A Survey of the Web Ontology Landscape

Taowei David Wang[1], Bijan Parsia[2], James Hendler[1]

[1] Department of Computer Science,
University of Maryland, College Park, MD 20742, USA,
{tw7, hendler}@cs.umd.edu
[2] The University of Manchester, UK
bparsia@cs.man.ac.uk

Abstract. We survey nearly 1300 OWL ontologies and RDFS schemas. The collection of statistical data allows us to perform analysis and report some trends. Though most of the documents are syntactically OWL Full, very few stay in OWL Full when they are syntactically patched by adding type triples. We also report the frequency of occurrences of OWL language constructs and the shape of class hierarchies in the ontologies. Finally, we note that of the largest ontologies surveyed here, most do not exceed the description logic expressivity of $\mathcal{ALC}$.

1 Introduction

The Semantic Web envisions a metadata-rich Web where presently human-readable content will have machine-understandable semantics. The Web Ontology Language (OWL) from W3C is an expressive formalism for modelers to define various logical concepts and relations. OWL ontologies come in three species: Lite, DL, and Full, ordered in increasing expressivity. Every Lite ontology is also a DL ontology, and every DL ontology is also a Full ontology. OWL Lite and OWL DL are the species that use only the OWL language features in the way that complete and sound reasoning procedures exist. OWL Full, on the other hand, is undecidable. While OWL recently became a W3C recommendation in 2004, people have been working with it a few years, and many interesting ontologies already exist on the Web. We are interested in evaluating these ontologies and see if there are interesting trends in modeling practices, OWL construct usages, and OWL species utilization.

2 Related Work

Using statistics to assess ontologies is not a new idea. Several approaches to create benchmarking for Semantic Web applications have exploited the statistical measures to create better benchmarks. Wang and collegues describe an algorithm to extract features of instances in a real ontology in order to generate domain-specific data benchmark that resembles the real ontology [16]. A method to count the types of triples of instances is employed, and the distribution of these triples is used to create the synthetic data. Tempich and Volz surveyed 95 DAML ontologies and collected various usage information regarding classes, properties, individuals, and restrictions [15]. By examining

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
2006		**00-00-2006 to 00-00-2006**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
A Survey of the Web Ontology Landscape	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of Maryland,Department of Computer Science ,College Park,MD,20742**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**13**	

these numbers, they were able to cluster the ontologies into 3 categories of significant difference.

In [11], Magkannaraki et al. looked at a collection of existing RDFS schemas and extracted statistical data for size and morphology of the RDFS vocabularies. Here we attempt a similar survey for both OWL and RDFS files. However, our focus is primarily on OWL, and the data from RDFS documents serve as a good measuring stick for comparisons.

Bechhofer and Volz studied a sample of 277 OWL ontologies and found that most of them are, surprisingly, OWL Full files [2]. They showed that many of these OWL Full ontologies are OWL Full because of missing type triples, and can be easily patched syntactically. Here we collect a much larger size of samples, and we apply similar analysis to attempt to patch these OWL Full files. In addition, we show how many OWL Full files can be coerced into OWL Lite and OWL DL files. With the expressivity binning of the surveyed ontologies, we show that the number of OWL Lite files that makes use of OWL Lite's full expressivity is relatively small.

3 Methodology

Here we describe the steps taken to collect the ontologies from the Web, how the data was then gleaned, and how we analyzed the data. Our goal was to analyze the various aspects of ontological documents, not RDF documents that make use of ontologies or schemas. Inspite of FOAF [3] (and DOAP [4] and RSS) being a large percentage of the semweb documents out there, they exhibit almost no ontological variance, being primarily data with a thin schema, and are not in the scope of this study.

3.1 Ontology Collection

We used several Web resources to collect the ontologies and schemas. We collected just the URIs at this stage, as our analysis tools will retrieve documents from the web given dereferenceable URIs. First, we used the Semantic Web Search engine Swoogle [7] to obtain a large number of semantic documents that Swoogle classify as ontologies. Using `sort:ontology` [5] as the search term, we were able to crawl on the list 4000+ files. They were a mixture of OWL, DAML, RDF, and RDFS documents. Since we are interested primarily in OWL ontologies, and wanted to get a representatively large sample to perform our analysis, we also searched on Google [6]. Using the search term `owl ext:owl`, we were able to obtain 218 hits [7] at the time of data collection (February 9, 2006). We also collected OWL ontologies from well-known repositories: Protégé OWL

[3] `http://xmlns.com/foaf/0.1/index.rdf`

[4] `http://usefulinc.com/doap`

[5] Swoogle 2005 `http://swoogle.umbc.edu/2005/` allows this type of search. The new Swoogle 2006, which was released after the survey was completed, does not.

