Combining RDF and Part of OWL with Rules: Semantics, Decidability, Complexity

Herman J. ter Horst

Philips Research, Eindhoven, The Netherlands
herman.ter.horst@philips.com

Abstract. This paper extends the model theory of RDF with rules, placing an emphasis on integration with OWL and decidability of entailment. We start from an abstract syntax that views a rule as a pair of rule graphs which generalize RDF graphs by also allowing rule variables in subject, predicate and object positions. We include RDFS as well as a decidable part of OWL that weakens D-entailment and OWL Full. Classes can be used as instances. Almost all examples in the DAML set of test rules are covered by our approach.

For a set of rules R, we define a general notion of R-entailment. Extending earlier results on RDFS and OWL, we prove a general completeness result for R-entailment. This result shows that a restricted form of application of rules that introduce blank nodes is sufficient to determine R-entailment. For rules that do not introduce blank nodes, we prove that R-entailment and R-consistency are decidable and in PSPACE. For rules that do not introduce blank nodes and that satisfy a bound on the size of rule bodies, we prove that R-consistency is in P, that R-entailment is in NP, and that R-entailment is in P if the target RDF graph is ground.

1 Introduction

There is much interest in combining the standard Semantic Web languages RDF and OWL with facilities for expressing and reasoning with rules. There is not yet a standard Semantic Web language for rules. The purpose of this paper is to extend the model theory of RDF [8] with rules, while integrating OWL. We focus specifically on decidability of entailment and on exploring the computational complexity of entailment.

It is well known that OWL Full entailment is undecidable and that OWL DL entailment is decidable and NEXPTIME-complete [9]. OWL DL is integrated with rules in SWRL [10]. Consistency and entailment for SWRL are undecidable [10]. OWL DL's direct model-theoretic semantics [16] has been extended for SWRL [10]. OWL DL's RDF-compatible semantics and correspondence theorem [16] have not been extended to SWRL.

In this paper we present basic definitions of an RDF-compatible semantics of rules. We combine this semantics with a non-standard semantics involving the OWL vocabulary, with lower computational complexity than OWL DL, so that there is greater scope for arriving at a decidable combination of ontologies and rules. In [11] the *pD** semantics was defined as a weakened variant of OWL Full. In [12] the *pD** semantics was extended to apply to a larger subset of the OWL vocabulary, which includes `FunctionalProperty`, `InverseFunctionalProperty`, `sameAs`, `SymmetricProperty`, `TransitiveProperty`, `inverseOf`, `equivalentClass`, `equivalentProperty`, `hasValue`, `someValuesFrom`, `allValuesFrom`, `differentFrom` and `disjointWith`. The *pD** semantics is in line with and extends the 'if-semantics' of RDFS [8], and is weaker than the 'iff-semantics' of D-entailment and OWL Full. As an example, the *pD** semantics assumes, like RDFS, that if c is a subclass of d, then each instance of c is an instance of d, but does not assume, like OWL does, that the converse condition also holds. While classes can be used as instances, fewer entailments are supported that relate to datatypes or entire classes or properties. The *pD** semantics seems to be sufficient for many applications where an ontology is used in combination with data relating to instances. There is a complete set of simple entailment rules for *pD** entailment which extend the standard entailment rules for RDFS; *pD** consistency is in P, while *pD** entailment is NP-complete, and in P if the target RDF graph has no blank nodes [12].

The model-theoretic semantics for RDF integrated with rules described in this paper includes the *pD** semantics. We show that the decidability and complexity results for the *pD** semantics can be extended to include a large class of rules. The resulting combination includes meta-modeling expressivity and uses a simple, uniform framework involving (entailment) rules also for RDFS and part of OWL. This leads to a relatively low threshold for implementation.

In this paper we describe some of the background to *pD** entailment, but refer to [12] for the underlying model theory. See [12], Sections 1.8 and 5.1, for an extensive discussion and comparison of the *pD** semantics and the semantics of OWL DL and Full. This paper does not contain complete proofs.[1]

2 Abstract Syntax, Examples, Overview, Discussion

2.1 Abstract Syntax for Rules

A rule is viewed as a pair of rule graphs; a rule graph is a set of *triple patterns*[2] which generalize RDF triples [13] by also allowing rule variables in subject, predicate and object positions. If $\rho = (\rho_\mathrm{l}, \rho_\mathrm{r})$ is a rule, then ρ_l is called the body or left-hand side of the rule, and ρ_r is called the head or right-hand side of the rule. We impose the common condition that each rule variable in the head of a rule also appears in the body of the rule. We also require that the body of a rule cannot contain blank nodes. It should be noted, however, that in an application of a rule, a rule variable in the body of the rule can be matched with a blank node in an RDF graph.

[1] A version of this paper with complete proofs is available on request.
[2] This term is used in the same way by [18].

If the body and the head of a rule ρ are both nonempty, then ρ will be called a *proper rule* and will be written informally as: `IF` ρ_l `THEN` ρ_r.

If the body of a rule ρ is empty, then the rule is viewed as specifying certain axioms. In this case ρ is called an *axiom rule*, written as: `AXIOMS` ρ_r.

If the head of a rule ρ is empty, then the rule is interpreted as specifying that a certain pattern of RDF statements should be viewed as inconsistent. In this case, ρ is called an *inconsistency rule*, written informally as: `NOT` ρ_l. Compare the standard equivalence of the formulas $P \Rightarrow Q$ and $\neg P \vee Q$.

In the following examples we use the N-Triples syntax for RDF [6] for bodies and heads of rules, writing rule variables for example as ?x.

