The Not-So-Easy Task of Computing Class Subsumptions in OWL RL

Markus Krötzsch

Department of Computer Science, University of Oxford, UK,
markus.kroetzsch@cs.ox.ac.uk

Abstract. The lightweight ontology language OWL RL is used for reasoning with large amounts of data. To this end, the W3C standard provides a simple system of deduction rules, which operate directly on the RDF syntax of OWL. Several similar systems have been studied. However, these approaches are usually complete for instance retrieval only. This paper asks if and how such methods could also be used for computing entailed subclass relationships. Checking entailment for arbitrary OWL RL class subsumptions is co-NP-hard, but tractable rule-based reasoning is possible when restricting to subsumptions between atomic classes. Surprisingly, however, this cannot be achieved in any RDF-based rule system, i.e., the W3C calculus cannot be extended to compute all atomic class subsumptions. We identify syntactic restrictions to mitigate this problem, and propose a rule system that is sound and complete for many OWL RL ontologies.

1 Introduction

The lightweight ontology language OWL RL [16] is widely used for reasoning with large amounts of data, and many systems support query answering over OWL RL ontologies. Commercial implementations of (parts of) OWL RL include Oracle 11g [12], OWLIM [2], Virtuoso [4], and AllegroGraph [5].

What makes OWL RL so appealing to users and implementers alike are its favourable computational properties. As for all three lightweight profiles of OWL 2, typical reasoning tasks for OWL RL can be solved in polynomial time. Possibly even more important, however, is the fact that this low complexity can be achieved by relatively straightforward algorithms that perform bottom-up, rule-base materialisation of logical consequences until saturation. While it is also possible to use similar algorithms for OWL EL [14], a simple rule-based algorithm for OWL RL is already given in the W3C specification [16]. Various similar rule sets have been published for fragments of OWL RL (e.g., [9,7,20]).

Another advantage of these rule systems is that they operate directly on the RDF serialisation of OWL. OWL RL reasoning thus can easily be implemented on top of existing RDF databases that support some form of production rules to infer additional information. Basic rule-matching capabilities are found in most RDF stores, since they are similar to query matching. Indeed, *SPARQL rules* have been proposed as a natural rule extension of the SPARQL query language

that can fully express the OWL RL rule system [19]. It is important that neither value invention (*blank node creation*) nor non-monotonic negation are needed in OWL RL, as both features complicate rule evaluation significantly [19].

A common strategy for evaluating larger amounts of OWL RL data is to separate terminological information (schema-level axioms) from assertional information (facts), since the latter are typically significantly larger [13,8,20]. To further reduce the rules that need to be applied to the data, it would be useful to pre-compute terminological inferences, especially all subclass relationships.

Unfortunately, it is not known how to do this. To the best of our knowledge, no polynomial time algorithm has been published for computing schema entailments in OWL RL. As shown in Section 2, the W3C rule system is not complete for computing class subsumptions, and, more problematically, it is not possible to compute all class subsumptions without taking assertions into account.

However, it is still possible to obtain RDF-based rule systems that can discover more entailments than the W3C rules. Using various abbreviations introduced in Section 3, we present one such algorithm in Section 4. We identify ObjectHasValue as the main obstacle – if it does not occur in superclasses, our algorithm can compute all class subsumptions. In Section 5, we take a look at RDF-based rules and show how our results transfer to this setting.

In Section 6, we discuss the problem of computing class subsumptions in unrestricted OWL RL ontologies, where ObjectHasValue is allowed in superclasses. It is still possible to use polynomial time rule systems for this case, but it turns out that there is no *RDF-based* rule system for this task. This surprising result shows an inherent limitation of the expressive power of RDF-based rules.

Most of our presentation is based on the Functional-Style Syntax of OWL [17]. This yields a more concise presentation (compared to the RDF serialisation) and still is close to the actual language. We assume basic familiarity with the syntax and semantics of OWL on the level of the *OWL Primer* [6]. If not stated otherwise, we generally consider the Direct Semantics of OWL, but we also mention some results about the RDF-based Semantics. When writing OWL entities in axioms, we tacitly use prefixes for suggesting abbreviated IRIs. Some proofs are omitted for reasons of space; they are found in a technical report [15].

2 Challenges of Schema Reasoning in OWL RL

Before we focus on the task of computing class subsumptions for OWL RL, it is worth pointing out some limitations and challenges that do not seem to be well known, even among practitioners and implementers.

Checking Entailment of OWL RL Axioms is Not Tractable The W3C specification mentions that *Class Expression Subsumption* in OWL RL is PTime-complete w.r.t. the size of the ontology [16, Section 5]. This might evoke the impression that one could check in polynomial time whether an OWL RL class inclusion axiom is entailed by an OWL RL ontology. This is not the case.[1]

[1] The 2012 update of the OWL 2 specification will correct this; see Section 7.

Proposition 1. *Given an OWL RL ontology $\mathcal{O}$ and an OWL RL* SubClassOf *axiom A, checking whether $\mathcal{O}$ entails A is co-*NP*-hard.*

Proof. We show this by reducing 3SAT unsatisfiability to OWL RL entailment checking. An instance of the 3SAT problem is a set of propositional clauses $\{(L_{11} \vee L_{12} \vee L_{13}), \ldots, (L_{n1} \vee L_{n2} \vee L_{n3})\}$, where each L_{ij} is a propositional variable or a negated propositional variable. The question whether the conjunction of these clauses is satisfiable is NP-complete. For each propositional variable p, we introduce two new class names T_p and F_p. To each literal L_{ij}, we assign a class name $c(L_{ij})$ as follows: if $L_{ij} = p$, then $c(L_{ij}) := T_p$; if $L_{ij} = \neg p$, then $c(L_{ij}) := F_p$. For every clause $(L_{i1} \vee L_{i2} \vee L_{i3})$, we define a class expression C_i as ObjectUnionOf($c(L_{11})$ $c(L_{12})$ $c(L_{13})$). Now let A be the axiom SubClassOf(ObjectIntersectionOf($C_1 \ldots C_n$) owl:Nothing), and let $\mathcal{O}$ be the ontology that consists of the axioms DisjointClasses(T_p F_p) for every propositional variable p. Clearly, A is an OWL RL axiom and $\mathcal{O}$ is an OWL RL ontology. However, $\mathcal{O}$ entails A if and only if the given instance of 3SAT has *no* solution. Indeed, if A is not entailed, then $\mathcal{O}$ has a model where an individual e is an instance of each of the class expression C_i. We can construct a propositional truth assignment as follows: if e is an instance of T_p, then p is mapped to true; otherwise p is mapped to false. It is easy to see that this is a solution to the 3SAT instance, since e cannot be an instance of T_p and F_p for any p. □

Another way to find hardness proofs is to use DataSomeValuesFrom in the subclass, together with datatypes such as XML Schema boolean, which is admissible in OWL RL subclasses. Moreover, similar problems can be found for other axiom types; e.g., checking entailment of hasKey axioms is also intractable.