[6] `http://www.google.com`

[7] As noted in [2], the number of search results returned by Google is only an estimate. Furthermore, Google has since changed how OWL files are indexed, and the numbers returned today are orders of magnitudes larger.

Library [8], DAML Ontology Library, [9], Open Biological Ontologies repository [10], and SchemaWeb [11].

Since we collected our URIs from several resources, some URIs appeared more than once in our collection. We first pruned off these duplicate URIs. Next, we threw away the unsuitable data for our analysis. We pruned off all the DAML files as they are not the focus of this study. We threw away the various test files for OWL from W3G and test files for Jena [4]. Though these are valid ontologies or schema files, they were created specifically for the purpose of testing, and do not resemble realistic ontological documents. Around 1000 WordNet RDFS files were also dropped. While WordNet as a whole is useful, each separate WordNet RDFS file does not preserve the meaning of that specific fragment. Finally, we discard any URIs that no longer existed. At the end, we had 1276 files. We looked at each of the documents to see if the OWL or the RDFS namespaces are defined to determine whether they are OWL ontologies or RDFS schemas. Of the 1275 collected, 688 are OWL ontologies, and 587 are RDFS schemas. Resolving these URIs, We keep local copies of these documents for future references.

Table 1. Sample Statistics Collected

Basic Statistics	**Dynamic Statistics**
No. Defined/Imported Classes	No. Subsumptions
No. Defined/Imported Properties	No. Multiple Inheritance in Class Hierarchy
No. Defined/Imported Instances	Graph Morphology of the Class Hierarchy
DL Expressivity	Depth, Bushiness of the Class Hierarchy
No. Individual (Type/Property) Assertions	Depth, Bushiness of the Property Hierarchy
OWL Species	Whether the Ontology is Consistent
No. of Symmetric Properties	No. Unsatisfiable Classes

3.2 Statistics Collection

We used the OWL ontology editor SWOOP [9] as a framework for automating the analysis tasks. For each URI we collected a set of statistics of that document. There were two types of statistics we collected. The first set contains the statistics that do not change when a reasoner processes the ontology. We call this set static statistics, and it includes, for example, number of defined classes, what ontologies are imported (if any), or which of the OWL species the document belongs to. On the other hand, a second set of statistics changes depending on whether a reasoning service is present. We call this set dynamic statistics. For example, the number of concepts that have more than one parent may change when reasoning is applied since new subsumption relationships can be discovered by the reasoner. Because dynamic statistics change, we collected both the

[8] `http://protege.stanford.edu/plugins/owl/owl-library/`
[9] `http://www.daml.org/ontologies/`
[10] `http://obo.sourceforge.net/main.html`
[11] `http://www.schemaweb.info/`

told (without reasoning), and the inferred (with reasoning) versions. Our method is to load each URI into SWOOP, collect the static statistics and the told dynamic statistics, then turn on the Pellet [13] reasoner and collect the inferred dynamic statistics. We list a few selected categories that are relevant to our discussion in Table 1.

For each OWL ontology, we also collect what OWL constructs are used. We do this by inserting each ontology into a Jena model and check all triples for OWL vocabulary. There are 38 boolean values, one for each OWL construct, for each ontology. Note that we are not keeping track of the usage of OWL:Thing and OWL:Nothing. RDF and RFDS vocabulary such as `rdfs:subClassOf` are also not collected.

4 Results

Here we report the analysis performed, results from our analysis, and what trends we discover.

4.1 OWL Species, DL Expressiveness, Consistency

There are several reasons that make an ontology OWL Full. Bechhofer and Volz discusses each reason in detail in [2]. Here we summarize them into 4 categories to facilitate discussion.

