Extending the RDFS Entailment Lemma

Herman J. ter Horst

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

Abstract. We complement the RDF semantics specification of the W3C by proving decidability of RDFS entailment. Furthermore, we show completeness and decidability of entailment for RDFS extended with datatypes and a property-related subset of OWL.
The RDF semantics specification provides a complete set of entailment rules for reasoning with RDFS, but does not prove decidability of RDFS entailment: the closure graphs used in the completeness proof are infinite for finite RDF graphs. We define partial closure graphs, which can be taken to be finite for finite RDF graphs, which can be computed in polynomial time, and which are sufficient to decide RDFS entailment.
We consider the extension of RDFS with datatypes and a property-related fragment of OWL: `FunctionalProperty`, `InverseFunctionalProperty`, `sameAs`, `SymmetricProperty`, `TransitiveProperty`, and `inverseOf`. In order to obtain a complete set of simple entailment rules, the semantics that we use for these extensions is in line with the 'if-semantics' of RDFS, and weaker than the 'iff-semantics' defining D-entailment and OWL (DL or Full) entailment. Classes can be used as instances, the use of `FunctionalProperty` and `TransitiveProperty` is not restricted to obtain decidability, and a partial closure that is sufficient for deciding entailment can be computed in polynomial time.

1 Introduction

The language RDF (Resource Description Framework) plays a foundational role in the W3C's Semantic Web vision. The RDF semantics specification [2] provides a model-theoretic description of the semantics of RDF and RDFS (RDF Schema). This specification also contains the RDFS entailment lemma, describing a complete set of entailment rules, used to implement RDF reasoners. The RDFS entailment lemma moreover makes clear to people, in syntactic terms, what RDFS entailment is. This paper extends the RDFS entailment lemma in three directions: decidability of entailment, reasoning with datatypes, and reasoning with property-related vocabulary such as `owl:FunctionalProperty`.

Decidability of RDFS entailment is a fundamental issue, which has not yet been settled. The existence of a complete set of entailment rules does not imply decidability of entailment; compare first-order logic, with Gödel's completeness theorem and Church's theorem showing undecidability. The completeness proof for RDFS [2] makes use of closure graphs which are infinite for finite graphs, in view of the container membership properties `rdf:_i`. The proof shows that

S.A. McIlraith et al. (Eds.): ISWC 2004, LNCS 3298, pp. 77–91, 2004.

in order to decide whether a finite graph H rdfs-entails a finite graph G, it is sufficient to check whether a finite subgraph of the closure of H simply entails G. However, since the closure of H is infinite, and therefore the number of candidate subgraphs is infinite, this is not a finite algorithm. In this paper we define partial closure graphs, which can be taken to be finite for finite graphs and which can be computed in polynomial time, and prove that these can be used to decide rdfs-entailment. Comparing the closure graphs described in [2], to decide whether a graph H rdfs-entails a graph G, it is allowed to omit the addition to H of the axiomatic triples for the URIs `rdf:_i` that do not appear in either H or G, as long as these axiomatic triples are added to H for at least one URI `rdf:_i`.

The RDF semantics specification [2] defines a notion of reasoning with datatypes, called D-entailment, for which no completeness or decidability result is known. We show that a completeness and decidability result can be obtained for a somewhat weaker semantics of datatypes, by making a relatively small modification to the entailment rules: rule rdf2 is replaced by a rule rdf2-D, with the same effect not only for the datatype `XMLLiteral` but for all datatypes in a datatype map D. This result requires a mild assumption, that datatype maps are 'discriminating', as will be explained below.

For OWL (Web Ontology Language) [5] no complete set of entailment rules is known. OWL Full entailment is known to be undecidable. For OWL DL, restrictions were imposed on the use of the language to obtain decidability. OWL DL entailment and OWL Lite entailment are known to be in `NEXPTIME` and `EXPTIME`, respectively [3]. Several people have stated that `owl:FunctionalProperty` is the single most useful element of the OWL vocabulary. We consider reasoning with RDFS extended with this and other property-related vocabulary: `InverseFunctionalProperty`, `sameAs`, `SymmetricProperty`, `TransitiveProperty`, and `inverseOf`. We show that 12 simple entailment rules, in combination with the 18 entailment rules for RDFS, form a complete set. Just like for the notion of datatype reasoning that we use, the semantics considered is in line with the 'if-semantics' of RDFS, rather than the stronger 'iff-semantics' underlying D-entailment and OWL (DL or Full) entailment. In a similar way as for RDFS, we prove that entailment can be decided with a partial closure graph which can be computed in polynomial time, and that the problem to decide whether a finite graph H entails a finite graph G is in `NP`, and in `P` when G has no blank nodes. No restrictions are imposed: for example, classes can be used as instances, and `FunctionalProperty` and `TransitiveProperty` can be freely combined.

In this paper, the descriptions of the proofs are largely restricted to places where the finiteness of a partial closure or the considered semantic extensions of RDFS lead to a difference with the proof of the RDFS entailment lemma.[1]

2 RDF Graphs and Simple Entailment

This section summarizes terminology, introduces notations, and recalls the interpolation lemma which characterizes simple entailment [2] [4].

[1] A version of this paper with complete proofs is available on request.

2.1 URI references, blank nodes, literals. Let U denote the set of *URI references* and B the set of *blank nodes*, which is assumed to be infinite. Let L be the set of *literals*; 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 shall write l as a pair, $l = (s, t)$. The sets U, B, L_{p} and L_{t} are pairwise disjoint. A *name* is an element of $U \cup L$, and a *vocabulary* is a subset of $U \cup L$. RDF has a special datatype URI, called `rdf:XMLLiteral`, which will also be written as X. An *XML literal* is a typed literal of the form (s, X).

2.2 RDF graphs. An *(RDF) graph* G is defined to be a subset of the set

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

The elements (s, p, o) of an RDF graph are called *RDF triples*, which consist of a *subject*, a *predicate* (or *property*), and an *object*, respectively. We shall write triples as $s\,p\,o$ and introduce one-letter abbreviations for several standard URI references (see Tables 1 and 4), to obtain a shortening of definitions and proofs. The notation can be viewed as an abbreviation of the N-Triples notation [1].[2]

Denoting the projection mappings on the three factor sets of the product set given in (1) by π_i, the *set of nodes* of an RDF graph G is

$$nd(G) = \pi_1(G) \cup \pi_3(G),$$

which is a subset of $U \cup B \cup L$. The set of *blank nodes* of G is denoted by

$$bl(G) = nd(G) \cap B.$$

The *vocabulary of a graph* G, which will be denoted by $V(G)$, is the set of names that occur as subject, predicate or object of a triple in G.

