Abstract

This is a specification of a model-theoretic semantics for RDF and RDFS, and some basic results on entailment. It does not cover reification or special meanings associated with the use of RDF containers. This document was written with the intention of providing a precise semantic theory for RDF and RDFS, and to sharpen the notions of consequence and inference in RDF. It reflects the current understanding of the RDF Core working group at the time of writing. In some particulars this differs from the account given in Resource Description Framework (RDF) Model and Syntax Specification, and these exceptions are noted.

Status of this Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. The latest status of this document series is maintained at the W3C.
This work is part of the W3C Semantic Web Activity. It has been produce by the the RDF Core Working Group which is chartered to address a list of issues raised since RDF 1.0 was issued. This document reflects the Working Group's recent deliberation of some of these issues, but the Working Group has not yet decided whether or how to integrate this document with the RDF 1.0 specification ultimately.

This document is a W3C Working Draft for review by W3C members and other interested parties. It is a draft document and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use W3C Working Drafts as reference material or to cite them as other than "work in progress". A list of current public W3C Working Drafts can be found as part of the W3C Technical Reports and Publications.

Please address feedback on this document to www-rdf-comments@w3.org, a mailing list with public archive). The editors and the Working Group plan to address feedback in future revisons of this document.

Table of Contents

0. Introduction
   0.1 Model-theoretic semantics
   0.2 Graph Syntax
1. Interpretations
   1.1 Technical Note
   1.2 URIs, resources and literals
   1.3 Ground RDF Interpretations
   1.4 Example
2. Unlabeled nodes as existential assertions
   2.1 Comparison with logic
3. Entailment in RDF
   3.1 Skolemization
   3.2 Merging RDF graphs
4. RDFS interpretations
5. RDFS entailment
6. RDF containers
Appendix A. Summary of model theory
Appendix B. Acknowledgements
References
0. Introduction

0.1 Model-theoretic semantics

A model-theoretic semantics for a language assumes that the language refers to a 'world', and describes the minimal conditions that a world must satisfy in order to assign an appropriate meaning for every expression in the language. The idea is to provide an abstract, mathematical account of the properties that any such world must have, making as few assumptions as possible about its actual nature or intrinsic structure; model theory tries to be metaphysically and ontologically neutral. It is typically couched in the language of set theory simply because that is the normal language of mathematics; for example, this semantics assumes that URIs denote things in a set IR called the 'universe'; but the use of set-theoretic language here is not supposed to imply that the things in the universe are set-theoretic in nature.

The chief utility of such a semantic theory is not to suggest any particular processing model, or to provide any deep analysis of the nature of the things being described by the language (in our case, the nature of resources), but rather to provide a technical tool to analyze the semantic properties of proposed operations on the language; in particular, the extent to which they preserve meaning.

There are several aspects of meaning in RDF which are ignored by this semantics. It treats URIs as simple names, ignoring aspects of meaning encoded in particular URI forms [RFC 2396]. It does not provide any analysis of time-varying data or of changes to URI denotations. It does not support some of the uses of rdf containers described in [RDFMS], such as the use of properties applied to containers to indicate properties of the members, and it ignores reification. Some of these may be covered by future extensions of the model theory.

0.2 Graph syntax

Any semantic theory must be attached to a syntax. Of the several syntactic forms for RDF, we have chosen the RDF graph as described in [RDFMS] as the primary syntax, largely for its simplicity. We understand linear RDF notations such as N-Triples and rdf/xml [RDF/XML] as lexical notations for describing an RDF graph.

An RDF graph is a partially labeled directed graph in which nodes are either unlabeled - these are also called anonymous or blank nodes - or else labeled with either URIs or literals; arcs are labeled with URIs; and distinct labeled nodes have different labels. The model theory assigns interpretations directly to the graph, which is taken as being the primary RDF syntax, in the sense that two RDF documents, in whatever lexical form, are syntactically equivalent if and only if they map to the same RDF graph. We will refer to
this as the 'graph syntax' to avoid ambiguity, since the bare term 'syntax' is often assumed to refer to a lexicalization.

We use the N-triples syntax described in [RDFTestCases] to describe RDF graphs. Note that while this syntax uses bNode expressions to identify a particular unlabeled node, those expressions are not considered to be the label of that node. In particular, two N-triples documents which differ only in their bNode expressions will be understood to describe the same RDF graph. (bNode expressions play a role in an ntripleDoc analogous to that played by bound variables in a logical expression. They are local to the document and serve only to indicate a 'connection' between other expressions.)