1. **(Syntactic OWL Full)** In this category, the document contains some syntactic features that make the ontology OWL Full. This category includes ontologies that are missing `rdf:type` assertions for its classes, properties, individuals, or itself (untyped ontology). Missing type triples is easily amended as proposed in [2]. Our tool Pellet can generate a patch in RDF/XML to add to the original document to eliminate this type of OWL Fullness.
   Another way to be in OWL Full is to have structural sharing. Here we discuss the sharing of a restriction as an example, but any bnode sharing is likely to lead to OWL Full. An OWL Restriction in RDF is represented as a bnode. A modeler can reuse an existing restriction by referring to the bnode ID. However, doing so will make the ontology OWL Full. On the other hand, if the same modeler creates a new restriction with the same semantics instead of referring to the existing one, structural sharing is avoided.
2. **(Redefinition of Built-In Vocabulary)** Documents that attempt to redefine known vocabulary (such as those in the OWL or RDFS specification) will be in OWL Full. Attempting to add new terms in known namespaces (OWL, RDF, RDFS, etc.) will place the document under OWL Full as well, even innocuous statements such as subclassing `rdf:label`.
3. **(Mixing Classes, Properties, and Individuals)** In OWL DL, the sets of `owl:Class`, `owl:Property`, and `owl:Individual` must be disjoint. The ontologies that use, for example, classes as instances or classes as properties do not respect such disjointness, and are classified as OWL Full documents. Some authors do intend to use instances as classes, for example, for metamodeling purposes. However, there are many other cases where simply an oversight had occurred. We also mention

that in RDFS semantics, the set of `rdfs:Class` and `rdf:Property` are not assumed to be disjoint, therefore any RDFS schema will be considered as a OWL Full file. Though if the schema does not use classes and properties interchangeably, patching up with type triples will likely take the RDFS document out of OWL Full.

4. **(Need for Beyond OWL DL)** This group uses OWL constructs to create an ontology that has expressivity going beyond what OWL DL has to offer. Examples are those that declare a DatatypeProperty to be inverse functional (e.g. `FOAF`), or those that declare cardinality restrictions on transitive properties.

Table 2. Number of Documents in Each Species (species determined by Pellet)

Species	RDFS	Lite	DL	Full	Error
Count	587	199	149	337	3

Now we have a better idea of the syntactic and semantic elements that make an OWL ontology OWL Full, we are ready to look at our data. By looking at the namespaces declared in each document, we decide which files are in RDFS, and which ones in OWL. Using Pellet as an OWL species validation tool, we obtain the distribution of each OWL species in Table 2. Note that since RDFS does not enforce the disjointness of the set of classes, the set of properties, and the set of instances, the RDFS files are technically OWL Full.

We inspected the results Pellet outputs. Out of 924 OWL Full files (including RDFS), 863 can be patched. 30 OWL and 31 RDFS documents can not. Of the 863 patchable ones, 115 become OWL DL, 192 become OWL Lite, and the remaining 556 documents are RDFS. Table 3 shows the updated counts.

Table 3. Number of Documents in Each Species (After Patching)

Species	RDFS(DL)	Lite	DL	Full	Error
Count	556	391	264	61	3

Though Table 3 resembles Table 2, there is one important difference. Note that we use RDFS(DL) [5] instead of RDFS in this case to emphasize that RDFS(DL) assumes the disjointness of classes and properties, and is a proper subset of OWL Lite. Of the 307 OWL Full documents that can be patched, 63% become OWL Lite documents, and just 37% become OWL DL. Two observations can be made. First, The majority (91%) of the OWL Full documents (from Table 2) can be turned into a decideable portions of the languages by adding type triples. Secondly, the majority of RDFS documents (95%) can transition to OWL easily by adding type triples and use OWL vocabulary instead of RDFS vocabulary.

Of the 30 OWL documents that cannot be patched, nearly all of them contain problems of redefining built-in vocabulary. One ontology contains structural sharing. There

are 8 ontologies that mix the usage of instances, classes, or properties. And there are 2 cases where beyond OWL DL features are detected. In both of these cases, a DatatypeProperty is defined to be inverse functional.

Of the 31 RDFS documents that cannot be patched, most contain wrong vocabulary, redefinition of known vocabulary, or liberal use built-in vocabulary (such as using `rdfs:subClassOf` on `xsd:time`).

Although species validation gives us a rough idea of the distribution of expressivity among ontologies, it is not a fine enough measure. OWL Lite has the same expressivity as the description logic $\mathcal{SHIF}(\mathcal{D})$, and OWL DL is equivalent to $\mathcal{SHOIN}(\mathcal{D})$. There is a large expressivity gap between RDFS(DL) and OWL Lite. We group the DL expressivity of the documents into bins in atttempt to find out how many ontologies make full use of OWL Lite's features.

