

A Possible Simplification of the Semantic Web Architecture

Bernardo Cuenca Grau^{*}
Maryland Information and Network Dynamics Lab.
8400 Baltimore Avenue
College Park, Maryland, USA
bernardo@mindlab.umd.edu

ABSTRACT

In the semantic web architecture, Web ontology languages are built on top of RDF(S). However, serious difficulties have arisen when trying to layer expressive ontology languages, like OWL, on top of RDF-Schema. Although these problems can be avoided, OWL (and the whole semantic web architecture) becomes much more complex than it should be. In this paper, a possible simplification of the semantic web architecture is suggested, which has several important advantages with respect to the layering currently accepted by the W3C Ontology Working Group.

Categories and Subject Descriptors

I.2.4 [Knowledge Representation Formalisms and Methods]: Representation Languages; F.4.1 [Mathematical Logic]: Model Theory

General Terms

Languages, Knowledge Representation, Web Technologies

Keywords

Semantic Web, Resource Description Framework (RDF), Resource Description Framework Schema (RDF-Schema), Ontology Web Language (OWL), Description Logics

1. INTRODUCTION

The semantic web [3] is an extension of the World Wide Web in which both data and its semantic definition can be processed by computer programs. The next generation of the Web will combine existing web technologies with knowledge representation formalisms in order to provide an infrastructure allowing data to be processed, discovered and filtered more effectively in the web. A set of new languages organized in a layered architecture will allow users and applications to write and share information in a machine-readable way and will enable the development of a new generation of technologies and toolkits.

In the semantic web architecture, Web ontology languages are built on top of RDF(S). However, serious difficulties have arisen when trying to layer expressive ontology languages, like OWL, on top of RDF-Schema [10][17]. Although these problems can

^{*}The author also works at the Institute of Robotics, Department of Computer Science, University of Valencia (Spain)

be avoided, OWL (and the whole semantic web architecture) becomes much more complex than it should be. In this paper, a possible simplification of the semantic web architecture is suggested. The paper is organized as follows: section 2 reviews the main features of RDF(S) model theory and of standard description logics formalisms. In section 3 the main design principles followed in the design of the semantic web languages are enumerated. Section 4 reviews the layering of OWL on top of RDF(S), while section 5 summarizes other proposals for layering the semantic web. Finally, in section 6 a possible simplification of the semantic web layering is suggested and its main features and advantages are discussed.

2. RDF AND RDF-SCHEMA (RDF(S)) SYNTAX AND SEMANTICS

RDF [6] and RDF-Schema [4][6] are at the base of the semantic web, so that all the other languages corresponding to the upper layers are built on top of them. RDF(S) defines both the syntax and semantics of the subsequent layers. However, RDF(S) is built upon a non-standard thesis of representation [9]. The syntactic and semantic characterization underlying RDF(S) differs significantly from the syntax and semantics of most first-order logical languages [10].

The syntax of RDF is based on triples, atomic chunks of information composed of three different elements $\langle S, P, O \rangle$. Each RDF triple:

- Is composed of a subject, a property and an object
- Represents a single fact, with a well-defined and complete meaning
- Is the minimum piece of knowledge that can be represented in RDF

Every RDF(S) document is equivalent to an (unordered) set of triples. However, although all these triples appear together in the document, each one is given a meaning, which is independent of the meaning of other triples, in the semantic characterization of the language.

RDF has a very limited set of syntactic constructs. No other syntactic construct except for triples is allowed, neither in RDF(S) nor in the remaining languages of the semantic web architecture. Moreover the meaning of the triple syntax must be retained in higher layers, i.e all the triples in RDF are assigned the same basic meaning so that richer languages are not allowed to modify this basic meaning and can only extend it.

RDF has been given a model-theoretic semantics, which provides formal meaning to its abstract syntax [6]. All the interpretations in RDF(S) model theory are relative to a set of names (URIs),

called the vocabulary of the interpretation. An RDF(S) interpretation is formally defined as a tuple $I = \{IR, IP, IEXT, IS, IL, LV\}$, composed of three sets and three mappings:

- Sets
 - IR is a non-empty set, called the domain of the interpretation
 - $IP \subseteq IR$ is the set of properties.
 - $LV \subseteq IR$ is the set of literal values
- Mappings
 - $IEXT$ is a mapping $IEXT : IP \rightarrow IR \times IR$ from IP into the set of pairs $\langle x, y \rangle$, with $x, y \in IR$
 - IS is a mapping $IS : URIrefs\ in\ V \rightarrow IR$
 - IL is a mapping from typed literals in V into IR

In RDF model theory all resources (individuals, classes and properties and even the language primitives) are elements of the domain of the interpretation. The function IS maps every name (URI) of the vocabulary to an element of the domain. All the names of a RDF vocabulary, even the names contained in the reserved vocabulary (language primitives) can appear at any position of a triple. This means that triples like:

rdf : type rdf : type rdf : Property.
rdfs : range rdfs : domain rdf : property.
rdfs : Class rdf : type rdfs : Class.

are allowed in RDF(S). These triples composed of language primitives define the metamodelling architecture of RDF(S) and can lead to unwanted consequences specially when layering more expressive languages like OWL on top of RDF(S) [15].

In RDF(S) all the names are mapped to a resource using IS . Thus, a resource is defined in RDF(S) as everything that has an identity, i.e., everything that can be referred to by a URI.

Names that occur in a RDF document in a property position of any triple are mapped by IS to a special subset of IR called IP . The elements of IP are then mapped to a set of ordered pairs of elements of the domain by the function $IEXT$. The set of ordered pairs corresponding to a property resource form the extension of the property. RDF(S) semantics distinguishes objects from their extensions. Objects representing individuals are defined to have empty extensions. Property objects are "special" in RDF because they have non-empty extensions. Class primitives are not fundamental primitives in RDF(S). The extension of a resource representing a class is defined through the property extension of $IS(rdf:type)$

The distinction between objects and their extensions allows for a certain "syntactic freedom" in RDF(S):

- The extension of a resource can contain the resource itself without violating the axioms of set theory. Extensions are sets and they do not contain themselves, but the resource resulting from the application of IS .
- Two properties or two classes can have the same instances and still be different entities. In the case of ordinary sets, equality is defined through the principle of extensionality: two sets are equal in axiomatic set theory iff they contain the same elements.

