Patching Syntax in OWL Ontologies

Sean Bechhofer[1] and Raphael Volz[2]

[1] University of Manchester, UK
http://www.cs.man.ac.uk
seanb@cs.man.ac.uk
[2] Institute AIFB, University of Karlsruhe, Germany
http://www.aifb.uni-karlsruhe.de
rvo@aifb.uni-karlsruhe.de

Abstract. An analysis of OWL ontologies represented in RDF/XML on the Web shows that a majority are OWL Full. In many cases this may not be through a desire to use the expressivity provided by OWL Full, but is rather due to syntactic errors or accidental misuse of the vocabulary. We present a "rogues gallery" of common errors encountered, and describe how robust parsers that attempt to cope with such errors can be produced.

1 Introduction

In Feb 2004, the World Wide Web Consortium (W3C) announced the release of the OWL Web Ontology Language Standard[3]. OWL is intended to provide the "Ontology Layer" of the oft-cited Semantic Web stack and in order to fulfill this, has an RDF carrier syntax [8] that will allow RDF-aware applications to make use of data marked up in OWL.

The publication of the OWL Recommendation does not, of course, mean that our work is done in terms of Ontology Languages. We must now support users and application builders in the development of ontologies and applications that use them. This is the rationale behind activities such as the Semantic Web Best Practices Working Group[4].

A key aspect of the OWL design is the layering of different language subspecies. All OWL documents can be represented using the same RDF/XML carrier concrete syntax, and OWL has three language variants with increasing expressivity: Lite, DL and Full. All Lite documents are DL, and all DL documents are Full, but the converse is not true. The constraints on the expressivity of OWL DL have been made in such a manner as to produce a language for which there exists sound and complete decision procedures. The task of species recognition – determining which particular species an ontology is in – thus becomes important, as an OWL DL application needs to know whether it can apply the appropriate semantics and inference procedures to an ontology. See [2] for a detailed discussion of the issues concerning parsing and recognition.

[3] http://www.w3.org/News/2004##item14
[4] http://www.w3.org/2001/sw/BestPractices/

The designers of OWL expect that a significant number of ontologies will use the Lite and DL subspecies. However, in a simple analysis reported here, we discover that the proportion of OWL Lite and DL ontologies currently on the web is small. It appears, however, that the reasons for this are often due to accidental syntactic errors or vocabulary misuse rather than modelling intention. For example, the rules about OWL DL are strict in their requirements for explicit typing of all classes used in the ontology. Such problems can be tackled through provision of tools supporting the production of ontologies, which ensure that correct and appropriate syntax is used.

As a parallel approach, we propose the use of robust parsers that can cope with syntactic errors and apply "fixups" or patches (as initially discussed in [2]) wherever necessary[5] in order to produce OWL DL ontologies. This approach is predicated on the assumption that we *are* interested in working with OWL DL ontologies – a hypothesis that we feel is valid (the number of implementors building OWL DL and Lite reasoners[6] bears witness to this).

The paper is structured as follows. Section 2 describes an experiment in analysing a number of Web ontologies in order to determine which particular species they fell into. Section 3 provides an analysis of the detailed reasons for membership (or non-membership) of a particular species – these reasons can be classified into a number of more general reasons (such as missing type triples) providing us with a rogues gallery of the "usual suspects" or kinds of syntactic malforms we may encounter that contribute towards non-membership of DL or Lite. Section 4 describes possible patches that can be applied to these classes of error in order to obtain OWL DL or Lite ontologies. Section 5 describes extensions to an OWL parser that provides a more robust tool that will attempt to cope with a number of different "error" conditions. We conclude with remarks on related work and a summary of our contribution.

2 Ontology Sources

Our initial motivation was to answer the question "how much OWL is there on the Web?" Of course, any RDF graph is, in fact, an OWL (Full) graph, but such documents can not necessarily be used with tools targeted at OWL DL or Lite. Thus, an alternative, and perhaps more interesting, question is "how much *OWL DL* is there on the Web?"

2.1 Ontology library

Our first port of call was the largest library of ontologies in the Web - the DAML ontology library[7] - which contained 282 ontologies of various formats at the time of our experiment. Of these 282 we selected those which we considered

[5] Of course, as discussed in the paper, we must be careful when applying patches in such situations as the semantics may be impacted.