Two RDF graphs G and G' are *equivalent* if there is a bijection $f : nd(G) \to nd(G')$ such that $f(bl(G)) \subset bl(G')$, such that $f(v) = v$ for each $v \in nd(G) \cap (U \cup L)$, and such that $s\,p\,o \in G$ if and only if $f(s)\,p\,f(o) \in G'$.

A *subgraph* of an RDF graph is a subset of the graph.

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

Given a partial function $h : B \rightharpoonup U \cup B \cup L$, an *instance* of a graph G is a graph obtained from G by replacing some (or all) blank nodes v in G by $h(v)$.

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

2.3 Simple interpretations. A *simple interpretation* I of a vocabulary V is a 6-tuple $I = (R_I, P_I, E_I, S_I, L_I, LV_I)$, 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:

$$E_I : \ P_I \ \to \ \mathcal{P}(R_I \times R_I),$$

[2] As in the expression $s\,p\,o \in G$, where G is an RDF graph, the context will always make clear what the triple is.

$$S_I : V \cap U \rightarrow R_I \cup P_I,$$
$$L_I : V \cap L_{\mathrm{t}} \rightarrow R_I.$$

Here $\mathcal{P}(X)$ denotes the power set of the set X, that is, the set of all subsets of X. The function E_I defines the *extension* of a property as a set of pairs of resources. The functions S_I and L_I define the interpretation of URI references and typed literals, respectively.

If I is a simple interpretation of a vocabulary V, then I also denotes a function with domain V, in the following way. For plain literals $l \in L_{\mathrm{p}} \cap V$, we have $I(l) = l \in LV_I$. For typed literals $l \in L_{\mathrm{t}} \cap V$, $I(l) = L_I(l)$. For URI references $a \in U \cap V$, $I(a) = S_I(a)$.

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 RDF graph, then I satisfies G if I_A satisfies G for some partial function $A : B \rightharpoonup R_I$, that is, if

$$(I_A(s), I_A(o)) \in E_I(I(p))$$

for each triple $s\,p\,o \in G$. If I is a simple interpretation and S a set of 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 $M(S)$.

2.4 Simple entailment. A set S of graphs *simply entails* a graph G if each simple interpretation that satisfies S also satisfies G. In this case we shall write

$$S \models G.$$

2.5 Interpolation lemma. *If S is a set of RDF graphs and G is an RDF graph, then $S \models G$ if and only if a subgraph of $M(S)$ is an instance of G.*

This lemma is proved in [2]. It shows that the simple entailment relation $S \models G$ between finite sets S of finite RDF graphs and finite RDF graphs G is decidable. It also shows that this problem is in `NP`: guess and check an instance function h. It is clear that this problem is in `P` when G is assumed to be ground. According to [2], this problem (without restrictive assumptions) is `NP`-complete.

3 RDFS Interpretations and D* Interpretations

As preparation for the proof of decidability of entailment for RDFS, possibly extended with datatypes, this section defines D* interpretations, a generalization of RDFS interpretations extending the semantics of `XMLLiteral` to all datatypes in a datatype map. We first consider RDFS interpretations.

3.1 RDFS interpretations. The *RDF* and *RDFS vocabulary*, $rdfV \cup rdfsV$, is a set of URI references listed with one-letter abbreviations in Table 1, plus 18 other URI references [2] (which play a smaller role in the semantics).

Table 1. Abbreviated notation for standard URI references

t	`rdf:type`	L	`rdfs:Literal`
P	`rdf:Property`	D	`rdfs:Datatype`
X	`rdf:XMLLiteral`	C	`rdfs:Class`
r_i	`rdf:_i`	s_C	`rdfs:subClassOf`
d	`rdfs:domain`	s_P	`rdfs:subPropertyOf`
r	`rdfs:range`	m	`rdfs:member`
R	`rdfs:Resource`	γ	`rdfs:ContainerMembershipProperty`

Let V be a vocabulary and I a simple interpretation of $V \cup \{t, C\}$ such that $I(t) \in P_I$, so that $E_I(I(t))$ is defined. In this case, the set C_I of *classes* of I is defined to be

$$C_I = \{a \in R_I : (a, I(C)) \in E_I(I(t))\},$$

and the *class extension function* $CE_I : C_I \to \mathcal{P}(R_I)$ of I is defined by

$$CE_I(a) = \{b \in R_I : (b, a) \in E_I(I(t))\} \ \ (a \in C_I).$$

An *rdfs-interpretation* of a vocabulary V is a simple interpretation I of $V \cup rdfV \cup rdfsV$ that satisfies the following conditions:

- I satisfies all triples in Table 2, plus 32 other ground triples [2]. These triples are together called the *RDF* and *RDFS axiomatic triples*.
- $a \in P_I$ if and only if $(a, I(P)) \in E_I(I(t))$
- $a \in R_I$ if and only if $(a, I(R)) \in E_I(I(t))$
- $a \in LV_I$ if and only if $(a, I(L)) \in E_I(I(t))$
- If $(a, b) \in E_I(I(d))$ and $a \in P_I$ and $b \in C_I$ and $(e, f) \in E_I(a)$, then $e \in CE_I(b)$ [3]
- If $(a, b) \in E_I(I(r))$ and $a \in P_I$ and $b \in C_I$ and $(e, f) \in E_I(a)$, then $f \in CE_I(b)$
- $E_I(I(s_P))$ is transitive and reflexive on P_I
- If $(a, b) \in E_I(I(s_P))$ then $a \in P_I$ and $b \in P_I$ and $E_I(a) \subset E_I(b)$
- If $a \in C_I$ then $(a, I(R)) \in E_I(I(s_C))$
- If $(a, b) \in E_I(I(s_C))$ then $a \in C_I$ and $b \in C_I$ and $CE_I(a) \subset CE_I(b)$
- $E_I(I(s_C))$ is transitive and reflexive on C_I
- If $a \in CE_I(I(\gamma))$ then $(a, I(m)) \in E_I(I(s_P))$
- If $a \in CE_I(I(D))$ then $(a, I(L)) \in E_I(I(s_C))$
- If $l = (s, X) \in V$ is a well-typed XML literal, then $L_I(l) = xml(l) \in LV_I$ and $(L_I(l), I(X)) \in E_I(I(t))$
- If $l = (s, X) \in V$ is an XML literal that is not well-typed, then $L_I(l) \notin LV_I$ and $(L_I(l), I(X)) \notin E_I(I(t))$

[3] This condition and the next condition have been slightly reformulated, to make clear that the definition uses the functions E_I and CE_I inside their domains, as required. See the explanation following the definition. Lemma 3.2 below shows that exactly the same class of rdfs-interpretations is defined as in [2].

Table 2. RDF and RDFS axiomatic triples

$t\,t\,P$	$s_P\,d\,P$	$r\,r\,C$	$X\,s_C\,L$	
$r_i\,t\,P$	$s_C\,d\,C$	$s_P\,r\,P$	$D\,s_C\,C$	
$t\,d\,R$	$m\,d\,R$	$s_C\,r\,C$	$r_i\,t\,\gamma$	
$d\,d\,P$	$t\,r\,C$	$\gamma\,s_C\,P$	$r_i\,d\,R$	
$r\,d\,P$	$d\,r\,C$	$X\,t\,D$	$r_i\,r\,R$	$(i = 1, 2, ...)$

Here the function xml assigns to each well-typed XML literal its value [4]. Note that the first axiomatic triple $t\,t\,P$ shows that for each rdfs-interpretation I, $I(t) \in P_I$, so that $E_I(I(t))$, C_I, and CE_I are defined, as is used in the statement of the remaining conditions. By using the functions E_I and CE_I, the definition assumes, implicitly, that certain values of I, such as $I(d)$ and $I(r)$, are in P_I, and that other values of I, such as $I(D)$, are in C_I. That this is allowed is demonstrated in the next lemma. This lemma also justifies some other conclusions that can be stated for each rdfs-interpretation, and that are constantly used in the manipulation of rdfs-interpretations.

3.2 Lemma. *The first condition in the definition of rdfs-interpretations ensures that each invocation of the functions E_I and CE_I in this definition is allowed. If I is an rdfs-interpretation of a vocabulary V, then*

- *$I(t)$, $I(d)$, $I(r)$, $I(s_C)$, $I(s_P)$, and $I(m)$ are in P_I and $P_I \subset R_I$*
- *$I(P)$, $I(R)$, $I(C)$, $I(L)$, $I(D)$, $I(X)$, and $I(\gamma)$ are in C_I*
- *$P_I = CE_I(I(P))$, $C_I = CE_I(I(C))$, $R_I = CE_I(I(R))$, and $LV_I = CE_I(I(L))$*
- *if (a, b) is in $E_I(I(d))$ or $E_I(I(r))$, then $a \in P_I$ and $b \in C_I$*
- *if $(a, b) \in E_I(I(t))$, then $b \in C_I$ and $a \in CE_I(b)$*
- *if $(a, b) \in E_I(I(d))$ and $(e, f) \in E_I(a)$, then $e \in CE_I(b)$*
- *if $(a, b) \in E_I(I(r))$ and $(e, f) \in E_I(a)$, then $f \in CE_I(b)$*

3.3 Datatype maps. Before defining D* interpretations, we summarize some terminology and introduce some notation related to datatype maps [4] [2]. A *datatype* d is defined by a nonempty set of strings $L(d)$, the *lexical space*, a nonempty set $V(d)$, the *value space*, and a mapping $L2V(d) : L(d) \to V(d)$, the *lexical-to-value mapping*. A *datatype map* is a partial function D from the set U of URI references to the class of datatypes. Each datatype map is required [2] to contain the pair (X, x), where X is `rdf:XMLLiteral` and x is the built-in XML literal datatype, defined by $L(x) = \{s : (s, X) \in L^+_X\}$, $V(x) = XV$, and $L2V(x)(s) = xml((s, X))$ for each $(s, X) \in L^+_X$. Here L^+_X is the set of well-typed XML literals [4].

Suppose that a datatype map D is given. The *D-vocabulary* is the *domain* $dom(D)$ of D, i.e., the set of URI references $a \in U$ such that $(a, d) \in D$ for some datatype d. The *range* of D, i.e., the set of datatypes d such that $(a, d) \in D$ for some $a \in U$, is denoted by $ran(D)$. The set of *D-literals* is the set of typed literals $(s, a) \in L_{\mathrm{t}}$ with $a \in dom(D)$. The set of all well-typed literals with type in D, i.e., the set of all *well-typed D-literals*, is denoted by L^+_D:

$$L^+_D = \{(s, a) \in L_{\mathrm{t}} : a \in dom(D), s \in L(D(a))\}.$$

The function val_D is defined to map each well-typed D-literal to its value:

$$val_D : L_D^+ \to \bigcup_{d \in ran(D)} V(d) \,,$$

$$val_D((s,a)) = L2V(D(a))(s) \quad (a \in dom(D), s \in L(D(a))) \,.$$

Injective datatype maps whose value spaces are disjoint and whose literal-to-value mappings are injective are of special importance in this paper; such datatype maps will be called *discriminating*. The assumption that a datatype map is discriminating is satisfied, for example, when it consists only of the four datatypes `rdf:XMLLiteral`, `xsd:string`, `xsd:integer`, and `xsd:boolean`.

3.4 Definition (D* interpretations). If D is a datatype map, a *D^* interpretation* of a vocabulary V is an rdfs-interpretation I of $V \cup dom(D)$ that satisfies the following conditions for each pair $(a,d) \in D$:

- $I(a) = d$
- I satisfies the following triples: $a\, t\, D$ and $a\, s_C\, L$
- if $l = (s,a') \in L_{\mathrm{t}} \cap V$ and $I(a') = d$ and $s \in L(d)$, then $L_I(l) = L2V(d)(s) \in LV_I$ and $(L_I(l), d) \in E_I(I(t))$
- if $l = (s,a') \in L_{\mathrm{t}} \cap V$ and $I(a') = d$ and $s \notin L(d)$, then $L_I(l) \notin LV_I$ and $(L_I(l), d) \notin E_I(I(t))$

If D is a datatype map, the triples $a\, t\, D$ and $a\, s_C\, L$ appearing in this definition are combined for $a \in dom(D)$ to form the *D-axiomatic triples*. It should be noted that D* interpretations generalize the `XMLLiteral`-related conditions on rdfs-interpretations.[4]

4 RDFS Entailment and D* Entailment

4.1 Definition (D* entailment). Given a datatype map D, a set S of graphs *D^* entails* a graph G if each D* interpretation I that satisfies S also satisfies G.

