Framework

The reasoning on the DLE framework is based on two reasoning paradigms over two distinct KBs that cooperate.

Table 1. Classification examples of some common entailment rules

Terminological Entailment Rules (T-entailments)	
rdfs8	$\langle c$ `type` `Class`$\rangle_T \rightarrow \langle c$ `subClassOf` `Thing`$\rangle_T$
rdfs11	$\langle c$ `subClassOf` $d\rangle_T \langle d$ `subClassOf` $k\rangle_T \rightarrow \langle c$ `subClassOf` $k\rangle_T$
rdfp12c	$\langle c$ `subClassOf` $d\rangle_T \langle d$ `subClassOf` $c\rangle_T \rightarrow \langle c$ `equivalentClass` $d\rangle_T$
rdfp13c	$\langle p$ `subPropertyOf` $q\rangle_T \langle q$ `subPropertyOf` $p\rangle_T \rightarrow \langle p$ `equivalentProperty` $q\rangle_T$
ext1	$\langle p$ `domain` $c\rangle_T \langle c$ `subClassOf` $d\rangle_T \rightarrow \langle p$ `domain` $d\rangle_T$
ext2	$\langle p$ `domain` $c\rangle_T \langle b$ `subPropertyOf` $p\rangle_T \rightarrow \langle b$ `domain` $c\rangle_T$
Hybrid Entailment Rules (H-entailments)	
rdfs2	$\langle p$ `domain` $c\rangle_T \langle x\ p\ y\rangle_A \rightarrow \langle x$ `type` $c\rangle_A$
rdfp8ax	$\langle p$ `inverseOf` $q\rangle_T \langle x\ p\ y\rangle_A \rightarrow \langle y\ q\ x\rangle_A$
rdfs9	$\langle c$ `subClassOf` $d\rangle_T \langle x$ `type` $c\rangle_A \rightarrow \langle x$ `type` $d\rangle_A$
rdfp1	$\langle p$ `type` `FunctionalProperty`$\rangle_T \langle x\ p\ y\rangle_A \langle x\ p\ z\rangle_A \rightarrow \langle y$ `sameAs` $z\rangle_A$
rdfp4	$\langle p$ `type` `TransitiveProperty`$\rangle_T \langle x\ p\ y\rangle_A \langle y\ p\ z\rangle_A \rightarrow \langle x\ p\ z\rangle_A$
rdfp14a	$\langle r$ `hasValue` $y\rangle_T \langle r$ `onProperty` $p\rangle_T \langle x\ p\ y\rangle_A \rightarrow \langle x$ `type` $r\rangle_A$
Exceptional Entailment Rules (E-entailments)	
rdfs4a	$\langle x\ p\ y\rangle_A \rightarrow \langle x$ `type` `Thing`$\rangle_A$
rdfs4b	$\langle x\ p\ y\rangle_A \rightarrow \langle y$ `type` `Thing`$\rangle_A$
rdfp6	$\langle x$ `sameAs` $y\rangle_A \rightarrow \langle y$ `sameAs` $x\rangle_A$
rdfp7	$\langle x$ `sameAs` $y\rangle_A \langle y$ `sameAs` $z\rangle_A \rightarrow \langle x$ `sameAs` $z\rangle_A$
rdfp11	$\langle x\ p\ y\rangle_A \langle x$ `sameAs` $x'\rangle_A \langle y$ `sameAs` $y'\rangle_A \rightarrow \langle x'\ p\ y'\rangle_A$

Definition 7. *The DLE framework consists of two distinct knowledge bases DLE =* ($\mathcal{DLKB}$, $\mathcal{RKB}$) *where:*

- $\mathcal{DLKB}$ *is the DL component's KB, with* $\mathcal{DLKB} = \langle\mathcal{T}\rangle$, *where* $\mathcal{T}$ *is the ontology TBox (concept and role axioms), and*
- $\mathcal{RKB}$ *is the rule engine's KB, with* $\mathcal{RKB} = \langle\mathcal{RB}, \mathcal{A}\rangle$, *where* $\mathcal{RB}$ *is the rule base of the rule engine that contains the entailment rules and* $\mathcal{A}$ *is the ABox of the ontology (instance and role assertions).*

The two KBs are distinct in the sense that the information flows only from the $\mathcal{DLKB}$ to the $\mathcal{RKB}$ (unidirectional) in order to populate the $\mathcal{RB}$ with entailments.

4.1 Reasoning on the DL Component

Basic DL reasoning problems include *class equivalence*, *concept subsumption*, *satisfiability* and *realization*. Since the $\mathcal{DLKB}$ does not consider ABoxes, the DL component of the DLE framework is not used for instance realization.