We bin the expressivity of the documents as follows. For simplicity, we ignore the presence of datatype, so $\mathcal{SHIF}(\mathcal{D})$ is considered the same as $\mathcal{SHIF}$. For all ontologies that contain nominals $\mathcal{O}$ or number restrictions $\mathcal{N}$, we put them in the most expressive bin (Bin 4). For example, $\mathcal{SHOIN}$ belongs to Bin 4. The next group Bin 3 contains the ones that make use of inverses $\mathcal{I}$ or complements $\mathcal{C}$ but not nominals or number restrictions. $\mathcal{SHIF}$ belongs to this group. Bin 2 consists of role hierarchies $\mathcal{H}$ or functional properties $\mathcal{F}$, but not the features Bin 4 or Bin 3 care about. Bin 2 would contain $\mathcal{ALHF}$, which is more expressive than RDFS(DL). Lastly, everything else will fall into the Bin 1, e.g. $\mathcal{AL}$. We expect the first two bins to contain all of the RDFS(DL) documents and some OWL Lite documents. The question is, of course, how many?

Table 4. Expressivity Binning

Bin	Bin 1 ($\mathcal{AL}$)	Bin 2 ($\mathcal{ALHF}$)	Bin 3 ($\mathcal{SHIF}$)	Bin 4 ($\mathcal{SHOIN}$)
Count	793	55	262	151

Table 4 shows the count of each expressivity bin. 14 OWL documents cannot be processed and are not included in this part of the analysis. The 848 documents in bin 1 and 2 consists of those that are less expressive than $\mathcal{SHIF}$. Subtracting 848 by the number of RDFS documents from Table 2, we reveal 261 documents that are OWL Lite. This is the number of OWL Lite files that do not make use of its full language expressivity. If we subtract this number from the number of OWL Lite documents in Table 3, we get 130. Therefore, the number of ontologies that make good use of OWL Lite features is less than 20% of the total number of OWL ontologies we surveyed here. This is an indication that the OWL Lite vocabulary guides users to create ontologies that are far less expressive than what OWL Lite can express. In fact, of the total number of OWL Lite documents (after patching), 67% use very little above RDFS(DL).

Out of the 688 OWL ontologies, 21 are inconsistent. 18 of the inconsistent ontologies are due to missing type on literal values. These are simple causes for inconsistency that can be detected syntactically. Data type reasoners should have a way to automatically fix it. The other three contain actual logical contradictions. There are also 17

consistent ontologies that contain unsatisfiable classes. 12 belong to bin 4, while the rest belong to bin 3.

4.2 Usage of OWL Constructs

In Table 5, we show, for each OWL construct, the number of ontologies that use it. The table is organized in 5 sections: Ontology, Class, Property, Individual, or Restriction-Related. Not surprisingly, `owl:Class`, `owl:ObjectProperty`, and `owl:DatatypeProperty` are used in many ontologies. `owl:ObjectProperty` occurs in 185 more ontologies than `owl:DatatypeProperty` does. One possible explanation is that modelers wish to use the semantically rich property types in OWL such as `owl:InverseFunctionalProperty`, `owl:SymmetricProperty`, `owl:TransitiveProperty`, and `owl:InverseOf`, which can only be used with `owl:ObjectProperty` in OWL DL. The fact that `owl:InverseOf` alone is used in 128 ontologies seem to support this hypothesis.

Looking at the Class-Related Constructs, we note that `owl:Union` (109) is used more often than `owl:IntersectionOf` (69). We believe the difference stems from the fact that OWL semantics assumes intersection by default when a modeler says 'A is a subclass of B' and in a different part of the document 'A is a subclass of C'. This is semantically equivalent to saying 'A is a subclass of (B and C)' in OWL. This means in these non-nested boolean cases, one can express an AND relationship without explicitly using 'owl:IntersectionOf'. Another possible contribution to the higher number of `owl:Union` is tool artifact. It is well-known that Protégé assumes union semantics for multiple range and domain axioms. That is, if one were to say 'R has domain A' and 'R has domain B', then Protégé assumes that the user means 'R has domain (A OR B)' and uses `owl:Union`. However, we are not sure how many ontologies were created by using Protégé.