However, there are two main non-standard features in the "free" syntax of RDF:

- The resources representing the language primitives can appear at any position inside a triple
- There is no separation of vocabularies. Classes, properties and individuals do not form disjoint collections in RDF(S). Therefore, the same element of a vocabulary can be used to denote for example a class and an individual, or even a class, a property and an individual at the same time
- Self-referential syntactic constructs are allowed

RDF and RDF-Schema are at the basis of the semantic web. All the languages in the upper layers have been designed to use the triple syntax in all their constructs and to extend the meaning assigned to triples by RDF model theory adding new semantic conditions to interpretations.

2.1 Standard Description Logics Semantics

Description logics (DL) [2][12] are a set of knowledge representation formalisms whose semantic characterization is based on standard first-order logics. Meaning is provided by interpretations, which define the formal semantics of the logic.

An interpretation in DL is a mathematical structure $I = \{\Delta^I, \cdot^I\}$ consisting of:

- A nonempty set Δ^I , called the domain of the interpretation. The domain is divided into two disjoint sets:
 - The abstract domain is the set of all the individuals
 - The concrete domain is composed of data values and is used to integrate datatypes in description logics
- An interpretation function \cdot^I that maps:
 - Every concept (class) name to a subset of Δ^I
 - Every role (property) name to a subset of $\Delta^I \times \Delta^I$
 - Every individual to an element of Δ^I

The interpretation function can be extended to complex concepts and roles and can be used to provide meaning to axioms in the knowledge base.

OWL semantics is based on DL, so that an OWL ontology corresponds to a DL terminology. However, standard DL semantics differs significantly from RDF(S) semantics [9]:

- The interpretation function maps individual names to elements of the domain. Names used in other ways do not denote an element of the domain. Class and property names are not mapped to resources: they are directly mapped to sets of resources and sets of ordered pairs of resources respectively. In DL a resource is no longer "everything that can be referred to by a URI". Only objects are considered as part of the domain. Consequently, a class cannot contain itself in DL, because classes themselves are not considered to be resources. Elements of the domain in DL do not have extensions, as in RDF(S)
- In DL, class primitives are as fundamental as property primitives. Classes are not defined in terms of properties, as in RDF(S). In RDF(S) all the semantic relationships are reducible to properties.

- In RDF(S), the type membership and the subclass relationships are not distinguished from other predicates and can occur at any position in RDF triples (i.e., they can be arguments for other predicates). However, in DL, these relationships are "special", and are embedded in the formalism. In DL the "type" relationship is clearly and unambiguously defined as "instance of" or "member of", while the "subclass" predicate is interpreted as "subset of". The consequence of the RDF(S) way of representing those primitives is that their meaning must be expressed explicitly through axioms. These axioms may become very complex when extending the language to the ontology layer.
- In DL, the vocabulary to which the interpretation refers is divided into disjoint sets. This means, for example, that classes cannot be properties or objects at the same time. In RDF(S) a resource can be both an individual and a class or even an individual, a class and a property

3. DESIGN PRINCIPLES OF THE SEMANTIC WEB

The semantic web has been designed according to the following principles [6] [16]:

- A resource is considered to be everything that has an identity, i.e. everything that can be referred to with a URI
- RDF and RDF-Schema semantics is largely based on SKIF semantics, a non-standard logical representation formalism
- RDF(S) syntax is based on triples and the way names of different kinds are introduced in those triples is rather free
- Every piece of knowledge representable in a semantic web document should be written (or translated) to triples
- All the triples share a common basic meaning. A document is an unordered set of triples and the meaning of each triple is independent from the others. Triples containing "special" URIs are given an extra meaning by adding semantic constraints to the interpretations whose vocabulary contains that particular URI
- All semantic web languages are forced to use RDF syntax only, and the meaning of RDF triples in all semantic web languages must be compatible with the meaning given to them in RDFS Model Theory
- The semantics of the languages of the ontology layer is based on very expressive description logics (standard FOL formalism)

4. LAYERING OWL ON TOP OF RDF-SCHEMA

Serious difficulties have arisen when trying to layer expressive ontology languages, like OWL on top of RDF(S) using this design premises. Languages based on very expressive DLs include powerful modelling primitives like concept constructors, restrictions and complex axioms. OWL richer theory of classes clashes with the underlying principles of RDF(S) in the same syntax and extended semantics layering [17][10][15]. Semantic integration problems arise when trying to make compatible the semantics derived from very expressive DLs with the triple syntax and the semantics already associated to those triples by RDF(S) model theory.

Although these semantic problems and paradoxes can be avoided, OWL becomes consequently much more complex than it should

be. Non-standard techniques (like the definition of comprehension principles [16]) must be used in order to make the DL semantics compatible with the RDF(S) semantics resulting from the translation of OWL to triples [16].

4.1 The solution adopted by the W3C Web Ontology Working Group

OWL combines features from different representation languages:

- Its semantics is largely based on very expressive description logics
- Its modelling syntax and surface structure have been influenced by frame-based systems
- OWL has been designed to keep the maximum upward compatibility with RDF and RDF-Schema in both syntax and semantics

The multiple influences underlying OWL specification led to many difficult tradeoffs. It has been specially conflictive to bring together satisfactorily the first and the third features. The W3C Web Ontology working group decided to:

- Provide OWL with a RDF-based syntax. This means that any OWL ontology can be written as a RDF graph, made of triples
- Ensure that the semantics of OWL ontologies is consistent with the semantics of RDF. This particular requirement implies two conditions:
 - On one hand, any OWL ontology written in abstract syntax notation is given a direct DL-like model-theoretic semantics; on the other hand, the same ontology can be written in terms of RDF triples and is assigned a RDF-compatible model-theoretic semantics. Both encodings of the same ontology must be semantically equivalent. As RDF triples have RDF meaning, this meaning together with the special conditions added to OWL interpretations in the OWL RDF-compatible semantics must be equivalent to the meaning assigned to the ontology by the direct model-theoretic semantics
 - If OWL is designed to be a semantic extension of RDF(S), then any RDF(S) document must be a valid OWL ontology. In other words, the free syntax of RDF(S) and its corresponding semantics is "exported" to OWL. This means, for example, that it would be possible to add number restrictions to the "subClassOf" relationship, because this relationship is considered syntactically as an "ordinary" property and can be used as the subject of RDF triples. These kind of assertions provide the language with an expressive power that goes beyond standard description logics semantics

