Increasing the Expressiveness of OWL Through Procedural Attachments

Dennis Taylor
CDM Technologies, Inc.
San Luis Obispo, California, USA

Abstract
The purpose of this paper is to provide an introduction to the OWL Web ontology language, a survey focused on the current state of the art in OWL inferencing capabilities, and a historical perspective on procedural attachments. The perspective is aligned with current OWL research. Several limitations of the OWL language and proposed extensions to overcome these limitations are discussed. A framework that provides empirical testing support for evaluating the effects of procedural attachments to the OWL inferencing capabilities is outlined. The examples presented suggest that it is possible to provide rule-based extensibility support for OWL that does not limit the ability of an OWL reasoner to perform consistency checking. Specifically, the framework is used to demonstrate with an example experiment the ability to provide support for the compound sub-property axiom.

Keywords: Google, inference engine, interoperability, ontology, OWL, OWL-DL, OWL-Full, OWL-Lite, OWL Reasoner, procedural attachment, World Wide Web.

1. The Semantic Web
One of the primary sources of information today is the World Wide Web, commonly referred to as the Web. It has revolutionized the way people communicate, exchange commodities, and even the way people think. The Web is accessed primarily by key-word searches through search engines such as Google, Yahoo, and MSN. Without these tools the Web may have never become the revolutionary technology that it is today. Key-word searches continue to improve, and give better access to the vast array of unstructured data on the Web. Regardless of this apparent success it has been recognized in the literature that key-word searches have several flaws. According to Parsons (2004) “… despite improvements in search engine technology, the difficulties remain essentially the same. It seems that the amount of Web content outpaces technological progress”. Parsons identifies several difficulties that key-word searches have not overcome. For example, key-word search results are highly sensitive to vocabulary. Also, key-word searches deliver only sites that contain words included in the search query and the results are limited to a list of single Web pages.

These difficulties can be considered semantic problems. Wood (1985) defines semantics as the scientific study of the relationships between signs and symbols and what they denote or mean. With a deeper understanding of the semantics of words a search could return sites that contain synonymous terms in addition to the words searched for. Modern search engines also return only lists of single Web pages. They could be further enriched by providing an ability to link concepts from multiple pages. For the most part these problems are difficult to address due to the unstructured nature of the Web. It is difficult to determine appropriate semantics for data that are unstructured. To properly determine the semantics of concepts in unstructured data most techniques require some defined context (Ceglowski and Cuadrado 2003). However, the majority of Web sites do not provide any context for the data found.

Some recent work has shown that it may be possible to gleam semantic information including
context from unstructured data. Cilibrasi and Vitanyi (2004) have shown promising results by performing automatic meaning discovery using Google’s search index. Advertisements displayed on Google’s g-mail facility appear to exhibit some level of semantic behavior. For example, an e-mail message containing the term *thesis* will automatically trigger several advertisements for book binding services. Admittedly this apparently semantic behavior may be little more than a clever use of algorithms.

Although more and more clever algorithms may lead to richer semantic usage of the Web, a sizable community of researchers has begun to pursue a more proactive approach. They propose to organize data in a more structured manner in preference to attempting to extract semantic information from unstructured data. This approach, commonly referred to as the Semantic Web, provides a framework that allows data to be shared across multiple boundaries. In this regard an understanding of the Semantic Web is critical to the value of this work.

2. **The Problem of Interoperability**

The fundamental aspects of the Semantic Web are external to the Web itself. They relate to a fundamental problem of Software Engineering in general, namely interoperability. It can be argued that the Web provides the first large scale example of how complex the problem of software integration really is.

Jackson (2005) describes the specifications of software as a boundary between the outside environment and the software. Adding a second software application to the environment shown in Figure 1 does not materially change the environment, if the two software applications are essentially isolated from each other (Figure 2). However, the situation is quite different if the two applications need to communicate and share data with each other.

![Figure 1: Single application](image1.png) ![Figure 2: Two communicating applications](image2.png)

The need for interoperability between two or more applications has become an increasingly common requirement and has therefore become the subject matter of a great deal of research. Arguably much of the success of the Java programming language is due to its J2EE interoperability capabilities provided by tools such as JBOSS, JMS, and SOAP. Using almost any standard programming technique results in clearly defined interfaces acting as a bridge or translator between the two systems (Figure 3). The complexity of a large number of interconnected systems can quickly become unmanageable (Figure 4).