A graph can be viewed as a set of triples corresponding to its edges; the correspondence between an ntripleDoc and an RDF graph is that the graph has one node for each uriref, bNode or literal identifier in the document, and one edge for each triple, directed from the node corresponding to the subject to the node corresponding to the object. Nodes corresponding to urirefs and literals are labeled with the corresponding URI or literal, and arcs are labeled with the property URI from the corresponding triple. Notice that this requires that all occurrences of a particular uriref or bNode identifier in an N-triples document be mapped to a single node in the corresponding graph. Other RDF lexicalizations may use other means of indicating the graph structure; for our purposes, the important syntactic properties of RDF graphs is that distinct nodes in an RDF graph are treated as distinct referring entities in the graph syntax.

Some definitions will be useful in what follows. An RDF graph will be said to be ground if every node in the graph is labeled. The vocabulary of a graph is the set of URIs that it contains. A graph which is like an RDF graph except that it has two or more nodes with the same label will be called an untidy graph. Untidy graphs may arise as a result of merging operations or be intermediate states in various operations on graphs, but we will assume that all RDF graphs are tidy unless otherwise noted. In graph syntax, an untidy graph is considered to be syntactically ill-formed.

1. Interpretations

1.1 Technical Note

We assume that there is no restriction on the domains and ranges of properties; in particular, a property may be applied to itself. When classes are introduced in RDFS, we will assume that they can contain themselves. This might seem to violate one of the axioms of standard (Zermelo-Fraenkel) set theory, the axiom of foundation, which forbids infinitely descending chains of membership. However, the semantic model given here distinguishes properties and classes as objects from their extensions - the sets of object-value pairs which satisfy the property, or things that are 'in' the class - thereby
allowing the extension of a property or class to contain the property or class itself without violating the axiom of foundation. In particular, this use of a class extension mapping allows classes to contain themselves. For example, it is quite OK for (the extension of) a 'universal' class to contain itself as a member, a convention that is often adopted at the top of a classification hierarchy. (If an extension contained itself then the axiom would be violated, but that case never arises.) The technique is described more fully in [Hayes&Menzel], which gives a semantics for an infinitary extension of full first-order logic.

Notice that the question of whether or not a class contains itself as a member is quite different from the question of whether or not it is a subclass of itself.

In what follows, the fact that two sets are given distinct labels does not imply that they are disjoint; we will explicitly state any disjointness or containment conditions as they arise. In the same spirit, the fact that one set is stated to be a subset of another should not be interpreted as saying that these sets cannot be identical, unless this is stated explicitly.

1.2 URIs, resources and literals

RDF uses two kinds of referring expression, URIs and literals. We make very simple and basic assumptions about these: they are both treated like logical constants, i.e. as expressions which are interpreted as having a single value. URI's are treated as simply denoting resources, and no assumptions are made about the nature of resources.

We do not take any position here on the way that node labels may be composed from other expressions, e.g. from relative URIs or Qnames; the model theory simply assumes that such lexical issues have been resolved in some way that is globally coherent, so that a single URI can be taken to have the same meaning wherever it occurs. Similarly, the model theory given here has no special provision for tracking temporal changes; it assumes, implicitly, that URIs have the same meaning whenever they occur. (If one wishes to apply RDF in a context where the referents of URIs (or names more generally) may change with time, then the current theory could be regarded as defining a 'snapshot' of the meaning of a changing representation.)

Similarly, the model theory makes no assumptions about the exact nature of literals, other than that they are syntactically distinguished from URIs. We assume a global set LV of literal values and a fixed mapping XL from the set of literals to LV. All interpretations will be required to conform to XL on literal nodes. This leaves open the question of the exact nature of literals, their language-sensitivity and so on, while acknowledging their special status as expressions with a 'fixed' interpretation.

(This memo uses the QName syntax to refer to certain URIs defined by RDF. The following namespace prefix and URI values are assumed throughout:
Prefix: rdf, namespace URI: http://www.w3.org/1999/02/22-rdf-syntax-ns#)
1.3 Ground RDF Interpretations

All interpretations will be relative to a set of URIs, called the vocabulary of the interpretation, so that one has to speak, strictly, of an interpretation of an RDF vocabulary, rather than of RDF itself. (For a lexicalized version of the language, we can think of the vocabulary of an interpretation, more traditionally, as being a subset of the URI-indicating expressions used by that lexicalization; e.g. for N-Triples, we can consider the vocabulary of an interpretation to be a set of urirefs.)