The use of DL TBox reasoning in the DLE framework makes redundant the T-entailments. These entailments generate T-triples (*Definition* 4), such as subclass and class equivalence relationships, domain and range restrictions, property equivalence, etc. Since these TBox semantics are handled by the DL reasoner, the T-entailments are ignored, decoupling the TBox inference procedure from any entailment-based approach. As we explain in the next section, each T-triple of an H-entailment is substituted with a query to the $\mathcal{DLKB}$. In that way, *we are not concerned about how to implement the TBox semantics in the DLE framework, but only how to use them at the instance level via ABox entailments*.

To exemplify, consider the *rdfs11* entailment for subclass transitivity. For every three concepts $C, D, E \in \mathcal{T}$ of the $\mathcal{DLKB}$:

$$\textit{if } \mathcal{T} \models C \sqsubseteq D \textit{ and } \mathcal{T} \models D \sqsubseteq E\textit{, then } \mathcal{DLKB} \models C \sqsubseteq E.$$

Practically, by querying the $\mathcal{DLKB}$ for the indirect superclasses of a concept, all the concepts that belong to the subclass transitive closure are returned (as well as their equivalents), thus the *rdfs11* entailment is natively supported.

Consider also the `intersectionOf` construct. A class C is contained in the intersection of the classes $C_1, \ldots, C_n$ by saying that C is a subclass of each class C_n. This is also the inference result of a DL reasoner:

$$\textit{if } \mathcal{T} \models C \equiv C_1 \sqcap \ldots \sqcap C_n\textit{, then } \mathcal{DLKB} \models C \sqsubseteq C_n.$$

Thus, by querying the DL reasoner for the superclasses of C, the intersection classes C_n are also returned (a similar approach is followed for the semantics of the `unionOf` construct). Notice that the iff semantics of class intersection (and union) require ABox reasoning and thus, the DLE framework handles them at the instance level by entailments. The same holds for class restrictions, e.g. $\forall R.C$. The DL reasoner is used also to facilitate entailment-free TBox consistency check, e.g. inconsistent subclass relationships of disjoint classes.

4.2 Transforming H-Entailments

Since we do not consider T-entailments, the $\mathcal{RKB}$ does not contain any T-triple and thus, the H-entailments would never be activated. In order to cope with the missing TBox triple information, we substitute each T-triple of the condition of an H-entailment with a query to the DL component in order to ground the TBox-related variables (*vars*) of the remaining A-triples. We call this procedure *entailment reduction*, because the H-entailments are *reduced* to domain-specific E-entailments, which we call *dse-entailments*. Therefore, for a specific H-entailment, it is possible to generate more than one dse-entailment. There are two advantages behind the reduction:

- Since the dse-entailments are generated based on the ontology TBox axioms, the $\mathcal{RB}$ will contain only the entailments that are needed, in contrast to the traditional EOR paradigm where all the entailments are preloaded. For example, a transitive role-free ontology will result in a transitive entailment-free $\mathcal{RB}$, reducing the number of the rules need to be checked in each cycle.
- The dse-entailments contain less conditional elements than the initial H-entailments, since the T-triples are removed. Thus, the dse-entailments have lower complexity and thus, they are activated faster by the rule engine.

DL queries. A T-triple t_t can be transformed into a TBox query for the DL component, denoted as $t_T \twoheadrightarrow \mathcal{DLQ}(t_T)$, in order to retrieve the ontology values that correspond to the variables of the T-triple. Similarly, the set T of the conditional T-triples of an H-entailment H can be transformed into a conjunction query, denoted as $\mathcal{DLQ}_H$, that corresponds to each T-triple, that is $T \twoheadrightarrow \mathcal{DLQ}(t1_T) \wedge \ldots \wedge \mathcal{DLQ}(tn_T), \forall tn_T \in T$.

To exemplify, using a predicate-like syntax for the DL queries, the T-triple $\langle p\ \texttt{domain}\ c\rangle_T$ of the *rdfs2* entailment can be transformed into the query:

$$\mathcal{DLQ}_{rdfs2} : \langle p\ \texttt{domain}\ c\rangle_T \twoheadrightarrow \texttt{domain}(var(p), var(c)),$$

retrieving all the properties of the $\mathcal{DLKB}$ with their domain constraints. Similarly, the T-triples of the *rdfp14a* entailment can be transformed into:

$$\mathcal{DLQ}_{rdfp14a} : \{\langle r\ \texttt{hasValue}\ y\rangle_T, \langle r\ \texttt{onProperty}\ p\rangle_T\} \twoheadrightarrow \texttt{hasValue}(var(r), var(y)) \wedge \texttt{onProperty}(var(r), var(p)),$$

retrieving the properties and their restriction values for every `hasValue` restriction.

T-dependency. We introduce the notion of *T-dependency* between an A-triple (t_A) and a T-triple (t_T), according to whether t_A has a variable component that exists in t_T.

Definition 8. *An A-triple t_A is T-dependent to a T-triple t_T, denoted as $t_A \curvearrowright t_T$, iff $\exists var(c) \in t_A : var(c) \in t_T$. Each such c variable of a T-dependent triple is called T-dependent variable and it is denoted as $[c]$ in the entailment rule.*

For example, both the $t2_A = \langle x\ p\ y\rangle_A$ and $t3_A = \langle x\ \texttt{type}\ c\rangle_A$ A-triples of the *rdfs2* h-entailment are T-dependent to $t1_T = \langle p\ \texttt{domain}\ c\rangle_T$, that is $t2_A \curvearrowright t1_T$ and $t3_A \curvearrowright t1_T$, since $var(p) \in t2_A$, $var(c) \in t3_A$ and $var(p), var(c) \in t1_T$. On the other hand, in the *rdfp1* H-entailment only the A-triples $t2_A = \langle x\ p\ y\rangle_A$ and $t3_A = \langle x\ p\ z\rangle_A$ are T-dependent to $t1_T = \langle p\ \texttt{type FunctionalProperty}\rangle_T$, since $\nexists var(c) \in t1_T : var(c) \in t4_A = \langle y\ \texttt{sameAs}\ z\rangle_A$. The T-dependency denotes the A-triples whose T-dependent variables should be grounded, due to the removal of the T-triple on which they depend.

Generating the dse-entailments. An H-entailment H is reduced by removing the T-triples of its condition and applying a $\mathcal{DLQ}_H$ query. The results of the query are used to ground the T-dependent variables of the A-triples, generating domain-specific versions of H (dse-entailments). The H-entailment reduction can be considered as the procedure of grounding the T-dependent variables of a *pseudo-rule*.

Definition 9. *The pseudo-rule PR_H for the H-entailment H, is the entailment-like rule we obtain after the removal of any T-triple of the H's condition, and it is of the form*

$$\langle [s_i]\ p_i\ o_i\rangle_A \ldots \langle s_k\ [p_k]\ o_k\rangle_A \ldots \langle s_n\ p_n\ [o_n]\rangle_A \rightarrow \langle [s_l]\ p_l\ o_l\rangle_A \ldots \langle s_m\ [p_m]\ o_m\rangle_A \ldots \langle s_o\ p_o\ [o_o]\rangle_A \ldots \langle s_u\ p_u\ o_u\rangle_A,$$

where $[x]$ denotes the T-dependent variables of the entailment. The pseudo-rule is actually a template rule which can be loaded in the $\mathcal{RB}$ as a valid rule (dse-entailment) after the grounding of its T-dependent variables.

For example, the pseudo-rule PR_{rdfs2} for the *rdfs2* entailment that we obtain after the removal of its T-triples is the following:

$$\langle x\ [p]\ y\rangle_A \rightarrow \langle x\ \texttt{type}\ [c]\rangle_A,$$

where [p] and [c] are the two T-dependent variables of the entailment. Thus, based on the $\mathcal{DLQ}_{rdfs2}$ query, we can ground ($G[PR_{rdfs2}(p, c)]$) the T-dependent variables as:

$$\forall(p, c) \in \mathcal{DLQ}_{rdfs2} : G[\langle x\ [p]\ y\rangle_A \rightarrow \langle x\ \texttt{type}\ [c]\rangle_A],$$

generating as many rules as the pairs (*property*, *domain*) are in the ontology. For example, for two properties p_k and p_m with the classes c_k and c_m as domain constraints, respectively, we will obtain the following dse-entailments:

$$\langle x\ p_k\ y\rangle_A \rightarrow \langle x\ \texttt{type}\ c_k\rangle_A,$$
$$\langle x\ p_m\ y\rangle_A \rightarrow \langle x\ \texttt{type}\ c_m\rangle_A.$$

One of the advantages of the entailment reduction is that the complexity of the dse-entailments is lower than the corresponding H-entailments. The time needed for the *rdfs2* H-entailment to be activated is $O(n^2)$, where n is the size of the partial closure graph under construction [17], whereas the reduced one can be handled in $O(pn)$, where p is the number of the grounded entailments generated from an H-entailment. Similarly, the *rdfp14a* H-entailment requires $O(n^3)$ time, while the reduced entailment runs in $O(pn)$. Generally, if $O(n^t)$ is the complexity of an H-entailment, where t is the number of triples of the condition, the reduced entailments have $O(pn^{t-k})$ complexity, where k is the number of T-triples (that are removed). We should mention that:

- The reduction results in a $\mathcal{RB}$ that contains more entailments than the initial H-entailments, since for each H-entailment more than one rule might be generated (p rules). However, in section 5 we show that such an $\mathcal{RB}$ terminates the ABox reasoning procedure faster than the corresponding generic H-entailment $\mathcal{RB}$.
- The $\mathcal{RB}$ contains only ABox-related entailments. Thus, only updates related to instances can be handled. We elaborate further on this in section 7.