[6] `http://www.w3.org/2001/sw/WebOnt/impls`

[7] `http://daml.org/ontologies/`

Collection	Total	Full	DL	Lite
DAML Library	76	63 (83%)	3 (4%)	10 (13%)
Google	201	174 (86%)	19 (10%)	8 (4%)

Table 1. Species Breakdown

to be *candidates* for OWL DL ontologies. Our heuristic to judge whether an ontology is a candidate was based on usage of the OWL namespace. If an RDF graph fails to mention the OWL namespace, we can be sure that it will not be an OWL DL graph, as there can be no explicit typing as required by the OWL DL rules[8].

Unfortunately most of the ontologies in this library (193 ontologies) were definitely not candidates as they were no longer available, had XML errors, or used previous Web ontology languages such as OIL or DAML-ONT. Seventy-six ontologies, however, were available, were valid XML, and used elements from the OWL vocabulary.

2.2 Google

This small set of 76 ontologies was not considered satisfactory, so we chose to expand our collection by searching for OWL ontologies using Google. A simple search for `OWL` is of course too broad and leads to more than 4 million web pages, most of which are not ontologies. Two refined searches for `filetype:.owl owl` and `filetype:.rdf owl` provided 264[9] more appropriate documents, of which 56 documents were suitable, e.g. they were HTML pages, and 15 documents were no longer available online. In conclusion we actually found 201 candidate ontologies. Obviously, these ontologies do not necessarily constitute all available ontologies, since we missed those that departed from the syntactic search pattern used or were not made publicly available[10].

3 Analysis

A breakdown of the species distribution of the candidate ontologies from the DAML Library and Google search is shown in Table 1. This initial analysis reveals that a majority of the ontologies found are not directly usable by OWL DL or OWL Lite-aware systems. Our hypothesis, however, is that most of these ontologies are not intentionally OWL Full, but become OWL Full through various errors made by the designer of the ontologies. We therefore had a closer

[8] An exception to this, of course, is the trivially empty graph, relatively uninteresting in this context.

[9] The reader may note that the Google estimate displayed with the search results is not correct.

[10] For example ontologies held in repositories such as Sesame [4] will not necessarily be apparent to Google unless the owners of the repository have taken steps to publish information about them.

look at the 174 OWL Full ontologies found by Google and the 63 OWL Full ontologies found in the DAML library[11]. We could identify 20 different reasons why those ontologies were OWL Full (cf. Table 2). As discussed before, these reasons are seen as errors for OWL DL and OWL Lite processors. The errors can be aggregated into five more general categories of fault, which are described below.

Note that here we are making an assumption that our desired intention *is* that ontologies should be in the OWL DL or Lite subspecies. Thus we use the term "error" to describe a situation where an ontology does not belong to DL or Lite.

Missing typing: This category occurred most frequently and in almost all of the OWL Full ontologies. A good example for missing typing is when a property was not specified to be a datatype or object property. Missing type information often occurred together for various types of elements, for example in 154 (89%) of the OWL Full ontologies found by Google;

Namespace problems: Namespace problems occurred in about half of all full ontologies, but more frequently in the ontologies from the DAML library. Namespace problems are (i) a violation of the namespace separation mandated in OWL, (ii) usage of the OWL namespace itself (one may not declare classes, properties or instances in the OWL namespace) and a (iii) redefinition of elements of OWL itself, e.g. OWL class.

Wrong vocabulary: 32 (18%) of the OWL Full ontologies found by Google were using the wrong vocabulary. But this was the second most frequent (78%) error category for the DAML ontologies. Usage of the wrong vocabulary, for example `rdf:Class` instead of `owl:Class`, `rdf:Property` instead of `owl:Property` or usage of `owl:sameAs` on classes, can be accounted to mistakes in legacy migration. With no exceptions all of the ontologies in this category also had missing type information and namespace problems, which are both direct results of using wrong vocabulary;

OWL Full: Around 20% of the ontologies in both data sets were definitely OWL Full, since either (i) complex object properties were declared to be transitive[12] or (ii) illegal subproperty statements were made. The latter refers to making a datatype property a subproperty of an object property or vice versa[13].