It was proved to be impossible to define OWL as a true "same syntax, semantic extension" of RDF(S) while keeping the reasoning services (OWL entailment) decidable. The Web Ontology working group defined three sublanguages of OWL [16], each of which satisfies most (but not all) the requirements:

- OWL full is a true "same syntax, semantic extension" of RDF(S) so that all RDF graphs are allowed. As a consequence, its semantics is beyond the standard DL framework and the language happens to be undecidable. Moreover, it has yet to be proved that the RDF MT compatible semantics for OWL Full gives a coherent meaning to OWL full

- In OWL-DL and OWL-Lite, however, some constraints are applied in order to keep the language in a standard DL framework, and to keep the decidability of the reasoning services [16][10]:
 - Only certain RDF triples and certain combinations of those triples are allowed.
 - The vocabulary of the interpretations is divided into disjoint sets, so that the same URI cannot be used for example as an individual and a property name at the same time. As a result, the domain of the interpretation is also partitioned into disjoint sets.
 - An abstract syntax is defined for the language, so that OWL-Lite and OWL-DL can be defined as the subset of RDF graphs generated by the translation of their respective abstract syntaxes to triples.

With these restrictions the direct model-theoretic and the RDF-complatable semantics for OWL-DL and OWL-Lite are equivalent (OWL-full has neither a direct model-theoretic semantics, nor an abstract syntax).

4.2 Discussion

In the author's opinion this layering of OWL on top of RDF(S) is not truly satisfactory, for several reasons:

1. The future semantic web languages built on top of the ontology layer (e.g. rules languages) will probably not be layered on top of OWL full. OWL full is too expressive (namely undecidable) and its semantic formalization is rather complex, cumbersome and non-standard. As a result OWL full happens to be both difficult to implement and to understand and it is foreseeable that it won't be integrated in many applications. Consequently OWL-full is not the best language to stay at the basis of future and probably more expressive semantic web languages.
2. OWL-DL has been designed to be equivalent to the SHOIN(D) [16][10] description logics. Although OWL-DL is decidable, the entailment problem is computationally intractable in the worst-case and there exists currently no "practical" (easy to implement and with a good computational behaviour in average) reasoning algorithm to solve the satisfiability problem in the (whole) SHOIN(D) logics. The design of efficient tableau-based reasoning algorithms combining nominals, inverse roles, number restrictions, transitive roles and role hierarchies remains an open problem in description logics. Therefore, the only OWL sublanguage that could be easily implemented and optimised happens to be OWL-Lite.
3. The division of OWL in dialects is not clear. While in OIL (Ontology Inference Layer) the different sublanguages were layered according exclusively to their expressive power, in OWL there are two different kinds of layering criteria:
 - The relationship between OWL-Lite and OWL-DL is similar to the relationship between OIL dialects: OWL-DL is more expressive than OWL-Lite and they both share the same underlying semantic formalism
 - The relationship between OWL-DL and OWL-Full is far more complex and unclear
4. The "same syntax, extended semantics" requirement for the semantic web forces OWL specifications to be far more complex than they would be otherwise (e.g. comprehension principles)

5. Two different logical formalisms have been used to define the semantics of the semantic web languages. RDF and RDF-Schema are largely (but not completely) based on the SKIF semantics [7][9], while OWL has been given a standard description logics (direct) model-theoretic semantics. The compatibility requirements between both formalisms force the semantic web to be complex and difficult to understand

5. OTHER PROPOSALS FOR LAYERING THE SEMANTIC WEB

The current layering of OWL on top of RDF(S) has been considered by several authors [17][9][15] to be rather unclear and non-satisfactory. Other approaches for layering the semantic web have been recently suggested. There are mainly two ways to improve the layering of the semantic web:

1. To change some aspects in the design principles of OWL and to leave the RDF(S) specifications unaltered. This approach is based on the idea that some decisions made during the early development of OWL and, in general, of the semantic web layer cake have proven to be bad design decisions up to a certain extent. Particular, in the semantic web architecture all the languages are forced to use RDF syntax so that they are not allowed, neither to extend it nor, to change the basic meaning assigned to triples by RDF(S) model theory.
2. To modify the specifications of RDF(S) in order to make them more compatible with the underlying description logics formalism of OWL-DL. In the semantic web two different formalisms have been used to define the semantics of the languages; RDF(S) is based on an unusual model-theoretic semantics, while OWL (direct) model-theoretic semantics is based on standard description logics. The layering of the semantic web could be made clearer and simpler if the whole semantic web architecture was based on a standard first order logic formalism.

5.1 Syntax and semantic extension

In this approach, both the syntax and semantics of OWL are defined to be an extension of RDF(S) syntax and semantics respectively [17]. This solution is based on the idea that most of the problems and semantic paradoxes that occur when trying to layer OWL on top of RDF(S) in the "same-syntax, semantic extension" approach could be easily avoided if some of the OWL syntactic constructs (e.g. restrictions) could be written using non RDF/XML syntax.

If some "conflictive" constructs of OWL are not translated to triples, then these constructs wouldn't be given a RDF meaning and their semantics would be completely defined by the OWL model theory.

In this proposed layering each new language would share a common syntactic and semantic core with RDF, but would be allowed to include new syntactic constructs and to define their meaning.

5.2 Stratifying RDF-Schema:RDFS(FA)

RDFS(FA) (RDF-Schema Fixed Layer Metamodelling Architecture) [13][15][14] is a sublanguage of RDF-Schema, which has been assigned a (quite) standard description logics semantics. The main design goal of RDFS(FA) has been to define a language for the schema layer with both a standard first-order logic semantics and a clear metamodelling architecture. Moreover, the designers of RDFS(FA) have tried to minimise the changes made to RDFS in order to achieve a maximum reuse of existing RDFS documents and tools.