---

1 See [http://www.w3.org/2001/sw/](http://www.w3.org/2001/sw/).

2 Java 2 Platform Enterprise Edition (J2EE).
To further illustrate this point imagine the following situation. If four prisoners speaking different languages were to be incarcerated in the same room it would make little sense for each prisoner to attempt to learn the languages of the other three prisoners.

3. Ontologies

According to Szolovitz and Ohno-Machado (2005) an ontology “... is a formal, explicit specification of a shared conceptualization. In other words, an ontology describes the concepts in the domain and the relationships that hold between these concepts. It is a shared vocabulary that can be used to model a domain (i.e., the objects and/or concepts that exist, their properties and relations”).

Figure 5: Two applications sharing the same semantics

If two software applications share the same semantics for the objects in their respective systems then they will be able to communicate within a meaningful context (Figure 5). Naturally, the entry of other applications into the environment shown in Figure 5 will require the addition of further concepts that must be described and shared within the domain ontology. Although the notion of an ontology is quite appealing there are several practical issues that need to be resolved. One of those issues is the need for a standard language to represent domain knowledge in the form of an ontology. The language recommended for this purpose by the W3C working
group is OWL\textsuperscript{3}.

OWL has enjoyed a fair amount of interest since its introduction. One of the first large ontologies implemented in OWL was the GALEN biomedical terminology knowledge base (KB), which contains many biomedical concepts including anatomy, drugs, diseases, signs and symptoms (Rogers and Rector 2006). The GALEN KB has been used in numerous applications in the medical field (Rogers and Rector 1996, Rosse et al. 1998, Wroe and Cimino 2001) and serves as one of the primary examples of the real world application of ontologies. The medical field incorporates a very large collection of shared information that is ideal for ontology representation. For instance the GALEN KB has been used to store diabetological terminology to assist in the diagnosis and treatment of Diabetes (Birkmann et al. 1997).

Originally developed under the direction of the Defense Advanced Research Projects Agency (DARPA) as an extension to the Resource Description Framework (RDF), the OWL ontology language has become widely recognized as a powerful means of representing knowledge.

4. OWL

The OWL Web Ontology Language is designed for use by applications that need to process the content of information instead of just presenting data to humans. OWL facilitates greater machine interpretability of Web content than that supported by XML, RDF, and RDF Schema (RDF-S) by providing additional vocabulary along with formal semantics. OWL has three increasingly-expressive sublanguages: OWL Lite; OWL DL; and, OWL Full.

OWL is a language specification that allows ontologies to be expressed in the form of a description logic. Of particular interest among the OWL documents provided by W3C is the OWL Use Cases and Requirements document found at http://www.w3.org/TR/webont-req/. This document outlines the essential requirements for OWL. The requirements specified are general enough to be good requirements for any appropriate ontology language. The first two requirements are essential for maintaining semantic clarity.

R1. Ontologies as distinct resources

R2. Unambiguous concept referencing with URIs

The first requirement addresses the need for every OWL ontology to be a unique reference. The second reiterates the importance of uniqueness by adding the need for unambiguous concept references. In reference to URIs W3C states: “… Uniform Resource Identifiers (URIs, aka URLs) are short strings that identify resources in the Web: documents, images, downloadable files, services, electronic mailboxes, and other resources.” OWL is ideal for the Semantic Web because every reference is a URI allowing any OWL document to reference any other by way of the Web. It is essential that the concepts in an ontology remain unambiguous, since a software program will not be able to distinguish one concept from another if there are ambiguities.

\textsuperscript{3} The OWL language is a revision of the DAML+OIL Web ontology language. DAML+OIL was developed by the "US/UK ad hoc Joint Working Group on Agent Markup Languages" that was jointly funded by the Defense Advanced Research Projects Agency (DARPA) in the US and the European Union (EU). The World Wide Web Consortium (W3C) created the Web Ontology Working Group, which published the first draft of the OWL language specification in July 2002. The OWL specifications became a formal W3C recommendation in February 2004.
4.1 OWL Basics

While it is not intended to describe the syntax of OWL in detail, some of the basic concepts will be presented to facilitate further discussion. Since OWL is an extension of RDF it inherits the triple notation used by RDF. This triple notation is quite simple consisting of a subject, object, and predicate (Figure 6).