OWL/RDF Irregularities: 20 (11%) of the Google ontologies and 15 (24%) of the DAML ontologies contained various OWL/RDF irregularities. This refers to (i) Unused RDF Triples, (ii) Malformed Lists, (iii) Anonymous

[11] More detailed results concerning the analysis can be found at: `http://owl.man.ac.uk/patching`.

[12] In order to ensure that OWL DL is decidable, there are extra side conditions on the use of transitive properties. *Complex properties* are those that are functional, are involved in cardinality restrictions, or have a complex inverse or superproperty. Such properties cannot be declared to be transitive. See [8] for details.

[13] With a single exception, the definitely Full ontologies found by Google also had also namespace problems and were missing type information.

Class Creation, (iv) Structure Sharing and (v) Malformed Restrictions. With no exceptions all of the ontologies in this category also had missing type information and namespace problems, which comes at no surprise since these errors are directly effected by the OWL/RDF irregularities.

Note that the categories are not disjoint – an ontology could have both missing type information and illegal sub property axioms. Note also that the summary figures (e.g. Missing Typing) summarise *any* of the subcategories referred to, e.g. untyped class, untyped individual etc.

We can identify several other problems that could theoretically occur, but were not found in the data set. These include particular RDF malforms such as malformed descriptions or `owl:AllDifferent` axioms, the use of Inverse Functional Data Properties [8], cycles in bnodes or `owl:sameAs` applied to Object or Datatype Property.

Error Type	DAML Library		Google	
	Absolute	Relative	Absolute	Relative
OWL Full (Total)	63	100%	174	100%
Missing Typing	**63**	**100%**	**167**	**96%**
Untyped Ontology	5	7%	24	13%
Untyped Object Property	34	53%	131	75%
Untyped Individual	37	58%	137	78%
Untyped Datatype	18	28%	10	5%
Untyped Data Property	18	28%	134	77%
Untyped Class	58	92%	148	85%
Untyped Functional Property	1	1%	4	2%
Namespace Problems	**41**	**65%**	**76**	**43%**
Redefinition of built in vocabulary	23	36%	50	28%
OWL Namespace Used	14	22%	35	20%
Namespace Separation Violated	33	52%	68	39%
Wrong Vocabulary	**49**	**77%**	**32**	**18%**
SameAs with Class	0	0%	2	1%
RDF Property Used	9	14%	27	15%
RDF Class Used	48	76%	23	13%
OWL Full	**13**	**20%**	**31**	**17%**
Illegal Sub Property	14	22%	28	16%
Complex role declared Transitive	1	1%	3	1%
OWL/RDF Irregularities	**15**	**23%**	**20**	**11%**
Unused Triples	12	19%	10	5%
Structure Sharing	0	0%	4	2%
Malformed Restriction	9	14%	2	1%
Malformed List	2	3%	8	4%
Anonymous Class Creation	3	4%	8	4%

Table 2. Frequent Error Types

4 Patching

Adopting the terminology as introduced in [2], errors can be classified into *external* and *internal* errors. *Internal* errors are those where the triples correspond to an OWL ontology, but the ontology makes use of expressivity outside of the required species. *External* errors are those where the RDF graph is in some way incorrectly formed.

We can also identify a third class of *syntactic* errors to cover those situations where there are problems with the underlying concrete presentation of the ontology – for example the XML concrete syntax is malformed. Such situations are likely to be best dealt with by the underlying XML parsers – we do not devote much attention to these here, and concentrate mainly on addressing *external* errors such as missing type information. In addition, we would expect that as more and more ontologies are produced by tools, malformed XML will become a thing of the past. Similarly, if ontologies are held in RDF repositories (such as Sesame [4]) we would hope to see less malformed XML as the repository takes responsibility for the production of the concrete representation. Some *internal* errors, in general those arising from imports, can be tackled, and we discuss these here.

4.1 Missing Type Information

Fortunately the most frequent error of missing type information is also the most easy to deal with. OWL DL requires that URI references used in particular contexts (e.g. as a class) must be explicitly typed as such. This requirement for complete information about the typing of resources is effectively a restriction on the *syntax*. Contrast this with the *semantics* [8] of OWL, which applies an open world assumption, allowing incomplete information.