RDFS entailment coincides with D* entailment when the datatype map D is assumed to consist of only the type `rdf:XMLLiteral`. We shall use the notation

$$S \models_{\mathrm{s}} G$$

for D* entailment (and also for the special case of RDFS entailment).

[4] This correspondence with the `XMLLiteral`-related conditions would be exact when the phrase "if $l = (s,a') \in L_{\mathrm{t}} \cap V$ and $I(a') = d$ " starting the last two conditions on D* interpretations would in both cases be simplified to "if $l = (s,a) \in L_{\mathrm{t}} \cap V$ ". In fact, this change can be made without leading to a change in entailments, as can be seen by checking the proof of the D* entailment lemma described below; only minor changes to the proof are needed. With the definition chosen, D* interpretations can be related in a simple way to the D-interpretations of [2]: If I is an rdfs-interpretation of a vocabulary V, then I is a D-interpretation of V if and only if I is a D* interpretation of V that satisfies $CE_I(d) = V(d)$ for each $d \in ran(D)$.

Table 3. D* entailment rules

	If G contains		where	then add to G
lg	$v\,p\,l$		$l \in L$	$v\,p\,b_l$
gl	$v\,p\,b_l$			$v\,p\,l$
rdf1	$v\,p\,w$			$p\,t\,P$
rdf2-D	$v\,p\,l$		$l = (s,a) \in L_D^+$	$b_l\,t\,a$
rdfs1	$v\,p\,l$		$l \in L_{\mathrm{p}}$	$b_l\,t\,L$
rdfs2	$p\,d\,u$	$v\,p\,w$		$v\,t\,u$
rdfs3	$p\,r\,u$	$v\,p\,w$	$w \in U \cup B$	$w\,t\,u$
rdfs4a	$v\,p\,w$			$v\,t\,R$
rdfs4b	$v\,p\,w$		$w \in U \cup B$	$w\,t\,R$
rdfs5	$v\,s_P\,w$	$w\,s_P\,u$		$v\,s_P\,u$
rdfs6	$v\,t\,P$			$v\,s_P\,v$
rdfs7	$p\,s_P\,q$	$v\,p\,w$	$q \in U$	$v\,q\,w$
rdfs8	$v\,t\,C$			$v\,s_C\,R$
rdfs9	$v\,s_C\,w$	$u\,t\,v$		$u\,t\,w$
rdfs10	$v\,t\,C$			$v\,s_C\,v$
rdfs11	$v\,s_C\,w$	$w\,s_C\,u$		$v\,s_C\,u$
rdfs12	$v\,t\,\gamma$			$v\,s_P\,m$
rdfs13	$v\,t\,D$			$v\,s_C\,L$

4.2 Definition (D* entailment rules). Given a datatype map D, the D* entailment rules are defined in Table 3. They consist of exactly the 18 rules defined in [2] for RDFS, the only difference being that rule rdf2 is replaced by the more general rule rdf2-D. In words, the first rule lg ('literal generalization') says that if G contains $v\,p\,l$, where l is a literal, then add $v\,p\,b_l$ to G, where b_l is a blank node allocated to l by this rule. Here *allocated to* means that the blank node b_l has been created by an application of rule lg on the same literal l, or if there is no such blank node, in which case it must be a new node which is not yet in the graph. In rule rdfs1, b_l is a blank node that is allocated by rule lg to the plain literal $l \in L_{\mathrm{p}}$. In rule rdf2-D, b_l is a blank node that is allocated by rule lg to the well-typed D-literal $l \in L_D^+$. Note that rule rdf2-D is a direct generalization of entailment rule rdf2 from [2], which has the same effect only for well-typed XML literals $l \in L_X^+$. If D contains only the datatype `rdf:XMLLiteral`, then rule rdf2-D becomes exactly rule rdf2.

The notion of XML clash from [2] is generalized in a straightforward way to any datatype map.

4.3 Definition (D-clash). Given a datatype map D, a *D-clash* is a triple $b\,t\,L$, where b is a blank node allocated by rule lg to an ill-typed D-literal.

We turn to completeness and decidability. As was also done in earlier versions of [2], we extract from the completeness proof the notions of closure and Herbrand interpretation, as well as a satisfaction lemma. In view of the finiteness of a partial closure, the closure notion is of practical interest and is no longer just an 'abstraction in the proof'. The Herbrand construction of [2] will need to be refined, to deal with the finiteness of a partial closure and with datatypes.

4.4 Definition (partial and full RDFS and D* closures). Suppose that G is an RDF graph and D a datatype map. In the definitions that follow, the axiomatic triples containing the URI references r_i (i.e., the four triples $r_i\, t\, P$, $r_i\, t\, \gamma$, $r_i\, d\, R$, and $r_i\, r\, R$) are treated in a special way. Suppose that K is a nonempty subset of the positive integers $\{1, 2, ...\}$ chosen in such a way that for each $r_i \in V(G)$ we have $i \in K$. The *partial D* closure* $G_{\mathrm{s},K}$ of G is defined in the following way. In the first step, all RDF and RDFS axiomatic triples and D-axiomatic triples are added to G, except for the axiomatic triples including r_i such that $i \notin K$. In the next step, rule lg is applied to each triple containing a literal. Then, rules rdf2-D and rdfs1 are applied to each triple containing a well-typed D-literal or a plain literal, respectively. Finally, the remaining D* entailment rules are applied until the graph is unchanged. In addition to the partial D* closure $G_{\mathrm{s},K}$ obtained in this way, the *full D* closure* G_{s} of G is defined by taking $G_{\mathrm{s}} = G_{\mathrm{s},K}$ where K is the full set $\{1, 2, ...\}$. When the datatype map D consists of only the datatype `rdf:XMLLiteral`, a partial and a full D* closure of an RDF graph G are called a *partial* and a *full RDFS closure* of G, respectively. With respect to a partial (or full) D* closure $G_{\mathrm{s},K}$ and a literal $l \in V(G) \cap L$, the unique blank node allocated by rule lg to l is denoted by b_l.

4.5 Lemma. *Let D be a finite datatype map. If G is a finite RDF graph, then each partial D* closure $G_{\mathrm{s},K}$ of G is finite for K finite, and a partial D* closure of G can be computed in polynomial time.*

Proof. The graph obtained from a finite graph G in the first step of the definition of partial closure is clearly finite when K is finite. Then, rule lg adds only finitely many new triples, leading to a finite graph G' containing G. In the remaining steps, no new names or blank nodes are added to G', so it follows that there exist finite sets $U_0 \subset U$, $B_0 \subset B$, and $L_0 \subset L$ such that

$$G_{\mathrm{s},K} \;\subset\; U_0 \cup B_0 \;\times\; U_0 \;\times\; U_0 \cup B_0 \cup L_0\,. \tag{2}$$

Hence $G_{\mathrm{s},K}$ is a finite graph.

To prove that a partial D* closure can be computed in polynomial time, choose a finite, nonempty set $K \subset \{1, 2, ...\}$ such that $i \in K$ for each $r_i \in V(G)$. Let $g = |G|$, $d = |D|$, and $k = |K|$. We prove that $G_{\mathrm{s},K}$ can be computed in time polynomial in g, d, and k. It will follow that a partial D* closure can be computed in time polynomial in g, since k can clearly be taken to be $3g + 1$, and d can be viewed as a bounded parameter. The first step of the closure construction process adds $48 + 4k$ RDF and RDFS axiomatic triples and $2d$ D-axiomatic triples. In the next step, rule lg adds at most g triples. It follows that the graph G' resulting from these steps has at most $2g + 2d + 4k + 48$ triples, so that the sets U_0, B_0, and L_0 in (2) can be chosen to satisfy $|U_0 \cup B_0 \cup L_0| \leq 3(2g + 2d + 4k + 48)$, and $|G_{\mathrm{s},K}| \leq 27(2g + 2d + 4k + 48)^3$. Therefore, in the remaining steps at most $27(2g + 2d + 4k + 48)^3$ rule applications can add a new triple to the partial closure graph under construction. For each of the entailment rules used, it can be determined whether a successful rule application exists in at most linear or quadratic time as a function of the size of the partial closure graph under construction (cf. Table 3). For example, for rule rdf1 linear time is sufficient, and

for rule rdfs2 quadratic time is sufficient. It follows that $G_{s,K}$ can be computed in time polynomial in g, d, and k.

4.6 Definition (D* Herbrand interpretation). Suppose that D is a discriminating datatype map. A *D* Herbrand interpretation* of a graph G is a simple interpretation $S_K(G)$ of $V(G_s)$ defined as follows. K is a nonempty subset of $\{1, 2, ...\}$ chosen such that for each $r_i \in V(G)$ we have $i \in K$. R_I is defined as the union of four pairwise disjoint sets:

$$val_D(V(G_s) \cap L_D^+)$$

$$V(G_s) \cap (L - L_D^+)$$

$$V(G_s) \cap (U \cup B - dom(D))$$

$$ran(D)$$

The remainder of the definition makes use of the function

$$sur : R_I \to nd(G_{s,K}) \cap (U \cup B)$$

('surrogate'), which replaces the elements of R_I by non-literal nodes in $G_{s,K}$. The function *sur* is defined as follows, using the assumption on D:

- If $l \in V(G_s) \cap L_D^+$, then $sur(val_D(l)) = b_l$.
- If $l \in V(G_s) \cap (L - L_D^+)$, then $sur(l) = b_l$.
- If $v \in V(G_s) \cap (U \cup B - (dom(D) \cup \{r_i : i \notin K\}))$, then $sur(v) = v$.
- If $i \notin K$, then $sur(r_i) = r_n$, where $n \in K$ is fixed.
- If $(a, d) \in D$, then $sur(d) = a$.

The other parts of the $S_K(G)$ are defined in the following way:

$$LV_I = \{x \in R_I : sur(x)\, t\, L \in G_{s,K}\}$$

$$P_I = \{x \in R_I : sur(x)\, t\, P \in G_{s,K}\}$$

$$S_I(v) = v \quad (v \in V(G_s) \cap U - dom(D))$$

$$S_I(a) = D(a) \quad (a \in V(G_s) \cap dom(D))$$

$$L_I(v) = v \quad (v \in V(G_s) \cap L_t - L_D^+)$$

$$L_I(l) = val_D(l) \;\; (l \in V(G_s) \cap L_D^+)$$

$$E_I(p) = \{(s, o) : s \in R_I, o \in R_I, sur(s)\, sur(p)\, sur(o) \in G_{s,K}\} \quad (p \in P_I)$$

Note that for each $v \in V(G_s) \cap (U - \{r_i : i \notin K\})$, we have $sur(I(v)) = v$, and that $sur(I(r_i)) = r_n$ for $i \notin K$. Moreover, if $l \in V(G_s) \cap L$, then $sur(I(l)) = b_l$.

4.7 D* satisfaction lemma. *Let G be an RDF graph and D a discriminating datatype map. If the partial D* closure $G_{s,K}$ of G does not contain a D-clash, then $S_K(G)$ is a D* interpretation that satisfies $G_{s,K}$.*

Proof. In the proof the D* Herbrand interpretation $S_K(G)$ of G is denoted by I. Define the function $A : bl(G_{s,K}) \to R_I$ by $A(b_l) = val_D(l)$ if $l \in L_D^+$, $A(b_l) = l$ if $l \in L - L_D^+$, and by $A(b) = b$ for other blank nodes b. Rule rdf1 shows that for

each triple $v\,p\,w$ in $G_{\mathrm{s},K}$, $G_{\mathrm{s},K}$ contains the triple $p\,t\,P = sur(I(p))\,t\,P$, so that $I(p) \in P_I$. In order to prove that I_A satisfies $v\,p\,w$, i.e. that $(I_A(v), I_A(w)) \in E_I(I(p))$, it is sufficient to prove that $sur(I_A(v))\,p\,sur(I_A(w)) \in G_{\mathrm{s},K}$, since $sur(I(p)) = p$. In order to prove that this triple is indeed in $G_{\mathrm{s},K}$, note first that

$$sur(I_A(v)) = v \quad (v \in nd(G_{\mathrm{s},K}) - L).$$

Namely, as is easy to see, this holds in each of the following cases where $v \in nd(G_{\mathrm{s},K}) - L$: $v \in U$, $v \in B$ allocated to a literal in L_D^+ or in $L - L_D^+$, other blank nodes v. Moreover, we have $sur(I_A(l)) = b_l$ for each $l \in nd(G_{\mathrm{s},K}) \cap L$, as was already noted. It follows that if $v\,p\,w \in G_{\mathrm{s},K}$, then $sur(I_A(v))\,p\,sur(I_A(w)) \in G_{\mathrm{s},K}$, as desired: the only case that needs checking is where $w \in L$. In this case rule lg shows that $v\,p\,b_w \in G_{\mathrm{s},K}$, as required: there is no entailment rule that can be applied with a literal in object position but that cannot be applied with a blank node in object position, so rule applications used to derive $v\,p\,w$ can be followed in parallel with other rule applications to provide a derivation of $v\,p\,b_w$.

Note that $S_K(G)$ satisfies all axiomatic triples, even though $G_{\mathrm{s},K}$ may not contain all of them: if $i \notin K$ then $sur(r_i) = r_n$, and the axiomatic triples including r_i are also satisfied. For example, the triple $r_i\,t\,P$ is satisfied because $sur(r_i)\,t\,P = r_n\,t\,P \in G_{\mathrm{s},K}$.

We turn to the assumptions on D* interpretations. Suppose that $(a, d) \in D$. If $l = (s, a') \in V(G_{\mathrm{s}}) \cap L_{\mathrm{t}}$ and $I(a') = d$, then the assumption on D shows that $a = a'$. Suppose therefore that $l = (s, a)$ is a well-typed D-literal in $V(G_{\mathrm{s}})$. Then $I(l) = val_D(l) = L2V(d)(s) = A(b_l)$. In order to prove that $I(l) \in LV_I$, we need to prove that $sur(I(l))\,t\,L = b_l\,t\,L \in G_{\mathrm{s},K}$. Rule rdf2-D shows that the triple $b_l\,t\,a$ is in $G_{\mathrm{s},K}$. With the axiomatic triple $a\,s_C\,L$ and rule rdfs9 it follows that $b_l\,t\,L \in G_{\mathrm{s},K}$. We also have $(L_I(l), d) = (I_A(b_l), I_A(a)) \in E_I(I(t))$, since $sur(I_A(b_l))\,t\,sur(I_A(a)) = b_l\,t\,a \in G_{\mathrm{s},K}$.

Suppose that $(a, d) \in D$ and that $l = (s, a) \in V(G_{\mathrm{s}})$ is a D-literal that is not well-typed. Then $A(b_l) = l$. Suppose that $(L_I(l), d) \in E_I(I(t))$; then $b_l\,t\,a = sur(L_I(l))\,t\,sur(d) \in G_{\mathrm{s},K}$, so that with the axiomatic triple $a\,s_C\,L$ and rule rdfs9 it follows that $b_l\,t\,L \in G_{\mathrm{s},K}$: $G_{\mathrm{s},K}$ contains a D-clash. Suppose that $L_I(l) \in LV_I$; then, again, $b_l\,t\,L = sur(L_I(l))\,t\,L \in G_{\mathrm{s},K}$: $G_{\mathrm{s},K}$ contains a D-clash.

We show that $a \in R_I$ if and only if $(a, R) \in E_I(t)$. If $(a, I(R)) \in E_I(t)$, then clearly $a \in R_I$. Suppose that $a \in R_I$. It is sufficient to prove that $sur(a)\,t\,R \in G_{\mathrm{s},K}$. If $a = val_D(l)$, $l \in V(G_{\mathrm{s}}) \cap L_D^+$, then $sur(a) = b_l$ and $G_{\mathrm{s},K}$ contains triples $v\,p\,l$ and $v\,p\,b_l$. Rule rdfs4b shows that $sur(a)\,t\,R = b_l\,t\,R \in G_{\mathrm{s},K}$. If $a = l \in V(G_{\mathrm{s}}) \cap (L - L_D^+)$, then $sur(l) = b_l$ and $G_{\mathrm{s},K}$ contains triples $v\,p\,l$ and $v\,p\,b_l$. Again, rule rdfs4b shows that $sur(l)\,t\,R = b_l\,t\,R \in G_{\mathrm{s},K}$. If $a \in V(G_{\mathrm{s}}) \cap (U \cup B - (dom(D) \cup \{r_i : i \notin K\}))$, then $sur(a) = a$ and $G_{\mathrm{s},K}$ contains, by rule rdf1, a triple of the form $a\,p\,v$ or $v\,p\,a$, so that by rule rdfs4a or rdfs4b, we have $sur(a)\,t\,R = a\,t\,R \in G_{\mathrm{s},K}$. If $a = r_i$ and $i \notin K$, then $sur(r_i)\,t\,R = r_n\,t\,R \in G_{\mathrm{s},K}$ in view of $r_n\,t\,P \in G_{\mathrm{s},K}$ and rule rdfs4a. If $a = d \in ran(D)$, then D contains a pair (a', d) and $G_{\mathrm{s},K}$ contains the axiomatic triple $a'\,t\,D$. Rule rdfs4a shows that $G_{\mathrm{s},K}$ contains the triple $sur(d)\,t\,R = a'\,t\,R$.

4.8 D* entailment lemma. *Let D be a discriminating datatype map, S a set of RDF graphs, and G an RDF graph. Then, $S \models_{\mathrm{s}} G$ if and only if there*

is a graph H that can be derived from $M(S)$ combined with RDF and RDFS axiomatic triples and D-axiomatic triples, by application of the D entailment rules, and that either simply entails G or contains a D-clash.*

The next statement gives a more precise version of this lemma.

4.9 D* entailment lemma (alternative statement). *Let D be a discriminating datatype map, S a set of RDF graphs, and G an RDF graph. Let H be a partial D* closure $M(S)_{\mathrm{s},K}$ of $M(S)$ and suppose that $i \in K$ for each $r_i \in V(G)$. Then, $S \models_{\mathrm{s}} G$ if and only if either H simply entails G or H contains a D-clash.*

4.10 Corollary. *Let D be a finite discriminating datatype map. The D* entailment relation $S \models_{\mathrm{s}} G$ between finite sets S of finite RDF graphs and finite RDF graphs G is decidable. This problem is in* NP*, and in* P *when G is ground.*

Proof of corollary. Take a partial D* closure $H = M(S)_{\mathrm{s},K}$ of $M(S)$ such that K is finite and $i \in K$ for each $r_i \in V(G)$, and check whether this graph contains a D-clash or whether this graph simply entails G. By Lemma 4.5, H can be computed in time polynomial in $|M(S)| + |G|$. The proof is concluded by using the remarks following the interpolation lemma (see 2.5).