4.3 Basic Reasoning Steps in a Forward Chaining DLE Framework

The E-entailments are the only entailments that are predefined in the DLE framework, since they cannot be reduced, following the approach of the convectional EOR paradigm. Assuming that E_A and PR are the sets of the A-entailments and the pseudo-rules (reduced H-entailments) that will be supported by the DLE-based implementation, the algorithm of Fig. 1 depicts the reasoning methodology using a forward chaining rule engine. Initially, the TBox of the ontology is loaded into the DL reasoner in order to classify the ontology (lines 1 and 2). Then, the ABox is loaded into the rule engine (line 3) in order to create the internal rule engine representation, for example triple facts. Moreover, the predefined E-entailments are loaded into the $\mathcal{RB}$ (line 4). In order to generate the dse-entailments, i.e. the grounded pseudo-rules, we conduct the necessary $\mathcal{DLQ}_H$ queries to the DL component in order to retrieve the values for the T-dependent variables of the H entailments. The resulting rules are loaded into the $\mathcal{RB}$, populating it with the domain specific dse-entailments (lines 5, 6 and 7). Finally, the rule engine runs and materializes the semantics in the form of derived triples.

```
BEGIN
1. $\mathcal{T} \leftarrow load(TBox)$
2. $classify(\mathcal{DLKB})$
3. $\mathcal{A} \leftarrow load(ABox)$
4. for each $e_A \in E_A$ do $\mathcal{RB} \leftarrow load(e_A)$
5. for each $pr_H \in PR$ do
6.    for each $(x_1,\ldots,x_n) \in \mathcal{DLQ}_H$ do
7.       $\mathcal{RB} \leftarrow load(G[pr_H(x_1,\ldots,x_n)])$
8. $\mathcal{RKB}.run()$
END
```

Fig. 1. The reasoning steps involved in a production rule-based DLE system

5 Testing the ABox Reasoning Performance

We conducted experiments to test the scalability of the dse-entailments against the conventional implementation of the same set of entailments in the same rule engine. We used three rule engines (Bossam [33], the *forwardRETE* rule engine of Jena [28] and Jess [22]) and developed six prototype implementations: three DLE-based, using the Pellet reasoner as the DL component, and three generic, i.e. direct implementation of the entailments following the conventional EOR paradigm. Each prototype was tested on the UOBM [26], Vicodi and wine ontologies[1]. Table 2 depicts the number of entailment rules that each implementation involves. Notice that:

- Our intention is to test the behaviour of a rule engine following the DLE and the generic EOR paradigms and not to compare these two paradigms on different rule engines, since each rule engine has a different performance.
- We are not concerned about the completeness of reasoning, since it depends on the number of the implemented entailments[2]. We want only to test the scalability of the prototypes in terms of rule application time and memory utilization.
- The response time of queries over the ABoxes between a DLE-based and the corresponding generic implementation in the same rule engine is the same, since both approaches result in the same KB (inferred triples).

Moreover we should mention that a fair comparison of the DLE prototypes with existing EOR systems that use the same rule engines is not feasible since this requires the implementation of the same set of entailments that the reasoners support, as well as to follow the same implementation principles or potential optimizations. However, the Bossam OWL reasoner does not provide the full set of the supported entailments, and the Jena OWL reasoner and OWLJessKB implement some semantics internally without entailments, such as the class intersection (Jena reasoner) or using `defqueries` and `deffunctions` (OWLJessKB). In order to conduct a fair comparison, we re-implemented directly the same set of entailments in the three rule engines. The experiments ran on Windows XP with 3.2 GHz processor, 2 GB RAM and 1.2 GB maximum Java heap space.

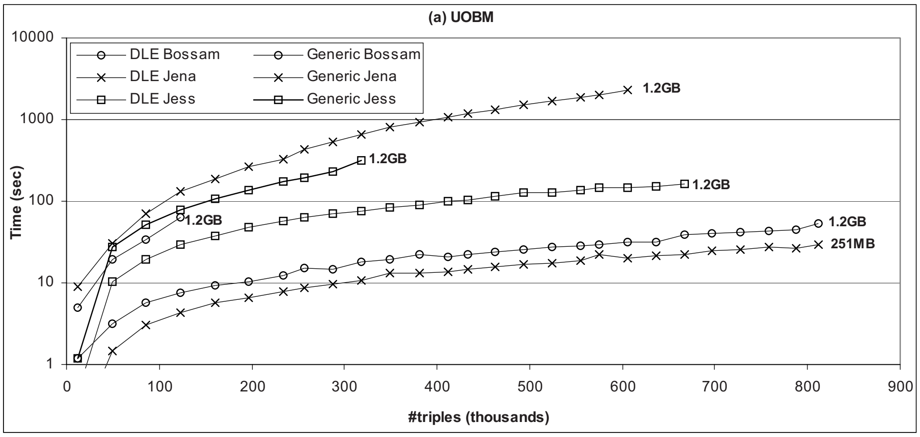
[1] We obtained Vicodi and wine from kaon2.semanticweb.org/download/test_ontologies.zip
[2] A set of OWL entailment rules can be found in http://www.agfa.com/w3c/euler/owl-rules

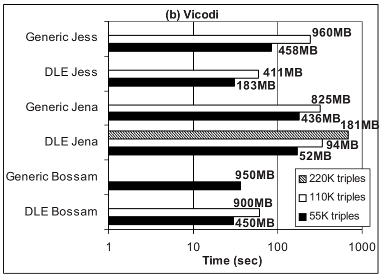
Table 2. The number of the entailments involved in the DLE and Generic implementations

	DLE Implementations (dse-entailments + exceptional)	Generic Implementations (generic entailments)
UOBM	323	34 (16 terminological + 13 hybrid + 5 exceptional)
Vicodi	1,164	
wine	592	

Fig. 2 depicts the ABox reasoning performance of each prototype in each ontology dataset. Each graph displays also the memory requirements of each implementation.
UOBM. We used a dataset of almost 810,000 triples (Fig. 2 (a)). DLE Bossam reasoned considerably faster than the Generic Bossam. In particular, it reasoned on five times more triples until it reached the memory limit. DLE Jena displayed a notably better performance than the Generic Jena, processing almost 200,000 more triples without reaching the memory limit. Finally, DLE Jess managed to process faster a dataset twice the size of the one processed by the Generic Jess.

Vicodi. These experiments were performed on three datasets (Fig. 2 (b)). DLE Jess demonstrated a better performance than the Generic Jess both in terms of reasoning time and memory utilization. DLE Jena processed the first two datasets in almost





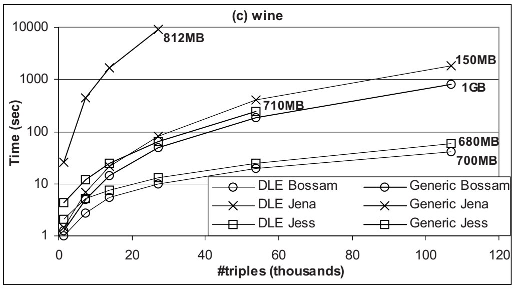


Fig. 2. Results on ABox reasoning

the same time to the Generic Jena but with better memory utilization, enabling the processing also of the third dataset without reaching the memory limit. The same behaviour observed in DLE Bossam that managed to process the first two datasets without reaching the memory limit. In contrast to DLE Jess, DLE Bossam and DLE Jena seem to be affected by the number of the dse-entailments (1,159) of their rule base. However, the memory utilization remains still in lower levels than the generic prototypes.

Wine. The wine experiments used a dataset of about 110,000 triples (Fig. 2 (c)). DLE Bossam processed the dataset significantly faster than the Generic Bossam, using half of the available memory. Generic Jena displayed a poor reasoning performance, while the DLE Jena managed to load the dataset in a reasonable time limit. Finally, Generic Jess processed only half triples than DLE Jess before reaching the memory limit.

Fig. 3 (a) depicts the TBox reasoning times of the prototypes. Since the DLE implementations are based on Pellet for TBox reasoning, they have the same TBox reasoning performance. Except for the wine ontology, Pellet achieves faster TBox inferencing than the generic entailment-based approaches. Bear in mind that the generic prototypes had been implemented with a limited number of T-entailments (16 entailments), while Pellet performs full TBox reasoning. The average dse-entailments generation time of the three DLE-prototypes is 250 ms (lines 5, 6 and 7 in Fig. 1).

In order to give a gist about the ABox reasoning performance of Pellet and KAON2, we present in Fig. 3 (b) the time needed by Pellet, KAON2 and DLE Jena to retrieve the instances on some datasets (since KAON2 performs reasoning on demand and thus, a query is required). We forced Pellet to completely realize the ABoxes which is close to the total materialization approach of our six prototypes, since we used forward chaining rule engines. For TBoxes with medium and large complexity, i.e. many class restriction, intersection or equivalence axioms, such as the UOBM and wine ontologies, Pellet does not perform well compared to the DLE approach, emphasizing the need for scalable implementations. On simple TBoxes, such as the Vicodi ontology which contains only simple subclass axioms, Pellet performs better, but KAON2 depicts the best performance, exploiting the ability to reason on demand. However, on the other two ontologies of medium and large TBox complexity, the DLE implementation performs better, especially on UOBM, even if it follows the

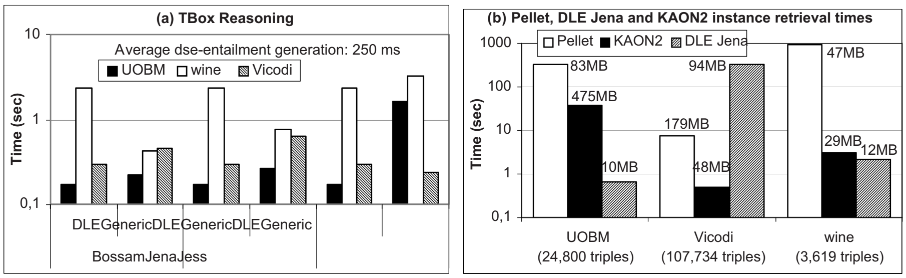


Fig. 3. (a) TBox reasoning times, and (b) Pellet, KAON2 and Jena DLE instance retrieval times

complete materialization approach. Notice that the results of Fig. 3 (b) cannot be considered as a fair comparison, due to the limited semantics that the DLE prototype supports and the different rule paradigm that it follows. A comparison of KAON2 with a backward chaining DLE implementation would be more meaningful. However, Fig. 3 (b) gives a gist about the weak and strong points of each reasoning paradigm.

6 Related Work

To the best of our knowledge, the existing EOR systems, such as OWLJessKB [23], Bossam [33], BaseVISor [27], Jena [28] and OWLIM [21], follow the same approach: the asserted ontological knowledge is transformed into facts and TBox and ABox entailment rules are applied. Although some systems offer the possibility to attach a DL reasoner for both TBox and ABox OWL reasoning, e.g. Jena, none of them has considered the possibility of using a DL reasoner in parallel with the rule engine for OWL reasoning. The DLE framework works towards this idea, combining the TBox inferencing capabilities of DL reasoners with the scalability of the EOR paradigm.

In [43] and [25], the entailments were enhanced with a dependency information, denoting the rules that should be checked after the firing of each entailment. Although this improves the performance, it is very difficult to manage such rule bases, since any modification needs the reconfiguration of the correlations. Furthermore, our experiments have shown that it is not the number of the rules that matters most, but the complexity of their condition. Although the $\mathcal{RB}$ of the DLE approach contains more rules than the conventional EOR paradigm, the inferencing procedure terminates faster. However, we believe that a combination of the DLE with the approach of [43] and [25] would increase even more the performance.

KAON2 [19] reduces OWL into disjunctive Datalog [18]. Its reasoning procedure is based totally on rules and it is focused mainly on query answering. As it was mentioned in [35], DL reasoners have better classification performance on complex TBoxes than KAON2. The DLE framework tries to embed this DL TBox efficiency into the EOR paradigm. The ABox performance of KAON2 depends on the TBox, having an increased performance on simple TBoxes.

Instance Store [15] combines a DL reasoner with a relational database. The idea is to use the DL reasoner for TBox reasoning and the database to store the ABox. The limitation of this approach is that it deals only with role-free ontologies, i.e. ontologies that contain only axioms of the form $i : C$. However, the use of a database is the only solution when the ABox exceeds the size of main memory (see also [43] and Owlgres [36] for DL-Lite). DLE has been only tested in main memory.

In [5] the embedding of different RDF entailments (including eRDFS) in F-Logic is presented that can be used to extend RDF or to align RDF and OWL DL. The DLE approach is focused only on the OWL language, defining an OWL reasoning framework based both on a DL reasoner and a rule engine.

The notion of generating ABox rules in the conventional EOR paradigm was briefly introduced in [30] and [31], where both the TBox and ABox reasoning are performed only by a rule engine (RDF MT). DLE extends and enhances our previous research, defining a new EOR reasoning paradigm (dedicated to OWL ontologies) using DL reasoning for TBox inferencing and ABox entailment generation.

7 Conclusions and Future Work

In this paper we presented an approach for embedding the TBox inferencing capabilities of DL reasoners into the EOR paradigm, resulting in an OWL-oriented reasoning framework. In that way we are able to capture OWL TBox semantics without applying TBox entailments, as well as to enhance the EOR scalability in terms of reasoning time and memory utilization. This is achieved by generating "engine-friendly" ABox entailments, with less conditional elements in their body (thus less complex) than the corresponding generic entailments of the traditional EOR paradigm.

We tested three DLE-based implementations against the three traditional EOR implementations, using three well known rule engines. The experiments have shown that although the DLE prototypes need to apply more rules than the corresponding EOR, they achieve better reasoning performance (at least on the tested ontologies) in terms of rule application time and memory utilization. We conclude that a DLE approach can considerably enhance the performance of existing EOR systems (regarding OWL reasoning), such as the Jena, OWLJessKB and Bossam OWL reasoners. More experiments, however, need to be conducted in order to investigate the impact of the number of the dse-entailments on the reasoning performance.

Although we define an entailment-free TBox framework, the DLE still depends on entailments for ABox reasoning. Thus, it is still a rule-based approach with all the modelling strengths and weaknesses comparing it to the DL paradigm, such as the limited modelling capabilities with incomplete information, the closed-world reasoning or the Unique Name Assumption (UNA) (see [4][34][38] for details).