![Figure 6: RDF triple notation](image)

Anything expressed in an OWL ontology uses the triple graph notation of RDF. The fundamentals of OWL can be subdivided into: classes; data types; properties; property restrictions; and, individuals. Instead of adhering to the RDF naming convention for a triple of subject, predicate and object, OWL has chosen a more fitting terminology. The subject of a triple is always a class. The object of a triple can be a class or data type and the predicate of a triple is a property. A property can either be an object property (Figure 7) that links two classes or a data type property (Figure 8) that links a class to a data type.

![Figure 7: Object property](image) ![Figure 8: Data type property](image)

Properties may also contain restrictions. One type of restriction is a value restriction. Such a restriction can limit the domain or range of a property. A restriction of a properties range will limit the classes or data types that can be used as the range for that property when applied to a particular class description. For example a Human Child class could have a restriction on the property hasParent to restrict the range of the property to be Human. In this way a Human Child could never have a Tree for a parent.

![Figure 9: Family taxonomy](image)

Another restriction type is the cardinality restriction that limits the number of values of a
property that a class can have. Cardinality can be expressed as a minimum, maximum or both. For example a Spider would have both a maximum and a minimum cardinality of eight legs. An individual in OWL may be an instance of several classes or none of the classes defined. However, an individual will have a set of assertions that define its place in the concept hierarchy as well as its associations to other individuals. Once again these characteristics are all defined using the triple notation. By building a series of these triples with various restrictions a description of a concept can emerge. An ontology that contains some of the concepts that describe a family will be presented. This ontology has a fairly straightforward taxonomy (Figure 9) and will be used throughout the rest of this paper to discuss various features and limitations of OWL.

As shown in Figure 9, a person class is defined as the subclass of owl:Thing. The classes Male and Female are then defined as subclasses of the class Person. These are the only defined classes in this simple family taxonomy. For this ontology there are many more properties defined than classes. The following is a list of properties defined with their domain and range restrictions.

<table>
<thead>
<tr>
<th>Property</th>
<th>Domain</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>uncleOf</td>
<td>Male</td>
<td>Person</td>
</tr>
<tr>
<td>auntOf</td>
<td>Female</td>
<td>Person</td>
</tr>
<tr>
<td>nieceOf</td>
<td>Female</td>
<td>Person</td>
</tr>
<tr>
<td>newphewOf</td>
<td>Male</td>
<td>Person</td>
</tr>
<tr>
<td>siblingOf</td>
<td>Person</td>
<td>Person</td>
</tr>
<tr>
<td>brotherOf</td>
<td>Male</td>
<td>Person</td>
</tr>
<tr>
<td>sisterOf</td>
<td>Female</td>
<td>Person</td>
</tr>
<tr>
<td>parentOf</td>
<td>Person</td>
<td>Person</td>
</tr>
<tr>
<td>motherOf</td>
<td>Female</td>
<td>Person</td>
</tr>
<tr>
<td>fatherOf</td>
<td>Male</td>
<td>Person</td>
</tr>
<tr>
<td>childOf</td>
<td>Person</td>
<td>Person</td>
</tr>
</tbody>
</table>

The properties brotherOf and sisterOf are subProperties of siblingOf. The motherOf and parentOf properties are also the subProperties of the parentOf property. More detailed descriptions of OWL can be found in the OWL semantics and abstract syntax section of the W3C Web site.

### 4.2 OWL as an Ontological Medium

There are additional benefits for using OWL as an ontological medium. These added benefits derive from OWL’s semantics and allow OWL reasoners to perform various inferencing functions, including but are not limited to:

- Checking Ontology Consistency to ensure that any instances of an ontology meet all of the restrictions of that ontology and do not produce any contradictions.
- Concept satisfiability, which checks if it is possible for a class to have any
instances. If a class is unsatisfiable, then defining an instance of the class will cause the entire ontology to be inconsistent.

- Classification, which computes the subclass relationships between every named class to create the complete class hierarchy. The class hierarchy can be used to answer queries for all or only the direct subclasses of a class.
- Realization, which finds the most specific classes that an individual belongs to (i.e., computes the direct types for each of the individuals).