`owl:Imports` appears in 221 OWL documents. This seems to suggest that a good number of ontologies are being reused. However, we do not know how widely an ontology is being imported, nor do we know how many ontologies are being imported. Many institutions that create a suite of ontologies often have heavy use of imports among these ontologies (e.g. SWEET JPL [12]). However cross-institutional ontology sharing seems less common.

There are 253 OWL ontologies that have at least 1 defined individual in this survey. However, Table 5 shows that very few Individual-Related OWL constructs are used. Though `owl:SameAs` is used much more often than the others.

4.3 Tractable Fragments of OWL

There has recently been interest in finding useful yet tractable fragments of OWL in the community [13]. Recent proposals for tractable Description Logics include $\mathcal{EL}++$ [1] and *DL-Lite* [3]. $\mathcal{EL}++$ is an extension of $\mathcal{EL}$, which is used to model certain medical domains. *DL-Lite*, on the other hand, is designed for query answering. We inspect our

[12] http://sweet.jpl.nasa.gov/ontology/

[13] http://owl-workshop.man.ac.uk/Tractable.html

Table 5. OWL Construct Usage

Construct	Count	Construct	Count
Ontology-Related Constructs		*Class-Related Constructs*	
owl:Ontology	567	owl:Class	580
owl:OntologyProperty	0	owl:ComplementOf	21
owl:BackwardCompatibleWith	0	owl:DeprecatedClass	2
owl:Imports	221	owl:DisjointWith	97
owl:InCompatibleWith:	1	owl:EquivalentClass	77
owl:PriorVersion	8	owl:IntersectionOf	69
owl:VersionInfo	305	owl:OneOf	43
Individual-Related Constructs		owl:Union	109
owl:AllDifferentFrom	6	*Property-Related Constructs*	
owl:DifferentFrom	5	owl:AnnotationProperty	28
owl:DistinctMembers	6	owl:DataRange	14
owl:SameAs	18	owl:DatatypeProperty	277
Restriction-Related Constructs		owl:DeprecatedProperty	2
owl:AllValuesFrom	118	owl:EquivalentProperty	25
owl:Cardinality	120	owl:FunctionalProperty	114
owl:hasValue	48	owl:InverseFunctionalProperty	30
owl:MaxCardinality	60	owl:InverseOf	128
owl:MinCardinality	99	owl:ObjectProperty	462
owl:onProperty	263	owl:SymmetricProperty	20
owl:Restriction	263	owl:TransitiveProperty	39
owl:SomeValuesFrom	85		

OWL ontologies to see how many fall into the expressivities the two languages provide. We also look at how many OWL ontologies fall into RDFS(DL). Because Pellet's DL expressivity checker checks on normalized models, and is not very fine grained (starts with $\mathcal{AL}$), we use expressivity as reported by SWOOP.

Table 6. Tractable fragments of OWL and how many of each fragment appears in this survey.

Fragment	RDFS(DL)	*DL-Lite*	$\mathcal{EL}++$	Non-Tractable
Count	230	94	56	287

Table 6 confirms that many OWL files are in RDFS(DL). Of the other two more expressive fragments, the number of *DL-Lite* documents nearly doubles that of $\mathcal{EL}$++. We also look at the OWL constructs for the ontologies that fall into these two fragments. Table 7 shows the highlight. Although conjunction is the only logical connective the two fragments allow fully, `owl:Intersection` not widely used. The $\mathcal{EL}$++ ontologies have a much higher percentage in using restrictions and object Properties than *DL-Lite*. However, much higher percentage of *DL-Lite* files use datatype property. The large disparity in the number of $\mathcal{EL}$s++ that use datatype property and object property is surprising. Finally, we note that *DL-Lite* does not allow cardinality greater than one.

However, it does allow for functionality. All the *DL-Lite* documents that make use of cardinality restrictions are only using cardinality of 1.

Table 7. OWL construct usage for *DL-Lite* and $\mathcal{EL}++$

Constructs	*DL-Lite*	$\mathcal{EL}++$
owl:Intersection	1(1%)	3(5%)
owl:Restriction	35 (37%)	36 (64%)
owl:ObjectProperty	45 (48%)	43(77%)
owl:DatatypeProperty	44 (0.47%)	4 (7%
owl:FunctionalProperty	20 (20%)	0 (0%)
owl:Cardinality	21 (22%)	0 (0%)
owl:SomeValuesFrom	0(0%)	33(60%)

4.4 Shape of Class Hierarchy

When we think of defined vocabularies in schemas and ontologies, we often think of the structure as a tree, where each class is a node, and each directed edge from a parent to a node denotes subsumption. It may be because of our experience as seeing the terms being displayed as tree widgets in our ontology editing tools such as SWOOP or Proétegé or because trees are easier to mentally visualize. However, the vocabulary hierarchy can be all kinds of more general graph structures. In Figure 1 we show the kinds of graph structure a defined set of vocabulary can take shape. The black-dotted circle denotes the top concept (e.g. owl:Thing in OWL ontologies). List, lists, tree, and trees should be familiar to the reader. Multitrees can be seen as a directed acyclic graph (DAG) where each node can have a tree of ancestors and a tree of children. There cannot be a diamond structure in a mulitree [8]. If a diamond structure exists, then it is a general DAG. We can consider the categories list, lists, tree, trees, multitree, and DAG as a strictly ordered list in increasing order of graph complexity.

We point out that a general graph (where cycles exist) is possible. However, because the edges represent subsumptions, all the nodes on the cycle are semantically equivalent. Some paths on the cycle may not be obvious, but sound and complete reasoners will always discover them. Therefore when a reasoner is present, no cyclic graphs of subsumption hierarchies can appear. There can be cycles in a told structure, though these are easy to detect syntactically. In addition, because turning on reasoning services will discover these equivalences and more subsumptions, the graph morphology may change between the told and the inferred structure. Below we show scatterplots of the graph morphological changes in the OWL documents. The scatterplots are fashioned using Spotfire [14].

In Figure 2, each square represents an OWL document, and the size of the square indicates how many classes are in the document. Using the grid point (x,y) closest to each document and referring to the two axes, we can find out what morphology the class

[14] `http://www.spotfire.com/`

hierarchy is in. The vertical axis indicates the morphology in the told structure. The horizontal axis indicates the morphology in the inferred structure. The data points do not lie strictly on an intersection of the grid lines because we have jittered the positions of the data points to avoid occlusions. The jittering also gives a better idea of how many datapoints are in each grid intersection.

If an ontology is inconsistent when reasoner is turned on, the class hierarchy will collapse, and there are no structures. We use the category INCONSISTENT to denote this case. The None structure denotes that the ontology contains no classes, hence there are no structures. In Figure 2, note the clusters along the diagonal. These indicate that most ontologies retain their told morphology after a reasoner has been applied. However, 75 of them did change, 21 of which became inconsistent. 42 ontologies went up to a more complex structure (e.g. from trees to multitrees). Of the 42 that went up in graph complexity, 25 came from trees to either DAGs or multitrees. 3 multitrees and 3 lists became DAGs. 5 ontologies that had lists as the told strucure had the tree or trees strucure when reasoning is turned on. 6 lists became multitrees. The graph morphological changes in increasing graph complexity indicate that more subsumptions are discovered. The ones in decreasing graph complexity means that equivalences are discovered. The most interesting ones are the ontologies that discover multiple inheritance in the inferred structure when there was none in the told structure. These are the list, lists, tree, and trees that became multrees or DAGs. This indicates that some interesting modeling is at work here, and there are 34 of them.

Figure 2 shows the same scatterplot, but for the RDFS documents. We do not expect there to be many, if any, changes in graph morphology because every subclass relationship must be explicitly asserted. In this graph, we clearly see that no RDFS class strucure has changed as a result of a reasoning service.

Because the morphology changes between the told and the inferred structures can give indication on which classes are undermodeled or heavily modeled, to be able to compare them side-by-side and interactively explore them can be potentially useful to modelers and users. Current ontology editors and visualizers, such as the ones described in [9] [12] [10] [14], do not directly support this task.

Here we look at the distribution of the largest ontologies in this survey. Of the 19 ontologies that have more than 2000 classes, 14 have the expressivity of $\mathcal{ALC}$ or lower. 2 have the expressivity $\mathcal{SHF}$, 2 have $\mathcal{S}$, and 1 has $\mathcal{SHOIF}(\mathcal{D})$. In the top right corner of Figure 2, we see that there are a number of large OWL ontologies sitting in the (DAG, DAG) position. To explore further, we plotted the inferred graph morphology against OWL species in Figure 3. The upper right corner shows that many large ontologies belong to the OWL Lite species, and their class structures are DAGs. There are 6 ontologies with more than 10000 classes in this survey, 5 of the 6 are in the (DAG, Lite) cluster. Of these 5, 4 have DL expressivity of $\mathcal{ALC}$, 1 has the the expressivity of $\mathcal{S}$. The combination of the most generalized graph structure and the least expressive species is interesting because it suggests that these ontologies are modeling fairly complex domains where the class structures are DAGS. However, none of the OWL DL features are used in the modeling process. Whether the modelers purposely intended to stay in OWL Lite (for fear of computational complexity in reasoning), or that OWL Lite provides all the constructs they needed is unclear.

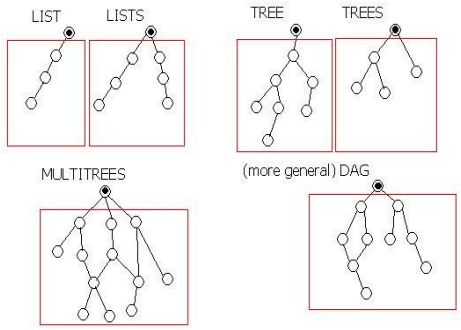

Fig. 1. Possible graph morphology of class hierarchies.

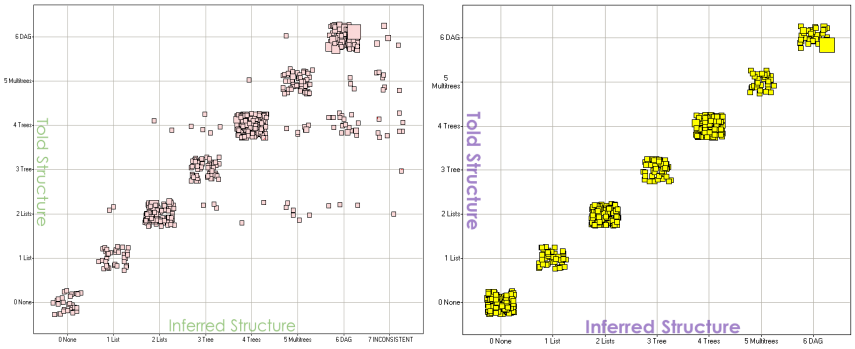

Fig. 2. Scatterplots of the graph morphology of OWL documents (on left), and the RDFS documents (right).

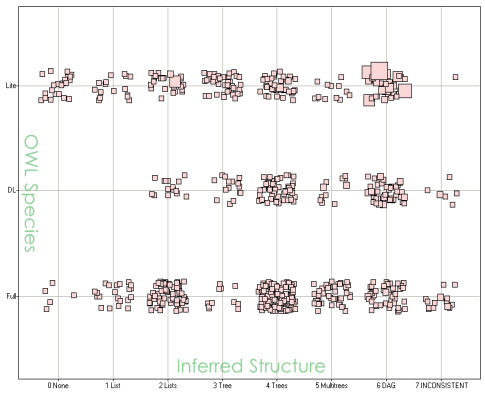

Fig. 3. Scatterplot of the graph morphology of OWL documents against OWL species.

5 Future Work

The future work includes a survey on a larger pool of ontologies. For example, many DAML files can be converted to OWL without any loss of semantics. The only major difference between the two languages is that DAML has qualified number restrictions. It would be an interesting to see how many DAML files uses qualified number restrictions. In addition, the newly released Swoogle 2006 claims to have indexed many more semantic documents, including over 10000+ ontologies.

We see in this study that a fairly large number of ontologies use imports. It would be interesting to find out which ontologies are being imported and by how many others, what percentage of imports are not used by ontologies developed in the same institution. Related to this issue is finding out which are the most popularly used ontologies by RDF files (such as people's FOAF files). Another issue related to imports is to find out how many terms are being used in an ontology without importing the ontologies the terms are defined in.

It would also be interesting to attempt to partition the OWL ontologies using the modularity framework outlined in [6]. Partitionability of an ontology indicates that there are, informally, self-contained domains that can be separated, and possibly reused by other ontologies. The number of ontologies that can be partitioned and the distribution of the sizes of the partitions can shed some light about practitioners' modeling practices in terms of how often/many disjoint domains are used in an ontology.

6 Conclusions

As use OWL grows, assessments of how the language is being used and how modeling trends begin to emerge is both useful and interesting to the community. By collection nearly 1300 ontological documents from the Web and analyzing the statistics collected from them, we were able to note several trends and make interesting observations. There are higher percentage of OWL DL and OWL Lite files than it was previously reported in [2]. Most of the OWL Full files surveyed here can be syntactically patched. Of the patched OWL Full files, roughly one-third becomes OWL DL two-thirds become OWL Lite. In addition, by adding type triples, most of the RDFS files can easily transition to OWL files.