An interpretation I on the vocabulary V is defined by:

1. A non-empty set IR of resources, called the domain or universe of I.
2. A subset IP of IR corresponding to properties
3. A mapping IEXT from IP into the powerset of IRx(IR union LV) (i.e. the set of sets of pairs <x,y> with x in IR and y in IR or LV)
4. A mapping IS: V -> IR

IEXT(x) is a set of pairs, i.e. a binary relational extension, called the extension of x. This trick of distinguishing a relation as an object from its relational extension allows a property to occur in its own extension, as noted earlier.

The denotation of a ground RDF graph in I is then given recursively by the following rules, which extend the interpretation mapping I from labels to graphs:

<table>
<thead>
<tr>
<th>Condition</th>
<th>Denotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>if E is a literal then I(E) = XL(E)</td>
<td></td>
</tr>
<tr>
<td>if E is a URI then I(E) = IS(E)</td>
<td></td>
</tr>
<tr>
<td>if E is an asserted triple s p o . then I(E) = true if &lt;I(s),I(o)&gt; is in IEXT(I(p)), otherwise I(E) = false.</td>
<td></td>
</tr>
<tr>
<td>if E is a ground RDF graph then I(E) = false if I(E') = false for some asserted triple E' in E, otherwise I(E) = true.</td>
<td></td>
</tr>
</tbody>
</table>

The use of the phrase "asserted triple" is a deliberate weasel-worded artifact, to allow an RDF graph or document to contain triples which are being used for some non-assertional purpose. Strict conformity to the RDF 1.0 specification [RDFMS] assumes that all triples in a document are asserted triples, but making the distinction allows RDF parsers and inference engines to conform to the RDF syntax and to respect the RDF model theory...
without necessarily being fully committed to it.

### 1.4 Example

For example, for the vocabulary \{a, b, c\} the following is a small interpretation, where we use integers to indicate the 'things' in the universe:

IR= \{1, 2\}; IP = \{1\}

IEXT: 1->\{<1,2>,<2,1>\}

IS: a->1,b->1,c->2

![Diagram](image.png)

**Figure 1**: An example of an interpretation

which makes these triples true:

- a b c.
- c a a.
- c b a.

and these triples false:

- a c b.
For example, $I(a b c .) = true$ if $<I(a),I(c)>$ is in $IEXT(I(b))$, i.e. if $<1,2>$ is in $IEXT(1)$, which is $\{<1,2>,<2,1>\}$ and so does contain $<1,2>$ and so $I(a b c .) = true$.

$I(a c b .) = true$ if $<I(a),I(b)>$, i.e.$<1,2>$, is in $IEXT(I(c))$; but $I(c)=2$ and $IEXT$ is not defined on 2, so the condition fails and $I(a c b .) = false$.

### 2. Unlabeled nodes as existential assertions

We could treat unlabeled nodes exactly like URIs, semantically speaking, by extending the IS mapping to include them as well as URIs. That would amount to adopting the view that an unlabeled node is equivalent to a node with an unknown label. However, it seems to be more in conformance with [RDFMS] to treat unlabeled nodes as existentially bound variables. This will require some definitions, as the theory so far provides no meaning for unlabeled nodes.

Suppose $I$ is an interpretation and $A$ is a mapping from some set of unlabeled nodes to the domain of $I$, and define $I+A$ to be an extended interpretation which is like $I$ except that it uses $A$ to give the interpretation of unlabeled nodes. Define $anon(E)$ to be the set of unlabeled nodes in $E$. Then we can extend the above rules to include the two new cases that are introduced when unlabeled nodes occur in the graph:

<table>
<thead>
<tr>
<th>Condition</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>If $E$ is an unlabeled node then $<a href="E">I+A</a> = A(E)$</td>
<td></td>
</tr>
<tr>
<td>If $E$ is an RDF graph then $I(E) = true$ if $<a href="E">I+A'</a> = true$ for some mapping $A'$ from $anon(E)$ to $IR$, otherwise $I(E) = false$.</td>
<td></td>
</tr>
</tbody>
</table>

This effectively treats all unlabeled nodes as existentially quantified in the RDF graph in which they occur. Notice that since two nodes cannot have the same label, there is no need to specify the 'scope' of the quantifier within a graph. (However, it is local to the graph.) If we were to apply the semantics directly to N-Triples syntax, we would need to indicate the quantifier scope, since in this lexicalization syntax the same bNode identifier may occur several times. The above rule amounts to the N-triple convention that would place the quantifiers just outside, i.e. at the outer edge of, the N-triple document corresponding to the graph.