These problems are hardly surprising from a logical perspective, since checking the entailment of A from $\mathcal{O}$ is equivalent to checking the consistency of $\mathcal{O} \cup \{\neg A\}$, where $\neg A$ is the negation of the axiom A (as a logical formula). For the latter to be in OWL RL, one needs to impose different syntactic restrictions on A. The check is then possible in PTime. A particularly relevant case where this is possible is that A is a subclass relationship between two class names. This is the task that we will focus on in the remainder of this work.

The W3C Rule System is Not Complete for Class Subsumption Checking The W3C specification states a completeness theorem for its rule system, which asserts completeness only for entailment of assertional axioms. However, the rule system contains a number of rules that would not be necessary to achieve this especially in Table 9, entitled *The Semantics of Schema Vocabulary*. This might lead to the wrong impression that the rules can infer all schema axioms, or at least all class subsumptions. The following example illustrates that this is wrong.

Example 1. We consider an ontology of three axioms:

SubClassOf(:A :B) (1)

SubClassOf(:A :C) (2)

SubClassOf(ObjectIntersectionOf(:B :C) :D) (3)

This ontology clearly entails SubClassOf(:A :D). However, this is not entailed by the W3C rule system. The only entailment rules that refer to ObjectIntersectionOf are rules *cls-int1*, *cls-int2*, and *scm-int* in [16]. The former two rules are only applicable to individual instances. Rule *scm-int* can be used to infer SubClassOf(ObjectIntersectionOf(:B :C) :B) and SubClassOf(ObjectIntersectionOf(:B :C) :C) – the rule can be viewed as a schema-level version of *cls-int2*. However, there is no rule that corresponds to *cls-int1* on the schema level, so one can not infer SubClassOf(:A ObjectIntersectionOf(:B :C)).

This example extends to many other types of axioms. For example, one cannot infer all entailed property domains or ranges if some class subsumptions have been missed.

Assertions Can Not be Ignored when Checking Class Subsumptions Since many OWL RL ontologies are dominated by assertional axioms (i.e., data), it would be useful if this part of the ontology would not be relevant if one is only interested in computing class subsumptions. This cannot work in general, since assertions can cause the ontology to become inconsistent, which in turn leads to the entailment of arbitrary class subsumptions. However, even for consistent ontologies, assertions cannot be ignored when computing class subsumptions, as shown in the next example.

Example 2. We consider an ontology of three axioms:

$$\text{InstanceOf(:B :b)} \tag{4}$$

$$\text{SubClassOf(:A ObjectHasValue(:P :b))} \tag{5}$$

$$\text{SubClassOf(ObjectSomeValuesFrom(:P :B) :C)} \tag{6}$$

This ontology entails SubClassOf(:A :D): every instance of :A has a :P successor :b (5), that is an instance of :B (4); thus the subclass in (6) is a superclass of :A. Without (4) this entailment would be missed.

The relevant assertional information in this example was directly given, but it is clear that it could also be the result of more complicated reasoning. Therefore, it is not possible in general to compute all class subsumptions of an ontology without also computing a significant amount of fact entailments as well. Theorem 3 in Section 4 below identifies a case where assertions can be ignored.

3 A Simpler OWL RL

OWL is a very rich language that provides many redundant syntactic constructs for the convenience of ontology engineers. When specifying a rule system for OWL RL, this abundance of syntax precludes a concise presentation – the W3C calculus already counts 78 rules. To avoid this problem, we introduce various simplification rules that allow us to restrict to a much smaller number of features.

$$\mathbf{R_{ca1}}\ \frac{\mathsf{ClassAssertion(}\ C\ a\ \mathsf{)}}{\mathsf{SubClassOf(}\ \mathsf{ObjectOneOf(}a\mathsf{)}\ C\ \mathsf{)}} \qquad \mathbf{R_{ca2}}\ \frac{\mathsf{SubClassOf(}\ \mathsf{ObjectOneOf(}a\mathsf{)}\ C\ \mathsf{)}}{\mathsf{ClassAssertion(}\ C\ a\ \mathsf{)}}$$

$$\mathbf{R_{pc1}}\ \frac{\mathsf{ObjectPropertyAssertion(}\ P\ a\ b\ \mathsf{)}}{\mathsf{SubClassOf(}\ \mathsf{ObjectOneOf(}a\mathsf{)}\ \ \mathsf{ObjectSomeValuesFrom(}\ P\ \mathsf{ObjectOneOf(}b\mathsf{)}\ \mathsf{)}\ \mathsf{)}}$$

$$\mathbf{R_{pc2}}\ \frac{\mathsf{SubClassOf(}\ \mathsf{ObjectOneOf(}a\mathsf{)}\ \ \mathsf{ObjectSomeValuesFrom(}\ P\ \mathsf{ObjectOneOf(}b\mathsf{)}\ \mathsf{)}\ \mathsf{)}}{\mathsf{ObjectPropertyAssertion(}\ P\ a\ b\ \mathsf{)}}$$

Fig. 1. Rules for expressing assertions as class subsumptions

Syntactic simplifications can affect the semantics of an ontology. On the one hand, they may introduce auxiliary vocabulary symbols that had not been defined by the original ontology. On the other hand, any syntactic transformation of expressions has an impact on the RDF-based Semantics of OWL, which entails only axioms about expressions that are syntactically present in the ontology. Adding a new expression, even if tautological, thus changes entailments. However, we expect applications that rely on the RDF-based Semantics to tolerate this effect. Alternatively, it is possible to view our syntactic simplifications as a mere abbreviation scheme for a much larger number of rules.

Lists in Axioms We generally assume that all lists of classes or properties in OWL axioms have been binarised, that is, broken down into lists of length two. This is always possible by introducing additional axioms; we omit the details of this frequently used technique. We point out that this simplification is least essential for our rules. It is easy to generalise all rules we state for binary lists to lists of arbitrary length.

Datatypes and Data Properties We omit all features related to datatypes from our presentation. It is not hard to add them. In essence, datatypes in OWL RL behave like classes for which certain subsumptions are given upfront (e.g., *decimal* is a superclass of *integer*), and for which some disjointness axioms are known (e.g., *rational* is disjoint from *string*). Likewise, datatype literals behave like individual names for which the membership in some class (i.e., datatype) is known. This auxiliary information can be added when loading an ontology. Thereafter, datatypes can be treated like classes, using the same rule system.

Assertions as Class Inclusions Class and property assertions can be expressed as class inclusion axioms, and indeed many rules for assertions are just special forms of the rules needed for terminological axioms. To avoid unnecessary repetition, we generally use the respective forms interchangeably. This is formalised by the rules in Fig. 1. The rules are applied top to bottom: whenever the axioms above the line are found, the axioms below the line are added. C denotes a class expression (not necessarily a class name), and P denotes an object property expression. We use a and b for individual names (IRIs).

We will make use of the shorter syntactic form of assertions whenever possible. Note that the class subsumptions in rules $\mathbf{R_{pc1}}$ and $\mathbf{R_{pc2}}$ are not in OWL RL,

Table 1. Syntactic simplifications for OWL RL constructs

ObjectMaxCardinality(0 P)	ObjectAllValuesFrom(P owl:Nothing)
ObjectMaxCardinality(0 P C)	ObjectAllValuesFrom(P ObjectComplementOf(C))
ObjectHasValue(P d)	ObjectSomeValuesFrom(P ObjectOneOf(c))
ObjectOneOf(a_1 ... a_n)	ObjectUnionOn(ObjectOneOf(a_1) ... ObjectOneOf(a_n))
EquivalentClasses(C_1 ... C_n)	SubClassOf(C_1 C_2), ..., SubClassOf(C_n C_1)
DisjointClasses(C_1 ... C_n)	SubClassOf(ObjectIntersectionOf(C_i C_j) owl:Nothing) for all $1 \leq i < j \leq n$
ObjectPropertyDomain(P C)	SubClassOf(ObjectSomeValuesFrom(P owl:Thing) C)
ObjectPropertyRange(P C)	SubClassOf(owl:Thing ObjectAllValuesFrom(P C))
EquivalentObjectProperties(P_1 ... P_n)	SubObjectPropertyOf(P_1 P_2), ..., SubObjectPropertyOf(P_n P_1)
InverseObjectProperties(P Q)	SubObjectPropertyOf(P ObjectInverseOf(Q)), SubObjectPropertyOf(Q ObjectInverseOf(P))
SymmetricObjectProperty(P)	SubObjectPropertyOf(P ObjectInverseOf(P))
TransitiveObjectProperty(P)	SubObjectPropertyOf(ObjectPropertyChain(P P) P)
FunctionalObjectProperty(P)	SubClassOf(owl:Thing ObjectMaxCardinality(1 P))
InverseFunctionalObjectProperty(P)	SubClassOf(owl:Thing ObjectMaxCardinality(1 ObjectInverseOf(P)))
AsymmetricObjectProperty(P)	DisjointObjectProperties(P ObjectInverseOf(P))
SameIndividual(a_1 ... a_n)	SubClassOf(ObjectOneOf(a_1) ObjectOneOf(a_2)), ..., SubClassOf(ObjectOneOf(a_n) ObjectOneOf(a_1))
NegativeObjectPropertyAssertion(P a b)	SubClassOf(ObjectOneOf(a) ObjectComplementOf(ObjectSomeValuesFrom(P ObjectOneOf(b))))
DifferentIndividuals(a_1 ... a_n)	SubClassOf(ObjectOneOf(a_i) ObjectComplementOf(ObjectOneOf(a_j))) for all $1 \leq i < j \leq n$

Table 2. Subclasses (**CL**) and superclasses (**CR**) in syntactically simplified OWL RL

CL ::= **Class** | ObjectIntersectionOf(**CL CL**) | ObjectUnionOf(**CL CL**) | ObjectOneOf(**Individual**) | ObjectSomeValuesFrom(**Property CL**)

CR ::= **Class** | ObjectIntersectionOf(**CR CR**) | ObjectComplementOf(**CL**) | ObjectAllValuesFrom(**Property CR**) | ObjectMaxCardinality(1 **Property CL**) | ObjectSomeValuesFrom(**Property** ObjectOneOf(**Individual**)) | ObjectOneOf(**Individual**)

which does not allow ObjectOneOf in superclasses. Allowing this does not increase reasoning complexity as long as one restricts to exactly one individual in ObjectOneOf (indeed, this is also done in the tractable OWL EL profile). We will therefore introduce such expressions whenever this simplifies presentation.

Syntactic Sugar Many OWL features are directly expressible in terms of others. Table 1 lists straightforward syntactic transformations. C, D, E denote class expressions, P, Q object property expressions, and all lower-case letters denote individuals. Below, we can therefore disregard all features on the left of this table. As before, some of the expressions that we use here are not in OWL RL. Besides ObjectOneOf superclasses as discussed above, OWL RL also disallows owl:Thing to be used as a subclass. Again, introducing this does not complicate reasoning. The main motivation for leaving out owl:Thing in the standard is that it may lead to a big number of "uninteresting" entailments, since every individual is an instance of owl:Thing.

In summary, we therefore consider only OWL class inclusion axioms of the form SubClassOf(**CL** **CR**), where **CL** and **CR** are defined as in Table 2.

$$\mathbf{C_{sco}} \quad \frac{\mathsf{SubClassOf}(\ C\ \ D\ )\quad \mathsf{SubClassOf}(\ D\ \ E\ )}{\mathsf{SubClassOf}(\ C\ \ E\ )}$$

$$\mathbf{C_{init}} \quad \frac{C \text{ a class expression in the ontology}}{\mathsf{SubClassOf}(\ C\ \ C\ )\quad \mathsf{SubClassOf}(\ C\ \ \mathsf{owl{:}Thing}\ )}$$

$$\mathbf{C_{int-}} \quad \frac{\mathsf{SubClassOf}(\ C\ \ \mathsf{ObjectIntersectionOf}(\ D_1\ D_2\ )\ )}{\mathsf{SubClassOf}(\ C\ \ D_1\ )\quad \mathsf{SubClassOf}(\ C\ \ D_2\ )}$$

$$\mathbf{C_{int+}} \quad \frac{\mathsf{SubClassOf}(\ C\ \ D_1\ )\quad \mathsf{SubClassOf}(\ C\ \ D_2\ )}{\mathsf{SubClassOf}(\ C\ \ \mathsf{ObjectIntersectionOf}(\ D_1\ D_2\ )\ )}$$

$$\mathbf{C_{com-}} \quad \frac{\mathsf{SubClassOf}(\ C\ \ D\ )\quad \mathsf{SubClassOf}(\ C\ \ \mathsf{ObjectComplementOf}(D)\ )}{\mathsf{SubClassOf}(\ C\ \ \mathsf{owl{:}Nothing}\ )}$$

$$\mathbf{C_{uni+}} \quad \frac{\mathsf{SubClassOf}(\ C\ \ D\ )\quad \text{where } D = D_1 \text{ or } D = D_2}{\mathsf{SubClassOf}(\ C\ \ \mathsf{ObjectUnionOf}(\ D_1\ D_2\ )\ )}$$

$$\mathbf{C_{sa}} \quad \frac{\mathsf{SubClassOf}(\ \mathsf{ObjectOneOf}(c)\ \ \mathsf{ObjectOneOf}(d)\ )}{\mathsf{SubClassOf}(\ \mathsf{ObjectOneOf}(d)\ \ \mathsf{ObjectOneOf}(c)\ )}$$

Fig. 2. OWL RL inference rules for class subsumptions

4 A Rule-Based Classification Calculus for OWL RL

In this section, we specify a class subsumption algorithm for OWL RL, and we introduce conditions under which it is complete. Using the simplifications of the previous section, the only OWL axioms that we need to consider are SubClassOf, SubObjectPropertyOf, DisjointObjectProperties, IrreflexiveObjectProperty, and HasKey. Remaining expressive features are those given in Table 2 (for class expressions) and ObjectPropertyChain (for property inclusions).

Figures 2 and 3 specify a rule system for deriving class subsumptions, where we use the same notation for rules as above. The rules in Fig. 2 apply to class subsumptions and, using the correspondences of Fig. 1, also to assertions. In contrast, the rules in Fig. 3 are only applicable to specific assertions and are not generalised to other subsumptions. For example, rule $\mathbf{P_{inv-}}$ is sound, but the following generalisation to class inclusions would of course be wrong:

$$\frac{\mathsf{SubClassOf}(\ C\ \ \mathsf{ObjectSomeValuesFrom}(\ \mathsf{ObjectInverseOf}(P)\ D\ )\ )}{\mathsf{SubClassOf}(\ D\ \ \mathsf{ObjectSomeValuesFrom}(\ P\ C\ )\ )}$$

As an additional condition for applying rules, we require that all class and property expressions in the conclusion also occur in the ontology. This is a restriction only for the rules $\mathbf{C_{int+}}$, $\mathbf{C_{uni+}}$, $\mathbf{P_{svf+}}$, and $\mathbf{P_{inv+}}$, since they are the only rules that derive expressions that are not mentioned in their premise.

For an OWL ontology $\mathcal{O}$, we say that a class subsumption $\mathsf{SubClassOf}(\ C\ D\ )$ is *inferred* by the rule system if one of the following axioms is derived by applying the rules exhaustively to $\mathcal{O}$:

- $\mathsf{SubClassOf}(\ C\ \ D\ )$,
- $\mathsf{SubClassOf}(\ C\ \ \mathsf{owl{:}Nothing}\ )$, or
- $\mathsf{ClassAssertion}(\ \mathsf{owl{:}Nothing}\ \ c\ )$ for some individual c.

$$\mathbf{P_{avf-}}\ \frac{\mathsf{ObjectPropertyAssertion}(\ P\ c\ d\ )\quad \mathsf{ClassAssertion}(\ \mathsf{ObjectAllValuesFrom}(\ P\ E\ )\ c\ )}{\mathsf{ClassAssertion}(\ E\ d\ )}$$

$$\mathbf{P_{svf+}}\ \frac{\mathsf{ObjectPropertyAssertion}(\ P\ c\ d\ )\quad \mathsf{ClassAssertion}(\ E\ d\ )}{\mathsf{ClassAssertion}(\ \mathsf{ObjectSomeValuesFrom}(\ P\ E\ )\ c\ )}$$

$$\mathbf{P_{inv-}}\ \frac{\mathsf{ObjectPropertyAssertion}(\ \mathsf{ObjectInverseOf}(P)\ c\ d\ )}{\mathsf{ObjectPropertyAssertion}(\ P\ d\ c\ )}$$

$$\mathbf{P_{inv+}}\ \frac{\mathsf{ObjectPropertyAssertion}(\ P\ d\ c\ )}{\mathsf{ObjectPropertyAssertion}(\ \mathsf{ObjectInverseOf}(P)\ c\ d\ )}$$

$$\mathbf{P_{spo}}\ \frac{\mathsf{ObjectPropertyAssertion}(\ P\ c\ d\ )\quad \mathsf{SubObjectPropertyOf}(\ P\ Q\ )}{\mathsf{ObjectPropertyAssertion}(\ Q\ c\ d\ )}$$

$$\mathbf{P_{spc}}\ \frac{\begin{array}{c}\mathsf{SubObjectPropertyOf}(\ \mathsf{ObjectPropertyChain}(\ P\ Q\ )\ R\ )\\ \mathsf{ObjectPropertyAssertion}(\ P\ c\ d\ )\quad \mathsf{ObjectPropertyAssertion}(\ Q\ d\ e\ )\end{array}}{\mathsf{ObjectPropertyAssertion}(\ R\ c\ e\ )}$$

$$\mathbf{P_{dp}}\ \frac{\begin{array}{c}\mathsf{DisjointObjectProperties}(\ P\ Q\ )\\ \mathsf{ObjectPropertyAssertion}(\ P\ c\ d\ )\quad \mathsf{ObjectPropertyAssertion}(\ Q\ c\ d\ )\end{array}}{\mathsf{ClassAssertion}(\ \mathsf{owl{:}Nothing}\ c\ )}$$

$$\mathbf{P_{ip}}\ \frac{\mathsf{ObjectPropertyAssertion}(\ P\ c\ c\ )\quad \mathsf{IrreflexiveObjectProperty}(P)}{\mathsf{ClassAssertion}(\ \mathsf{owl{:}Nothing}\ c\ )}$$

$$\mathbf{P_{key}}\ \frac{\begin{array}{c}\mathsf{HasKey}(\ E\ (P_1 \ldots\ P_n)\ ()\ )\\ \mathsf{ClassAssertion}(\ E\ c\ )\quad \mathsf{ClassAssertion}(\ E\ d\ )\\ \mathsf{ObjectPropertyAssertion}(\ P_1\ c\ e_1\ )\ \ldots\ \mathsf{ObjectPropertyAssertion}(\ P_n\ c\ e_n\ )\\ \mathsf{ObjectPropertyAssertion}(\ P_1\ d\ e_1\ )\ \ldots\ \mathsf{ObjectPropertyAssertion}(\ P_n\ d\ e_n\ )\end{array}}{\mathsf{SubClassOf}(\ \mathsf{ObjectOneOf}(c)\ \ \mathsf{ObjectOneOf}(d)\ )}$$

$$\mathbf{P_{fun}}\ \frac{\begin{array}{c}\mathsf{ClassAssertion}(\ \mathsf{ObjectMaxCardinality}(\ 1\ P\ D\ )\ c\ )\\ \mathsf{ObjectPropertyAssertion}(\ P\ c\ e_1\ )\quad \mathsf{ObjectPropertyAssertion}(\ P\ c\ e_2\ )\\ \mathsf{ClassAssertion}(\ D\ e_1\ )\quad \mathsf{ClassAssertion}(\ D\ e_2\ )\end{array}}{\mathsf{SubClassOf}(\ \mathsf{ObjectOneOf}(e_1)\ \ \mathsf{ObjectOneOf}(e_2)\ )}$$

Fig. 3. OWL RL inference rules that are specific to property assertions

The first condition corresponds to a direct derivation, the second captures the case that C is inconsistent (necessarily empty), and the third case occurs whenever $\mathcal{O}$ is inconsistent. The inference rules in [16] use a special conclusion *false* to encode ontology inconsistency, but this just a minor difference.

Theorem 1. *The rule system is sound, that is, if a class subsumption A is inferred from $\mathcal{O}$, then $\mathcal{O}$ entails A under the Direct Semantics and under the RDF-based Semantics of OWL.*

However, the rule system is not complete. The following examples illustrate two interesting cases that are not covered.

Example 3. We consider an ontology of four axioms:

InstanceOf(:D :d) (7)

SubClassOf(:D :C) (8)

SubClassOf(:C ObjectHasValue(:P :a)) (9)

InstanceOf(ObjectMaxCardinality(1 ObjectInverseOf(:P) owl:Thing) :a) (10)

From this, the axiom SubClassOf(:C :D) follows. Indeed, (9) and (10) together imply that :C can have at most one instance; by (7) and (8), this instance is :d and thus contained in :D. However, this is not entailed by our rule system. Axioms (9) and (10) can be represented as follows:

SubClassOf(:C ObjectSomeValuesFrom(:P ObjectOneOf(:a))) (11)

InstanceOf(ObjectAllValuesFrom(ObjectInverseOf(:P) ObjectOneOf(:e)) :a) (12)

where :e is an auxiliary individual. Using $\mathbf{C}_{\mathbf{sco}}$ and the rules of Fig. 1, we can derive ObjectPropertyAssertion(:P :d :a) from (7), (8), and (11). By applying rule $\mathbf{P}_{\mathbf{inv}+}$, we obtain ObjectPropertyAssertion(ObjectInverseOf(:P) :a :d). Together with (12), $\mathbf{P}_{\mathbf{avf}-}$ implies ClassAssertion(ObjectOneOf(:e) :d). The same could be derived for any other instance :d' of :C, showing that all such instances must be equal, and instances of :D. However, we cannot derive SubClassOf(:C :D).

Example 4. Consider the ontology of the following axioms:

SubClassOf(:C ObjectHasValue(:Q :b)) (13)

ObjectPropertyRange(:Q :D) (14)

ObjectPropertyDomain(:Q :D) (15)

SubClassOf(:D ObjectHasValue(:P :a)) (16)

SubObjectPropertyOf(ObjectPropertyChain(:P ObjectInverseOf(:P)) :R) (17)

SubClassOf(ObjectSomeValuesFrom(:R owl:Thing) :E) (18)

These axioms imply SubClassOf(:C :E). Indeed, axioms (16) and (17) together imply that every pair of instances of :D is connected by property :R. Axioms (14) and (15) in turn imply that :Q only connects instances that are in :D. Thus, we find that :Q is a subproperty of :P, which is an interesting inference in its own right. Combining this with (13) and (18), we obtain the claimed entailment SubClassOf(:C :E). Again, this is not inferred by our inference rules.

Examples 3 and 4 illustrate two very different forms of semantic interactions that lead to entailments not inferred by our rule system. It is interesting to note, however, that ObjectHasValue plays a crucial role in both cases. Indeed, we find that this feature is involved in every situation where an entailment is missed:

Theorem 2. *If $\mathcal{O}$ is an OWL RL ontology that does not contain* ObjectHasValue *in superclasses, then the rule system is complete for $\mathcal{O}$. More precisely, let $\mathcal{O}'$ be the syntactic simplification of $\mathcal{O}$ as discussed above. If* :A *and* :B *are class names and $\mathcal{O}$ entails* SubClassOf(:A :B) *under Direct Semantics, then* SubClassOf(:A :B) *is inferred from $\mathcal{O}'$ by the rule system.*

Even in this case, we need to take assertions into account during reasoning, since they might make the ontology inconsistent. For consistent ontologies, however, the rule system can be simplified further.

Theorem 3. *For a consistent OWL RL ontology $\mathcal{O}$ that does not contain* ObjectHasValue *in superclasses, all entailed subsumptions between class names can be computed using the rules of Fig. 2 only, without taking assertions into account.*

More precisely, let $\mathcal{O}_t \subseteq \mathcal{O}$ be the set of all axioms in $\mathcal{O}$ that do not use ClassAssertion*,* ObjectPropertyAssertion*,* SameIndividual*,* DifferentIndividuals*, or* NegativeObjectPropertyAssertion*. Let $\mathcal{O}'_t$ be the syntactic simplification of $\mathcal{O}_t$. If $\mathcal{O}$ entails* SubClassOf(:A :B) *under Direct Semantics, and* :A *and* :B *are class names, then* SubClassOf(:A :B) *or* SubClassOf(:A owl:Nothing) *is derived from $\mathcal{O}'_t$ by the rules of Fig. 2.*

This tells us that the computation of class subsumptions in OWL RL is indeed very simple for consistent ontologies without ObjectHasValue in superclasses. Provided that this situation can be assumed, it would therefore be feasible to pre-compute class subsumptions without taking assertions into account. In data-centric ontologies, this can lead to a much smaller set of axioms. In addition, the set of rules that need to be applied becomes relatively small as well.

5 RDF-Based Rule Systems

We have formulated rules above using the Functional-Style Syntax of OWL. In this section, we explain how these results transfer to RDF-based rules in the style of the W3C specification [16], which act on the RDF serialisation of OWL [18]. This also prepares the ground for discussing the limitations of such rules in the next section.

Definition 1. *An* RDF-based rule *is a first-order implication of the form*

$$T(s_1, t_1, u_1) \wedge \ldots \wedge T(s_n, t_n, u_n) \rightarrow T(s, t, u)$$

where $s_{(i)}$, $t_{(i)}$, and $u_{(i)}$ are RDF terms (i.e., IRIs, literals, or blank nodes) or first-order logic variables. All variables are assumed to be universally quantified.

An (RDF-based) rule system *is a finite set $\mathcal{R}$ of RDF-based rules. $\mathcal{R}$ is applied to an RDF graph by considering RDF triples $\langle s\ p\ o \rangle$ as facts $T(s, p, o)$. A rule system $\mathcal{R}$ is applied to an OWL ontology $\mathcal{O}$ by applying it to the RDF serialisation of $\mathcal{O}$ [18].*

A rule system $\mathcal{R}$ is a sound and complete classification system *for a class $\mathcal{C}$ of ontologies if, for every ontology $\mathcal{O}$ in $\mathcal{C}$ and all class names* :A *and* :B*:*

SubClassOf(:A :B) *is entailed by $\mathcal{O}$*
if and only if
$\mathcal{R}$ infers T(:A, rdfs:subClassOf, :B) from $\mathcal{O}$.

The main features of RDF-based rule systems are thus as follows:

- Finiteness: the set of rules is finite
- Monotonicity: the larger the set of (derived) facts, the larger the number of rules that can be applied
- No value invention: applying rules does not introduce new terms
- Uniformity: the applicability of rules does not depend on the IRIs of the entities it is applied to, but only on the statements that these IRIs occur in
- Triple-based: the only relation symbol used in inferences is the ternary T

The W3C OWL RL inference rules do not constitute a rule system in the sense of Definition 1, since they are not finite in number. The reason is that the rules support list-based OWL features for lists of arbitrary length, leading to an infinite number of possible patterns. A rule system that is not restricted to be finite can trivially solve all reasoning tasks: for every ontology $\mathcal{O}$ for which we require an inference $T(s, p, o)$, we can add a rule $T_{\mathcal{O}} \rightarrow T(s, p, o)$, where $T_{\mathcal{O}}$ is the triple pattern that corresponds to the RDF serialisation of $\mathcal{O}$. It is also clear, however, that the infinite W3C rule system does not use this potential power.

Theorem 4. *The rules of Section 4 give rise to an RDF-based sound and complete classification system for (the syntactic simplification of) OWL RL ontologies without* ObjectHasValue *in superclasses.*

This result is based on the rule system from Section 4, which we already know to be sound and complete. These rules can be easily expressed in the RDF serialisation of OWL based on the T predicate. For example, $\mathsf{C}_{\mathsf{int}+}$ can be written as follows, where question marks denote variables as in [16]:

$$\begin{aligned} T(?x, \mathsf{rdfs{:}subClassOf}, ?y_1) \wedge T(?x, \mathsf{rdfs{:}subClassOf}, ?y_2) \wedge \\ T(?c, \mathsf{owl{:}intersectionOf}, ?l) \wedge T(?l, \mathsf{rdf{:}first}, ?y_1) \wedge T(?l, \mathsf{rdf{:}rest}, ?l') \wedge \\ T(?l', \mathsf{rdf{:}first}, ?y_2) \wedge T(?l', \mathsf{rdf{:}rest}, \mathsf{rdf{:}nil}) \\ \rightarrow T(?x, \mathsf{rdfs{:}subClassOf}, ?c) \end{aligned}$$

This is the rule that is mainly missing from [16]. Note how the check for the existence of the expression ObjectIntersectionOf(D_1 D_2) that is required for applying $\mathsf{C}_{\mathsf{int}+}$ is performed naturally as part of the rule application. This does not work for the rules in Fig. 1, which do not require existence of all class expressions in the consequence. Either they are created as part of the syntactic simplification stage, or the remaining rules need to be extended to allow for multiple alternative forms of each premise. The latter approach is taken in the W3C specification.

6 Rule-Based Classification of Unrestricted OWL RL

So far, we have seen that the (RDF-based) rule systems for OWL RL can be extended to obtain a sound and complete approach for computing class subsumptions, as long as we disallow ObjectHasValue in superclasses. The natural

question is whether this can be extended to arbitrary OWL RL ontologies. The answer is *no*, as we will show in this section.

This negative result is caused by the use of RDF as a basis for deduction rules. Without this restriction, it is not hard to find rule-based classification systems that are sound and complete, though not necessarily efficient. Namely, one can always check SubClassOf(C D) by adding an axiom InstanceOf(C e) for a new individual e, and checking whether InstanceOf(D e) using the existing rule-based approaches for instance retrieval. The problem is that this executes a single check, based on a modification of the ontology, rather than computing all class subsumptions at once. Tests for different classes may lead to different inferences. To address this, we can augment each inferred axiom with the test class C for which we have assumed InstanceOf(C e) (the IRI of e is immaterial and does not need to be recorded). The rules are restricted to the case that all premises can be derived under the same assumption, and computation can proceed concurrently without interactions between independent assumptions. Similar solutions and various optimisations have been suggested and implemented for OWL EL [14,10]. Unfortunately, however, it is not obvious how to represent such an extended form of inferences in RDF without introducing new entities or blank nodes.

The main result of this section is that this problem is not just due to an overly naive approach for extending the rules, but rather an inherent limitation of RDF-based rules:

Theorem 5. *There is no RDF-based sound and complete classification system for OWL RL.*

This is a very general result, since it makes a statement about *every* conceivable RDF-based rule system. A technique for proving such results has been developed in the context of rule-based reasoning for OWL EL [14]. Recalling this in full detail is beyond the scope of this paper. Instead, we extract the main insights on the level of ontological reasoning (Lemma 1), and present the crucial steps for applying this approach to our scenario.

The argumentation is based on the analysis of derivations in rule systems, which can be represented as *proof trees*. The key observation is that every proof tree of a rule system can be used to construct further proof trees by selectively renaming constants (i.e., IRIs). This renaming leads to proof trees that derive the same conclusion, applying rules in the same order, but based on a renamed input ontology. The renaming may not be uniform, that is, multiple occurrences of the same constant might be renamed differently. The proof uses the fact that such renaming may destroy entailments. Ontologies that are particularly sensitive in this respect are called *critical*:

Definition 2. *A* renaming *of an ontology $\mathcal{O}$ is an ontology $\mathcal{O}'$ that is obtained from $\mathcal{O}$ by replacing occurrences of entity names by fresh entity names, where neither the old nor the new entities have a special semantics in OWL. A renaming is* uniform *if all occurrences of one entity have been replaced in the same way; otherwise it is* non-uniform.

An ontology $\mathcal{O}$ is critical *for an axiom A if A is entailed by $\mathcal{O}$, and A is not entailed by any non-uniform renaming of $\mathcal{O}$.*

Roughly speaking, an ontology is critical if all of its axioms are really needed for obtaining the desired conclusion. The next result explains the significance of critical ontologies $\mathcal{O}$. It states that, given a sound and complete classification system, it must be possible to decompose $\mathcal{O}$ into sets $\mathcal{O}_{\text{split}}$ and $\mathcal{O} \setminus \mathcal{O}_{\text{split}}$. The requirement is that both sets "touch" in at most 3 axioms, which reflects the arity of RDF triples. By finding a critical ontology for which this is not possible, we can show that there is no sound and complete classification system.

Lemma 1. *Suppose that $\mathcal{R}$ is a sound and complete classification system with at most ℓ atoms $T(s,t,u)$ in the premise of any rule, and consider an ontology $\mathcal{O}$ that is critical for an axiom* SubClassOf(C D) *with class names C and D.*

For every $\mathcal{O}' \subseteq \mathcal{O}$ with $|\mathcal{O}'| > 3(\ell+1)$, there is $\mathcal{O}_{split} \subseteq \mathcal{O}$ such that:

- $|\mathcal{O}' \cap \mathcal{O}_{split}| \geq 4$,
- $|\mathcal{O}' \cap (\mathcal{O} \setminus \mathcal{O}_{split})| \geq 4$,
- *at most 3 axioms in $\mathcal{O}_{split}$ share any vocabulary symbols with $\mathcal{O} \setminus \mathcal{O}_{split}$.*

This result is obtained as a summary of various insights from [14]. Lemma 1 requires critical ontologies to contain at least $3(\ell+1)$ axioms. Since ℓ depends on the rule system we consider, we require critical OWL RL ontologies of arbitrary size. The following definition provides this crucial ingredient.

Definition 3. *For every natural number $k \geq 0$, we define an ontology $\mathcal{O}_k$ as follows. We consider class names A, B, D_i ($i = 0, \ldots, k+1$), property names V, W, P_i ($i = 0, \ldots, k$), Q_i ($i = 0, \ldots, k+1$), and individual names a, b, c, d_i ($i = 1, \ldots, k+1$). The ontology $\mathcal{O}_k$ consists of the following axioms:*

SubClassOf(D_i ObjectHasValue(P_i d_{i+1})	$0 \leq i \leq k$	(19)
SubClassOf(D_i ObjectAllValuesFrom(P_i D_{i+1})	$0 \leq i \leq k$	(20)
SubClassOf(D_0 ObjectHasValue(W a))		(21)
SubClassOf(D_0 ObjectAllValuesFrom(W A))		(22)
SubClassOf(A C)		(23)
SubClassOf(C ObjectHasValue(Q_{k+1} b))		(24)
SubClassOf(A ObjectHasValue(V c))		(25)
SubClassOf(D_{k+1} ObjectHasValue(V c))		(26)
InverseFunctionalObjectProperty(V)		(27)
SubObjectPropertyOf(ObjectPropertyChain(P_i Q_{i+1}) Q_i)	$0 \leq i \leq k$	(28)
SubClassOf(ObjectHasValue(Q_0 b) B)		(29)

Lemma 2. *For every $k \geq 0$, $\mathcal{O}_k$ entails* SubClassOf(D_0 B).

This can be seen as follows: If D_0 does not contain any instances, then the statement clearly holds. If D_0 contains some instance o, then it is the start of a

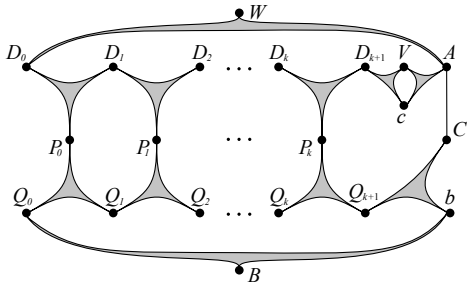

Fig. 4. Illustration of syntactic dependencies in $\mathcal{O}_k$

property chain $P_0, \ldots, P_k$ (through individuals $d_1, \ldots, d_{k+1}$) due to (19) and (20). In particular, d_k is an instance of D_{k+1}, hence, by (26), d_k has a V successor c. Similarly, by (21) and (22), a is an instance of A. By (23), (24), and (25), a thus has a Q_{k+1} successor b and a V successor c. Since V is inverse functional (27), a must therefore be equal to d_k, so d_k has a Q_{k+1} successor b. Applying axioms (28) to the chain of d_i elements, we find that d_i has the Q_i successor b for all $1 \leq i \leq k$. Accordingly, the instance o of D_0 has Q_0 successor b. By (29), o thus is an instance of B. Since this reasoning applies to every instance o of D_0, we find that SubClassOf(D_0 B) as claimed.

It is not hard to see that this entailment is no longer valid if any two occurrences of symbols within $\mathcal{O}_k$ are renamed in a non-uniform way:

Lemma 3. *For every $k \geq 0$, $\mathcal{O}_k$ is critical for* SubClassOf(D_0 B).

To complete the proof of Theorem 5 we thus need to argue that there are critical ontologies that cannot be split as in Lemma 1.

Lemma 4. *Consider $\mathcal{O}_k$ and let $\mathcal{O}'$ be the subset of all axioms* (20)*. For every $\mathcal{O}_{split} \subseteq \mathcal{O}$ such that $|\mathcal{O}_{split} \cap \mathcal{O}'| \geq 4$ and $|\mathcal{O}' \setminus \mathcal{O}_{split}| \geq 4$, there are at least* 4 *axioms in $\mathcal{O}_{split}$ that share vocabulary symbols with $\mathcal{O} \setminus \mathcal{O}_{split}$.*

Proof. We can illustrate the syntactic dependencies in an ontology by means of a (hyper)graph where each axiom is an edge between all entities that it refers to. Figure 4 shows part of the according graph for $\mathcal{O}_k$, showing only axioms (20), (22)–(26), (28), and (29). A subset of these axioms thus corresponds to a subset of edges, and the shared vocabulary symbols are shared nodes.

The assumptions on $\mathcal{O}_{\text{split}}$ require that $\mathcal{O}_{\text{split}}$ contains at least 4 of the axioms (20) (upper row in Fig. 4), and also misses at least 4 of the axioms (20). Using the illustration in Fig. 4, it is not hard to see that this requires $\mathcal{O}_{\text{split}}$ to share signature symbols with at least 4 axioms not in $\mathcal{O}_{\text{split}}$. □

Thus, for any rule system $\mathcal{R}$ with at most ℓ atoms in rule premises, the ontology $\mathcal{O}_k$ for $k = 3(\ell + 1)$ is critical but does not satisfy the conditions of Lemma 1. Thus, $\mathcal{R}$ cannot be a sound and complete classification system.