As we have seen in the previous section almost all of the OWL Full ontologies tested failed to belong to OWL DL primarily because of such missing information.

There are a number of situations that we can identify as *Class Contexts* – those where a URI is expected to be an `owl:Class`. Such situations include:

- The subject or object of an `rdfs:subClassOf` triple.
- The object of an `rdf:type` triple.
- The object of an `owl:someValuesFrom` or `owl:allValuesFrom` triple where the subject of the triple is also the subject of an `owl:onProperty` triple with an `owl:ObjectProperty` as the object.
- The object of an `rdf:domain` triple.
- The object of an `rdf:range` triple where the subject is an `owl:ObjectProperty`.

In these cases, we can be fairly sure that the intention *is* that the URI is intended to be an `owl:Class` and can add an appropriate `rdf:type` triple.

Similarly, we can identify *Object Property Contexts*, those where an `owl:ObjectProperty` is expected. These include:

– The object of an `owl:onProperty` triple where the subject of the triple is also the subject of an `owl:someValuesFrom` or `owl:allValuesFrom` triple with an `owl:Class` as the object.
– The subject of an `rdf:range` triple where the object is an `owl:Class`.

Data Property Contexts encapsulate those situations where an `owl:DatatypeProperty` is expected. These include:

– The object of an `owl:onProperty` triple where the subject of the triple is also the subject of an `owl:someValuesFrom` or `owl:allValuesFrom` triple with a `Datatype`[14] as the object.
– The subject of an `rdf:range` triple where the object is a `Datatype`.

Note that the identification of `owl:ObjectProperty` and `owl:DatatypeProperty` usage requires some analysis of the surrounding context.

These possible contexts may interact and it may be the case that it is **not** possible to disambiguate. For example, consider the following fragment:

```
<rdf:Property rdf:about='#p'>
  <rdf:range rdf:resource='#D'/>
</rdf:Property>
```

One possible patch here would be to consider `p` as an `owl:ObjectProperty` and `D` as an `owl:Class`. An alternative would be to consider `p` as a `owl:DatatypeProperty` and `D` as an `rdfs:Datatype`. Either would provide a valid OWL DL ontology.

Patching is, of course, not always possible (as the ontology may use expressiveness that results in an OWL Full ontology). In the following fragment:

```
<rdf:Description rdf:about="#a">
  <rdf:type rdf:resource="#A"/>
</rdf:Description>
<rdf:Description rdf:about="#A">
  <rdf:type rdf:resource="#B"/>
</rdf:Description>
```

There is no combination of type triples that we can add here to produce an OWL DL ontology as there is an implicit use of classes as instances (and thus a violation of the requirement for a *separated vocabulary* [8]) as `A` is both given a type and used as a type. This example serves to illustrate that there may not always be an appropriate patch that can be applied.

The order in which the contexts are addressed has an impact. For example, if `rdf:Property` (rather than `owl:ObjectProperty`) has been used in an ontology, analysing the use of the property in `owl:someValuesFrom` or `owl:allValuesFrom` triples is likely to yield more information than, for example `owl:cardinality` triples (which do not tell us anything about the characteristics of the property). Thus the entire context of the ontology can prove useful in disambiguating such situations.

Another "hard" case is where the intention is that a property is used for annotation. Take the following ontology:

[14] When we refer to `Datatype`s we mean either an `rdfs:Datatype` or a known XML Schema Datatype.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<rdf:RDF xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xml:base="http://owl.man.ac.uk/ontologies/whatever"
    xmlns="http://owl.man.ac.uk/ontologies/whatever#">
<owl:Ontology rdf:about=""/>
<owl:Class rdf:about="#A"/>
<rdf:Description rdf:about="#A">
  <author rdf:resource="http://www.cs.man.ac.uk/~seanb"/>
</rdf:Description>
</rdf:RDF>
```

In this example, `A` is explicitly declared to be an `owl:Class`. However, we have no type for the property `author`. In this particular case, by choosing to type `author` as an `owl:AnnotationProperty` we can produce an OWL DL ontology. Choosing to type `author` as an `owl:ObjectProperty` would also require us to treat `http://www.cs.man.ac.uk/~seanb` as an individual (and thus provide a type), but would then push us into OWL Full due to the treatment of `A` as class and instance. Thus in this case, to obtain an OWL DL ontology, we need to treat `author` as an `owl:AnnotationProperty`.

In situations like this, it is unlikely that an automated approach will be appropriate – we are in a situation where *user input* is needed in order to make the ultimate decision as to how to resolve the problem. Tools can, however, provide a degree of support in determining what needs to be done and suggest appropriate course of action.

4.2 Import of OWL or RDF schemas

A common occurrence in OWL ontologies is the import of OWL or RDF schemas. This is not necessary in order to produce OWL ontologies, and in fact will *always* result in an OWL-Full (and not OWL-DL) ontology as the import of the schema results in triples using terms from the *disallowed vocabulary* (see Section 4 of [8] for details) as subjects in triples. Although it may be the case that importing the OWL schema *is* sometimes required or intended – for example if we wish to extend the OWL Schema itself – in the majority of cases this is unnecessary.

Resources in RDF namespace OWL does not completely forbid the use of URIs from the RDF and RDF(S) namespaces. For example, `rdf:Bag` can be used as a Class in an OWL DL ontology. Addition of type triples is, however, necessary in certain situations in order to ensure that such RDF schemas are OWL DL. For example, we need to say that `rdf:Bag` is being treated as an `owl:Class`. Similarly, the RDF container properties `rdf:_1`, `rdf:_2`, etc. can be used as properties, but must be typed appropriately.

4.3 Use of vocabularies without appropriate import

This also belongs to the category of missing type information, but is worth special attention. A number of standard vocabularies are often used in ontologies without the necessary definitions. For example, in the ontologies analysed, we encounter Dublin Core properties such as `dc:author` or `dc:title` used without import of the Dublin Core schema or explicit definition of the property. Even if the schema at `http://purl.org/dc/elements/1.1` *is* included, however, this does not necessarily alleviate the position, as that particular schema is itself an OWL Full document due to the use of `rdf:Property` rather than a more specific OWL property. In general, we might expect a property from the Dublin Core to be treated as an `owl:AnnotationProperty`.

4.4 Misuse of `owl:sameAs`

According to the OWL semantics [8], the `owl:sameAs` property should be used to represent the fact that two *individuals* are the same. Using `owl:sameAs` to relate two Classes is thus **not** the same thing as relating them using `owl:equivalentClass` (the latter states that the classes are extensionally equivalent, e.g. have the same members while the former asserts that the two classes are to be interpreted as the same object). Using `owl:sameAs` to relate two Classes results in an OWL Full ontology. The situation is similar with properties, although here we would expect to see `owl:equivalentProperty` rather than `owl:sameAs`. It is likely that in such situations, the modeller may have intended to use `owl:equivalentClass` or `owl:equivalentProperty` rather than `owl:sameAs`.

4.5 `xml:base` and `xmlns` confusion

The use of `xml:base` and the default namespace `xmlns` often causes confusion and can be the source of missing type information. The confusion arises due to the fact that in an RDF/XML file, elements are resolved with respect to the default namespace, whereas attribute values are resolved with respect to the XML base [7]. If no `xml:base` attribute is explicitly set, then this is the URI where the file is retrieved from. If `xml:base` is not set and the document is moved, this can then cause problems as the URIs of the classes and properties declared in the ontology may not match those used in the ontology. For example, assume the following ontology is available at `http://owl.man.ac.uk/ontologies/base`:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns="http://owl.man.ac.uk/ontologies/base#">
<owl:Class rdf:about="#A"/>
<A rdf:about="#a"/>
</rdf:RDF>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<rdf:RDF xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xml:base="http://owl.man.ac.uk/ontologies/broken"
    xmlns="http://owl.man.ac.uk/ontologies/broken#">
<owl:Ontology rdf:about=""/>
<!-- B should be a class -->
<owl:Class rdf:about="#A">
  <rdfs:subClassOf rdf:resource="#B"/>
</owl:Class>
</rdf:RDF>
```

Fig. 1. Not OWL DL

This particular ontology will validate as OWL DL. If, however, we move the file to `http://owl.man.ac.uk/ontologies/base2` and then attempt to validate, validation will fail as the `owl:Class` typing triple will now apply to the URL `http://owl.man.ac.uk/ontologies/base2#A`. Use of an `xml:base` declaration would alleviate this problem.