In section 4 we mentioned that DLE is a unidirectional framework able to handle updates only on instances. However, in the hybrid paradigm, there is the notion of the *bidirectional* combination, allowing the rule program to alter the ontological information of the DL component [8][20][46]. This would be an interesting extension of the DLE framework and we plan to implemented it by indexing appropriately the dse-entailments in order to modify the $\mathcal{RB}$ according to TBox updates.

We are working on releasing a stable, DIG-compliant and more complete, in terms of supported ABox entailments, Jena-based DLE system, since with the Jena rule engine we achieved the best combination of memory utilization and reasoning performance. We plan also to implement a DLE system using the backward-chaining rule engine of Jena. As a practical application of the DLE framework, we consider the domain of OWL-S Semantic Web Service discovery and composition, where there is the need of efficient TBox reasoning on complex TBoxes, and scalable ABox reasoning on service advertisements that point to TBox concepts (inputs/outputs).

Acknowledgments. This work was partially supported by a PENED program (No. 03EΔ73), jointly funded by the European Union and the Greek government (GSRT).

References

1. Antoniou, G., Damasio, C.V., Grosof, B., Horrocks, I., Kifer, M., Maluszynski, J., Patel-Schneider, P.F.: Combining Rules and Ontologies, REWERSE Deliverables, REWERSE-DEL-2005-I3-D3 (2005)
2. Baader, F.: The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, Cambridge

3. Baader, F., Sattler, U.: An overview of tableau algorithms for description logics. Studia Logica 69(1), 5–40
4. de Bruijn, J., Lara, R., Polleres, A., Fensel, D.: OWL DL vs. OWL Flight: Conceptual Modeling and Reasoning for the Semantic Web. In: International Conference on World Wide Web, pp. 623–632. ACM Press, New York (2005)
5. de Bruijn, J., Heymans, S.: Logical foundations of (e)RDF(S): Complexity and reasoning. In: International Semantic Web Conference (+ 2ndASWC), pp. 86–99. Springer, Heidelberg (2007)
6. Donini, F.M., Lenzerini, M., Nardi, D., Schaerf, A.: AL-log: Integrating Datalog and Description Logics. Intelligent and Cooperative Information Systems, 227–252 (1998)
7. Drabent, W., Henriksson, J., Maluszynski, J.: HD-rules: A Hybrid System Interfacing Prolog with DL-reasoners. In: Applications of Logic Programming to the Web, Semantic Web and Semantic Web Services, CEUR-WS, vol. 287, pp. 76–90 (2007)
8. Eiter, T., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining Answer Set Programming with Description Logics for the Semantic Web. Knowledge Representation and Reasoning, 141–151 (2004)
9. Eiter, T., Ianni, G., Schindlauer, R., Tompits, H.: NLP-DL: A Knowledge-Representation System for Coupling Nonmonotonic Logic Programs with Description Logics. In: International Semantic Web Conference, Galway, Ireland (2005)
10. Grant, J., Beckett, D.: RDF Test Cases (2004), http://www.w3.org/TR/rdf-testcases/
11. Grosof, B.N., Horrocks, I., Volz, R., Decker, S.: Description Logic Programs: Combining Logic Programs with Description Logic. In: WWW, pp. 48–57. ACM Press, New York (2003)
12. Haarslev, V., Moller, R.: Racer: A Core Inference Engine for the Semantic Web. In: International Workshop on Evaluation of Ontology-based Tools, pp. 27–36 (2003)
13. Haarslev, V., Moller, R.: An Empirical Evaluation of Optimization Strategies for ABox Reasoning in Expressive Description Logics. Description Logics, 22 (1999)
14. Hayes, P.: RDF Semantics (2004), http://www.w3.org/TR/rdf-mt/
15. Horrocks, I., Li, L., Turi, D., Bechhofer, S.: The Instance Store: Description Logic Reasoning with Large Numbers of Individuals. Description Logics, 104, 31–40 (2004)
16. Horst, H.J.: Extending the RDFS Entailment Lemma. In: Proceedings of the International Semantic Web Conference, pp. 77–91. Springer, Heidelberg (2004)
17. Horst, H.J.: Completeness, Decidability and Complexity of Entailment for RDF Schema and a Semantic Extension Involving the OWL Vocabulary. Journal of Web Semantics 3(2-3), 79–115 (2005)
18. Hustadt, U., Motik, B., Sattler, U.: Reducing SHIQ- Description Logic to Disjunctive Datalog Programs. Knowledge Representation and Reasoning, Canada, pp. 152–162 (2004)
19. KAON2, http://kaon2.semanticweb.org/
20. Kattenstroth, H., May, W., Schenk, F.: Combining OWL with F-Logic Rules and Defaults, Applications of Logic Programming to the Web. In: Semantic Web and Semantic Web Services. CEUR-WS, vol. 287, pp. 60–75 (2007)
21. Kiryakov, A., Ognyanov, D., Manov, F.: OWLIM - A Pragmatic Semantic Repository for OWL. In: Scalable Semantic Web Knowledge Base Systems, pp. 182–192. Springer, Heidelberg (2005)
22. Jess, http://herzberg.ca.sandia.gov/
23. Kopena, J.: OWLJessKB, http://edge.cs.drexel.edu/assemblies/software/owljesskb/
24. Levy, A.Y., Rousset, M.: Combining Horn Rules and Description Logics in CARIN. Artificial Intelligence 104(1-2), 165–209 (1998)