5.2.1 RDFS(FA) overview

An interpretation in RDFS(FA) is a pair $I = \{IR, IE\}$ where IR is the domain and IE the interpretation function. In RDFS(FA) interpretations, the universe of discourse is divided into four serveral strata or layers, so that each individual, user-defined class or property, and each language primitive are mapped into a certain stratum of the interpretation domain.

1. The instance stratum is composed of objects. Every individual name in the vocabulary is mapped to an element of the domain corresponding to this layer
2. The ontology stratum is composed of both sets of objects (user-defined classes) and sets of ordered pairs of objects (user-defined properties).
3. The Language stratum defines a language for specifying ontologies
4. The metalanguage stratum is used to define the language layer

The ontology and the instance strata define the modelling capabilities of the language, while the two upper layers define its meta-modelling architecture.

It is not the purpose of this paper to analyse in detail the specification of RDFS(FA). A full discussion of the language can be found in [13][15][14]. Nevertheless, as the simplification of the semantic web architecture suggested in this paper is largely based on the RDFS(FA) approach, it is particularly interesting to discuss the main similarities and differences between RDFS(FA) and both RDF(S) model theory and standard description logics formalism.

5.2.2 RDF(S) and RDFS(FA)

Although RDFS(FA) has been designed with the purpose of introducing as few changes as possible in the specification of RDF-Schema, the semantic characterization of RDFS(FA) alters significantly some of the basic assumptions of RDF(S) model theory. Therefore, RDFS(FA) not only simplifies the metamodelling architecture of RDF(S), but also changes the very basics of RDF(S) data model and formal semantics.

In this section, we will consider separately the new features introduced in RDFS(FA). First of all, we will consider the changes introduced in the basics of RDF(S) model theory and then the consequent simplification of the metamodelling architecture of the language will be analysed.

With respect to the basics of the data model and formal semantics, there are several main differences between RDF(S) and RDFS(FA):

1. The vocabulary (set of URIs) of the RDFS(FA) interpretations is partitioned into disjoint sets, as in OWL-DL. As a consequence of this particular design decision, the free syntax of RDF(S) is explicitly constrained in the RDFS(FA) specification. The set of RDFS(FA) graphs can be mapped to a subset of RDF(S) graphs; however, not every valid RDF(S) graph can be mapped to a syntactically correct RDFS(FA) graph. The constraints introduced by RDFS(FA) do alter significantly the syntax and data model of RDF(S) in several important ways:
 - If a URI is used in a property position inside a triple, it cannot be used in the subject position of another triple. The same URI cannot be used at the same time as a class, property or individual
 - Self-referential constructs are suppressed in RDFS(FA)

2. The names of the vocabulary are mapped directly from property or class symbols to sets of pairs of elements and sets of elements respectively. In RDFS(FA) class and property primitives are fundamental primitives. Classes and properties are not objects of the domain: they are mapped directly to sets of resources and to sets of pairs of resources respectively in the adjacent lower stratum. In RDF model theory, classes and properties are first mapped via IS to objects in the domain and then mapped by either ICEXT or IEXT to their extensions. Thus, in RDFS(FA) a class or property is not different from its extension. Consequently, a class cannot contain itself as in RDFS. This also means that the principle of extensionality in set theory applies in a standard way, i.e. two classes are equal if and only if they contain exactly the same instances (in RDF(S) two classes and properties with identical extensions could be mapped via IS to different objects in the domain)
3. Type properties are just instance-of relationships and can only be used to link resources belonging to different layers. Specifically, a "property" can only be used in RDFS(FA) to state that an element or a set belonging to a certain layer is a member of a set in the adjacent upper layer
4. Triples are not treated in a uniform way in the semantic specification of RDFS(FA). In RDF(S) all the triples share a common basic meaning. Triples containing certain language primitives are assigned an extended meaning through the addition of semantic constraints to interpretations. However, in RDFS(FA), as in DL, each triple is given a particular meaning depending on the kind of URIs included in it. There is no basic shared meaning between all the valid triples of the data model.

With respect to the language primitives and their relationships, RDFS(FA) simplifies the metamodelling architecture of RDF-Schema suppressing some of its most confusing aspects. Note that some of the features of RDFS(FA) metamodelling architecture arise as a direct consequence of the general design decisions of the language, discussed above.

- RDFS(FA) disallows the arbitrary use of built-in language primitives, i.e some of the (metamodelling) triples containing two or more language primitives that were valid in RDF(S) have been suppressed in RDFS(FA). The metamodelling architecture in RDF(S) is thus much richer than in RDFS(FA)
- A modelling primitive in RDF(S) might correspond to several language primitives in RDFS(FA) (contained in different strata).
- Language primitives are not instances of themselves
- As in RDF-Schema the language primitives are interpreted either as classes or as properties, but:
 - Language primitives in stratum "i" interpreted as classes are mapped directly into sets of resources in stratum "i-1"
 - Language primitives interpreted as properties are also mapped directly into sets of resources in the adjacent lower stratum

The relationships among different language primitives are defined so that language primitives can only be linked with other primitives layered in a different stratum using an "instance of" (type) relationship.

5.2.3 RDFS(FA) and Standard Description Logics

RDFS(FA) semantics is largely embedded in a standard description logics framework. RDFS(FA) semantics has been conceived to be compatible with OWL-DL and OWL-Lite direct model-theoretic semantics. However, there are two important features in which RDFS(FA) and OWL differ:

1. The domain of the interpretation in RDFS(FA) is different from standard description logics. In RDFS(FA) interpretations not only sets of objects and pairs of objects are considered, but also sets of sets of objects and sets of sets of pairs of objects, etc..
2. Standard DL semantics modelling primitives are not related to each other as in RDFS(FA). OWL does not have a meta-modelling architecture where the language primitives are treated as meta-classes and meta-properties respectively.

The use of RDFS(FA) instead of RDF(S) implies several important advantages. First, OWL can be much more easily layered on top of RDFS(FA) than on top of RDF(S). If RDFS were replaced by RDFS(FA) the semantics of OWL could be greatly simplified. Second, the organization of the semantic web architecture would become clearer as a whole; all the languages of the semantic web would be based on a single, clear standard formalism. The price to pay when replacing RDF(S) by RDFS(FA) is a loss of expressive power. RDFS(FA) is less expressive than RDF-Schema.

6. A POSSIBLE REORGANIZATION OF THE SEMANTIC WEB TOWER ARCHITECTURE

6.1 Motivation

The layering suggested in this paper is largely based on the RDFS(FA) approach, but represents a further simplification of RDFS(FA), in the same way as RDFS(FA) can be seen as a restricted sublanguage of RDF-Schema. The solution discussed here is motivated by the following ideas:

1. In order to replace RDF-Schema by RDFS(FA) in the schema layer of the semantic web, the RDF syntactic and semantic specifications should also be modified. RDFS(FA) cannot be easily layered on top of RDF in the way both RDFS(FA) and RDF have been designed. In order to properly layer RDFS(FA) on top of the metadata layer, a sublanguage of RDF must be defined
2. It is unclear whether or not the metamodelling architecture provides additional expressive power to RDFS(FA). In the way RDFS(FA) has been specified, there is no plausible reason, apart from the (apparent) compatibility with RDF-Schema, to define a metamodelling architecture for the language. Metamodelling doesn't seem to enhance the inference capabilities of the language. As far as we know no "interesting" entailments have been suggested as a proof of the utility of the metamodelling structure of RDFS(FA). Moreover it is also unclear:
 - What kind of reasoning algorithms could be defined for RDFS(FA).
 - If standard DL reasoning algorithms are used, how metamodelling could be integrated in the algorithms to provide new interesting inference capabilities

3. The ontology and the schema layers share most of their aims and scope. The main difference between both layers is the semantic formalism they are based on. In the current semantic web RDF-Schema can be considered both as a simple ontology language and as a bridge between RDF and the ontology layer. However, in the layering suggested in this paper there is no plausible reason for the schema layer to exist.

6.2 Methodology

The methodology followed in this paper is composed of the following steps:

1. A sublanguage of RDFS(FA), called RDFS(DL) is defined and its correspondence to a simple description logic is established
2. It is argued that RDFS(FA) can be replaced by RDFS(DL) without a significant loss of expressivity
3. The schema layer is suppressed from the semantic web tower architecture and the language RDFS(DL) is promoted to the ontology layer as the simplest ontology language for the semantic web
4. The Organization of the OWL dialects is altered in two important ways:
 - OWL-Full is suppressed
 - OWL-Lite is layered on top of RDFS(DL)
5. A sublanguage of RDF, called RDF(DL) is defined and RDFS(DL) is layered on top of it

6.3 RDF-Schema(DL)

RDF-Schema(DL) is a sublanguage of RDFS(FA) where:

- The relationships between language primitives have been suppressed
- Its semantics covers the strata 0 and 1 and the language itself exists in stratum 2
- The language primitives are not stratified into different strata of the language. This implies that there are no different "versions" of the same modelling primitive in different strata.

RDF-Schema(DL) is the result of suppressing the metamodelling architecture of RDFS(FA). This design decision is motivated by the fact that metamodelling in RDFS(FA) doesn't seem to increase the expressivity and the inference capabilities of the language. Metamodelling in RDFS(FA) can be thought as a way to make the language as similar as possible to RDF-Schema. However, the fundamental assumptions of RDFS(FA), like the explicit division of vocabularies and the constraints applied to the free syntax of RDF(S) do change the language completely. In this new framework, metamodelling reveals itself as a rather "artificial" feature of the language. In this particular sense, RDFS(DL) is clearer than RDFS(FA).

6.3.1 RDFS(DL) Syntax and Semantics

A RDFS(DL) vocabulary V is a set of URIs (a vocabulary) composed of the following disjoint sets:

- V_C is the set of concept (class) names
- V_D is the set of datatype names
- V_{RA} is the set of (object) property names

- V_{RD} is the set of datatype property names
- V_I is the set of individual names

As in RDF, a datatype "d" is defined by two sets and one mapping: $L(d)$ (lexical space), $V(d)$ (value space), and $L2V(d)$ the mapping from the lexical space to the value space. In this context a datatype theory T is a partial mapping from URIs to datatypes.

A RDF-Schema(DL) interpretation with respect to T with vocabulary $V_C, V_D, V_{RA}, V_{RD}, V_I$ is a tuple $I = \{R, EC, ER, L, S, LV\}$ where:

- R is a non-empty set representing the (abstract) domain of I
- $LV \subseteq R$ is the distinguished subset representing the literal values of I
- $EC: V_C \rightarrow P(O)$, and $V_D \rightarrow P(LV)$ is a mapping from class names in the vocabulary to subsets of the abstract domain and from datatype names in the vocabulary to subsets of the datatype domain LV .
- ER maps object property names to a set of ordered pairs of elements in the abstract domain $V_{RA} \rightarrow P(O \times O)$ and datatype property names to ordered pairs, whose first element is a member of the abstract domain, and whose second component is a datatype value $V_{RD} \rightarrow P(O \times LV)$
- L maps typed literals to elements of the datatype domain $L: TL \rightarrow LV$. Let T be a datatype theory, d a datatype and d' a datatype name (a URIref), then:
 - If $T(d') = d$ then $EC(d') = V(d) \subseteq LV$
 - If $T(d') = d$ then $L('v', d') \in V(d)$, where 'v' is a string
 - If $T(d') = d$ and $v \in L(d)$ then $L('v', d') = L2V(d)(v)$, where 'v' is a string
 - If $T(d') = d$ and $v \in L(d)$ then $L('v', d') \in R - V(d)$
- S maps individual names to objects in the abstract domain of the interpretation. $S: V_I \rightarrow O$

The abstract syntax and semantics of RDFS(DL) is summarized in tables 1 and 2

6.4 Reasoning in RDFS(DL)

The core reasoning service in the semantic web languages is entailment. We say that a RDFS(DL) (or an OWL) ontology O_1 entails O_2 ($O_1 \models O_2$) if and only if all the interpretations that satisfy O_1 also satisfy O_2 . However, entailment is not a standard DL reasoning service. The standard DL tableaux-based and structural reasoning algorithms have been conceived to solve the subsumption and the satisfiability problems. DL inference algorithms do not directly deal with ontology entailment. Therefore, in order to use these algorithms for reasoning in the semantic web languages entailment, has to be reduced to DL knowledge base satisfiability [8].

In [8] has been shown that both ontology entailment in OWL-DL and OWL-Lite can be reduced to the knowledge base satisfiability problem in the SHOIN(D) and SHIF(D) logics respectively.

Unfortunately, a similar reduction cannot be accomplished for RDFS(DL) entailment. A RDFS(DL) ontology is composed of a set of concept inclusion axioms $A \sqsubseteq B$, a set of role inclusion axioms $R \sqsubseteq S$, a set of datatype role inclusion axioms $U \sqsubseteq V$ and a set of facts, where A,B are atomic concepts, R and S atomic

ABSTRACT SYN.	DL SYN.	SEMANTICS
CLASS AXIOMS		
<i>SubClassOf</i> ($A_1 A_2$)	$A_1 \sqsubseteq A_2$	$EC(A_1) \subseteq EC(A_2)$
PROP.AXIOMS		
ObjectProperty(R) dom(A_1) ... dom (A_m)	$R \in V_{RA}$ $\geq 1R \sqsubseteq A_i,$ $\forall i \in [1, m]$ $\top \sqsubseteq \forall R.A_i$	$ER(R) \subseteq R \times R$ $ER(R) \subseteq$ $EC(A_i) \times R$ $ER(R) \subseteq$ $R \times EC(A_i)$ $ER(R_1) \subseteq ER(R_2)$
range(A_1) ... range(A_n)		
<i>subPropertyOf</i> ($R_1 R_2$)	$R_1 \sqsubseteq R_2$	$ER(R_1) \subseteq ER(R_2)$
DatatypeProperty(U) dom(D_1) ... dom (D_m)	$R \in V_{RD}$ $\geq 1R \sqsubseteq D_i,$ $\forall i \in [1, m]$ $\top \sqsubseteq \forall U.D_i$	$ER(D) \subseteq R \times LV$ $ER(U) \subseteq$ $EC(D_i) \times R$ $ER(U) \subseteq$ $R \times EC(D_i)$ $ER(U_1) \subseteq ER(U_2)$
range(D_1) ... range(D_n)		
<i>subPropertyOf</i> ($U_1 U_2$)	$U_1 \sqsubseteq U_2$	
FACTS		
Individual(o) type(C_1)...type(C_n) value($R_1 o_1$)...value($R_n o_n$)	$o \in C_i$ $[o, o_i] \in ER(R)$	$S(o) \in$ $EC(C_i)$ $[S(o), S(o_i)] \in$ $ER(R_i)$
value($U_1 v_1$)...value($U_n v_n$)	$[o, v_i] \in EC(U_i)$	$[S(o), S(v_i)]$ $\in ER(U_i)$

Table 1: RDFS(DL) Axioms

ABSTRACT SYN.	DL SYN.	SEMANTICS
CLASSES		
A (URIref) owl:Thing	A \top	$A \in \Delta^I$ $EC(owl:Thing)$ $= R$
CONSTRUCTORS		
(None)		
RESTRICTIONS		
(None)		
DATATYPES		
d' (URIref)	d'	$EC(d') = V(d)$ $\subseteq LV$
ABSTRACT PROP.		
R (URIref)	R	$ER(R) \subseteq R \times R$
DATATYPE PROP.		
U (URIref)	U	$ER(U) \subseteq R \times LV$
OBJECTS		
o (URIref)	o	$o^I \in R$
DATA VALUES (d)		
d (Literal)	d	$S(d) \in LV$

Table 2: RDFS(DL) semantics

roles, U and V datatype properties and where cycles might occur in the terminology.

Let O_1 and O_2 be two RDFS(DL) ontologies and assume we want to check if $O_1 \models O_2$. It can be shown that $O_1 \models O_2$ if and only if $O_1 \models A_j$ for every axiom A_j in O_2 . Then [8]:

- If A_j is of the form $A \sqsubseteq B$, entailment checking is equivalent to checking the unsatisfiability of $O_1 \cup \{(A \sqcap \neg B)\}$. Therefore, in order to reduce entailment to satisfiability for this kind of axioms RDFS(DL) should include atomic negation
- If A_j is of the form $R \sqsubseteq S$, then entailment checking is equivalent to verify the unsatisfiability of:

$$O_1 \cup \{(B \sqcap \exists R(\forall S^-. \neg B)\}$$

where B is a new atomic concept and S^- is the inverse role of S. Thus, in order to reduce entailment to satisfiability for these axioms RDFS(DL) should include full existential restrictions, value restrictions, concept negation and role inversion

- If A_j is a fact, then RDFS(DL) would need to include negation and value restrictions
- If A_j is a datatype role inclusion axiom, then nominals would be required in order to reduce entailment to satisfiability. It is worth pointing out here that this specific aspect, which would also affect the reduction of OWL-Lite entailment, is not considered in [8] when dealing with OWL-Lite. In that case, OWL-Lite entailment would be reduced to SHOIF(D) KB satisfiability (not to SHIF(D)). Therefore, if datatype role inclusions are considered, the logics corresponding to OWL-Lite does not differ significantly from SHOIN(D). In that case, there would be, for example, no reason to avoid nominals ("one-of" constructor) in OWL-Lite and, most important, there would be no plausible reason for OWL-Lite to exist separately from OWL-DL .

Therefore, the following aspects should be considered for RDFS(DL) (and also for RDFS(FA)!):

- If entailment is considered as the core reasoning service then, in order to use standard DL reasoning algorithms for RDFS(DL), the language seems to have to be extended up to the *ALCHOI* logic, which is computationally intractable in the worst case.
- If entailment is taken as the main reasoning service, then new DL algorithms could be designed for solving directly the entailment problem. These algorithms might have polynomial complexity (RDFS(DL) does not contain restrictions and concept union, which are considered as the main sources of complexity)
- If satisfiability or subsumption of concepts with respect to a knowledge base are considered as the core inference services, then standard DL algorithms could be used to reason in RDFS(DL)
- The main source of complexity in RDFS(DL) is the role hierarchy. The possibility of replacing the role hierarchy by other constructors could be reconsidered. If, for example, RDFS(DL) is made equivalent to the ALC description logics and if the concept inclusion axioms are constrained to be acyclic, then entailment could be reduced to knowledge

base satisfiability, which is in PSPACE under these specific conditions. This would make possible to have a polynomial logic in the bottom of the ontology layer and to use standard tableaux-based reasoning algorithms.