Unfortunately there is not much we can do in terms of applying patches here. The resolution of URIs is, in general, dealt with by the XML parser and the information regarding the base and default namespaces may not actually be available to the RDF processing phase. However, it is worth mentioning here as this *is* an issue that those publishing ontologies need to be aware of and sensitive to.

5 Implementation

The OWL Validator [2] provided with the WonderWeb OWL API [3] checks the species of OWL ontologies represented using RDF/XML. It does this through a combination of syntactic checks on the structure of the triple graph, and further checks on the expressiveness used in the ontology[15].

The Validator reports any violations of the OWL DL/Lite rules encountered during the parse – this information can then be used to try and provide patches for such violations. The implementation of the Validator has been extended to provide such a Patcher, applying the heuristics described in the paper in order to try and obtain OWL DL ontologies.

[15] The Validation and Patching services described in this paper are available at: `http://owl.man.ac.uk/services.shtml`

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<rdf:RDF xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
 <owl:Ontology rdf:about="">
   <owl:imports>
     <owl:Ontology rdf:about="http://owl.man.ac.uk/ontologies/broken"/>
   </owl:imports>
 </owl:Ontology>
<owl:Class rdf:about="http://owl.man.ac.uk/ontologies/broken#B"/>
</rdf:RDF>
```

Fig. 2. OWL DL

5.1 Additions

A number of different approaches can be taken to producing "patches". In the case of errors of omission, such as missing type information, we can try and provide a collection of triples that need to be added to the ontology in order to provide OWL DL. We cannot always assume, however, that we have access to the source of the ontology – indeed in most cases we will not as the ontologies are referred to via URLs. In order to cope with this, we can produce a new ontology which uses `owl:imports` to import the checked ontology and then adds the necessary type triples. For example, say that at `http://owl.man.ac.uk/ontologies/broken` we find the RDF shown in Figure 1. The results of the patcher will be a new OWL ontology as shown in Figure 2.

Addition of extra triples can be considered to be a "safe" manipulation of the graph. In this case, any entailments (in the RDF sense) that held will continue to hold, due to the *monotonicity* of RDF entailment [6].

5.2 Deletions

For situations such those described in Sections 4.2 (the import of the OWL schema), the change required to the ontology is effectively a *deletion*, e.g. removing the offending `owl:import` triple. Again, if the ontology source is not within our control, removing such a triple is impossible. We can, however, instruct the parser to ignore particular import statements when parsing. In contrast to the addition of triples, after deletions, RDF entailments from the original graph may no longer hold. We must be sure that this is not done silently – the ontology obtained when we ignore an `owl:import` triple is *not* the same ontology as that obtained when the triple is processed. Note, however, that in terms of entailments that can be drawn using OWL DL semantics, the question is moot, as the original graph is not OWL DL, and thus cannot have the OWL DL semantics applied to it.

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns="http://www.daml.org/tools/tools-ont#">
  <Project rdf:ID="DAML">
    <name>DAML</name>
  </Project>
  <Tool rdf:ID="SiRPAC">
    <name>Simple RDF Parser And Compiler (SiRPAC)</name>
    <description>RDF parser, from W3C</description>
    <site>http://www.w3.org/RDF/Implementations/SiRPAC/</site>
    <category>RDF Parser</category>
  </Tool>
  <Tool rdf:ID="SiLRI">
    <name>Simple Logic-based RDF Interpreter (SiLRI)</name>
    <description>a main-memory logic-based inference engine</description>
    <site>http://www.ontoprise.de/download/</site>
    <uses rdf:resource="#SiRPAC"/>
    <category>Inference Engine</category>
  </Tool>
  ...
</rdf:RDF>
```

Fig. 3. DAML Tools Ontology (fragment)

Similarly, for vocabulary misuse such as the `owl:sameAs` problem described in Section 4.4 we can tell the parser to treat `owl:sameAs` triples as if they were `owl:equivalentClass` or `owl:equivalentProperty` as appropriate (depending on context).

Once the processor has identified all the possible patches it can apply, it attempts to revalidate the ontology in the light of those patches and reports its findings.