4.3 Consistency Checking

In systems like OWL where concepts are intended to be unambiguously defined it is very important that there is a way to maintain consistency of the information described. Inconsistencies in a stand-alone system can often pose problems but do not typically become an area of concern until interoperability between systems is necessary. Controlling inconsistencies in the Semantic Web is the primary goal of some OWL inference engines (Zou et al. 2004). They typically tackle the problem of eliminating inconsistencies in two stages. The focus of the first stage is to detect inconsistencies, while the second stage addresses the resolution of inconsistencies. One eventual goal of the Semantic Web is to enable trust of various systems. Once an inconsistency has been identified steps have to be taken to resolve the inconsistency.

For example, if one source states that a Human is the parent of a Human Child and another source asserts that there is a Human Child with a Tree as a parent then one of these sources must be wrong. It is obvious to a person that the second source is wrong since clearly trees do not have human children. Identifying which source is the most trustworthy may be an appropriate means of resolving the inconsistency. Some research has been conducted on the various uses of trust in the Semantic Web (Klyne 2002, Golbeck et al. 2003). In this paper we will be concerned only with the detection of inconsistencies and in particular the ability for an inference engine to perform consistency checking on an OWL document.

There are several inference capabilities of OWL reasoners that have received a great deal of attention. One of these is consistency checking. This capability is used to ensure that asserted facts in an OWL ontology are not contradictory. The simplest example of two contradictory statements is: A is true, and A is false. Since A cannot be both true and false, the statements are contradictory. While consistency checking is essentially involves the verification that an ontology has no contradictory statements, W3C provides the following more formal description of consistency checking.

**Definition:** The translation of a separated OWL vocabulary, $V' = VO + VC + VD + VI + VOP + VDP + VAP + VXP$, written $T(V')$, consists of all the triples of the form

$v$ rdf:type owl:Ontology . for $v$ _ VO,
$v$ rdf:type owl:Class . for $v$ _ VC,
$v$ rdf:type rdfs:Datatype . for $v$ _ VD,
$v$ rdf:type owl:Thing . for $v$ _ VI,
$v$ rdf:type owl:ObjectProperty . for $v$ _ VOP,
$v$ rdf:type owl:DatatypeProperty . for $v$ _ VDP,
$v$ rdf:type owl:AnnotationProperty . for $v$ _ VAP, and
$v$ rdf:type owl:OntologyProperty . for $v$ _ VXP.
**Definition:** Let D be a datatype map. An Abstract OWL interpretation, I, with respect to D with vocabulary consisting of VL, VC, VD, VI, VDP, VIP, VAP, VO, satisfies an OWL ontology, O, if:

1. each URI reference in O used as a class ID (datatype ID, individual ID, data-valued property ID, individual-valued property ID, annotation property ID, annotation ID, ontology ID), belongs to VC (VD, VI, VDP, VIP, VAP, VO, respectively);
2. each literal in O belongs to VL;
3. I satisfies each directive in O, except for Ontology Annotations;
4. there is some o _ R with <o,S(owl:Ontology)> _ ER(rdf:type) such that for each Ontology Annotation of the form Annotation(p v), <o,S(v)> _ ER(p) and that if O has name n, then S(n) = o; and
5. I satisfies each ontology mentioned in an owl:imports annotation directive of O.

**Definition:** A collection of abstract OWL ontologies and axioms and facts is consistent with respect to datatype map D if there is some interpretation I with respect to D such that I satisfies each ontology and axiom and fact in the collection.

If this definition seems confusing then it may be replaced by a simpler alternative definition that is based on the formalism of the description logic, as follows.

An OWL ontology can be divided into two distinct parts, namely the terminological box (Tbox) and the assertional box (Abox). The TBox contains hierarchical concept definitions, whereas the Abox contains assertions that state where in the hierarchy individuals belong, and the associations between those individuals.

(1) Every employee is a person ……… belongs in the TBox, while the statement:

(2) Bob is an employee …………….. belongs in the ABox.