25. Li, H., Wang, Y., Qu, Y., Pan, J.Z.: A Reasoning Algorithm for pD*. In: 1st Asian Semantic Web Conference, pp. 293–299. Springer, Heidelberg (2006)
26. Ma, L., Yang, Y., Qiu, Z., Xie, G., Pan, Y., Liu, S.: Towards a Complete OWL Ontology Benchmark. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 125–139. Springer, Heidelberg (2006)
27. Matheus, C., Dionne, B., Parent, D., Baclawski, K., Kokar, M.: BaseVISor: A Forward-Chaining Inference Engine Optimized for RDF/OWL Triples. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, Springer, Heidelberg (2006)
28. McBride, B.: Jena, Implementing the RDF Model and Syntax Specification. In: International Workshop on the Semantic Web, CEUR-WS, vol. 40 (2001)
29. McGuinness, D.L., Harmelen, F.: OWL Web Ontology Language Overview, W3C Recommendation, http://www.w3.org/TR/owl-features/
30. Meditskos, G., Bassiliades, N.: Rule-based OWL Ontology Reasoning Using Dynamic ABOX Entailments. In: 18th European Conference on Artificial Intelligence (ECAI), pp. 731–732. IOS Press, Patras (2008)
31. Meditskos, G., Bassiliades, N.: A Rule-Based Object-Oriented OWL Reasoner. IEEE Transactions on Knowledge and Data Engineering 20(3), 397–410 (2008)
32. Mei, J., Lin, Z., Boley, H.: ALC: An Integration of Description Logic and General Rules. In: Proceedings of the Web Reasoning and Rule Systems, pp. 163–177. Springer, Heidelberg (2007)
33. Minsu, J., Sohn, J.C.: Bossam: An Extended Rule Engine for OWL Inferencing. In: Rules and Rule Markup Languages for the Semantic Web, pp. 128–138 (2004)
34. Motik, B., Horrocks, I., Rosati, R., Sattler, U.: Can OWL and Logic Programming Live Together Happily Ever After? In: International Semantic Web Conference, pp. 501–514 (2006)
35. Motik, B., Sattler, U.: A Comparison of Reasoning Techniques for Querying Large Description Logic ABoxes. In: Logic for Programming Artificial Intelligence and Reasoning, pp. 227–241 (2006)
36. Owlgres, http://pellet.owldl.com/owlgres
37. Pan, J.Z., Horrocks, I.: RDFS(FA) and RDF MT: Two Semantics for RDFS. In: International Semantic Web Conference, pp. 30–46. Springer, Heidelberg (2003)
38. Patel-Schneider, P.F., Horrocks, I.: A Comparison of Two Modelling Paradigms in the Semantic Web. In: International Conference on World Wide Web, pp. 3–12. ACM Press, New York (2006)
39. Rosati, R.: Towards expressive KR systems integrating datalog and description logics. In: Workshop on Description Logics, CEUR-WS, vol. 22, pp. 160–164 (1999)
40. Rosati, R.: Semantic and Computational Advantages of the Safe Integration of Ontologies and Rules. In: Proc. Principles and Practice of Semantic Web Reasoning, pp. 50–64 (2005)
41. Rosati, R.: DL+log: Tight Integration of Description Logics and Disjunctive Datalog. In: Principles of Knowledge Representation and Reasoning, pp. 68–78. AAAI Press, Menlo Park (2006)
42. Sagonas, K., Swift, T., Warren, D.S.: XSB as an Efficient Deductive Database Engine. ACM SIGMOD Record 23(2), 442–453 (1994)
43. Sesame, http://openrdf.org/
44. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A Practical OWL-DL Reasoner. Journal of Web Semantics 5(2), 51–53 (2007)
45. Tsarkov, D., Horrocks, I.: Fact++ description logic reasoner: System description. In: Proceedings of Automated Reasoning, pp. 292–297. Springer, Heidelberg (2006)
46. Wang, K., Billington, D., Blee, J., Antoniou, G.: Combining Description Logic and Defeasible Logic for the Semantic Web. In: Rules and Rule Markup Languages for the Semantic Web, pp. 170–181. Springer, Heidelberg (2004)