The outcome of the process is thus a combination of things:

- A report on the errors encountered during the parse/validation.
- A collection of type triples that need to be added to the ontology.
- Identification of imports that cause problems (such as the OWL Schema).

As an example of this process, the DAML pages provide an ontology of tools at `http://www.daml.org/tools/tools.owl`. This is a vanilla RDF (and thus OWL Full) file that does not validate as OWL DL due to the presence of a number of untyped properties and classes. A fragment of the source is shown in Figure 3. In this case, an analysis of the RDF suggests that `Tool` is intended to be an `owl:Class`, properties such as `name` and `site` are instances of `DatatypeProperty`, and `uses` is an `owl:ObjectProperty`. After adding these triples to the ontology (using the `owl:imports` mechanism as described above), we find that the resulting graph does indeed validate as OWL DL.

Collection	Attempted	Patched by Type Triples	Patched by Schema Handling
DAML Library	50	16 (32%)	1 (2%)
Google Search	140	78 (56%)	2 (1%)

Table 3. Results of Patching Non-DL Ontologies

5.3 Results

The ontologies analysed in the initial experiment were run through the Patcher in order to see whether OWL DL ontologies could be produced. Results are summarised in Table 3. Of the 63 candidates from the DAML Library, 13 were definitely OWL Full due to the expressiveness used (for example subproperty axioms were asserted between Object and Datatype Properties), leaving 50 potentially available for patching. Of the ontologies gathered using Google, 31 were definitely OWL Full, leaving 140 potentially available for patching.

6 Related Work

Approaches to parsing OWL in RDF are described in [2, 1]. The application described here extends the approach of trying to construct an abstract syntax tree representing the OWL ontology. In contrast, the Jena recognizer attempts to classify nodes according to their occurrence within the RDF graph. The graph validates as a particular species if the classification of the nodes meets certain conditions. This approach may well also be amenable to patching as described here as the node categories provide pointers to the expected types of the nodes.

BBN's OWL Validator[16] is a tool to "...check OWL markup for problems beyond simple syntax errors...". It is able to spot some missing type errors, but does not (as yet) supply detailed information on how these errors could or should be addressed.

7 Conclusion

The widespread use of ontology languages like OWL is at a rather early stage, and a number of the "rough edges" still need to be smoothed out. Providing tools that support the production of ontologies that conform to the rules and conditions relating to OWL sublanguages is, we feel, a useful step in the right direction. This paper can be seen as complementary to Appendix E of the OWL Reference [5] and the Working Group Note on Parsing [1]. The former provides "Rules of Thumb" to ensure that ontologies are in OWL Lite or DL, while the latter describes how to parse OWL ontologies in RDF/XML. This work describes how, in some sense, one might *retrospectively* apply the Rules of Thumb to existing ontologies.

[16] `http://owl.bbn.com/validator/`

Our results are encouraging. As we see in Section 5, over half of the searched ontologies that were originally found to be OWL Full can in fact be "retrofitted" to the OWL DL requirements through the addition of the missing type triples or judicious handling of imports. For those in the DAML Library, we were able to handle around a third. The solutions we describe here are admittedly rather simple – the latter example in Section 4.1 gives a glimpse of the complex dependencies that may arise from missing types and we do not (as yet) offer automated solutions to such problems. However the results reported show that even with simple approaches, we are able to handle a significant number of ontologies.

A key message here is that care should be taken when applying such heuristics – the underlying semantics of the ontology *are* being changed (see Sections 5.1 and 5.2), and tools must ensure that users are aware of the fact that this is happening. We are not advocating that applications should feel free to arbitrarily rewrite ontologies they find on the web – this is likely to compromise interoperability. Rather, the procedures we describe here can form a first cut in a process of "cleaning up" information on the Web.

An analogy can be made with HTML processors. A significant proportion of HTML pages which are available on the web do not, in fact, conform to schema such as the XHTML standard. It is often the case that, for example, closing tags are missing. HTML parsers such as those found in browsers have been carefully honed to try and deal with these situations. Indeed, it is unlikely that the web would have met with the success it has if browsers were not able to handle poorly structured documents. We should be careful not to stretch this analogy too far, however. In general, HTML pages are targeted at a human interpreter – humans are fairly robust in terms of their ability to deal with incomplete or dirty information. Software agents are a different matter, and care must be taken when applying patching heuristics to ensure that the end user is aware that such an approach is being taken. In addition, even with what seems to be "broken" or dirty information, it may be the case that the original syntactic presentation *is* exactly what was intended.