We showed that majority of OWL Lite documents fall into the bins of very inexpressive ontologies. The number of ontologies that contain interesting logical contradictions in this survey is small. But they all have high expressivity. In OWL construct analysis, we showed that `owl:intersection` is used in fewer ontologies than `owl:union`. `owl:ObjectProperty` is more prevalent than `owl:DatatypeProperty`. Though about one-third of the ontologies contain instances, very few instance constructs are being used currently. Looking at the graph morphologies, we are able to see where the interesting modeling practices occur. In addition, we conjecture that tools that presents/exploits the changes between told and inferred structures may allow users to gain understanding otherwise hard to obtain. We also observe that the largest of the OWL files have the characteristic that they have a high graph-morphological complexity and relatively low DL expressivity.

7 Acknowledgments

This work was supported in part by grants from Fujitsu, Lockheed Martin, NTT Corp., Kevric Corp., SAIC, the National Science Foundation, the National Geospatial Intelligence Agency, DARPA, US Army Research Laboratory, and NIST. Special thanks to Evren Sirin and Aditya Kalyanpur for their insightful discussions.

References

1. Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the el envelope. *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05)*, 2005.
2. Sean Bechhofer and Raphael Volz. Patching syntax in owl ontologies. *Proceedings of the 3rd International International Semantic Web Conference*, 2004.
3. Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. *DL-Lite*: Tractable description logics for ontologies. *Proceedings of American Association for Artificial Intelligence (AAAI05)*, 2005.
4. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson. Jena: Implementing the semantic web recommendations. *Proceedings of the 13th World Wide Web Conference*, 2004.
5. Bernardo Cuenca Grau. A possible simplification of the semantic web architecture. *Proceedings of the 13th International World Wide Web Conference (WWW2004)*, 2004.
6. Bernardo Cuenca-Grau, Bijan Parsia, Evren Sirin, and Aditya Kalyanpur. Modularity and web ontologies. 2006. To Appear in Proceedings of the 10th *International Conference on Principles of Knowledge Representation and Reasoning (KR2006)*.
7. Li Ding et al. Swoogle: A search and metadata engine for the semantic web. *Proceedings of the Thirteenth ACM Conference on Information and Knowledge Management*, 2004.
8. G. W. Furnas and J.Zacks. Multitrees: Enriching and reusing hierarchical structure. *Proceedings of ACM CHI 1994 Conference on Human Factors in Computing Systems*, 1994.
9. A. Kalyanpur, B. Parsia, and J. Hendler. A tool for working with web ontologies. *Int. J. on Semantic Web and Info. Syst.*, 1(1), 2004.
10. Thorsten Liebig and Olaf Noppens. OntoTrack: Combining browsing and editing with reasoning and explaining for OWL Lite ontologies. *Proceedings of the 3rd International International Semantic Web Conference*, 2004.
11. A. Magkanaraki, S. Alexaki, V. Christophides, and D. Plexousakis. Benchmarking rdf schemas for the semantic web. *Proceedings of the 1rd International International Semantic Web Conference*, 2002.
12. Natalya F. Noy, Michael Sintek, Stefan Decker, Monica Crubézy, Ray W. Fergerson, and Mark A. Musen. Creating semantic web content with protégé-2000. *IEEE Intelligent Systems*, 16(11):60–71, 2001.
13. Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical owl-dl reasoner. Submitted for publication to Journal of Web Semantics.
14. M.-A. D. Storey, M. A. Musen, J. Silva, C. Best, N. Ernst, R. Fergerson, and N. F. Noy. Jambalaya: Interactive visualization to enhance ontology authoring and knowledge acquisition in Protégé. *Workshop on Interactive Tools for Knowledge Capture (K-CAP-2001)*, 2001.
15. Christoph Tempich and Raphael Volz. Towards a benchmark for semantic web reasoners - an analysis of the daml ontology library.
16. Sui-Yu Wang, Yuanbo Guo, Abir Qasem, and Jeff Heflin. Rapid benchmarking for semantic web knowledge base systems. *Proceedings of the 4th International Semantic Web Conference(ISWC2005)*, 2004.