Two Proposals for a Semantic Web Ontology Language

Peter F. Patel-Schneider
Bell Labs Research, Murray Hill, New Jersey, U. S. A.
email: pfps@research.bell-labs.com

Abstract

Recent investigations have uncovered some problems in the relationship between DAML+OIL and the new RDF model theory. There are several ways of remedying these problems, resulting in greater or lesser compatability with RDF. One possible remedy is to diverge from the RDF philosophy only on statements about descriptions, resulting in a complex web ontology language containing a description logic and that is compatible with RDF. Another possible remedy is to remain compatible with RDF only on statements about individuals, resulting in a semantic web ontology language that is very close to standard description logics, but that is not fully compatible with RDF.

1 Introduction

Over the last year or so, several efforts have been underway to develop or revise representation formalisms suitable for use in the semantic web. These efforts include

- the revision of the Resource Description Framework (RDF) [7, 8] and the development of a model theory for RDF [5],
- the development of the DAML+OIL representation formalism [1, 6], and
- the W3C Web Ontology (WebOnt) Working Group [10], which is chartered to develop a formalism for representing ontology information in the semantic web.

In the course of developing a revision of DAML+OIL for presentation to the WebOnt Working Group, I came across some problems in the relationship between DAML+OIL and the newer RDF model theory. I designed two variations of DAML+OIL, one being an extension of RDF and one being very similar to standard description logics.

2 RDF and DAML+OIL

RDF is a very simple representation formalism, somewhat reminiscent of semantic networks of the 1970s. There is an extension of RDF, RDF Schema (RDFS) [2], that provides more facilities, such as a type hierarchy and domains and ranges for properties. RDF and RDFS are supposed to be the base for all semantic web representation formalisms.

Like in many of the early semantic network formalisms RDF and rdfs represent information about classes and properties, such as class memberships and domains and ranges of properties, in the same way that regular information is represented, i. e., as objects and relationships between these objects. In particular, RDF has a relationship, `rdf:type`, that represents class membership, and and RDFS has several objects that are metaclasses, including the class of all objects, `rdfs:Resource`, and the class of all classes, `rdfs:Class`.

RDF has several syntaxes, including an encoding in XML [3]. (RDFS uses the same syntax as RDF.) However, an RDF or RDFS knowledge base is essentially a collection of subject-prediate-object triples. Each such triple represents a fact, such as `John loves Mary`. Because information about classes is represented in the same way, some triples also have extra meaning. For example, `Student rdfs:subclassOf Person` represents a fact, namely a `rdfs:subclassOf` relationship between the object `Student` and the object `Person`, as well as representing a subclass relationship between two classes, and `John rdf:type Person` represents both a fact and the membership of `John` in the class `Person`.

Initally RDF and RDFS had only an informal semantics. Recently, however, a full model-theoretic semantics for RDF and RDFS has been developed by the W3C RDF Core Working Group [5] including a definition of entailment for both RDF and RDFS. The model theory accounts for the special meaning of triples like the ones above. In the model theory interpretations, every object, including classes and properties, denotes an element of the domain of discourse. Every relationship, including the instance-of and subclass relationships, shows up in interpretations.

RDF and RDFS use the same basic interpretations. RDFS interpretations have more built-in vocabulary, such as `rdfs:Class` and `rdfs:subClassOf`. RDFS interpretations also have more conditions, to ensure that the RDFS built-in vocabulary has its intended meaning.

DAML+OIL was developed to be a language for representing ontology information on the semantic web. It had to be upward compatible with RDFS and thus its syntax is given as a collection RDF triples. Even DAML+OIL descriptions are given as groups of RDF triples.

DAML+OIL has its own model theory [9], which predates the RDF model theory. Nevertheless, the DAML+OIL model theory is quite close to the RDF model theory, and can be easily modified to be an extension of the RDF model theory.