Support for the migration from vanilla RDF to OWL is also of interest here. As discussed in the introduction, the number of OWL DL/Lite ontologies currently on the web is small. Indeed the number of ontologies that even use the OWL vocabulary is small – many more schemas are currently represented using RDF. It would be useful if such schemas could be made accessible to OWL processors whenever possible. Again, we surmise that in a large proportion of the RDF schemas available on the web, the schemas are not inherently OWL Full due to the expressivity used, but are rather OWL Full because they do not meet the *syntactic* restrictions imposed by OWL DL. Translating these to valid OWL DL ontologies is not, as we have seen, simply a case of replacing vocabulary. However, with the application of appropriate heuristics, we can move towards the support of automatic migration of RDF vocabularies to OWL DL and Lite.

Another issue that we touch on here, but do not examine in much depth is that of the provision of ontology libraries. The success of the Semantic Web relies, in part, on the provision of ontologies and the *sharing* of those ontolo-

gies. Not only must we author the ontologies, but they must also be published and made available to applications. The initial analysis reported here used a somewhat crude mechanism (searching Google) in order to find OWL ontologies. There are a small number of ontology libraries on the web (for example the DAML ontology library[17] which is probably the largest) but these are not particularly comprehensive, and in the main are rather *lightweight*[18]. For example, our experience of encountering Dublin Core properties in OWL ontologies suggests that an available OWL DL or Lite schema for the Dublin Core properties[19] is likely to be a useful resource.

In conclusion, although the initial answer to the question "how much OWL DL is there on the Web?" is "not much", with the provision of some quite simple tool support, we believe we can increase this to at least "a little bit".

Acknowledgments This work was supported by the WonderWeb project (EU grant IST-2001-33052). Sean Bechhofer would like to thank Peter Patel-Schneider and Jeremy Carroll for their invaluable assistance in deciphering the minutiae of OWL syntax.

References

1. Sean Bechhofer. OWL Web Ontology Language Parsing OWL in RDF/XML. W3C Working Group Note, World Wide Web Consortium, January 2004. `http://www.w3.org/TR/owl-parsing`.
2. Sean Bechhofer and Jeremy J. Carroll. Parsing OWL DL: Trees or Triples? In *Proceedings of World Wide Web Conference, WWW2004*. ACM Press, May 2004.
3. Sean Bechhofer, Raphael Volz, and Phillip Lord. Cooking the Semantic Web with the OWL API. In *2nd International Semantic Web Conference, ISWC*, volume 2870 of *Lecture Notes in Computer Science*, Sanibel Island, Florida, October 2003. Springer.
4. Jeen Broekstra, Arjohn Kampman, and F. van Harmelen. Sesame: A Generic Architecture for Storing and Querying RDF. In Ian Horrocks and James Hendler, editors, *Proceedings of the International Semantic Web Conference, ISWC2002*, volume 2342 of *Lecture Notes in Computer Science*, pages 54–68. Springer-Verlag, June 2002.
5. Mike Dean and Guus Schreiber. OWL Web Ontology Language Reference. W3C Recommendation, World Wide Web Consortium, 2004. `http://www.w3.org/TR/owl-ref/`.
6. P. Hayes. RDF Semantics. W3C Recommendation, World Wide Web Consortium, 2004. `http://www.w3.org/TR/rdf-mt/`.
7. Jonathan Marsh. XML Base. W3C Recommendation, World Wide Web Consortium, 2004. `http://www.w3.org/TR/xmlbase/`.
8. P. Patel-Schneider, P. Hayes, and I. Horrocks. OWL Web Ontology Language Abstract Syntax and Semantics. W3C Recommendation, World Wide Web Consortium, 2004. `http://www.w3.org/TR/owl-semantics/`.

[17] `http://www.daml.org/ontologies`

[18] By lightweight here we mean ontologies that do not extend much beyond the expressivity supported by RDF Schema.

[19] `http://www.aifb.uni-karlsruhe.de/WBS/rvo/ontologies/dublincore.owl`