Notice that we have not changed the definition of an interpretation. The same interpretation that provides a truth-value for ground graphs also assigns truth-values to graphs with unlabeled nodes, even though it provides no interpretation for the unlabeled...
nodes themselves. Notice also that the unlabeled nodes themselves are perfectly well-defined entities with a robust notion of identity; they differ from other nodes only in not being assigned a direct model-theoretic interpretation, which means that they have no 'global' meaning (i.e. outside the graph in which they occur).

2.1. Comparison with formal logic

With this semantics, an RDF graph can be translated directly into a logical expression with the same meaning.

Each node labeled with a URI translates to a logical constant which is its label; each unlabeled node to a distinct variable. An arc labeled with p from a node n1 to a node n2 maps to an atomic assertion that the relation p holds true between the expressions s and o gotten by translating n1 and n2 respectively (written as (p s o) in KIF syntax); and finally, an RDF graph translates to the existential closure of the conjunction of the translations of all the arcs in the graph.

For example, the graph defined by the following N-triples document:

```
_:x a b .
c b _:x .
_:x a c .
```

translates to the logical expression (written in [KIF] syntax)

```
(exists (?y)(and (a ?y b)(b c ?y)(a ?y c)))
```

Notice that the resulting expression may contain the same symbol in both relation and object positions (e.g. 'b' in this example), which is considered syntactically illegal in many versions of logic. To map to a more conventional logical syntax, translate the triple

```
s p o .
```

into the form

```
(holds p s o)
```

where 'holds' is a three-place relation used to assert that a binary relation holds between two arguments. The above example would then map to the expression:

```
(exists (?y)(and (holds a ?y b)(holds b c ?y)(holds a ?y c)))
```

This use of 'holds' corresponds quite closely to the property/extension contrast in this model theory, since the first argument may be considered to be the relation itself, and (holds a b c) can be read as meaning ((IEXT a) b c).
3. Entailment in RDF

We say that I satisfies E if I(E)=true, and that a set S of expressions entails E if every interpretation which satisfies every member of S also satisfies E'. If {E} entails E' we will say that E entails E'.

Any process or technique which constructs a graph E from some other graphs S is said to be valid iff S always entails E, otherwise invalid. (Note that being an invalid process does not mean that the conclusion is false, and being valid does not guarantee truth. However, validity represents the best guarantee that any assertional language can offer: if given true inputs, it will never draw a false conclusion from them.)

In this section we give a few basic results about entailment in RDF. (Proofs are omitted. Note, these results do not take account of extra semantic restrictions imposed by RDFS; they apply to 'pure' RDF entailment only.) Since the language is so simple, entailment can be recognized by relatively simple syntactic comparisons. The two basic forms of valid proof step in RDF are, in logical terms, the inference from (P and Q) to P, and the inference from (foo baz) to (exists (?x) (foo ?x)).

**Conjunction Lemma.** If E is ground, then I satisfies E iff it satisfies every triple in E.

I.e. a ground graph is equivalent in meaning to the logical conjunction of its component triples.

To give a syntactic characterization of entailment we will need to define some relationships between RDF graphs. If E is an RDF graph, say that E' is a subgraph of E when every node and arc in E' are also in E. This corresponds to taking a subset of the triples constituting the graph. Obviously any subgraph of a tidy graph is tidy.

The following is an immediate consequence of the conjunction lemma:

**Subgraph Lemma.** If E and E' are ground, then E entails E' iff E' is a subgraph of E.

The following is proven by a (simple version) of the technique used to prove Herbrand's theorem in first-order logic, hence the name:

**Herbrand Lemma.** Any RDF graph has a satisfying interpretation.

This means that there is no such thing as an inconsistency or a contradiction in RDF, which is not surprising since the language does not contain negation. In fact, a stronger result, from which many of the subsequent results can be derived, is true:

**Strong Herbrand Lemma.** Any RDF graph has an interpretation which satisfies the graph and does not satisfy any ground triple not in the graph.

This emphasizes the extent to which ground RDF triples are independent of each other;
contrast this situation with a logic in which implications can be expressed. (However, RDFS does introduce some mutual constraints among ground triples.)

### 3.1 Skolemization

Skolemization is a syntactic transformation routinely used in automatic inference systems in which existential variables are replaced by 'new' functions applied to universal variables. While not itself strictly a valid operation, skolemization adds no new content to an expression, in the sense that a skolemized expression has the same entailments as the original expression provided they do not contain the new skolem functions.

In RDF, skolemization simplifies to the special case where an existential variable is replaced by a 'new' constant name, i.e. a URI which is guaranteed to not occur anywhere else. The following lemma shows that skolemization has the same properties in RDF as it has in conventional logics.

Say that $E'$ is an instance of $E$, with respect to a set $V$ of URIs, when $E'$ is like $E$ except that some of the unlabeled nodes in $E$ have been labeled with members of $V$. Obviously if $E$ is ground then it is its only instance. Notice that an instance may be untidy. A tidy ground instance with respect to any set which is disjoint from vocab($E$) is called a skolemization of $E$.

**Skolemization Lemma.** Suppose $sk(E)$ is a skolemization of $E$ with respect to $V$. Then $SK(E)$ entails $E$; and if $SK(E)$ entails $F$ and vocab($F$) is disjoint from $V$, then $E$ entails $F$.

### 3.2 Merging RDF graphs

Suppose $S$ is a set of RDF graphs, then their merge is the graph obtained by taking the union of the sets of all their triples and tidying the resulting set, i.e. identifying nodes labeled with the same URI or literal. Notice that one does not, in general, obtain the merge of a set of graphs by concatenating their corresponding N-triples documents and constructing the graph described by the merged document, since if some of the documents use the same bNode identifier, the merged document will describe a graph in which some of the unlabeled nodes have been merged.

**Merging lemma.** The merge of a set $S$ is entailed by $S$, and entails every member of $S$.

Notice that unlabeled nodes are not identified with other nodes in the merge, and indeed this reflects a basic principle of RDF graph inference: nodes with the same URI must be identified (i.e. the graph must be tidy), but unlabeled nodes must not be identified with other nodes or re-labeled with URIs, in order to ensure that the resulting graph is entailed by what one starts with. This is made precise in the following two lemmas (which follow directly from the strong Herbrand lemma):

**Anonymity lemma 1.** Suppose $E'$ is like $E$ except that at least one unlabeled node in $E$ is
labeled with a URI in E’. Then E does not entail E’.

**Anonymity lemma 2.** Suppose that E’ is like E except that two distinct unlabeled nodes in E have been identified in E’. Then E does not entail E’.

This means that there is no valid RDF inference process which can produce an RDF graph in which a single unlabeled node occurs in triples originating from several different graphs. (Of course, such a graph can be constructed, but it will not be entailed by the original documents. It must reflect the addition of new information about the identity of two unlabeled nodes. Such information can be represented in many other ways, e.g. in DAML+OIL, but it cannot be represented in RDF itself.)

The main result for RDF inference is:

**Interpolation Lemma.** S entails E iff a subgraph of the merge of S is an instance of E.

The interpolation lemma completely characterizes RDF entailment in syntactic terms. To tell whether a set of RDF graphs entails another, find a subgraph of their merge and replace URIs by unlabeled nodes to get the second. Of course, there is no need to actually construct the merge. If working backwards from the consequent E (the graph that may be entailed by the others), the most efficient technique would be to treat unlabeled nodes as variables in a process of subgraph-matching, allowing them to bind to 'matching' URI labels in the antecedent graph(s) in S, i.e. those which may entail the consequent graph. The interpolation lemma shows that this process is valid, and is also complete if the subgraph-matching algorithm is. The existence of complete subgraph-checking algorithms also shows that RDF is decidable, i.e. there is a terminating algorithm which will determine for any finite set S and any expression E, whether or not S entails E.

Notice however that such a variable-binding process would only be appropriate when applied to the conclusion of a proposed entailment. This corresponds to using the document as a goal or a query, in contrast to asserting it, i.e. claiming it to be true. If an RDF document is asserted, then it would be invalid to bind new values to any of its unlabeled nodes, since (by the anonymity lemmas) the resulting graph would not be entailed by the assertion.

It might be thought that the operation of changing a bound variable would be an example of an inference which was valid but not covered by the interpolation lemma, e.g. the inference of

```
_:xxx foo baz .
```

from

```
_:yyy foo baz .
```

Recall however that by our conventions, these two N-triples documents describe the same RDF graph, and any graph is both a subgraph and an instance of itself.
4. RDFS interpretations

RDF Schema [RDFS] introduces a special RDF vocabulary, with some fixed constraints on its interpretation; in particular, the notion of a 'class'. We can define all of RDFS in terms of a single RDF property rdf:type which relates entities to the classes which contain them. Following conventional usage, we understand a classification into a 'type' to mean membership in a set; the sets will be considered to be the extensions of resources called 'classes', using the same technique used earlier to relate properties to their extensions. RDFS provides names for several of the classes whose extensions we have already used in defining the model theory (IR, LV, IP) and a general naming scheme to allow user-defined classes. The one new class is the class of all classes, whose extension - the set of all classes - will be called IC. It will be convenient to define a mapping ICEXT (for the Class Extension in I) from classes to their extensions, in terms of the relational extension of rdf:type, as follows:

\[
\text{ICEXT}(x) = \{ y \mid <y,x> \text{ is in IEXT(I(rdf:type))} \}
\]

I.e. the class extension of x is the set of things that are related to x by the property rdf:type, i.e. which have the class x as their 'type'. This definition is restated below as the semantic rule for rdf:type.

An rdfs interpretation is then an RDF interpretation plus one extra subset of the universe, forming a fifth condition on an interpretation:

5. A subset IC of IR, containing classes

An rdfs interpretation I satisfies the following conditions, in addition to those for an rdf interpretation given earlier. The fifth of these is a restatement of the definition of ICEXT. The subsets referred to in subPropertyOf are sets of pairs.

Note, these definitions for rdf:domain and rdf:range reflect our current understanding of multiple domain and range restrictions rather than the wording in [RDFS] . They correspond more closely to the meanings assumed by DAML+OIL.[DAML]
ICEXT(I(rdfs:Resource)) = IR
ICEXT(I(rdf:Property)) = IP
ICEXT(I(rdfs:Class)) = IC
ICEXT(I(rdfs:Literal)) = LV

<x,y> is in IEXT(I(rdf:type)) iff x is in ICEXT(y)
<x,y> is in IEXT(I(rdfs:subClassOf)) iff ICEXT(x) is a subset of ICEXT(y)
<x,y> is in IEXT(rdfs:subPropertyOf) iff IEXT(x) is a subset of IEXT(y)
I(rdfs:ConstraintResource) is in IC
ICEXT(I(rdfs:ConstraintProperty)) is the intersection of
ICEXT(I(rdfs:ConstraintResource)) and IP
I(rdf:range) and I(rdf:domain) are in ICEXT(I(rdfs:ConstraintProperty))

If <x,y> is in IEXT(I(rdf:range)) and <u,v> is in IEXT(x) then v is in ICEXT(y)
If <x,y> is in IEXT(I(rdf:domain)) and <u,v> is in IEXT(x) then u is in ICEXT(y)

The model theory imposes no conditions on the interpretations of rdfs:seeAlso,
rdfs:isDefinedBy, rdfs:comment or rdfs:label.

## 5. RDFS entailment

RDFS introduces new constraints which must hold between RDF graphs. For example,
the fact that the subset relationship is transitive means that two subClassOf assertions may
entail a third, different, triple. It would seem, therefore, that RDFS entailment is
fundamentally different from RDF entailment, since it does not satisfy the strong
Herbrand lemma. However, rather than redefine the notions of entailment and validity for
RDFS, we can think of RDFS as defining a notion of 'closure' on an RDF vocabulary,
thereby reducing RDFS entailment to the RDF case.

Suppose E is an RDFS graph (i.e. an RDF graph possibly using RDFS terminology), then
its schema-closure is the graph gotten by adding triples according to the following rules:

1. Add the following triples (which are true in any RDFS interpretation):
   
rdfs:Resource rdf:type rdfs:Class.
2. Apply the following rules recursively to generate all legal RDF triples (i.e. until none of the rules apply or the graph is unchanged.) Here **xxx**, **aaa**, etc. stand for any uriref, bNode or literal expression, and **uuu** for any uriref or bNode (but not a literal).

<table>
<thead>
<tr>
<th>If E contains:</th>
<th>then add:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a xxx aaa yyy .</td>
<td>xxx rdf:type rdfs:Resource .</td>
</tr>
<tr>
<td>1b xxx aaa yyy .</td>
<td>aaa rdf:type rdf:Property .</td>
</tr>
<tr>
<td>1c xxx aaa uuu .</td>
<td>uuu rdf:type rdfs:Resource .</td>
</tr>
<tr>
<td>2 aaa rdfs:subPropertyOf bbb .</td>
<td>aaa rdfs:subPropertyOf ccc .</td>
</tr>
<tr>
<td>bbb rdfs:subPropertyOf ccc .</td>
<td></td>
</tr>
<tr>
<td>3a aaa rdfs:subPropertyOf bbb .</td>
<td>aaa rdf:type rdf:Property .</td>
</tr>
<tr>
<td>3b aaa rdfs:subPropertyOf bbb .</td>
<td>bbb rdf:type rdf:Property .</td>
</tr>
<tr>
<td>4 xxx aaa yyy .</td>
<td>xxx bbb yyy .</td>
</tr>
<tr>
<td>aaa rdfs:subPropertyOf bbb .</td>
<td></td>
</tr>
<tr>
<td>5 xxx aaa yyy .</td>
<td>xxx rdf:type zzz .</td>
</tr>
<tr>
<td>aaa rdf:domain zzz .</td>
<td></td>
</tr>
<tr>
<td>6 xxx aaa uuu .</td>
<td>uuu rdf:type zzz .</td>
</tr>
<tr>
<td>aaa rdf:range zzz .</td>
<td></td>
</tr>
<tr>
<td>7 xxx rdf:type uuu .</td>
<td>uuu rdf:type rdfs:Class .</td>
</tr>
<tr>
<td>8 xxx rdf:type rdfs:Class .</td>
<td>xxx rdfs:subClassOf rdfs:Resource .</td>
</tr>
<tr>
<td>9a xxx rdfs:subClassOf yyy .</td>
<td>xxx rdf:type rdfs:Class .</td>
</tr>
<tr>
<td>9b xxx rdfs:subClassOf uuu .</td>
<td>uuu rdf:type rdfs:Class .</td>
</tr>
<tr>
<td>10 xxx rdfs:subClassOf yyy .</td>
<td>xxx rdfs:subClassOf zzz .</td>
</tr>
<tr>
<td>yyy rdfs:subClassOf zzz .</td>
<td></td>
</tr>
<tr>
<td>11 xxx rdfs:subClassOf yyy .</td>
<td>aaa rdf:type yyy .</td>
</tr>
<tr>
<td>aaa rdf:type xxx .</td>
<td></td>
</tr>
<tr>
<td>12 aaa rdf:type</td>
<td>aaa rdf:type rdfs:ConstraintProperty .</td>
</tr>
<tr>
<td>rdfs:ConstraintResource .</td>
<td></td>
</tr>
<tr>
<td>aaa rdf:type rdf:Property .</td>
<td></td>
</tr>
</tbody>
</table>

It is easy to see that this process will terminate on any finite RDF graph, since there are
only finitely many possible triplets that can be formed from a given finite vocabulary. For example, the schema-closure of the graph consisting of the single triplet

foo bar baz.

contains the following N-triples:

foo bar baz.

foo rdf:type rdfs:Resource.

baz rdf:type rdfs:Resource.

bar rdf:type rdf:Property.

dfd:type rdf:type rdf:Property.

rdf:Property rdf:type rdfs:Class.

rdf:Property rdfs:subClassOf rdfs:Resource.

(The final three triples will be generated in every schema closure.)