With this distinction a better definition of Consistency Checking can be given as: the operation to check the consistency of an ABox with respect to a Tbox. If a Tbox defines a concept Employee as in (1) above then if the individual Bob is asserted to be an animal in the Abox, the Abox will not be consistent with respect to the Tbox since Bob is an animal and not a person. The distinction between Tbox and Abox will become more important later in this paper.

The ability to perform consistency checking and the other inference capabilities mentioned previously depend on what parts of the OWL language are being used. Certain expressive forms of the language can lead to a loss in inference capabilities. The distinction between what leads to possible inference and what does not has prompted a split in the OWL language. Before discussing these distinctions it is important that the reader has a grasp of the concepts of *decidability* and *tractability*, since they are both fundamental in understanding the capabilities of OWL.

**4.4 Decidable and Undecidable**

Decidability has been an area of study since the humble beginnings of computer science when Alonzo Church and Alan Turing independently hypothesized about the nature of computable
problems in what is commonly referred to as the Church-Turing thesis (Turing 1936, Church 1934). The thesis claims that any calculation that is possible can be performed by an algorithm running on a computer, provided that sufficient time and storage space are available. The most important result of this thesis is not so much what is computable but that there exist problems that are not computable and are therefore undecidable as far as a computer is concerned. This does not mean that it will take a long time for a computer to find the answer but that it is impossible for a computer to find an answer. One such problem is the Halting problem (Turing 1936). It is not the intention of this paper to provide a detailed account of this problem. Instead the reader is referred to the cited description or any book on computational complexity. However, it is very important to know when a problem is undecidable so that software developers do not waste their time trying to solve unsolvable problems. Later this will become important in understanding that some capabilities of OWL are undecidable.

4.5 Tractable

Another important concept in computer science and general problem solving is that of tractability. A tractable problem is one that is decidable within a reasonable amount of time and memory. Reasonable is a relative term but can be quantified to some degree: if a week or even a year is considered a reasonable period of time to answer a question than certainly a century would be an unreasonable period of time. Problems that fall into the category of intractable are those that are decidable but take an exponential amount of time to compute. As a problem becomes larger it takes an exponential amount of time longer to find the answer. The best way to comprehend this is to imagine the growth of an exponential problem. If an exponential problem has a size \( n \) equal to 20 then an exponential growth of \( n \) would be nearly a million. If each of those million possibilities takes one second then it will take about 12 days of computation time to solve the problem. Now consider a problem of size \( n \) equal to 40, only twice as large as the previous example. The time required to compute an answer for this problem is nearly 35,000 years. This would certainly be considered an unreasonable amount of time. Exponential problems are considered to be NP-Complete. Tractability will also become important in understanding some of the capabilities of OWL later in this paper.

4.6 Sub-Languages of OWL

The questions of decidability and tractability within OWL has lead to the distinctions of three sub-languages. Much of the ongoing academic work on OWL has been based on formalizing the capabilities of the language by proofs of decidability, presenting algorithms that are tractable, or proving that an algorithm is NP-Complete. It is readily seen that the OWL language in its most expressive form is highly undecidable and that the language in its most decidable form is not very expressive. The balance between expressiveness and decidability has created the necessary divisions in the language. This section will briefly describe the capabilities of each of the three sub-languages.

4.6.1 OWL-Lite

OWL-Lite is the least expressive of the sub-languages but maintains the most decidable and tractable capabilities. W3C has stated that “… OWL-Lite supports those users primarily needing a classification hierarchy and simple constraints.” OWL-Lite maintains all of the restrictions of OWL-DL and adds several more. It is ideal for a simple class hierarchy because it allows classes to be subclasses of others. In the family ontology example a Male would be a subclass of Person. It also allows classes to be
specified as equivalent to other classes. For example the Person class could be equivalent to a Human class.

OWL-Lite allows three basic restriction to be applied to classes. These include cardinality, allValuesFrom, and someValuesFrom. The cardinality restriction is limited to values of zero or one. A cardinality restriction of one would imply that a class has to have one value of the specified property. For example, the United States has one and only one president. A cardinality of zero implies that the class cannot have that property. A democracy cannot have a dictator. The United States would be considered a democracy since it does not have a dictator. The allValuesFrom and someValuesFrom restrictions provide a way to define a class from classical predicate logic and are equivalent to the classic (for-all) and (there exists) definitions. OWL-Lite makes an additional constraint on these restrictions that they must apply only to class names. A detailed description of these restrictions is not important for this work and is therefore omitted.