7 Conclusion

From a practical perspective, the main contribution of this work is to clarify the problems of OWL RL classification, and to propose rule systems for solving this task in relevant cases. The rules that we proposed produce sound conclusions on arbitrary OWL ontologies, under either of the two semantics of OWL. If the input is an OWL RL ontology where ObjectHasValue is not used in superclasses, the rule system is also guaranteed to be complete.

Our findings have also been brought to the attention of the OWL Working Group, which is preparing an editorial update of the OWL 2 specification at the time of this writing. This new version will correct the complexity claims about OWL RL. Extending the inference rules to be complete for computing class subsumptions in ontologies without ObjectHasValue, however, is beyond the scope of this editorial update. OWL RL tools can achieve completeness in this sense by adding, in essence, the one additional rule given after Theorem 4 (generalised to conjunctions of arbitrary arity). This does not affect official conformance.

Interestingly, ObjectHasValue in superclasses is the one OWL RL feature that complicates schema reasoning the most. This contrasts with OWL EL, where such expressions are as easy to handle as assertions [10]. The reason is that inverse properties, ObjectAllValuesFrom, and ObjectMaxCardinality, all of which allow for some complicated interactions with ObjectHasValue, are not in OWL EL.

Another interesting insight of this work is that there are practical problems in OWL RL reasoning that RDF-based rules are too inexpressive to solve. This limitation is due to the triple-based representation of RDF, which could be overcome by allowing predicates of higher arities as in Datalog [1] or RIF [11]. For keeping closer to features supported in RDF databases, it might be possible to use *quads* or *named graphs* for expressing 4-ary predicates, but it is doubtful if this would be an adequate use of these features. On the other hand, 4-ary relations are only needed as intermediate results during reasoning, so individual systems can implement solutions without referring to any language standard.

Another approach is to allow rules with value creation (*blank nodes* in rule heads) to encode n-ary relationships by introducing auxiliary entities. Value invention is problematic in general, as it can lead to non-termination and undecidability. Many works have studied conditions that ensure termination of bottom-up reasoning in the presence of value creation – see [3] for a recent overview – but it is unclear if any of these conditions would apply in our case.

Acknowledgements The research reported herein was supported by the EPSRC projects ConDOR and ExODA.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison Wesley (1994)
2. Bishop, B., Kiryakov, A., Ognyanoff, D., Peikov, I., Tashev, Z., Velkov, R.: OWLIM: a family of scalable semantic repositories. Semantic Web Journal 2(1), 33–42 (2011)

3. Cuenca Grau, B., Horrocks, I., Krötzsch, M., Kupke, C., Magka, D., Motik, B., Wang, Z.: Acyclicity conditions and their application to query answering in description logics. In: Proc. 13th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'12). pp. 243–253. AAAI Press (2012)
4. Erling, O.: Virtuoso, a hybrid RDBMS/graph column store. IEEE Data Eng. Bull. 35(1), 3–8 (2012)
5. Franz Inc.: AllegroGraph RDFStore: Web 3.0's Database (2012), http://www.franz.com/agraph/allegrograph/, accessed April 2012
6. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S. (eds.): OWL 2 Web Ontology Language: Primer. W3C Recommendation (27 October 2009), available at http://www.w3.org/TR/owl2-primer/
7. Hogan, A., Harth, A., Polleres, A.: Scalable authoritative OWL reasoning for the Web. Int. J. of Semantic Web Inf. Syst. 5(2), 49–90 (2009)
8. Hogan, A., Pan, J.Z., Polleres, A., Decker, S.: SAOR: template rule optimisations for distributed reasoning over 1 billion linked data triples. In: Proc. 9th Int. Semantic Web Conf. (ISWC'10). LNCS, vol. 6496, pp. 337–353. Springer (2010)
9. ter Horst, H.J.: Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary. J. of Web Semantics 3(2–3), 79–115 (2005)
10. Kazakov, Y., Krötzsch, M., Simančík, F.: Practical reasoning with nominals in the $\mathcal{EL}$ family of description logics. In: Proc. 13th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'12). pp. 264–274. AAAI Press (2012)
11. Kifer, M., Boley, H. (eds.): RIF Overview. W3C Working Group Note (22 June 2010), available at http://www.w3.org/TR/rif-overview/
12. Kolovski, V., Wu, Z., Eadon, G.: Optimizing enterprise-scale OWL 2 RL reasoning in a relational database system. In: Proc. 9th Int. Semantic Web Conf. (ISWC'10). LNCS, vol. 6496, pp. 436–452. Springer (2010)
13. Kotoulas, S., Oren, E., van Harmelen, F.: Mind the data skew: distributed inferencing by speeddating in elastic regions. In: Proc. 19th Int. Conf. on World Wide Web (WWW'10). pp. 531–540. WWW'10, ACM (2010)
14. Krötzsch, M.: Efficient rule-based inferencing for OWL EL. In: Walsh, T. (ed.) Proc. 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI'11). pp. 2668–2673. AAAI Press/IJCAI (2011)
15. Krötzsch, M.: The (not so) easy task of computing class subsumptions in OWL RL. Tech. rep., University of Oxford (2012), available from http://korrekt.org/page/OWLRL2012
16. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C. (eds.): OWL 2 Web Ontology Language: Profiles. W3C Recommendation (27 October 2009), available at http://www.w3.org/TR/owl2-profiles/
17. Motik, B., Patel-Schneider, P.F., Parsia, B. (eds.): OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax. W3C Recommendation (27 October 2009), available at http://www.w3.org/TR/owl2-syntax/
18. Patel-Schneider, P.F., Motik, B. (eds.): OWL 2 Web Ontology Language: Mapping to RDF Graphs. W3C Recommendation (27 October 2009), available at http://www.w3.org/TR/owl2-mapping-to-rdf/
19. Schenk, S., Staab, S.: Networked graphs: a declarative mechanism for SPARQL rules, SPARQL views and RDF data integration on the Web. In: Proc. 17th Int. Conf. on World Wide Web (WWW'08). pp. 585–594. ACM (2008)
20. Urbani, J., Kotoulas, S., Maassen, J., van Harmelen, F., Bal, H.: WebPIE: a Web-scale parallel inference engine using MapReduce. J. of Web Semantics 10, 59–75 (2012)