Instead of using tableaux-based algorithms, an interesting approach for reasoning in RDFS(DL) would be to use logic programming (LP) inference engines. RDFS(DL) can be mapped to equality-free Datalog Horn Logic [5] (the logical language composed of Horn rules not including, neither functions with arity greater than zero, nor the equality predicate) . In [5] the equality-free Datalog LP (def-LP) is also defined. The difference between this logic and the equality-free Datalog Horn Logic (def-Horn) is subtle, but important. The entailments in def-LP are limited to facts, while in def-Horn also rules can be entailed. Therefore, def-LP is a subset of def-Horn. In return to this loss of expressivity, the inference happens to be polynomial in the worst case. However, RDFS(DL) must be mapped to def-Horn and not to def-LP, because the entailments defined in RDFS(DL) are not restricted to simple facts.

6.5 Building the new ontology layer

RDFS(DL) is defined as a subset of both OWL-Lite and OWL-DL, and is equivalent to a simple (though non-conventional) DL. Instead of replacing RDF-Schema by RDFS(DL) in the schema layer, it is suggested in this paper to promote RDFS(DL) to the ontology layer and then suppress the schema layer from the semantic web tower.

In this context, the layering of OWL becomes similar to the layering of OIL. In this new framework, all the OWL dialects correspond to a particular description logic. OWL-Lite is defined as a subset of OWL-DL and RDFS(DL) as a subset of OWL-Lite. The semantic web developer could choose one of these languages for creating an ontology depending on the expressive power required by his/her application.

In this layering framework, RDFS(DL) would be layered directly on top of the metadata layer and both the remaining layers (like the rule level) and other OWL extensions would be based on OWL-DL.

6.6 RDF(DL)

RDFS(DL) cannot be defined as an extension of RDF, in the way RDF has been specified. RDF allows the programmer to use a free syntax, so that:

- A resource can appear both in a subject/object or in a property position.
- A URI belonging to RDF reserved vocabulary can appear as the subject of a RDF triple
- Self references are allowed

We suggest a sublanguage of RDF, called RDF(DL), based on a standard model-theoretic semantics. In order to carry out this task vocabularies in RDF are divided into three different parts, namely classes, properties and individuals. The domain of RDF(DL) interpretations is the set of all individuals, like in DL.

A RDF vocabulary V consists of five disjoint sets of URI references, V_C , V_D , V_I , V_{RA} and V_{RD} , where:

- V_C is the set of concept (class) names
- V_D is the set of datatype names
- V_{RA} is the set of (object) property names
- V_{RD} is the set of datatype property names

- V_I is the set of individual names

The syntax of RDF(DL) is also based on triples, but some constraints are introduced so that the way triples can be written is no longer completely free. In RDF(DL) close attention must be paid to which names belong to which part of the RDF vocabulary.

- Class URI references can only appear in the object position of the triples where `rdf:type` appears in the property position
- URIs belonging either to V_{RA} or to V_{RD} can only appear in property positions
- Individuals (names in V_I) can appear both in subject and object positions, but never in property positions.
- Literal values can only appear at the object position of triples where an element of V_{RD} appears in the property position

In RDF(DL) four different kinds of assertions can be used:

- An assertion of the form $C(a)$, where C is a class name and "a" an object name, states that "a" is an instance of C and is mapped to the triple `[a rdf:type C]`
- An assertion of the form $R(a,b)$, where R is an object property name and a,b are object names states that the objects denoted by a,b are linked by the role denoted by R and is mapped to the triples `[a R b]` and `[R rdf:type rdf:ObjectProperty]`
- An assertion of the form $U(v)$, where U is a datatype name and "v" a literal state that the value v belongs to the datatype U and is mapped to `[v rdf:type D]`
- An assertion of the form $U(a,v)$, where U is an datatype property name, a is an object name and v is a literal state that the object denoted by a is linked by the role denoted by U to the data value denoted by v . The assertion would be mapped to `[a U v]` and `[U rdf:type rdf:DatatypeProperty]`

Only the triples resulting of this mapping are given a model-theoretic semantics in RDF(DL).

A RDF(DL) interpretation of a vocabulary V is a tuple $I = \{IR, LV, .^I\}$ where,

- IR is a non-empty set called the domain of the interpretation
- $LV \subseteq IR$ is a distinguished subset of IR
- $.^I$ is a function that:
 - It maps every individual name "o" to an element of IR
 $o^I \in IR$
 - It maps every object property name R to $R^I \subseteq IR \times IR$
 - It maps every datatype property name U to $U^I \subseteq IR \times LV$
 - It maps every class name C to $C^I \subseteq IR$

A rdf interpretation satisfies:

- The assertion $C(a)$ if $a^I \in C^I$
- The assertion $R(a,b)$ if $\langle a^I, b^I \rangle \in R^I$
- The assertion $D(v)$ if $v^I \in D^I$
- The assertion $U(a,v)$ if $\langle a^I, v^I \rangle \in U^I$

An RDF(DL) interpretation satisfies a RDF(DL) graph if and only if it satisfies all the assertions in the graph.