2.2 Example: Uncle

The well-known uncle example displays a widely-used kind of rule that cannot be expressed in OWL:

```
IF   ?a ex:hasParent ?b .
     ?b ex:hasBrother ?c .
THEN ?a ex:hasUncle ?c .
```

2.3 Example: Entailment for RDFS and OWL

RDFS-entailment [8] is characterized by entailment rules (see Table 1 below), which can be viewed as being defined by proper rules. Along the same lines, the *pD** semantics [12] involving the OWL vocabulary is characterized by entailment rules, which can also be viewed as being defined by proper rules. The *pD** semantics can be realized by 1 axiom rule, 23 proper rules (none of which introduces blank nodes; cf. Table 2) and 2 inconsistency rules. We give two examples, the inconsistency rule for `differentFrom` and the proper rule for `FunctionalProperty` (cf. entailment rule rdfp1, see Table 2):

```
NOT ?v owl:differentFrom ?w .
    ?v owl:sameAs ?w .

IF   ?p rdf:type owl:FunctionalProperty .
     ?u ?p ?v .
     ?u ?p ?w .
THEN ?v owl:sameAs ?w .
```

2.4 Example: intersectionOf

Although the *pD** semantics does not explicitly include `unionOf` and `intersectionOf`, half of OWL's iff conditions for these constructs are available in an alternative way, by means of `rdfs:subClassOf` [12]. For example, the fact that a class c is contained in the intersection of the classes $c_1, \ldots, c_n$ can be expressed by saying that c is a subclass of each class c_j. The converse condition, and thereby OWL's complete semantic condition for `intersectionOf`, can be realized by adding a proper rule:

```
IF   ?x rdf:type c_1 .
     ...
     ?x rdf:type c_n .
THEN ?x rdf:type c .
```

2.5 Example: disjointProperties

Rules can be used for meta-modeling, for example to extend OWL. OWL's `disjointWith` primitive applies to classes: OWL does not have a similar notion for properties. Such a primitive can be added with an inconsistency rule and an axiom rule:

```
NOT ?p ex:disjointProperties ?q .
    ?a ?p ?b .
    ?a ?q ?b .

AXIOMS ex:disjointProperties rdfs:domain rdf:Property .
       ex:disjointProperties rdfs:range rdf:Property .
```

2.6 Example: someValuesFrom

The *pD** semantics [12] includes the complete iff condition for `hasValue` from the OWL semantics [16] (cf. entailment rules rdfp14a and rdfp14bx in Table 2 below), while including an if condition for `someValuesFrom` and `allValuesFrom` (cf. entailment rules rdfp15 and rdfp16 in Table 2). An additional proper rule can be used to obtain OWL's complete iff condition for `someValuesFrom`:[3]

```
IF   ?v owl:someValuesFrom ?w .
     ?v owl:onProperty ?p .
     ?u rdf:type ?v .
THEN ?u ?p  b  .
     b  rdf:type ?w .
```

This rule introduces a new blank node, which is denoted by b.

2.7 Example: Rules for Role-Value-Maps

A role-value-map [1] is a definition of a class in terms of the composite of certain properties p_i and q_j:

$$C = \{x : \forall y\ (x, y) \in p_1 \circ \ldots \circ p_m \Rightarrow (x, y) \in q_1 \circ \ldots \circ q_n\}$$

Role-value-maps arise in a number of applications and are difficult to combine with description logics. The inclusion $\subseteq$ can be written as: if $x \in C$ and $(x, y) \in p_1 \circ \ldots \circ p_m$, then $(x, y) \in q_1 \circ \ldots \circ q_n$. It is not difficult to see that this if condition can be expressed by a proper rule that introduces $n - 1$ new blank nodes.

[3] This proper rule induces entailment rule rdf-svx: see [12], Section 6.

2.8 Example: Airports and Map Points

For the final example we switch to the N3 syntax [2], which can be used to give a succinct representation of rules that introduce blank nodes. The following rule (by Mike Dean) from the DAML set of test rules[4] states that for each airport there is a map point with the same location, which is the underlying object of the airport, and has the appropriate label:

```
 { ?airport              a airport-ont:Airport;
                         airport-ont:latitude ?lat;
                         airport-ont:longitude ?lon;
                         airport-ont:name ?name }
 => { :layer map:object [a map:Point;
                         map:Location [a map:Location;
                           map:latitude ?lat; map:longitude ?lon];
                         map:underlyingObject ?airport;
                         map:label ?name].}.
```

This is the original version of the rule, which introduces two blank nodes. See [10] for an alternative representation of this rule in SWRL, which uses two `someValuesFrom` statements.

2.9 Overview and Discussion

In this paper we present basic definitions of an RDF-compatible semantics of rules and give a model-theoretic definition of *R-entailment*, which describes in a mathematical way what it means if a (source) RDF graph S entails a (target) RDF graph G with respect to a set of rules R. R-entailment is taken to be an extension of RDFS entailment, extending the meta-modeling capabilities of RDFS. R-entailment also incorporates pD^* entailment [12] and thereby part of OWL. Most examples in the DAML set of test rules (cf. Example 2.8) are covered by our approach; in this paper we do not consider the use of arithmetic, e.g. for conversion of different units. We prove a general completeness result for R-entailment, which shows that a restricted form of application of rules that introduce blank nodes is in general sufficient to determine R-entailment. For rules that do not introduce blank nodes, we prove that R-entailment and R-consistency are decidable and in PSPACE. For rules that do not introduce blank nodes and that satisfy a bound on the size of rule bodies, we prove that R-consistency is in P, that R-entailment is in NP, and that R-entailment is in P if target RDF graphs do not have blank nodes. These results are proved, as in [12], by showing that entailment rules can be used to form a partial closure graph H of the source graph S that is sufficient to decide consistency and entailment, that is polynomially bounded in size, and that can be computed in polynomial time if there is a bound on the size of rule bodies. S R-entails a target graph G if replacements can be made of the blank nodes in G that turn G into a subset of H; this can be checked with a non-deterministic guess, which is not needed if

[4] `http://www.daml.org/2003/06/ruletests/translation-1.n3`

G does not have blank nodes. S is R-inconsistent if H contains a set of triples that matches with an inconsistency rule.