However, when looked at as a standard model theory, the DAML+OIL model theory has a problem—there are too few entailments. As an example, John being a member of the intersection of Student and Employee and European does not entail that John is a member of the intersection of Employee and Student. The basic reason for this non-entailment is that DAML+OIL classes, like RDFS classes, are also objects and thus correspond to objects in the domain of interpretations, and not all models of John being a member of the intersection of Student and Employee and European even have a class that corresponds to the intersection of Employee and Student.

This is not a big problem if DAML+OIL could be considered on its own. It would be possible to define the usual description logic notions of subsumption and membership following from a knowledge base without requiring that the descriptions and classes therein actually exist in the knowledge base. However this solution would result in incompatibility with RDF and RDFS, counter to the requirements for DAML+OIL.

3 DAML+OIL as an Extension of RDFS

Making DAML+OIL an extension of rdfs, in the same way that RDFS is an extension of RDF, requires adding extra built-in vocabulary and conditions to RDFS interpretations. Interpretations that have this extra vocabulary and satisfy the extra conditions are DAML+OIL interpretations.

Many of these extra conditions require the presence of extra classes in interpretations. For example, an interpretation that contains classes `Student`, `Employee`, and `European` will also contain classes for all their intersections and unions as well as many other classes. Then if an interpretation makes `John` belong to the intersection of `Student` and `Employee` and `European`, that interpretation will have a class for the intersection of `Employee` and `Student`, and will have `John` as an instance. In this way the entailments missing from the model theory for DAML+OIL are incorporated into the model theory.

More classes than just unions and intersections have to exist in this way. DAML+OIL restrictions, like the restriction consisting of those objects that have at most one friend, also have to exist. If they do not then there will be missing entailments. So there must be a condition that requires that if a property is in an interpretations, then so is a restriction that requires of its instances that they have at most one of that property.

So far so good. However, what about classes that reference themselves, such as defining `Person` to be a subset of `Animal` that has all friends of `Person`s be `Person`s? For the same reasons, such self-referential classes, and restrictions, have to also exist in many interpretations even if they are not explicitly referenced.[1]

But now there is serious trouble. Remember that class membership in RDF and RDFS is just another property, so it is possible to use it in the same way that other properties, like `friend`, can be used. So it is possible to create a restriction whose definition states that it contains precisely those objects that do not belong to it. One such class, using a slight modification of DAML+OIL syntax, is

```
<Restriction rdf:about="_:1" maxCardinalityQ="0">
  <onProperty rdf:resource="rdf:type" />
  <hasClassQ>
     <oneOf rdf:parseType="daml:collection">
        <Restriction rdf:resource="_:1" />
     </oneOf>
  </hasClassQ>
</Restriction>
```

As this restriction only mentions vocabularly that is in all DAML+OIL interpretations, all DAML+OIL interpretations will have restrictions like this.

However, it is impossible to determine membership in such a restriction. If an object is an instance of this restriction, then it can't have an rdf:type relationship to the restriction, and thus it is not an instance of the restriction. If an object is not an instance of this restriction, then it must have an rdf:type relationship to the restriction, and thus it is an instance of this restriction.

This shows that that no DAML+OIL interpretation can have a restriction like this one. But all DAML+OIL interpretations must have a restriction like this one. Therefore there are no DAML+OIL interpretations, and so it is not possible to make DAML+OIL an extension of RDFS in this way.

4 Fixing the Situation

There are several ways out of this problem.

First, it would be possible, as suggested above, to just have DAML+OIL incompatible with RDFS. In essence, DAML+OIL would use RDF for its syntax, but have differing meanings. This is not a viable option, as RDF and RDFS are supposed to be the semantic base of the semantic web. Besides, the syntax of RDF has its own problems.

Another approach would be to simply give up on RDF and RDFS and to base a semantic web ontology language directly on XML. This would be easy to do, and has essentially been done in the Ontology Inference Layer (OIL) [4], which

[1]Note that this means that there has to be many classes that are subsets of "Animal" and that have their friends belong to the class.

has an XML syntax. However, this approach also violates the requirement that RDF and RDFS be the semantic base of the semantic web.

A third approach would be to build an extension of RDFS, in the same sense as first-order logic is an extension of propositional logic. This formalism would include RDFS as a subset, in a very strong sense. Given RDFS inputs, the formalism would work exactly the same as RDFS in all respects, at least for those RDFS constructs that have a well-defined meaning.[2] However, there would be syntactically-valid inputs for this formalism that are not syntactically-valid RDF.

5 A Complex Semantic Web Ontology Language

What could such a formalism look like? Well one possibility is to include the class constructs of DAML+OIL, but to use XML syntax for them instead of the RDF syntax used in DAML+OIL. So, one might say

```
<Class rdf:about="Foo">
  <rdfs:subClassOf>
    <toClass property="friend" class="Student"/>
  </rdfs:subClassOf>
</Class>
```

Looking at this as an extension of RDFS, one might be tempted to think of it as creating two RDF classes, one for `Foo` and one for `<toClass property="friend" class="Student"/>`, and relating them via an `rdfs:subClassOf` link. However, this approach would result in the above semantic paradox. Instead such constructs are treated as requiring that the class extension of the object `Foo` be a subset of the description logic-like extension of `<toClass property="friend" class="Student"/>`.

Constructs that are valid RDF are treated in the RDF manner so

```
<Class rdf:about="Foo">
  <rdfs:subClassOf>
    <Class rdf:about="Bar"/>
  </rdfs:subClassOf>
</Class>
```

results in two class objects, `Foo` and `Bar`, and a `rdfs:subClassOf` relationship between them.

In this way an extension of RDFS can be created. RDFS-like constructs are treated in the RDFS manner, resulting in object classes and relationships between them. Constructs that contain description logic-like constructs are treated

[2]Currently this excludes RDF reification and collections.

differently, so that classes corresponding to them do not have to be part of interpretations. The details of the construction are too complex to be given in full here, but can be found at http://lists.w3.org/Archives/Public/www-webont-wg/2002Jan/att-0061/01-swol.text.

In order to retain maximum compatibility with RDFS, DAML+OIL built-in classes and properties, such as `TransitiveProperty`, are class objects in the domain of interpretations just like `rdf:Property`. The meanings of these classes and properties are captured in conditions on interpretations, just as they are in the model theory for RDFS.

6 Remaining Problems

The situation is not as good as it could be, however. Because interpretations contain all the RDFS stuff, they have to have all the conditions from the RDFS model theory, which is considerably more complex than standard model theories. Interpretations are even more complex than required by RDFS because they also contain the DAML+OIL built-in classes, such as `TransitiveProperty`, and their semantic conditions. All this complexity means that it is possible that there are semantic paradoxes still present.

Further, such classes can show up in conditional constructs. For example, one could say

```
<rdf:Property rdf:about="bar">
  <rdfs:subPropertyOf rdf:resource="bbb" />
</rdf:Property>

<rdf:Property rdf:about="baz">
  <rdfs:subPropertyOf rdf:resource="bbb" />
</rdf:Property>

<Person rdf:about="Jake">
  <rdf:type>
     <Restriction property="foo" toClass="TransitiveProperty">
  </rdf:type>
  <rdf:type>
     <Restriction property="foo">
       <hasClass>
         <oneOf> bar baz </oneOf>
       </hasClass>
  </rdf:type>
</Person>

<Person rdf:about="John">
```

```
  <bar rdf:resource="Jill" />
  <bar rdf:resource="Mary" />
</Person>
<Person rdf:about="Jill">
  <bar rdf:resource="Susan" />
</Person>
<Person rdf:about="Mary">
  <bar rdf:resource="Susan" />
</Person>
```

which would entail that `John` was `bbb`-related to `Susan` because either `bar` or `baz` is transitive.