The schema-closure of E has all the entailments which arise from the meanings of RDFS already included in it, so it reduces RDFS entailment to RDF entailment. Say that a set S of graphs s-entails a graph E if every RDFS interpretation which satisfies S also satisfies E. Then the basic result connecting RDFS to RDF inference is:

**Schema Lemma.** S s-entails E iff the schema-closure of the merge of S entails E.

### 6. RDF containers

Basic properties of RDF containers can be stated in terms of the semantics already described. A container is a resource which is a member of one of three subclasses rdf:Bag, rdf:Seq or rdf:Alt of the class rdfs:Container, which is the domain of a countable set of properties indicated by rdf:_1, rdf:_2, etc., all of which in turn are members of the class rdfs:ContainerMembershipProperty. All of this can be said in an obvious way by suitable collections of RDF triples using the RDFS vocabulary. As mentioned earlier, uses of containers may well go beyond the rather basic meanings sanctioned by this model theory. For example, with this understanding of containers, a triple with a container as a subject does not entail any assertion with a member of the container as a subject.
### Appendix A: Summary of model theory

#### RDF/RDFS model theory summary

<table>
<thead>
<tr>
<th>0. Domains and mappings of interpretation I</th>
</tr>
</thead>
<tbody>
<tr>
<td>vocab(I): set of URIs ; LV: (global) set of literal values ; IR: set of resources (universe); IP: subset of IR (properties) ; IC: subset of IR (classes).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>XL: literals -&gt; LV; IS: vocab(I) -&gt; IR; IEXT: IP -&gt; subsets of (IR x (IR union LV)); ICEXT: IC -&gt; subsets of IR</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>1. Basic equations</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>E is:</strong></td>
</tr>
<tr>
<td>a literal</td>
</tr>
<tr>
<td>a uri (uriref)</td>
</tr>
<tr>
<td>an asserted triple &lt;s p o&gt;</td>
</tr>
<tr>
<td>any other triple</td>
</tr>
<tr>
<td>a ground RDF graph</td>
</tr>
<tr>
<td>an unlabeled node (bNode)</td>
</tr>
<tr>
<td>an RDF graph</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2. Class extensions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>E is:</strong></td>
</tr>
<tr>
<td>rdfs:Resource</td>
</tr>
<tr>
<td>rdf:Property</td>
</tr>
<tr>
<td>rdfs:Class</td>
</tr>
<tr>
<td>-----------------</td>
</tr>
<tr>
<td>rdfs:Literal</td>
</tr>
<tr>
<td>rdfs:ConstraintResource</td>
</tr>
<tr>
<td>rdfs:ConstraintProperty</td>
</tr>
</tbody>
</table>

3. Property extensions

E is:  
I(E) is in IP; <x,y> is in IEXT(I(E)) iff:

<table>
<thead>
<tr>
<th>rdf:type</th>
<th>x is in ICEXT(y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>rdfs:subClassOf</td>
<td>ICEXT(x) is a subset of ICEXT(y)</td>
</tr>
<tr>
<td>rdfs:subPropertyOf</td>
<td>IEXT(x) is a subset of IEXT(y)</td>
</tr>
</tbody>
</table>

E is:  
I(E) is in CP; <x,y> is in IEXT(I(E)) iff:

<table>
<thead>
<tr>
<th>rdfs:range</th>
<th>if &lt;u,v&gt; is in IEXT(x) then v is in ICEXT(y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>rdfs:domain</td>
<td>if &lt;u,v&gt; is in IEXT(x) then u is in ICEXT(y)</td>
</tr>
</tbody>
</table>

**Acknowledgments**

This document has benefited from inputs from many members of the RDF Core Working Group. Particular thanks to Dan Connolly for clarifying the relationship between RDF and RDFS, Ora Lassilla for the idea of using graph syntax, Sergey Melnick for the translation into logic, and Jos deRoo and Graham Klyne for finding errors in earlier drafts.

The use of an explicit extension mapping to allow self-application without violating the axiom of foundation was suggested by Chris Menzel. Peter Patel-Schneider found an error in an earlier draft.
References

[KIF]

[Hayes&Menzel]
Patrick Hayes, Christopher Menzel, A Semantics for the Knowledge Interchange Format, 6 August 2001 (Proceedings of 2001 Workshop on the IEEE Standard Upper Ontology)

[DAML]
Frank van Harmelen, Peter F. Patel-Schneider, Ian Horrocks (editors), Reference Description of the DAML+OIL (March 2001) ontology markup language

[RDF/XML]

[RDFMS]

[RDFSchema]

[RDFTestCases]
Art Barstow, Dave Beckett (editors), RDF Test Cases, 12 September 2001 (W3C Working Draft).

[RFC 2396]