Proof of D entailment lemma (completeness part).* Let $H = M(S)_{\mathrm{s},K}$ be a partial D* closure of $M(S)$ such that $i \in K$ for each $r_i \in V(G)$, and let I be the associated D* Herbrand interpretation $S_K(M(S))$. If H does not contain a D-clash, then I is a D* interpretation which satisfies H, so I satisfies S. By $S \models_{\mathrm{s}} G$, I satisfies G. Choose $A : bl(G) \to R_I$ such that I_A satisfies G. It follows that for each triple $s\,p\,o \in G$, $E_I(I(p))$ contains the pair $(I_A(s), I_A(o))$. Therefore, for each triple $s\,p\,o \in G$, H contains the triple

$$sur(I_A(s))\, sur(I(p))\, sur(I_A(o)).$$

The assumption that $i \in K$ for each $r_i \in V(G)$ implies that if $v \in V(G) \cap U$, then $sur(I(v)) = v$. Moreover, if $l \in nd(G) \cap L$ then $sur(I(l)) = b_l$. Define the mapping $h : bl(G) \to nd(H)$ to be $h = sur \circ A$, and extend h to all of $nd(G)$ by $h(v) = v$ for each $v \in nd(G) - bl(G)$. It follows that

- if $s\,p\,o \in G$ and o is not in L, then $h(s)\,p\,h(o) \in H$
- if $l \in L$ and $s\,p\,l \in G$, then $h(s)\,p\,b_l \in H$.

In the second of these two cases, it also follows that $h(s)\,p\,h(l) = h(s)\,p\,l \in H$, in view of rule gl. Hence there is a subgraph of H that is an instance of G, with respect to the instance mapping h. It follows that H simply entails G.

5 pD* Interpretations and pD* Entailment

In this section, we extend the above results on RDFS entailment and D* entailment to a subset of OWL related to properties (see Table 4). We do not obtain the full power of the 'iff-semantics' of the W3C's OWL semantics specification [5]. We consider a weaker semantics, in line with the 'if-semantics' of RDFS. This leads to simple and useful entailment rules, which can be used to extend RDF reasoners. This weaker semantics, which is computationally less complex, seems to be sufficient for many applications of this subset of OWL.

Table 4. Abbreviations of property-related vocabulary

F	`FunctionalProperty`	S	`SymmetricProperty`
F'	`InverseFunctionalProperty`	T	`TransitiveProperty`
s	`sameAs`	i	`inverseOf`

Table 5. P-axiomatic triples

$F\, s_C\, P$	$S\, s_C\, P$	$s\, t\, P$	$i\, d\, P$
$F'\, s_C\, P$	$T\, s_C\, P$	$i\, t\, P$	$i\, r\, P$

5.1 Definition (pD* interpretations). The *P-vocabulary*, denoted by *propV*, is a set of URI references listed in Table 4, together with one-letter abbreviations.

Let D be a datatype map. A *pD* interpretation* of a vocabulary V is a D* interpretation I of $V \cup propV$ that satisfies the following conditions:

- I satisfies the triples in Table 5, which are called the *P-axiomatic triples*.
- If $p \in CE_I(I(F))$ and $(a,b),(a,c) \in E_I(p)$ then $(b,c) \in E_I(I(s))$
- If $p \in CE_I(I(F'))$ and $(a,c),(b,c) \in E_I(p)$ then $(a,b) \in E_I(I(s))$
- If $p \in CE_I(I(S))$ and $(a,b) \in E_I(p)$ then $(b,a) \in E_I(p)$
- If $p \in CE_I(I(T))$ and $(a,b),(b,c) \in E_I(p)$ then $(a,c) \in E_I(p)$
- $E_I(I(s))$ is an equivalence relation on R_I
- If $(p,q) \in E_I(I(i))$ then $(a,b) \in E_I(p)$ if and only if $(b,a) \in E_I(q)$
- If $a \in C_I$ and $(a,b) \in E_I(I(s))$ then $(a,b) \in E_I(I(s_C))$
- If $p \in P_I$ and $(p,q) \in E_I(I(s))$ then $(p,q) \in E_I(I(s_P))$

5.2 Lemma. *The first condition in the definition of pD* interpretations ensures that each invocation of the functions E_I and CE_I in this definition is allowed. If I is a pD* interpretation of a vocabulary V, then $I(s)$ and $I(i)$ are in P_I, and $I(F)$, $I(F')$, $I(S)$, and $I(T)$ are in C_I.*

5.3 Definition (pD* entailment). Given a datatype map D, a set S of RDF graphs *pD* entails* an RDF graph G if each pD* interpretation I that satisfies S also satisfies G. In this case, we shall write

$$S \models_{\mathrm{p}} G.$$

It is clear that each OWL interpretation [5] is a pD* interpretation, so that if $S \models_{\mathrm{p}} G$, then S OWL Full entails G.

5.4 Definition (P-entailment rules). See Table 6.

5.5 Definition (partial and full pD* closures). Suppose that G is an RDF graph and D a datatype map. The *partial pD* closure* $G_{\mathrm{p},K}$ of G, and the *full pD* closure* G_{p} of G, are defined in a way similar to the definition of the partial and full D* closures $G_{\mathrm{s},K}$ and G_{s} of G (see 4.4). The only differences are in the first and last steps. In the first step, the 8 P-axiomatic triples are also added to G. In the last step, the P-entailment rules are used as well.

Table 6. P-entailment rules

	If G contains		where	then add to G
rdfp1	$p\,t\,F$	$u\,p\,v$ $u\,p\,w$	$v \in U \cup B$	$v\,s\,w$
rdfp2	$p\,t\,F'$	$u\,p\,w$ $v\,p\,w$		$u\,s\,v$
rdfp3	$p\,t\,S$	$v\,p\,w$	$w \in U \cup B$	$w\,p\,v$
rdfp4	$p\,t\,T$	$u\,p\,v$ $v\,p\,w$		$u\,p\,w$
rdfp5a	$v\,p\,w$			$v\,s\,v$
rdfp5b	$v\,p\,w$		$w \in U \cup B$	$w\,s\,w$
rdfp6	$v\,s\,w$		$w \in U \cup B$	$w\,s\,v$
rdfp7	$u\,s\,v$	$v\,s\,w$		$u\,s\,w$
rdfp8a	$p\,i\,q$	$v\,p\,w$	$w \in U \cup B$ and $q \in U$	$w\,q\,v$
rdfp8b	$p\,i\,q$	$v\,q\,w$	$w \in U \cup B$ and $q \in U$	$w\,p\,v$
rdfp9	$v\,t\,C$	$v\,s\,w$		$v\,s_C\,w$
rdfp10	$p\,t\,P$	$p\,s\,q$		$p\,s_P\,q$

5.6 Lemma. *Let D be a finite datatype map. If G is a finite RDF graph, then each partial pD* closure $G_{\mathrm{p},K}$ of G is finite for K finite, and a partial pD* closure of G can be computed in polynomial time.*

Proof. This is proved as for partial D* closures (see 4.5). For the last part of the proof, note that for rules rdfp1, rdfp2 and rdfp4, the existence of a successful rule application can be detected in time $O(|G''|^3)$, where G'' is the partial closure graph under construction. The other P-entailment rules can be handled in linear or quadratic time, just like the D* entailment rules.

5.7 Definition (pD* Herbrand interpretation). Given a discriminating datatype map D and an RDF graph G, a *pD* Herbrand interpretation* $P_K(G)$ is defined in a similar way as a D* Herbrand interpretation $S_K(G)$ (see 4.6). The only differences are that G_{s} is replaced by G_{p} and $G_{\mathrm{s},K}$ by $G_{\mathrm{p},K}$.

5.8 pD* satisfaction lemma. *Let G be an RDF graph and D a discriminating datatype map. If the partial pD* closure $G_{\mathrm{p},K}$ of G does not contain a D-clash, then $P_K(G)$ is a pD* interpretation that satisfies $G_{\mathrm{p},K}$.*

Proof. We consider here only the most complicated addition to be made to the proof of the D* satisfaction lemma (see 4.7), which is reflexivity. To prove reflexivity of $E_I(I(s))$ on R_I, where $I = P_K(G)$, the four parts of the definition of R_I need to be considered separately. Suppose first that $l \in V(G_{\mathrm{p}}) \cap L_D^+$. Then by rule rdfp5b, $G_{\mathrm{p},K}$ contains the triple $b_l\,s\,b_l = sur(val_D(l))\,s\,sur(val_D(l))$, so that $(val_D(l), val_D(l)) \in E_I(s)$, as desired. Suppose next that $l \in V(G_{\mathrm{p}}) \cap (L - L_D^+)$. Again by rule rdfp5b, $G_{\mathrm{p},K}$ contains the triple $b_l\,s\,b_l = sur(l)\,s\,sur(l)$, so that $(l, l) \in E_I(I(s))$, as desired. Suppose next that $a \in V(G_{\mathrm{p}}) \cap (U \cup B - (dom(D) \cup \{r_i : i \notin K\}))$. Then, $sur(a) = a$ and $G_{\mathrm{p},K}$ contains, by rule rdf1, a triple of the form $a\,p\,v$ or $v\,p\,a$, so that by rules rdfp5a and rdfp5b, it follows that $G_{\mathrm{p},K}$ contains the triple $a\,s\,a = sur(a)\,s\,sur(a)$, so that $(a, a) \in E_I(I(s))$. If $i \notin K$, then $sur(r_i) = r_n$ and $G_{\mathrm{p},K}$ contains the axiomatic triple $r_n\,t\,P$, so that by rule rdfp5a, $sur(r_i)\,s\,sur(r_i) = r_n\,s\,r_n \in G_{\mathrm{p},K}$ and $(r_i, r_i) \in E_I(I(s))$. If $d \in ran(D)$, then D contains a pair (a, d) and $G_{\mathrm{p},K}$ contains the axiomatic triple $a\,t\,D$. In view

of rule rdfp5a, it follows that $G_{\mathrm{p},K}$ contains the triple $a\,s\,a = sur(d)\,s\,sur(d)$, so that $(d,d) \in E_I(I(s))$.

5.9 pD* entailment lemma. *Let D be a discriminating datatype map, S a set of RDF graphs, and G an RDF graph. Then, $S \models_{\mathrm{p}} G$ if and only if there is a graph H that can be derived from $M(S)$ combined with RDF and RDFS axiomatic triples and D-axiomatic triples and P-axiomatic triples, by application of D* entailment rules and P-entailment rules, and that either simply entails G or contains a D-clash.*

5.10 pD* entailment lemma (alternative statement). *Let D be a discriminating datatype map, S a set of RDF graphs, and G an RDF graph. Let H be a partial pD* closure $M(S)_{\mathrm{p},K}$ of $M(S)$ and suppose that $i \in K$ for each $r_i \in V(G)$. Then, $S \models_{\mathrm{p}} G$ if and only if either $H \models G$ or H contains a D-clash.*

5.11 Corollary. *Let D be a finite discriminating datatype map. The pD* entailment relation $S \models_{\mathrm{p}} G$ between finite sets S of finite RDF graphs and finite RDF graphs G is decidable. This problem is in NP, and in P when G is ground.*

Proof. Most of the work to be added to the proof for D* entailment involves the validity of the P-entailment rules, and this does not present problems.

6 Conclusion

In this paper we proved that rdfs-entailment is decidable, by using partial closure graphs that are finite for finite RDF graphs. It was also proved that these partial closure graphs can be computed in polynomial time, and that the problem 'H rdfs-entails G', with H and G finite RDF graphs, is in NP, and in P when G is assumed to have no blank nodes. Results on RDFS completeness, decidability, and complexity were extended to include reasoning with datatypes and a property-related subset of OWL.

Acknowledgment. I would like to thank Warner ten Kate, Jan Korst, and the anonymous referees for comments that helped to improve the manuscript.

References

1. J. Grant, D. Beckett (Eds.), RDF Test Cases, W3C Recommendation, 10 February 2004, http://www.w3.org/TR/2004/REC-rdf-testcases-20040210/
2. P. Hayes (Ed.), RDF Semantics, W3C Recommendation, 10 February 2004, http://www.w3.org/TR/2004/REC-rdf-mt-20040210/
3. I. Horrocks, P.F. Patel-Schneider, Reducing OWL Entailment to Description Logic Satisfiability, Proceedings of the 2nd International Semantic Web Conference, Springer-Verlag LNCS 2870, 2003, pp. 17-29.
4. 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/
5. 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/