6.7 A new vision of the Semantic Web

What have we achieved? First, and most important the layering of the semantic web has become clearer, easier to understand and formalise. The layering of OWL dialects is now analogous to the organization of OIL dialects. In this approach the ontology layer is composed of three different languages:

1. RDFS(DL)
2. OWL-Lite
3. OWL-DL

These languages are organised according to their expressivity: OWL-DL is the most expressive language and is equivalent to the SHOIN(D) description logics. Satisfiability checking for this logic is known to be NEXP-TIME complete. No "practical" tableau-based reasoning algorithm for this logic has been developed yet. OWL-Lite provides enough expressive power for most applications and its abstract syntax can be mapped to the SHIF(D) description logics. Reasoning services, though intractable in the worst-case, can be implemented efficiently using highly-optimised algorithms that work fairly well in average for most applications and use cases. Finally, RDFS(DL) stays at the bottom of the ontology layer. Although its expressive power is rather limited it can still be useful for many applications. In this framework, the semantics of OWL-DL and OWL-Lite would be greatly simplified: only a direct model-theoretic semantics would be required. On the other hand, the schema layer is suggested to be suppressed from the semantic web layer cake. Two main reasons support this suggestion:

- The schema layer has always been difficult to understand and formalise.
- The semantic specification of RDF-Schema requires non-standard model-theoretic semantics and has non-standard metamodelling architecture, which make the language difficult to understand, implement and formalise. Its aims and scope have never been clearly distinguished from those of the ontology layer

RDF-Schema can be conceived both as a simple ontology language and as a "bridge" between the metadata and the ontology layers. However, its motivation has not been properly justified and supported. There is no clear difference between the motivations and goals of the schema and the ontology layers; RDF-Schema has always been considered in practice as a simple ontology language which shares many features with OWL. The only important difference between RDF-Schema and OWL happens to be the underlying semantic characterization. While OWL-Lite and OWL-DL are based on very expressive description logics, RDF-Schema has non-standard model-theoretic semantics.

In this approach, the metadata and the ontology layers share a standard description logics semantics, with RDF(DL) corresponding to an assertional (ABox) formalism, while OWL and RDFS(DL) would correspond to a terminological (TBox) language. In this new framework the semantic web documents directly correspond to sets of TBox axioms (OWL documents) and ABox facts (RDF(DL)) in a DL knowledge base. With this approach, the layers at the bottom of the semantic web could be nicely integrated in a common and coherent formalism.

7. CONCLUSION

In this paper an alternative layering for the semantic web is defined. This approach has several advantages with respect to the layering accepted by the Web Ontology Working Group:

- The semantic web tower has become clearer, easier to understand and formalise
- The remaining layers of the semantic web and the extensions of OWL can be more easily defined using this approach.
- The semantic web could benefit from 15 years of description logics research in both the metadata and the ontology layer
- In the new layering paradigm RDF(DL) would correspond to a ABox formalism in DL, while RDFS(DL) and OWL would be used to define the TBox.
- Some expressive power is lost in the lower layers of the semantic web when using this approach. However:
 - The languages on top of the ontology layer will be as expressive as they would be following the standard approach. New languages and extensions would hardly be layered on top of OWL-full, which means that the additional expressive power provided by RDF(S) model theory will probably not be used in the languages built on top of the ontology layer
 - The additional capabilities of RDF and RDF-Schema have not been widely used so far by modellers. For example, the only modelling capability of RDF not provided by RDF(DL) is the ability to use properties in subject position of triples. Nevertheless it is hard to find any examples of RDF graphs that use this specific feature of the language.
- The semantic specification of OWL would be greatly simplified. Thus, for example, no RDF-compatible model theoretic semantics would be required. All the languages in this solution would be given a direct model-theoretic semantics. This makes OWL tools easier to implement
- Users can choose to use RDF(DL) and RDFS(DL) on one hand or full RDF(S) on the other.

8. ACKNOWLEDGMENTS

The author would like to thank Bijan Parsia for insightful comments and discussions.

9. REFERENCES

- [1] F. Baader, I. Horrocks, and U. Sattler. Description logics as ontology languages for the semantic web. In D. Hutter and W. Stephan, editors, *Festschrift in honor of Jörg Siekmann*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 2003.
- [2] F. Baader and W. Nutt. Basic description logics. In F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors, *The Description Logic Handbook: Theory, Implementation, and Applications*, pages 43–95. Cambridge University Press, 2003.
- [3] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 2001.
- [4] D. Brickley and R. V. Guha. Resource description framework (rdf) schema specification 1.0: Rdf schema. *W3C Working Draft*, 2003.
- [5] B. Groszof, I. Horrocks, R. Volz, and S. Decker. Description logics programs, combining logic programs with description logics. In *Proc. of the 2003 International World Wide Web Conference (WWW2003)*, Budapest, 2003.
- [6] P. Hayes. Resource description framework (rdf) semantics. *W3C Working Draft*, 2003.
- [7] P. Hayes and C. Menzel. A semantics for the knowledge interchange format. In *Proc. of IJCAI 2001 Workshop on the IEEE Upper Ontology*, 2001.
- [8] I. Horrocks and P. F. Patel-Schneider. Reducing OWL entailment to description logic satisfiability. In *Proc. of the 2003 Description Logic Workshop (DL 2003)*, 2003.
- [9] I. Horrocks and P. F. Patel-Schneider. Three theses of representation in the semantic web. In *Proc. of the Twelfth International World Wide Web Conference (WWW 2003)*, 2003.
- [10] I. Horrocks, P. F. Patel-Schneider, and F. V. Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 2003.
- [11] G. Klyne and J. J. Carroll. Resource description framework (rdf) concepts and abstract syntax. *W3C Working Draft*, 2003.
- [12] D. Nardi and R. Brachman. An introduction to description logics. In F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors, *The Description Logic Handbook: Theory, Implementation, and Applications*, pages 43–95. Cambridge University Press, 2003.
- [13] J. Z. Pan and I. Horrocks. Metamodelling architecture of web ontology languages. In *Proc. of the 2001 Semantic Web Working Symposium (SWWS)*, 2001.
- [14] J. Z. Pan and I. Horrocks. RDFS(FA): A DL-ised sub-language of RDFS. In *Proc. of the 2003 Description Logic Workshop (DL 2003)*, 2003.
- [15] J. Z. Pan and I. Horrocks. RDFS(FA) and RDF MT: Two semantics for RDFS. In *Proc. of the 2003 International Semantic Web Conference (ISWC 2003)*, 2003.
- [16] P. F. Patel-Schneider, P. Hayes, and I. H. (editors). Owl web ontology language semantics and abstract syntax. *W3C Candidate Recommendation*, 2003. Work in progress.
- [17] P. F. Patel-Schneider, P. Hayes, and I. Horrocks. Layering the semantic web, problems and directions. In *Proc. of the 2002 International Semantic Web Conference (ISWC 2002)*, 2002.