The rules considered here are analogous to and simpler than datalog rules; triple patterns take the place of first-order atoms. Datalog and description logics do not use blank nodes. The PSPACE complexity of R-entailment for rules that do not introduce blank nodes compares favorably with the complexity of OWL DL (NEXPTIME-complete [9]) and datalog (EXPTIME-complete [4]); the latter results form points of comparison for combination formalisms (cf. Section 5). The data complexity of pure datalog is P (in fact P-complete [4]). If rules do not introduce blank nodes, then R-entailment (and R-consistency) with respect to a fixed set of rules is also in P. This compares favorably with the coNP-hard data complexity reported for systems that extend a description logic with datalog rules (see Section 5). The gain in complexity can be 'understood' in part by noting that part of OWL is captured as part of R-entailment with (pD^* entailment) rules by the results of [12].

3 Background

This section summarizes part of the material used from [13] [8] [12].

3.1 URI References, Blank Nodes, Literals

The symbol U denotes the set of *URI references*, B denotes the set of *blank nodes*, i.e. (existentially quantified) variables, and L denotes the set of *literals*, i.e. data values such as strings and integers. L is the union of the set L_p of *plain literals* and the set L_t of *typed literals*. A typed literal l consists of a lexical form s and a datatype URI t: we write l as a pair, $l = (s, t)$. The sets U, B, L_p and L_t are pairwise disjoint. A *vocabulary* is a subset of $U \cup L$. The symbol T denotes the set of all *RDF terms*, i.e. $T = U \cup B \cup L$. The notion 'RDF term' is used in the same way by [18].

3.2 Generalized RDF Graphs

A *generalized RDF graph* G [12] is defined to be a subset of the set

$$U \cup B \ \times \ U \cup B \ \times \ U \cup B \cup L\,. \tag{1}$$

The elements (s, p, o) of a generalized RDF graph are called *generalized RDF statements* or *generalized RDF triples*, which consist of a *subject*, a *predicate* (or *property*) and an *object*, respectively. We write triples as $s\,p\,o$. RDF graphs [13] [8] require properties to be URI references; generalized RDF graphs, which also allow properties to be blank nodes, were introduced in [12] to solve the problem that the standard set of entailment rules for RDFS [8] is incomplete.

If the projection mappings on the three factor sets of the product set given in (1) are denoted by π_i, the *set of RDF terms* of a generalized RDF graph G is

$$T(G) = \pi_1(G) \cup \pi_2(G) \cup \pi_3(G).$$

The set of *blank nodes* of a generalized RDF graph G is denoted by $bl(G) = T(G) \cap B$. The *vocabulary* of a generalized RDF graph G is the set $V(G) = T(G) \cap (U \cup L)$. Two generalized RDF graphs G and G' are *equivalent* if there is a bijection $f : T(G) \to T(G')$ such that $f(bl(G)) \subseteq bl(G')$, such that $f(v) = v$ for each $v \in V(G)$, and such that $s\,p\,o \in G$ if and only if $f(s)\,f(p)\,f(o) \in G'$.

A generalized RDF graph is *ground* if it has no blank nodes.

Given a partial function $h : B \rightharpoonup T$, an *instance* of a generalized RDF graph G is the generalized RDF graph G_h obtained from G by replacing the blank nodes v in G and the domain of h by $h(v)$.

Given a set S of generalized RDF graphs, a *merge* of S is a generalized RDF graph that is obtained by replacing the generalized graphs G in S with equivalent generalized graphs G' that do not share blank nodes and by taking the union of these generalized graphs G'. The merge of a set of generalized RDF graphs S is uniquely defined up to equivalence. A merge of S will be denoted by $M(S)$.

3.3 Simple Interpretations

A *simple interpretation* [8] I of a vocabulary V is a 6-tuple $I = (R_I, P_I, E_I, S_I, L_I, \mathrm{LV}_I)$, where R_I is a nonempty set, called the set of *resources*, P_I is the set of *properties*, LV_I is the set of *literal values*, which is a subset of R_I that contains at least all plain literals in V, and where E_I, S_I and L_I are functions: $E_I : P_I \to \mathcal{P}(R_I \times R_I)$, $S_I : V \cap U \to R_I \cup P_I$, $L_I : V \cap L_{\mathrm{t}} \to R_I$. Here $\mathcal{P}(X)$ denotes the power set of the set X, i.e. the set of all subsets of X. If I is a simple interpretation of a vocabulary V, then I also denotes a function with domain V, in the following way. For $l \in L_{\mathrm{p}} \cap V$, we have $I(l) = l \in \mathrm{LV}_I$. For $l \in L_{\mathrm{t}} \cap V$, $I(l) = L_I(l)$. For $a \in U \cap V$, $I(a) = S_I(a)$.

If $E = \; s\,p\,o$ is a ground triple, then a simple interpretation I of a vocabulary V is said to *satisfy* E if $s, p, o \in V, I(p) \in P_I$ and $(I(s), I(o)) \in E_I(I(p))$. If G is a ground RDF graph, then I satisfies G if I satisfies each triple $E \in G$.

Given a simple interpretation I and a partial function $A : B \rightharpoonup R_I$, a function I_A is defined that extends I by using A to give an interpretation of blank nodes in the domain of A. If $A(v)$ is defined for $v \in B$, then $I_A(v) = A(v)$. If G is any generalized RDF graph, then I *satisfies* G if I_A satisfies G for some function $A : bl(G) \to R_I$, i.e. if, for each triple $s\,p\,o \in G$, we have $I_A(p) \in P_I$ and $(I_A(s), I_A(o)) \in E_I(I_A(p))$. If I is a simple interpretation and S a set of generalized RDF graphs, then I satisfies S if I satisfies G for each G in S; it is not difficult to see that I satisfies S if and only if I satisfies $M(S)$.

3.4 RDFS Entailment and D^* Entailment

The notion of D^* entailment [11] [12] generalizes RDFS entailment [8] to include reasoning with datatypes from a given datatype map D [8]. If D contains only the standard datatype `rdf:XMLLiteral`, then D^* entailment coincides exactly with RDFS entailment. Table 1 lists a complete set of entailment rules for D^* entailment. In this table, prefixes such as `rdf:` are omitted from e.g. the URI `rdf:type`. These rules consist of the 18 rules defined in [8] for RDFS, with two differences that affect rules rdf2 and rdfs7. Rule rdfs7x corrects an error overlooked in [8] and [11] (see [12], Section 1.5): it differs from rule rdfs7 in that it can produce generalized RDF triples with blank nodes in predicate position when applied to ordinary RDF triples. To handle datatypes, rule rdf2 is replaced by the more general rule rdf2-D. Use is made of new blank nodes b_l, called *surrogate blank nodes*, allocated by rule lg ('literal generalization') to literals l. In rule rdfs1, b_l is a blank node allocated by rule lg to a plain literal $l \in L_{\mathrm{p}}$. In rule rdf2-D, b_l is a blank node allocated by rule lg to a well-typed D-literal l: $l \in L_D^+$. The only inconsistencies that can arise for the D^* semantics are D-clashes, which generalize XML-clashes [8]: given a datatype map D, a *D-clash* is a triple b `type` `Literal`, where b is a blank node allocated by rule lg to an ill-typed D-literal l: $l \in L_D - L_D^+$.

Table 1. D^* entailment rules [12]

	If G contains	where	then add to G
lg	$v\,p\,l$	$l \in L$	$v\,p\,b_l$
gl	$v\,p\,b_l$	$l \in L$	$v\,p\,l$
rdf1	$v\,p\,w$		p `type` `Property`
rdf2-D	$v\,p\,l$	$l = (s, a) \in L_D^+$	b_l `type` a
rdfs1	$v\,p\,l$	$l \in L_{\mathrm{p}}$	b_l `type` `Literal`
rdfs2	p `domain` u		
	$v\,p\,w$		v `type` u
rdfs3	p `range` u		
	$v\,p\,w$	$w \in U \cup B$	w `type` u
rdfs4a	$v\,p\,w$		v `type` `Resource`
rdfs4b	$v\,p\,w$	$w \in U \cup B$	w `type` `Resource`
rdfs5	v `subPropertyOf` w		
	w `subPropertyOf` u		v `subPropertyOf` u
rdfs6	v `type` `Property`		v `subPropertyOf` v
rdfs7x	p `subPropertyOf` q		
	$v\,p\,w$	$q \in U \cup B$	$v\,q\,w$
rdfs8	v `type` `Class`		v `subClassOf` `Resource`
rdfs9	v `subClassOf` w		
	u `type` v		u `type` w
rdfs10	v `type` `Class`		v `subClassOf` v
rdfs11	v `subClassOf` w		
	w `subClassOf` u		v `subClassOf` u
rdfs12	v `type` `Container-`		
	`MembershipProperty`		v `subPropertyOf` `member`
rdfs13	v `type` `Datatype`		v `subClassOf` `Literal`

D^* entailment is weaker than D-entailment [8]. For example, with regard to the XML Schema datatype `xsd:boolean`, the three triples $a\ p$ `true`, $a\ p$ `false`, b `type` `boolean` D-entail the triple $a\ p\ b$, but this is not a D^* entailment. It is possible to 'recover' certain missing D-entailments by using meta-modeling statements. See [12], Section 1.7, for an example that uses the pD^* semantics. It is also possible to use rules and R-entailment for this purpose.

3.5 *pD** Entailment

The notion of *pD** entailment was introduced in [11] [12] as a variant of OWL entailment, weakening OWL Full (see Section 1). The 18 D^* entailment rules of Table 1 become complete for *pD** entailment by adding the 23 P-entailment rules of Table 2 [12]. In addition to rule rdfp15, there is a second entailment rule for `someValuesFrom`, called rdf-svx (see 2.6 and [12], Section 6) and analogous to rule rdfp16 for `allValuesFrom`. This rule introduces a new blank node and can be added for R-entailment; it is not supported by the *pD** semantics because the proof of decidability given in [12] does not extend to the use of this rule. For the *pD** semantics, in addition to D-clashes, another type of inconsistency is formed by P-clashes: a *P-clash* is either a combination of two triples of the form v `differentFrom` w, v `sameAs` w, or a combination of three triples of the form v `disjointWith` w, u `type` v, u `type` w.

4 Rules and *R*-Entailment

In this section we define a model-theoretic semantics integrating RDF with rules and extend the completeness, decidability and complexity results obtained in [12]. Starting in 4.4, we make the combination with the *pD** semantics for OWL [12]. It is also possible to start from simple entailment or RDFS entailment, which would simplify some results; for example, P-entailment rules could be subsumed under R-entailment rules. However, an advantage of the chosen setup is that a closer connection is obtained to the semantic conditions of OWL.

4.1 Definition (Rule Graph)

In our definition of rules we use a set of *rule variables* X which is assumed to be disjoint from the set T of RDF terms: $X \cap T = \emptyset$. Rule variables will also briefly be called *variables.* A *rule graph* G is defined to be a set of *triple patterns*, i.e. a subset of the product set

$$U \cup B \cup X \ \times \ U \cup B \cup X \ \times \ U \cup B \cup L \cup X \, . \tag{2}$$

Given a rule graph G, we denote the union of the projection mappings π_i on the three factor sets of the product set given in (2), applied to G, by

$$\pi(G) = \pi_1(G) \cup \pi_2(G) \cup \pi_3(G).$$

The set of variables of a rule graph G is denoted by $\mathrm{var}(G) = \pi(G) \cap X$, the set of blank nodes of G by $bl(G) = \pi(G) \cap B$, and the vocabulary of G by $V(G) = \pi(G) \cap (U \cup L)$.

Two kinds of *instances* of rule graphs are defined, with respect to variables and with respect to blank nodes. Given a rule graph G and a function φ : $\mathrm{var}(G) \to T$, the instance of G with respect to φ is the generalized RDF graph G_φ obtained from G by replacing the variables $v \in \mathrm{var}(G)$ by $\varphi(v)$. Similarly,

Table 2. P-entailment rules [12]

	If G contains	where	then add to G
rdfp1	p type FunctionalProperty		
	$u\,p\,v$		
	$u\,p\,w$	$v \in U \cup B$	v sameAs w
rdfp2	p type Inverse-FunctionalProperty		
	$u\,p\,w$		
	$v\,p\,w$		u sameAs v
rdfp3	p type SymmetricProperty		
	$v\,p\,w$	$w \in U \cup B$	$w\,p\,v$
rdfp4	p type TransitiveProperty		
	$u\,p\,v$		
	$v\,p\,w$		$u\,p\,w$
rdfp5a	$v\,p\,w$		v sameAs v
rdfp5b	$v\,p\,w$	$w \in U \cup B$	w sameAs w
rdfp6	v sameAs w	$w \in U \cup B$	w sameAs v
rdfp7	u sameAs v		
	v sameAs w		u sameAs w
rdfp8ax	p inverseOf q		
	$v\,p\,w$	$w, q \in U \cup B$	$w\,q\,v$
rdfp8bx	p inverseOf q		
	$v\,q\,w$	$w \in U \cup B$	$w\,p\,v$
rdfp9	v type Class		
	v sameAs w		v subClassOf w
rdfp10	p type Property		
	p sameAs q		p subPropertyOf q
rdfp11	$u\,p\,v$		
	u sameAs u'		
	v sameAs v'	$u' \in U \cup B$	$u'\,p\,v'$
rdfp12a	v equivalentClass w		v subClassOf w
rdfp12b	v equivalentClass w	$w \in U \cup B$	w subClassOf v
rdfp12c	v subClassOf w		
	w subClassOf v		v equivalentClass w
rdfp13a	v equivalentProperty w		v subPropertyOf w
rdfp13b	v equivalentProperty w	$w \in U \cup B$	w subPropertyOf v
rdfp13c	v subPropertyOf w		
	w subPropertyOf v		v equivalentProperty w
rdfp14a	v hasValue w		
	v onProperty p		
	$u\,p\,w$		u type v
rdfp14bx	v hasValue w		
	v onProperty p		
	u type v	$p \in U \cup B$	$u\,p\,w$
rdfp15	v someValuesFrom w		
	v onProperty p		
	$u\,p\,x$		
	x type w		u type v
rdfp16	v allValuesFrom w		
	v onProperty p		
	u type v		
	$u\,p\,x$	$x \in U \cup B$	x type w

given a rule graph G and a partial function $h : B \rightharpoonup T$, the instance of G with respect to h is the rule graph G_h obtained from G by replacing the blank nodes v in G and the domain of h by $h(v)$. Given a rule graph G combined with $h : B \rightharpoonup T$ and $\varphi : \mathrm{var}(X) \to T$, $G_{h\varphi}$ is the instance of G_h with respect to φ.

4.2 Definition (Rule)

A *rule* is defined as a pair of rule graphs $\rho = (\rho_\mathrm{l}, \rho_\mathrm{r})$ that are not both empty and that satisfy the conditions $\mathrm{var}(\rho_\mathrm{r}) \subseteq \mathrm{var}(\rho_\mathrm{l})$ and $bl(\rho_\mathrm{l}) = \emptyset$. If $\rho = (\rho_\mathrm{l}, \rho_\mathrm{r})$ is a rule, then ρ_l is called its left-hand side or body, and ρ_r is called its right-hand side or head. Given a rule ρ, the set of variables of ρ is denoted by $\mathrm{var}(\rho) = \mathrm{var}(\rho_\mathrm{l})$, the set of blank nodes of ρ by $bl(\rho) = bl(\rho_\mathrm{r})$, and the vocabulary of ρ by $V(\rho) = V(\rho_\mathrm{l}) \cup V(\rho_\mathrm{r})$. If R is a set of rules, then $V(R) = \bigcup_{\rho \in R} V(\rho)$. A rule ρ is said to *introduce blank nodes* if $bl(\rho) \neq \emptyset$. A rule ρ is called *finite* if both ρ_l and ρ_r are finite. As was already mentioned in 2.1, a rule ρ is called a *proper rule* if ρ_l and ρ_r are both nonempty, an *axiom rule* if $\rho_\mathrm{l} = \emptyset$ and an *inconsistency rule* if $\rho_\mathrm{r} = \emptyset$.

4.3 Definition (Satisfaction)

Given a simple interpretation I (see 3.3) and a partial function $Z : X \rightharpoonup R_I$, a function I_Z is defined that extends I by setting $I_Z(v) = Z(v)$ if $Z(v)$ is defined for $v \in X$. If, in addition, a partial function $A : B \rightharpoonup R_I$ is given, a function I_{ZA} is defined that extends I_Z further by setting $I_{ZA}(v) = A(v)$ if $A(v)$ is defined for $v \in B$. If G is any rule graph, I a simple interpretation and $Z : \mathrm{var}(G) \to R_I$ a function, then I_Z is said to *satisfy* G if there is a function $A : bl(G) \to R_I$ such that for each triple pattern $s\,p\,o \in G$ we have $I_{ZA}(p) \in P_I$ and $(I_{ZA}(s), I_{ZA}(o)) \in E_I(I_{ZA}(p))$.

A simple interpretation I *satisfies a rule* ρ if $I(p) \in P_I$ for each $p \in U$ that appears in predicate position in a triple pattern in ρ_l or ρ_r, and if I also satisfies the following conditions:

- If ρ is an axiom rule, then I satisfies ρ_r.
- If ρ is a proper rule and $Z : \mathrm{var}(\rho) \to R_I$ a function such that I_Z satisfies ρ_l, then I_Z also satisfies ρ_r.
- If ρ is an inconsistency rule, then there is no function $Z : \mathrm{var}(\rho) \to R_I$ such that I_Z satisfies ρ_l.

4.4 Definition (*R*-Interpretations, *R*-Entailment)

If R is a set of rules and D a datatype map, an *R-interpretation* of a vocabulary V is a *pD** interpretation [12] of $V \cup V(R)$ that satisfies each rule $\rho \in R$.

Given a set of rules R, a set S of generalized RDF graphs is called *R-consistent* if there is an R-interpretation that satisfies S.

Given a set of rules R, the set of *R-axiomatic triples* is the generalized RDF graph obtained by taking the merge of the generalized RDF graphs ρ_r where ρ

ranges over the axiom rules in R, by adding the triples p `type Property` for each $p \in U$ appearing in predicate position in a triple pattern in a body or head of a rule $\rho \in R$, and by adding the triples v `type Resource` for each $v \in U \cap V(R)$.

Given a set of rules R, an *R-clash* is a generalized RDF graph that forms an instance $\rho_{l\varphi}$ of the body ρ_l of an inconsistency rule $\rho \in R$ for a function $\varphi : \mathrm{var}(\rho) \rightarrow T$.

Table 3. Three R-entailment rules (see 4.5 for the R-entailment rules rdfρ)

	If R contains	where	then add to G
lg-R	$v\,p\,l$	$l \in L$	b_l `type` `Resource`
rdf2-DR	$v\,p\,l$	$l = (s, a) \in L_D^+$	b_l `type` a
rdfs1-R	$v\,p\,l$	$l \in L_\mathrm{p}$	b_l `type` `Literal`

Given a datatype map D and a set of rules R, a set S of generalized RDF graphs *R-entails* a generalized RDF graph G if each R-interpretation I that satisfies S also satisfies G. In this case, we write $S \models_R G$.

4.5 Definition (*R*-Entailment Rules)

See Table 3 for the definition of the R-entailment rules lg-R, rdf2-DR and rdfs1-R, given a set of rules R and a datatype map D. In this table the phrase "If R contains $v\,p\,l$" stands for "If R contains a rule ρ such that ρ_l or ρ_r contains the triple pattern $v\,p\,l$". These rules are similar to rules lg, rdf2-D, rdfs1: see 3.4. For each proper rule $\rho \in R$, the R-entailment rules also include an entailment rule **rdf**ρ, defined in the following way. If a given generalized RDF graph G contains the triples in the instance $\rho_{l\varphi}$ of ρ_l for a function $\varphi : \mathrm{var}(\rho) \rightarrow T(G)$, where $\varphi(x) \in U \cup B$ for each $x \in (\pi_1(\rho_r) \cup \pi_2(\rho_r)) \cap X$, then rdf$\rho$ prescribes the following two steps:

- Replace the rule graph ρ_r with the instance ρ_{rh} of ρ_r by replacing the blank nodes b in ρ_r with blank nodes $h(b)$ that do not appear in G; here $h : bl(\rho_r) \rightarrow B$ is assumed to be an injective function.
- Add the triples in the instance $\rho_{rh\varphi}$ of ρ_{rh} to G.[5]

4.6 Definition (Partial and Full *R*-Closures)

The rule system described in this paper is declarative; the entailment rules of Tables 1 and 2 and the preceding definition can be applied in any order (cf. Theorem 4.10). However, in order to prove decidability, we consider a special

[5] Note, as an example, that the syntactic conditions imposed in Table 2 on the *pD** entailment rules (e.g. the condition $v \in U \cup B$ of rule rdfp1) are realized exactly by the general syntactic condition of the entailment rules rdfρ that arise from the corresponding proper rules.

way of applying the entailment rules. Suppose that D is a datatype map, R a set of rules and G a generalized RDF graph. Suppose that K is a nonempty subset of the positive integers $\{1, 2, ...\}$ chosen in such a way that for each container membership property [8] `rdf:_`$i \in V(G) \cup V(R)$ we have $i \in K$. The *partial R-closure* G_{RK} of G is defined in the following way, refining the definitions of partial D^* and pD^* closure [12]. In the first step, all RDF, RDFS, D-axiomatic triples and P-axiomatic triples [12] are added to G, except for the axiomatic triples that include `rdf:_`i such that $i \notin K$. Moreover, the R-axiomatic triples are added in such a way that G does not contain any blank node that appears in the merge of the generalized RDF graphs ρ_{r}, where ρ ranges over the axiom rules in R. In the next step, rules lg and lg-R are applied to each triple in G that contains a literal and to each triple pattern (in a rule in R) that contains a literal that does not appear in G, in such a way that distinct well-typed D-literals with the same value are associated with the same surrogate blank node b_l. Then, rules rdf2-D and rdfs1 are applied to each triple in G containing a well-typed D-literal or a plain literal, respectively. Next, rules rdf2-DR and rdfs1-R are applied to each triple pattern that appears in a rule in R and that contains a well-typed D-literal or plain literal that has not yet been handled by rules rdf2-D and rdfs1, respectively. The generalized RDF graph that has now been obtained is denoted by G_0. The partial R-closure G_{RK} is defined in a recursive way: $G_{RK} = \bigcup_{n=0}^{\infty} G_n$. Suppose that G_n has been defined. Then, G_{n+1} is the generalized RDF graph that extends G_n by making all possible applications to triples in G_n for each of the remaining D^* entailment rules, P-entailment rules and rule lg; moreover, for each entailment rule rdfρ arising from a proper rule $\rho \in R$, one application is made for each instance $\rho_{\mathrm{l}\varphi}$ of ρ_{l} for a function $\varphi : \mathrm{var}(\rho) \to T(G_n)$, where $\varphi(x) \in U \cup B$ for each $x \in (\pi_1(\rho_{\mathrm{r}}) \cup \pi_2(\rho_{\mathrm{r}})) \cap X$, such that $\rho_{\mathrm{l}\varphi} \subseteq G_n$, and, if ρ introduces blank nodes, such that there is no function $h : bl(\rho_{\mathrm{r}}) \to T(G_n)$ such that $\rho_{\mathrm{r}h\varphi} \subseteq G_n$.[6] This completes the definition of the partial closure G_{RK}. Theorem 4.11 shows that this restricted use of proper rules that introduce blank nodes is in general sufficient to determine R-entailment. It should be noted that applications of rule lg in the last, recursive step do not lead to new blank nodes b_l. The *full R-closure* G_R of G is defined by taking $G_R = G_{R\{1,2,...\}}$.

4.7 Lemma

Let D be a finite datatype map. If R is a finite set of finite rules that do not introduce blank nodes and G a finite generalized RDF graph, then each partial R-closure G_{RK} of G is finite for K finite, and of size bounded by a polynomial in $|G|$, $|K|$ and $\sum_{\rho \in R}(|\rho_l| + |\rho_r|)$. If there is a bound on the size of rule bodies (e.g. if R is fixed), then a partial R-closure of a finite generalized RDF graph G can be computed in polynomial time, and it is possible to determine in polynomial time if a finite generalized RDF graph contains an R-clash.

[6] For example: for the proper rule for `someValuesFrom` (see 2.6) no application needs to be made if (the term matched by) `?`u is already `?`p-related to a term of type `?`w.

Proof. This can be proved by refining the proof of Lemma 4.8 in [12]. □

In the remainder of this section D is a given datatype map.

4.8 Definition (*R*-Herbrand Interpretation)

Given a set of rules R and a generalized RDF graph G, an *R-Herbrand interpretation* $R_K(G)$ is defined in a similar way to a D^* Herbrand interpretation $S_K(G)$ (see [12], Definition 4.9). The only difference is that, throughout the definition, $V(G_\mathrm{s})$ is replaced by $V(G_R) \cup V(R)$, $bl(G_s)$ by $bl(G_R)$ and $G_{\mathrm{s},K}$ by G_{RK}.

4.9 *R*-Satisfaction Lemma

Let R be a set of rules and G a generalized RDF graph. If the partial R-closure G_{RK} of G does not contain an R-clash, P-clash or D-clash, then $R_K(G)$ is an R-interpretation that satisfies G_{RK}.

Proof. This can be proved by extending the proofs of the D^* and pD^* satisfaction lemmas (Lemmas 4.10 and 5.10 in [12]). □

4.10 Theorem (*R*-Entailment Lemma)

Let R be a set of rules, S a set of generalized RDF graphs and G a generalized RDF graph. Then, $S \models_R G$ if and only if there is a generalized RDF graph H that can be derived from $M(S)$ merged with RDF, RDFS, D-axiomatic triples, P-axiomatic triples and R-axiomatic triples, by application of D^* entailment rules, P-entailment rules and R-entailment rules, and that either contains an instance of G as a subset or contains an R-clash, P-clash or D-clash.

4.11 Theorem (*R*-Entailment Lemma: Alternative Statement)

Let R be a set of rules, S a set of generalized RDF graphs and G a generalized RDF graph. Let H be a partial R-closure $M(S)_{RK}$ of $M(S)$ and suppose that $i \in K$ for each `rdf:_`*i* $\in V(G)$*. Then, $S \models_R G$ if and only if either H contains an instance of G as a subset or H contains an R-clash, P-clash or D-clash.*

4.12 Corollary

If D is finite, then the R-entailment relation $S \models_R G$ between finite sets S of finite generalized RDF graphs, finite sets R of finite rules that do not introduce blank nodes, and finite generalized RDF graphs G is decidable and in PSPACE. If there is a bound on the size of rule bodies, then this problem is in NP, and in P if G is ground.

4.13 Theorem (R-Consistency Lemma)

Let R be a set of rules, S a set of generalized RDF graphs and H a partial R-closure of $M(S)$. Then, S is R-consistent if and only if H does not contain an R-clash, P-clash or D-clash.

4.14 Corollary

If D is finite, then the problem to determine if a finite set of finite generalized RDF graphs is R-consistent with respect to a finite set R of finite rules that do not introduce blank nodes is decidable and in PSPACE, and in P if there is a bound on the size of rule bodies.

Proof. The proof of Theorems 4.10, 4.11 and 4.13 builds further on the proof of Theorems 5.11, 5.12 and 5.15 of [12]. The proof of the corollaries is based on the computation of a partial R-closure $H = M(S)_{RK}$, following the steps described in Definition 4.6. Lemma 4.7 and its proof show that this computation can be done in polynomial space and that it can be done in polynomial time if there is a bound on the size of rule bodies. For Corollary 4.12, a non-deterministic guess is used of an instance function h such that $G_h \subseteq H$ (by Savitch's theorem, NPSPACE=PSPACE); this is not needed if G is ground. □

If R is allowed to vary without restrictions, then R-consistency is NP-hard, even if only inconsistency rules are used. This can be shown with a transformation from the standard NP-complete problem conjunctive boolean query.

5 Related Work

SWRL combines ontologies with rules by extending OWL DL with datalog rules, i.e. function-free Horn rules [10]. Rules may include DL atoms and `sameAs` and `differentFrom` statements; unlike the approaches that will be mentioned next, in SWRL rules cannot include non-DL atoms. Consistency and entailment for SWRL are undecidable, by a reduction from the domino problem [10]. For SWRL a prototype implementation has been described which makes use of first-order reasoning, necessarily without guarantee of completeness [10].

Several formalisms have been investigated which impose restrictions on the extension of a description logic with datalog rules in order to obtain decidable inference problems (cf. 2.9). $\mathcal{AL}$-log [5] allows the addition to rule bodies of atoms that specify that a constant or variable belongs to a class defined in the DL $\mathcal{ALC}$. The resulting combination is shown to be decidable and in NEXPTIME by using a tableau algorithm in combination with constrained SLD-derivation. As an example, the standard NP-complete problem graph 3-colorability is encoded with a knowledge base [5], which shows that data complexity of $\mathcal{AL}$-log is coNP-hard [3]. A similar encoding cannot be used with the approach described in this paper because it uses the union construct to express that each node in the input graph belongs to one of three colors (cf. 2.4).

The CARIN approach [14] includes a more powerful description logic ($\mathcal{ALCNR}$) and has more possibilities allowing concepts and roles in datalog rules. Much attention is devoted to the "existential entailment problem": for two or more Horn rules, it may occur that either the antecedents of one rule are satisfied or the antecedents of another rule are satisfied, while it is not known which of these possibilities occurs, so that all possibilities need to be considered, thus increasing the computational complexity. This contrasts with traditional Horn systems and the approach described in this paper, where the application of rules can be considered in isolation. In [14], several restrictions are discussed which guarantee decidability, leading to coNP-complete data complexity, i.e. complexity of inference in the number of ground facts.

One of these restrictions requires that each variable in a Horn rule appears in a non-DL-atom in the body of the rule. This "DL-safety" condition is also used to achieve decidability in [15] and [17], with formalisms that include increasingly expressive DLs. According to [15], DL-safety amounts to the condition that "the identity of all objects is known". R-entailment, on the other hand, allows variables in bodies of rules to be matched with blank nodes.

DLP captures part of OWL DL with datalog rules [7]. Datalog is EXPTIME-complete [4]. DLP does not include the same expressivity as R-entailment. For example, `sameAs` and `FunctionalProperty` are not supported by DLP. Unlike the R-semantics, the formalisms mentioned in this section do not include the full semantics of RDF and meta-modeling capabilities as provided by RDFS. For example, DLP is restricted to the "DAML+OIL subset of RDFS" [7].

This paper uses a simple, uniform approach which, unlike in e.g. [5], does not involve a hybrid system that incorporates two distinct reasoning paradigms. Just like RDF and OWL and unlike [5] [14] [17], this paper does not make a unique names assumption. To recapitulate, DLP seems to be the approach that is most similar to R-entailment; compared with other formalisms that combine ontologies and rules, R-entailment does not include the same expressivity but leads to improved complexity and adds meta-modeling expressivity.

6 Conclusion

In this paper we have defined a semantic extension of RDF that incorporates rules. We started from an abstract syntax that considers a rule as a pair of rule graphs which extend RDF graphs with the possibility to include rule variables. For a set of rules R, we defined a general notion of R-entailment, which extends RDFS and its meta-modeling capabilities. R-entailment also extends a decidable part of OWL that weakens D-entailment and OWL Full. We proved a general completeness result for R-entailment, which shows that a restricted form of application of rules that introduce blank nodes is in general sufficient to determine R-entailment. For rules that do not introduce blank nodes, we proved that R-entailment and R-consistency are decidable and in PSPACE. For rules that do not introduce blank nodes and that satisfy a bound on the size of rule bodies, we proved that R-consistency is in P, that R-entailment is in NP, and that R-entailment is in P if the target RDF graph is ground.

Acknowledgment. Many thanks to Warner ten Kate, Jan Korst and the anonymous reviewers for their useful comments about the manuscript.

References

1. F. Baader et al. (Eds.), *The Description Logic Handbook*, Cambridge, 2003.
2. T. Berners-Lee, S. Hawke, D. Connolly, Semantic Web Tutorial Using N3, May 2004, `http://www.w3.org/2000/10/swap/doc/`
3. M. Cadoli, L. Palopoli, M. Lenzerini, Datalog and Description Logics: Expressive Power, Proceedings of the 6th International Workshop on Database Programming Languages (DBPL1997), pp. 281-298, 1997.
4. E. Dantsin, T. Eiter, G. Gottlob, A. Voronkov, Complexity and Expressive Power of Logic Programming, ACM Computing Surveys, 33 (2001) 374-425.
5. F.M. Donini, M. Lenzerini, D. Nardi, A. Schaerf, $\mathcal{AL}$-log: Integrating Datalog and Description Logics, Journal of Intelligent Information Systems, 10 (1998) 227-252.
6. J. Grant, D. Beckett (Eds.), RDF Test Cases, W3C Recommendation, 10 February 2004, `http://www.w3.org/TR/2004/REC-rdf-testcases-20040210/`
7. B. Grosof, I. Horrocks, R. Volz, S. Decker, Description Logic Programs: Combining Logic Programs with Description Logic, Proceedings of the 12th International Conference on the World Wide Web (WWW2003), pp. 48-57, 2003.
8. P. Hayes (Ed.), RDF Semantics, W3C Recommendation, 10 February 2004, `http://www.w3.org/TR/2004/REC-rdf-mt-20040210/`
9. I. Horrocks, P.F. Patel-Schneider, Reducing OWL Entailment to Description Logic Satisfiability, Journal of Web Semantics 1 (2004) 345-357.
10. I. Horrocks, P.F. Patel-Schneider, S. Bechhofer, D. Tsarkov, OWL Rules: A Proposal and Prototype Implementation, J. Web Semantics 3 (2005) 23-40.
11. H.J. ter Horst, Extending the RDFS Entailment Lemma, Proceedings 3rd Int. Semantic Web Conference (ISWC2004), Springer LNCS 3298, pp. 77-91, 2004.
12. H.J. ter Horst, Completeness, Decidability and Complexity of Entailment for RDF Schema and a Semantic Extension Involving the OWL Vocabulary, Revised and extended version of [11], Journal of Web Semantics 3 (2005) 79-115.
13. G. Klyne, J. Carroll (Eds.), Resource Description Framework (RDF): Concepts and Abstract Syntax, W3C Recommendation, 10 February 2004, `http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/`
14. A.Y. Levy, M.-C. Rousset, Combining Horn Rules and Description Logics in CARIN, Artificial Intelligence 104 (1998) 165-209.
15. B. Motik, U. Sattler, R. Studer, Query Answering for OWL-DL with Rules, Journal of Web Semantics 3 (2005) 41-60.
16. P.F. Patel-Schneider, P. Hayes, I. Horrocks (Eds.), OWL Web Ontology Language Semantics and Abstract Syntax, W3C Recommendation, 10 February 2004, `http://www.w3.org/TR/2004/REC-owl-semantics-20040210/`
17. R. Rosati, On the Decidability and Complexity of Integrating Ontologies and Rules, Journal of Web Semantics 3 (2005) 61-73.
18. RDF Data Access Working Group, W3C, `http://www.w3.org/2001/sw/DataAccess/`