OWL-Lite allows properties to be transitive, symmetric, and inverses of other properties. When properties are defined with these restrictions an inference engine is able to automatically infer additional information about the individuals being processed. Examples of each of these characteristics were used to extend the Family ontology described previously.

  Transitive – If Bill is the sibling of Mary and Mary is the sibling of Bob then
  Bill is the sibling of Bob.

  Symmetric – If Bill is the sibling of Bob then
  Bob is also the sibling of Bill.

  Inverse - If Mary is the mother of Susan then
  Susan is the child of Mary.

The cardinality restrictions of OWL-Lite cannot be applied to properties that are also transitive. The restrictions provided by the semantics of OWL-Lite allow inference to be both decidable and tractable. Researchers at the University of Manchester have shown that OWL-Lite can be reduced to knowledge base satisfiability in the SHIF(D) description logic (Horrocks and Patel-Schneider 2004a). By showing that the capabilities of the various sub-languages are equivalent to certain description logic capabilities they have been able to tie OWL to the large body of work already produced in description logic.

4.6.2 OWL-DL

OWL-DL is a more expressive version of OWL-Lite. Every legal OWL Lite ontology is a legal OWL-DL ontology. Therefore, every valid OWL-Lite conclusion is also a valid OWL-DL conclusion. This is an important consequence. If part of an ontology may be classified as OWL-Lite then any inference that is possible on that part in isolation is still applicable to the ontology as a whole. Some inference problems can be performed on isolated parts of an ontology such as the classification of a single individual (Horrocks et al. 2000).

OWL-DL does not restrict the values specified for cardinality restrictions. However, it does maintain that properties that have cardinality restrictions are not transitive similarly
to OWL-Lite. Although the constraints on OWL-DL may appear to be arbitrary, they are actually largely based on description logic reasoning capabilities. W3C states that “… in particular, the OWL-DL restrictions allow the maximal subset of OWL-Full against which current research can assure that a decidable reasoning procedure can exist for an OWL reasoner”. The same group that has demonstrated that OWL-Lite can be reduced to knowledge base satisfiability in SHIF(D) description logic, has shown that OWL-DL can also be reduced in SHOIN(D), a more expressive description logic. Horrocks has published a survey outlining the state of art in description logic and current challenges that still exist ([Horrocks 2005]).

4.6.3 OWL-Full

OWL-Full contains all of the OWL language constructs and provides free, unconstrained use of RDF constructs. It allows classes to be treated as individuals and this allows it to be very expressive. For example, an ontology in OWL-Full would allow a particular type of aircraft, say a C-17, to represent a Class of that object but also to be an instance of the AirplaneType class. OWL-Full actually takes this one step further so that all data values are also considered to be part of the individual domain. The use of OWL-Full negates the guarantees provided by OWL-Lite and OWL-DL. The language is largely undecidable.

4.7 OWL Inference Engines

Since the W3C recommendation of OWL there have been numerous attempts at creating an OWL inference engine. All of the inference engines examined take one of two approaches. The first approach is based on Description Logic reasoners and the Tableaux algorithm. The second approach uses restricted first order logic theorem proofs.
The research group at the University of Manchester (UK) has specified some improvements currently used for intractable problems that can be reduced to constraint satisfaction problems (Horrocks and Patel-Schneider 2004a). Many of these improvements are variations of a commonly used algorithm for OWL inference known as the Tableaux algorithm. These algorithms provide a reasonable solution to OWL-DL reasoning.

Table 1 (Zou et al. 2004) provides a listing of the most popular inference engines and their capabilities as of 2004. It is interesting to note that this is a very active field as the Pellet inference engine used for the experiments in this paper now supports most consistency checking in OWL-DL instead of the mentioned OWL-Lite.