This sort of construct would enormously complicate reasoners for the formalism.

7 A Simpler Semantic Web Ontology Language

So the above situation is definitely not ideal. However, it appears that there is no ideal solution. It is not possible to completely follow the RDF philosophy of having all constructs show up in interpretations because that would result in semantic paradoxes. Following the RDF philosophy to the maximum extent possible results in a complex formalism that is hard to reason with, and may still be subject to paradoxes.

Perhaps it would be better to abandon the RDF philosphy and instead move to something more like description logic, but with XML or frame-like syntax. This, of course, is the approach taken in OIL.

An approach that is somewhat closer to the RDF philosophy is to retain RDF for statements about objects, but abandon RDF's inclusion of classes in the domain. This results in a web ontology language like the one described in http://lists.w3.org/Archives/Public/www-webont-wg/2002Jan/att-0061/02-swol-xml-rdf.text. that is very similar to standard description logics, differing only on surface syntax and a few minor semantic details.

This formalism has a *much* simpler model theory than the previous formalism. Gone are all the complex RDFS conditions on interpretations. Also gone are all the even-more-complex conditions on the DAML+OIL vocabulary.

Some price is paid for this, of course. New syntactical constructs have to be added for the definition of classes and different kinds of properties, as this can no longer be done using the RDF syntax. So instead of saying

```
<Property rdf:about="foo">
  <rdfs:domain rdf:resource="bar" />
</Property>
```

one has to use the specific construct

```
<domain>
  foo
  bar
</domain>
```

It is also not possible to query some meta-theoretic notions, so asking whether

```
<Property rdf:about="foo">
  <rdf:type rdf:resource="TransitiveProperty" />
</Property>
```

is entailed does not ask whether `foo` is transitive. However, such information can be obtained via

```
<TransitiveProperty rdf:about="foo" />
```

8 Discussion

It thus appears that there are two possible routes for developing a semantic web ontology language. The first route retains maximum RDFS compatability, but results in a complex formalism with difficult inference. The second route loses some RDFS compatability, but results in a much more familiar and simpler formalism that probably can be effectively implemented.

One task of the W3C WebOnt Working Group will be to decide which approach is better.

References

[1] DAML+OIL language. http://www.daml.org/language/, 2001.

[2] Resource description framework (RDF) schema specification 1.0. W3C Candidate Recommendation, 27 March 2000, http://www.w3.org/TR/rdf-schema, March 2000.

[3] Extensible markup language (XML). http://www.w3.org/XML/, 2001.

[4] Dieter Fensel, Ian Horrocks, Frank van Harmelen, Deborah L. McGuinness, and Peter F. Patel-Schneider. OIL: An ontology infrastructure for the semantic web. *IEEE Intellligent Systems*, 16(2), May 2001.

[5] Patrick Hayes. RDF model theory. W3C Working Draft, http://www.w3.org/TR/rdf-mt/, 2001.

[6] Ian Horrocks and Peter F. Patel-Schneider. The generation of DAML+OIL. In Carole Goble, Deborah L. McGuinness, Ralf Moeller, and Peter F. Patel-Schneider, editors, *Working Notes of the 2001 International Workshop on Description Logics*, July 2001. Available electronically as a CEUR publication at http://SunSite.Informatik.RWTH-Aachen.DE/Publications/CEUR-WS/.

[7] Resource description framework (RDF): Model and syntax specification. W3C Recommendation, 22 February 1999, http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/, February 1999.

[8] RDFCore working group. http://www.w3.org/2001/sw/RDFCore/, 2001.

[9] Frank van Harmelen, Ian Horrocks, and Peter F. Patel-Schneider. A model-theoretic semantics for DAML+OIL (March 2001). http://www.w3.org/TR/daml+oil-model, December 2001.

[10] Web-ontology (WebOnt) working group. http://www.w3.org/2001/sw/WebOnt/, 2001.