Some more detailed information about each of these systems as of 2004 is provided by Zou (2004). The Racer system implements SHIQ(D) using a Tableaux algorithm. It supports OWL-DL and both Tbox and Abox reasoning. The FaCT system implements SHIQ, but supports only Tbox reasoning. Pellet implements SHIN(D) and includes a complete OWL-Lite consistency checker supporting both Abox and Tbox queries. Vampire is a FOL theorem prover (Riazanov 2003). Otter is also an FOL theorem prover and is used in the Surnia inference engine. There are also a group of inference engines that take advantage of a subset of FOL known as Horn Logic. These engines typically take advantage of already known Horn Logic tools such as Jena, Jess, Triple, and XSB. Higher order first order logic has been experimented with in some systems through the use of Flora (Zou et al. 2004).

4.8 Limitations of OWL

Several research groups have identified missing capabilities of OWL. One of the primary groups is Horrocks’ team at the University of Manchester. Several of their findings will be briefly
presented in this section.

Complex Role Inclusion Axioms are not possible in any of the current sub-languages of OWL. These axiom take on two forms:

- **ownership** propagates from an aggregate to its parts (e.g., the owner of the car is also the owner of the car's parts);
- **localisation** propagates from a division to its aggregate (e.g., a trauma located in a part of a body structure is a trauma of the body structures (Horrocks and Patel-Schneider 2004a).

These are both valuable axioms that are not expressible in OWL. Work on the Galen medical terminology knowledge base has led to the need for expressing such axioms. The second axiom is very important so that trauma can be accurately diagnosed. If a patient has a trauma in a ventricle in his or her heart it would be useful if that trauma could be identified as a problem in the patient’s heart. Or in the case of a torn cartilage meniscus in the knee the system should be able to identify a knee injury. These are critical distinctions since doctors typically specialize in areas of the body. For instance the patient with a heart trauma will need a cardiologist while the patient with a knee injury may need an orthopedic surgeon. It would be a unfortunate to have to explicitly define the doctor who works on every sub-part of every body part. Several attempts have been made to include these axioms in OWL. One such attempt used Grammar Logic (Baldoni and Martelli 1998), but has been proven to be undecidable. Horrocks’ group has managed to show that with the RIQ description logic complex role inclusion axioms can be decidable. A benefit is that with some limitations of the SHIQ description logic it can be translated into RIQ. This restriction of SHIQ is not however identified within the W3C recommendation and therefore the distinction is not made. It is important to note that the satisfiability procedures for RIQ experiences enormous exponential growth, but has been validated on small examples within the Galen ontology.

Compound Sub-Property Axioms are also not currently possible in any of the sub-languages of OWL (Horrocks 2005). The compound sub-property axiom can be defined as the composition of several properties to assert another property. An example of this that will be used later in the experimental section of this paper shows that an uncle can be defined as the brother of a mother. Since the uncleOf property relies on the brotherOf property as well as the motherOf property it is not possible to express this in OWL.

Both the complex role inclusion axiom and the compound sub-property axiom discussed can be solved using other languages such as rule languages. This is a commonly recognized fact within the OWL and Semantic Web community. This limitation has lead to an apparent need for rule-based support that is somehow coupled with OWL. The logical foundations for the Semantic Web provide a diagrammatic view of the Semantic Web technologies and in this depiction they layer rules on top of the Ontological vocabulary in a segment labeled Logic (Figure 10). Since OWL provides both the Ontological vocabulary and some of the Logic there is a discrepancy between how rule-based support and OWL should be used in conjunction.

Several different approaches have been applied to add rule support to OWL. However, none of the approaches examined involves a completely extensible method such as the procedural attachment approach presented in this paper.

The Semantic Web Rule Llanguage (SWRL) is one of the most popular proposals for an OWL extension. SWRL combines OWL and ruleml and has been proposed by Horrocks and Patel-Schneider (2004b) to compensate for several of the lacking capabilities of OWL (Horrocks
2005). By exploiting the features of first order predicate logic in both description logic and rule languages this group believes that they will be able to extend the expressiveness of OWL-DL. The OWL inference engine Vampire was written as a general purpose first order logic theorem prover and has some support for SWRL (Tsarkov et al. 2004, Riazanov and Voronkov 2002).

Several other researchers, who also associated with the University of Manchester, have performed similar work and have provided a survey based on their perspective of the state of description logics (Hustadt and Motik 2005). They also discuss a combination of OWL-DL and rules, however, they define a subset of rules referred to as DL-safe to maintain decidability.
ERROR: stackunderflow
OFFENDING COMMAND: ~

STACK: