RDFS(FA): Connecting RDF(S) and OWL DL

Jeff Z. Pan* and Ian Horrocks†

* Department of Computing Science, University of Aberdeen, UK

Email: jpan@csd.abdn.ac.uk

† School of Computer Science, University of Manchester, UK

Email: horrocks@cs.man.ac.uk

Abstract

Semantic interoperability among Semantic Web (SW) languages is an important feature in knowledge engineering in the Semantic Web era. Recent research, however, has shown some issues on the compatibility between the semantics of the standard SW annotation language RDF (as well as its ontological extension RDFS) and that of the standard SW ontology language OWL DL. To address these issues, existing approaches either limit the extension of RDF(S) to only a property-related subset of OWL together with a weaker semantics [53], [54] or weaken the semantic connection between individual interpretations and class interpretations of URIs (and hence lose some intuitive inference) [12]. This paper proposes a novel modification of RDF(S) as a firm semantic foundation for many of the latest Description Logics-based SW ontology languages, including OWL DL. Its metamodeling architecture is very similar to that of UML and it imposes no limitation on its extensibility to more expressive Description Logics (such as OWL DL and OWL-Eu). As a result, the introduction of RDFS(FA) solidifies RDF(S)'s proposed role as the base of the Semantic Web and facilitates key knowledge engineering tasks, such as ontology reuse, in knowledge-based systems.

Index Terms

semantic interoperability, knowledge representation, ontology, metamodeling architecture, reuse

This is a revised and extended version of a paper titled 'RDFS(FA) and RDF MT: Two Semantics for RDFS' published in the 2nd International Semantic Web Conference (ISWC2003). The extended work is partially supported by the FP6 Network of Excellence EU project Knowledge Web (IST-2004-507842).

I. INTRODUCTION

The vision of the Semantic Web (SW) is to augment the syntactic Web with semantic markup, so that resources are more easily interpreted by programs (or 'intelligent agents'). Encoding semantic markups will necessitate the Semantic Web adopting an annotation language. To this end, the W3C (World Wide Web Consortium) community has developed a recommendation called Resource Description Framework (RDF) [30]. The development of RDF is an attempt to support effective creation, exchange and use of annotations on the Web.

Annotations alone, however, do not establish the semantics of what is being marked-up. In response to this need for more explicit meaning, ontologies [15], [55] have been proposed to provide shared and precisely defined terms and constraints to describe the meaning of resources through annotations — such annotations are called *machine-understandable annotations*. The advent of RDF Schema (RDFS) [6] represented an early attempt at a SW ontology language based on RDF. RDF and RDFS, or simply RDF(S), are intended to provide the foundation for the Semantic Web [32, Sec. 7.1]. As the constructors that RDFS provides for constructing ontologies are very primitive, more expressive SW ontology languages have subsequently been developed, such as OIL [21], DAML+OIL [24] and the W3C standard Semantic Web ontology language OWL [2],[1] which are all based on Description Logics (DLs) [1].

Knowledge-based systems in the Semantic Web era can/should make use of the power of the Semantic Web languages and technologies, in particular those related to ontologies, to support key tasks such as information retrieval and extraction ([16], [3], [33], [57], [29], [52], [9]), and information integration ([4], [34], [31]). In the OWL DL ontology language, an ontology corresponds to a DL knowledge base; i.e., an ontology contains not only knowledge about important concepts and relationships in a given domain, but also data (instances of these concepts and relationships) in the domain. Exploiting this logical foundation allows for the explication of information (or knowledge) that is only implicitly represented in the ontology; in practice this can be achieved with the help of ontology inference engines, such as FaCT [22], RACER [18], Pellet [49], FaCT++ DL [13], KAON2 [28] and FaCT-DG [39]. Interestingly, for a given set of data (knowledge about individuals and their relationships),

[1]There are three sub-languages of OWL, i.e., OWL Lite, OWL DL and OWL Full; cf. Section II-B for more details.

different information can be inferred given different contexts (background knowledge about classes and properties). Furthermore, with the help of semantically compatible SW languages, it is possible and desirable to infer useful information based on important knowledge (possibly described in different SW languages) that is often distributed across the Web and/or intranet(s). Indeed, semantic interoperability among SW languages, as addressed in this paper, is a crucial feature of knowledge engineering in the Semantic Web era. Without semantic interoperability, it is difficult or even impossible for Web resources to be shared and interpreted by programs in a meaningful way.

In order to allow for semantic interoperability, SW languages should at least be "compatible" with each other. RDF(S) has a key role in supporting such compatibility by providing a common basis on which more expressive SW languages can be built. Recent research, however, has revealed some problematical issues when trying to extend the RDF(S) semantics [19] to specify the meaning of OWL constructors; these issues include 'too few entailments', 'contradiction classes' and 'size of the universe' ([44], [45], [27]), all of which stem from the unusual characteristics of RDF(S) (cf. Section II-A). Furthermore, Motik [36] has shown that even adding only $\mathcal{ALC}$ (a DL much simpler than OWL DL) constructors to the metamodeling architecture of RDFS would already lead to undecidability. In short, the intended foundation of the Semantic Web and SW ontology languages does not seem to provide for the desired extensibility and semantic compatibility. This could seriously discourage potential users from adopting Semantic Web standards [5]. To address these issues, existing approaches either limit the extension of RDF(S) to only a property-related subset of OWL with a weaker semantics ([53], [54]), or weaken the semantic connection between the individual interpretation and class interpretation of a given URI [12], hence failing to propagate important inferences from metaclasses to classes (see Section VII for more details).

This paper proposes a novel modification of RDF(S) which provides a solid semantic foundation for many of the latest Description Logic-based SW ontology languages, and imposes no limitation on its extension to more expressive Description Logics (such as OWL DL and OWL-Eu [42]). After reviewing the design of RDF(S), and the needs of various applications and (potential) users, the following requirements for a sub-language of RDF(S) have been identified:

1) Ontologies in this sub-language should be RDF graphs.
2) It should enable the use of class URIrefs as property values, which is a feature of RDFS that is required in many applications [38].
3) It should provide a metamodeling architecture compatible with the layered metamodeling architecture of UML (Unified Modelling Language) [10], as UML is probably the most well known and widely accepted metamodeling architecture.
4) Its semantics should be compatible with the semantics of OWL DL [46].

This paper makes the following contributions:

1) After formally introducing the semantics of RDF(S) and OWL, it reviews in detail the syntactic and semantic mismatches between RDF(S) and OWL DL (Section III). This indicates people need two different inference engines to reason with RDF(S) and OWL DL ontologies. These mismatches motivate why we need a strong connection between RDF(S) and OWL DL.
2) It presents a sub-language of RDF(S), called RDFS(FA),[2] which satisfies the above requirements (Section IV to VI). In terms of the RDFS(FA) language, it substantially extends the conference version of the paper [41] with the following aspects: (i) It also covers datatypes and annotation properties, which are both useful in Semantic Web applications [42], [51], [38]. (ii) For the first time, it introduces the notion of RDFS(FA) ontologies, which makes it much easier to compare the RDFS(FA) and OWL DL ontology languages, and makes the bidirectional mapping (to be mentioned below) between them possible. (iii) It provides some rules of thumb on to help authors/users of RDFS(FA) ontologies quickly get the strata/layer number right. Although such numbers can/should be encapsulated by tools, this turns out to be very helpful because people can now easily play with their RDFS(FA) ontologies.
3) Most importantly, it identifies a bidirectional one-to-one mapping between RDFS(FA) axioms in strata 0-1 and OWL DL axioms, which enables RDFS(FA)-agents and OWL-DL-agents to communicate with each other (Section V). This also provides a significant insight on how to reason with RDFS(FA) as well as its extension OWL FA [43]. Such reasoning techniques make it possible to use one single inference engine to reason

[2] 'FA' for 'Fixed layered matamodeling Architecture'.

with RDFS(FA), OWL DL and OWL FA ontologies. This could significantly improve semantic interoperability of knowledge systems of the Semantic Web era.

4) Furthermore, it shows that introducing RDFS(FA) as a sub-language of RDF(S) clarifies the vision of the Semantic Web and solidifies RDF(S)'s proposed role as the foundation of the Semantic Web (Section VI).
5) Finally, it provides a discussion of related work,[3] and compare them with the RDFS(FA) approach (Section VII).

In short, we believe that the introduction of RDFS(FA) solidifies RDF(S)'s proposed role as the foundation of the Semantic Web, and facilitates key knowledge engineering tasks, such as ontology reuse, in knowledge-based systems.

II. Background

In this section provides a brief overview of the Semantic Web standards RDF(S) and OWL.

A. RDF and RDFS

Resource Description Framework (RDF) [30] is built upon earlier developments such as the Dublin Core [11] and the Platform for Internet Content Selectivity (PICS) [50] content rating initiative. An RDF statement (or RDF triple) is of the form: [subject property object .]. RDF-annotated resources (i.e., subjects) are usually named by Uniform Resource Identifier references (URIrefs). RDF annotates Web resources in terms of named properties. Values of named properties (i.e., objects) can be URIrefs of Web resources or literals, viz. representations of data values (such as integers and strings). A set of RDF statements is called an *RDF graph*.

RDF Schema (RDFS) can be seen as a first try to support expressing simple ontologies with RDF syntax. In RDFS, predefined Web resources $\mathrm{rdfs{:}Class}$, $\mathrm{rdfs{:}Resource}$ and $\mathrm{rdf{:}Property}$ can be used to declare classes, resources and properties, respectively. RDFS predefines the following meta-properties that can be used to represent background assumptions in ontologies: $\mathrm{rdf{:}type}$, $\mathrm{rdfs{:}subClassOf}$, $\mathrm{rdfs{:}subPropertyOf}$, $\mathrm{rdfs{:}domain}$ and $\mathrm{rdfs{:}range}$. At a glance, RDFS is a simple ontology langauge that supports only class and property hierarchies, as

[3]Including interesting recent work such as [53], [54] and [12], which was not available when RDFS(FA) was first proposed [41].

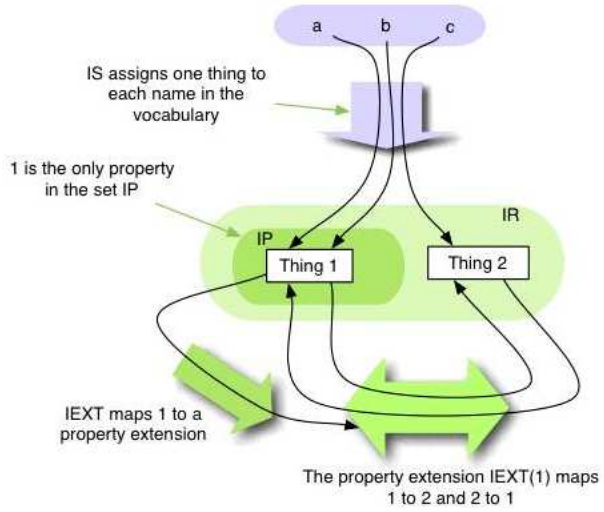

Fig. 1. A simple interpretation of **V** = {a,b,c} (from [19])

well as domain and range constraints for properties. According to the RDF Model Theory (RDF MT) [19], however, it is more complicated than that.

RDF MT provides semantics not only for RDFS ontologies, but also for RDF triples. RDF model theory is built on *simple interpretations*.[4] Given a set of URI references **V**, a *simple interpretation* I of **V** is defined by (i) an non-empty set **IR** of resources, called the *domain* (or *universe*) of I, (ii) a set **IP**, called the *set of properties* in I, (iii) a mapping $IEXT$, called the *extension function*, from **IP** to the powerset of **IR** $\times$ **IR**, and (iv) a mapping IS from URIrefs in **V** to **IR** $\cup$ **IP**. Given a triple $[\mathrm{s\ p\ o\ .}]$, I($[\mathrm{s\ p\ o\ .}]$) $= true$ if s,p,o $\in$ **V**, $IS(\mathrm{p}) \in$ **IP**, and $\langle IS(\mathrm{s}),\ IS(\mathrm{o})\rangle \in IEXT(IS(\mathrm{p}))$; otherwise, I($[\mathrm{s\ p\ o\ .}]$) $= false$. Given a set of triples S, I(S) $= false$ if I($[\mathrm{s\ p\ o\ .}]$) $= false$ for some triple $[\mathrm{s\ p\ o\ .}]$ in S, otherwise I(S) $= true$. I *satisfies* S, written as I $\models$ S if I(S) $= true$; in this case, we say I is a simple interpretation of S. Figure 1 presents a simple interpretation I of **V** = {a,b,c}, where the URIref b is simply interpreted as a property because $IS(\mathrm{b}) = 1 \in$ **IP**, and $IEXT(IS(\mathrm{b}))$, the extension of $IS(\mathrm{b})$, is a set of pairs of resources that are in **IR**, i.e., $\{\langle 1,2\rangle,\langle 2,1\rangle\}$. Since $\langle IS(\mathrm{a}), IS(\mathrm{c})\rangle \in IEXT(IS(\mathrm{b}))$, I($[\mathrm{a\ b\ c\ .}]$) $= true$; hence, we can conclude that I satisfies $[\mathrm{a\ b\ c\ .}]$.

Based on simple interpretations, RDF MT provides semantics for RDF triples and RDFS

[4]To simplify presentation, in this paper we do not cover blank nodes, which are identified by local identifiers instead of URIrefs.

statements by RDF-interpretations and RDFS-interpretations, respectively. These interpretations are simple interpretations that satisfy extra semantic conditions and axiomatic statements.Intuitively, RDF-interpretations require that $\mathbf{IP}$ to be a subset of $\mathbf{IR}$; i.e., all properties are resources. Similarly, RDFS-interpretations require that all classes are resources. Furthermore, RDFS-interpretations introduce the class extension function $ICEXT$, which works as follows: A class URIref is firstly mapped (by IS) to a resource in $\mathbf{IR}$ and then is further mapped (by $ICEXT$) to a set of resources which are instances of this class.

Definition 1 (**RDF-Interpretation**) Given a set of URI references $\mathbf{V}$ and the set $\mathbf{rdfV}$, called the *RDF vocabulary*, of URI references in the rdf: namespace, an RDF-interpretation of $\mathbf{V}$ is a simple interpretation I of $\mathbf{V} \cup \mathbf{rdfV}$ that satisfies:

1) for p$\in \mathbf{V} \cup \mathbf{rdfV}$, $IS(\mathrm{p}) \in \mathbf{IP}$ iff $\langle IS(\mathrm{p}), IS(\mathrm{rdf{:}Property})\rangle \in IEXT(IS(\mathrm{rdf{:}type}))$,
2) all the RDF axiomatic statements.[5] ◇

Condition 1 of Definition 1 implies that each member of $\mathbf{IP}$ is a resource in $\mathbf{IR}$, due to the definition of $IEXT$; in other words, RDF-interpretations require $\mathbf{IP}$ to be a subset of $\mathbf{IR}$. RDF axiomatic statements mentioned in Condition 2 are RDF statements about RDF built-in vocabularies in $\mathbf{rdfV}$; e.g., [$\mathrm{rdf{:}type\ rdf{:}type\ rdf{:}Property}$.] is an RDF axiomatic statement. According to Definition 1, any RDF-interpretation I should satisfy [$\mathrm{rdf{:}type\ rdf{:}type\ rdf{:}Property}$.], viz. $IS(\mathrm{rdf{:}type})$ should be in $\mathbf{IP}$.

Finally, the semantics of RDFS statements written in RDF triples is given in terms of RDFS-Interpretations. In particular, the following Condition 1) indicates that a 'class' is not a strictly necessary but convenient semantic construct [19] because the class extension function $ICEXT$ is simply a 'syntactic sugar' and is defined in terms of $IEXT$.

Definition 2 (**RDFS-Interpretation**) Given $\mathbf{rdfV}$, a set of URI references $\mathbf{V}$ and the set $\mathbf{rdfsV}$, called the *RDFS vocabulary*, of URI references in the rdfs: namespace, an RDFS-interpretation I of $\mathbf{V}$ is an RDF-interpretation of $\mathbf{V} \cup \mathbf{rdfV} \cup \mathbf{rdfsV}$ which introduces

- a set $\mathbf{IC}$, called the set of classes in I, and
- a mapping $ICEXT$ (called the *class extension function*) from $\mathbf{IC}$ to the set of subsets of $\mathbf{IR}$,

and satisfies the following conditions (let x,y,u,v be URIrefs in $\mathbf{V} \cup \mathbf{rdfV} \cup \mathbf{rdfsV}$)[6]

[5]Readers are referred to [19] for the list of the RDF axiomatic statements.
[6]We only focus on the core RDFS primitives here.

1) $IS(\mathrm{x}) \in ICEXT(IS(\mathrm{y}))$ iff $\langle IS(\mathrm{x}), IS(\mathrm{y})\rangle \in IEXT(IS(\mathrm{rdf{:}type}))$,
2) **IC** = $ICEXT(IS(\mathrm{rdfs{:}Class}))$ and **IR** = $ICEXT(IS(\mathrm{rdfs{:}Resource}))$,
3) if $\langle IS(\mathrm{x}), IS(\mathrm{y})\rangle \in IEXT(IS(\mathrm{rdfs{:}domain}))$ and $\langle IS(\mathrm{u}), IS(\mathrm{v})\rangle \in IEXT(IS(\mathrm{x}))$, then $IS(\mathrm{u}) \in ICEXT(IS(\mathrm{y}))$,
4) if $\langle IS(\mathrm{x}), IS(\mathrm{y})\rangle \in IEXT(IS(\mathrm{rdfs{:}range}))$ and $\langle IS(\mathrm{u}), IS(\mathrm{v})\rangle \in IEXT(IS(\mathrm{x}))$, then $IS(\mathrm{v}) \in ICEXT(IS(\mathrm{y}))$,
5) $IEXT(IS(\mathrm{rdfs{:}subPropertyOf}))$ is transitive and reflexive on **IP**,
6) if $\langle IS(\mathrm{x}), IS(\mathrm{y})\rangle \in IEXT(IS(\mathrm{rdfs{:}subPropertyOf}))$, then $IS(\mathrm{x})$,$IS(\mathrm{y}) \in$ **IP** and $IEXT(IS(\mathrm{x})) \subseteq IEXT(IS(\mathrm{y}))$,
7) $IEXT(IS(\mathrm{rdfs{:}subClassOf}))$ is transitive and reflexive on **IC**,
8) if $\langle IS(\mathrm{x}), IS(\mathrm{y})\rangle \in IEXT(IS(\mathrm{rdfs{:}subClassOf}))$, then $IS(\mathrm{x})$,$IS(\mathrm{y}) \in$ **IC** and $ICEXT(IS(\mathrm{x})) \subseteq ICEXT(IS(\mathrm{y}))$,
9) if $IS(\mathrm{x}) \in$ **IC**, then $\langle IS(\mathrm{x}), IS(\mathrm{rdfs{:}Resource})\rangle \in IEXT(IS(\mathrm{rdfs{:}subClassOf}))$,

and satisfies all the RDFS axiomatic statements.[7] ◇

Handling classes in this way can be counter-intuitive (cf. Proposition 3).

Proposition 3 The RDFS statements $[\mathrm{rdfs{:}Resource\ rdf{:}type\ rdfs{:}Class\ .}]$ and $[\mathrm{rdfs{:}Class\ rdfs{:}subClassOf\ rdfs{:}Resource\ .}]$ are true in all RDFS-interpretations.

Proof: For $[\mathrm{rdfs{:}Resource\ rdf{:}type\ rdfs{:}Class\ .}]$:

1) According to the definition of IS and Definition 1, for any resource x, we have $IS(\mathrm{x}) \in$ **IR**. Due to **IR** = $ICEXT(IS(\mathrm{rdfs{:}Resource}))$ and Condition 1 in Definition 2, $\langle IS(\mathrm{x}), IS(\mathrm{rdfs{:}Resource})\rangle \in IEXT(IS(\mathrm{rdf{:}type}))$. Since $\mathrm{rdf{:}Property}$ is a built-in resource, we have $\langle IS(\mathrm{rdf{:}Property}), IS(\mathrm{rdfs{:}Re\text{-}\ source})\ \rangle \in IEXT(IS(\mathrm{rdf{:}type}))$.
2) Due to [$\mathrm{rdf{:}type\ rdfs{:}range\ rdfs{:}Class.}$] (an RDFS axiomatic statement), $\langle IS(\mathrm{rdf{:}Property}), IS(\mathrm{rdfs{:}Resource})\rangle \in IEXT(IS(\mathrm{rdf{:}type}))$ and Condition 4 in Definition 2, we have $IS(\mathrm{rdfs{:}Resource}) \in ICEXT(IS(\mathrm{rdfs{:}Class}))$. Therefore, for any RDFS-interpretation I, we have I $\models$ [$\mathrm{rdfs{:}Resource\ rdf{:}type\ rdfs{:}\ Class\ .}$].

For $[\mathrm{rdfs{:}Class\ rdfs{:}subClassOf\ rdfs{:}Resource\ .}]$: According to the definition of **IC**, every class is its member, including $IS(\mathrm{rdfs{:}Class})$, viz.$IS(\mathrm{rdfs{:}Class}) \in$ **IC**. Due to Condition 9 of

[7]Again, readers are referred to [19] for a list of the RDFS axiomatic statements, which includes, e.g., [$\mathrm{rdf{:}type\ rdfs{:}range\ rdfs{:}Class\ .}$].

Abstract Syntax	DL Syntax	Semantics
Class(A)	A	$\mathsf{A}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
Class(owl:Thing)	$\top$	$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$
Class(owl:Nothing)	$\bot$	$\bot^{\mathcal{I}} = \emptyset$
intersectionOf($C_1, C_2, \ldots$)	$C_1 \sqcap C_2$	$(C_1 \sqcap C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$
unionOf($C_1, C_2, \ldots$)	$C_1 \sqcup C_2$	$(C_1 \sqcup C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$
complementOf(C)	$\neg C$	$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
oneOf($\mathsf{o}_1, \mathsf{o}_2, \ldots$)	$\{\mathsf{o}_1\} \sqcup \{\mathsf{o}_2\}$	$(\{\mathsf{o}_1\} \sqcup \{\mathsf{o}_2\})^{\mathcal{I}} = \{\mathsf{o}_1{}^{\mathcal{I}}, \mathsf{o}_2{}^{\mathcal{I}}\}$
restriction(R someValuesFrom(C))	$\exists R.C$	$(\exists R.C)^{\mathcal{I}} = \{x \mid \exists y.\langle x,y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
restriction(R allValuesFrom(C))	$\forall R.C$	$(\forall R.C)^{\mathcal{I}} = \{x \mid \forall y.\langle x,y\rangle \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$
restriction(R hasValue(o))	$\exists R.\{\mathsf{o}\}$	$(\exists R.\{\mathsf{o}\})^{\mathcal{I}} = \{x \mid \langle x, \mathsf{o}^{\mathcal{I}}\rangle \in R^{\mathcal{I}}\}$
restriction(R minCardinality(m))	$\geqslant mR$	$(\geqslant mR)^{\mathcal{I}} = \{x \mid \sharp\{y.\langle x,y\rangle \in R^{\mathcal{I}}\} \geq m\}$
restriction(R maxCardinality(m))	$\leqslant mR$	$(\leqslant mR)^{\mathcal{I}} = \{x \mid \sharp\{y.\langle x,y\rangle \in R^{\mathcal{I}}\} \leq m\}$
restriction(T someValuesFrom(u))	$\exists T.u$	$(\exists T.u)^{\mathcal{I}} = \{x \mid \exists t.\langle x,t\rangle \in T^{\mathcal{I}} \wedge t \in u^{\mathbf{D}}\}$
restriction(T allValuesFrom(u))	$\forall T.u$	$(\forall T.u)^{\mathcal{I}} = \{x \mid \exists t.\langle x,t\rangle \in T^{\mathcal{I}} \rightarrow t \in u^{\mathbf{D}}\}$
restriction(T hasValue(w))	$\exists T.\{w\}$	$(\exists T.\{w\})^{\mathcal{I}} = \{x \mid \langle x, w^{\mathbf{D}}\rangle \in T^{\mathcal{I}}\}$
restriction(T minCardinality(m))	$\geqslant mT$	$(\geqslant mT)^{\mathcal{I}} = \{x \mid \sharp\{t \mid \langle x,t\rangle \in T^{\mathcal{I}}\} \geq m\}$
restriction(T maxCardinality(m))	$\leqslant mT$	$(\leqslant mT)^{\mathcal{I}} = \{x \mid \sharp\{t \mid \langle x,t\rangle \in T^{\mathcal{I}}\} \leq m\}$
ObjectProperty(S)	S	$S^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
ObjectProperty(S' inverseOf(S))	S^-	$(S^-)^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
DatatypeProperty(T)	T	$T^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta_{\mathbf{D}}$

TABLE I

OWL CLASS AND PROPERTY DESCRIPTIONS

Definition 2, $\langle IS(\text{rdfs:Class}), IS(\text{rdfs:Resource})\rangle \in IEXT\ (IS(\text{rdfs:subClassOf}))$; hence, for any RDFS-interpretation I, we have I $\models$ [rdfs:Class rdfs:subClassOf rdfs:Resource.]□

The two RDFS statements in Proposition 3 suggest a strange situation, at least for some people, of rdfs:Class and rdfs:Resource as discussed in [40]: on the one hand, rdfs:Resource is an instance of rdfs:Class; on the other hand, rdfs:Class is a sub-class of rdfs:Resource. Hence is rdfs:Resource an instance of its sub-class? Some users find this counter-intuitive and thus hard to understand — this tricky relationships in RDF(S) ontologies indicate that RDF(S) is more complicated than it appears. Therefore, it is desirable to have a sub-language of RDF(S) that provides a more intuitive semantics, at least for its metamodeling architecture.

B. OWL

OWL is a standard (W3C recommendation) for expressing ontologies in the Semantic Web. The OWL language facilitates greater machine understandability of Web resources than that supported by RDFS by providing additional constructors for building class and property descriptions (vocabulary) and new axioms (constraints), along with a formal semantics. The OWL recommendation actually consists of three languages of increasing expressive power:

OWL Lite, OWL DL and OWL Full. *OWL Lite* and *OWL DL* are, like DAML+OIL, basically very expressive Description Logics (DLs); they are almost[8] equivalent to the $\mathcal{SHIF}(\mathbf{D}^+)$ and $\mathcal{SHOIN}(\mathbf{D}^+)$ DLs.[9] *OWL Full* provides the same set of constructors as OWL DL, but allows them to be used in an unconstrained way (in the style of RDF). It is easy to show that OWL Full is undecidable, because it does not impose restrictions on the use of transitive properties [23]. Therefore, OWL DL is the most expressive decidable sub-language of OWL.

Let $\mathbf{C}$, $\mathbf{R_I}$, $\mathbf{R_D}$ and $\mathbf{I}$ be the sets of URIrefs that can be used to denote classes, abstract properties, datatype properties and individuals respectively. An OWL DL *interpretation* is a tuple $\mathcal{I} = (\Delta^{\mathcal{I}}, \Delta_{\mathbf{D}}, \cdot^{\mathcal{I}}, \cdot^{\mathbf{D}})$ where the individual domain $\Delta^{\mathcal{I}}$ is a nonempty set of individuals, the datatype domain $\Delta_{\mathbf{D}}$ is a nonempty set of data values, $\cdot^{\mathcal{I}}$ is an individual interpretation function that maps

- each individual name $\mathtt{a} \in \mathbf{I}$ to an element $\mathtt{a}^{\mathcal{I}} \in \Delta^{\mathcal{I}}$,
- each class name $\mathsf{CN} \in \mathbf{C}$ to a subset $\mathsf{CN}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$,
- each abstract property name $RN \in \mathbf{R_I}$ to a binary relation $RN^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ and
- each datatype property name $TN \in \mathbf{R_D}$ to a binary relation $TN^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta_{\mathbf{D}}$,

and $\cdot^{\mathbf{D}}$ is a datatype interpretation function, which can be extended to provide semantics for OWL DL class and property descriptions shown in Table I, where $\mathsf{A} \in \mathbf{C}$ is a class URIref, $C, C_1, \ldots, C_n$ are class descriptions, $S \in \mathbf{R_I}$ is an *individual-valued* property URIref, R is an *individual-valued* property description and $\mathtt{o}, \mathtt{o}_1, \mathtt{o}_2 \in \mathbf{I}$ are individual URIrefs, u is a data range, $T \in \mathbf{R_D}$ is a *data-valued* property and $\sharp$ denotes cardinality.

An OWL DL ontology can be seen as a DL knowledge base [27], which consists of a set of *axioms*, including class axioms, property axioms and individual axioms.[10] Table II presents the abstract syntax, DL syntax and semantics of OWL axioms.

We conclude this section by a brief summary about the main differences, in terms of expressive power, between RDF(S) and OWL DL. RDF(S) is less expressive than OWL DL in that (i) it does not provide any constructors to construct class or property descriptions, and (ii) RDF(S) does not provide as many axioms about classes, properties and individuals

[8]They also provide annotation properties, which Description Logics don't.

[9]$\mathcal{SHOIN}(\mathbf{D}^+)$ provides two more class constructs than $\mathcal{SHIF}(\mathbf{D}^+)$, i.e., the nominals ($\mathcal{O}$) and number restriction ($\mathcal{N}$).

[10]Individual axioms are called *facts* in [48].

Abstract Syntax	DL Syntax	Semantics
Class(A partial $C_1 \ \ldots \ C_n$)	$\mathsf{A} \sqsubseteq C_1 \sqcap \ldots \sqcap C_n$	$\mathsf{A}^{\mathcal{I}} \subseteq C_1^{\mathcal{I}} \cap \ldots \cap C_n^{\mathcal{I}}$
Class(A complete $C_1 \ \ldots \ C_n$)	$\mathsf{A} \equiv C_1 \sqcap \ldots \sqcap C_n$	$\mathsf{A}^{\mathcal{I}} = C_1^{\mathcal{I}} \cap \ldots \cap C_n^{\mathcal{I}}$
EnumeratedClass(A $\mathsf{o}_1 \ \ldots \ \mathsf{o}_n$)	$\mathsf{A} \equiv \{\mathsf{o}_1\} \sqcup \ldots \sqcup \{\mathsf{o}_n\}$	$\mathsf{A}^{\mathcal{I}} = \{\mathsf{o}_1{}^{\mathcal{I}}, \ldots, \mathsf{o}_n^{\mathcal{I}}\}$
SubClassOf(C_1, C_2)	$C_1 \sqsubseteq C_2$	$C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$
EquivalentClasses($C_1 \ \ldots \ C_n$)	$C_1 \equiv \ldots \equiv C_n$	$C_1^{\mathcal{I}} = \ldots = C_n^{\mathcal{I}}$
DisjointClasses($C_1 \ \ldots \ C_n$)	$C_i \sqsubseteq \neg C_j$, $(1 \leq i < j \leq n)$	$C_1^{\mathcal{I}} \cap C_n^{\mathcal{I}} = \emptyset$, $(1 \leq i < j \leq n)$
SubPropertyOf(R_1, R_2)	$R_1 \sqsubseteq R_2$	$R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}}$
EquivalentProperties($R_1 \ \ldots \ R_n$)	$R_1 \equiv \ldots \equiv R_n$	$R_1^{\mathcal{I}} = \ldots = R_n^{\mathcal{I}}$
ObjectProperty(R super(R_1) ... super(R_n)	$R \sqsubseteq R_i$	$R^{\mathcal{I}} \subseteq R_i^{\mathcal{I}}$
domain(C_1) ... domain(C_k)	$\geqslant 1R \sqsubseteq C_i$	$R^{\mathcal{I}} \subseteq C_i^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
range(C_1) ... range(C_h)	$\top \sqsubseteq \forall R.C_i$	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times C_i^{\mathcal{I}}$
[Symmetric]	$R \equiv R^-$	$R^{\mathcal{I}} = (R^-)^{\mathcal{I}}$
[Functional]	$\mathsf{Func}(R)$	$\{\langle x, y\rangle \mid \sharp\{y.\langle x, y\rangle \in R^{\mathcal{I}}\} \leq 1\}$
[InverseFunctional]	$\mathsf{Func}(R^-)$	$\{\langle x, y\rangle \mid \sharp\{y.\langle x, y\rangle \in (R^-)^{\mathcal{I}}\} \leq 1\}$
[Transitive])	$\mathsf{Trans}(R)$	$R^{\mathcal{I}} = (R^{\mathcal{I}})^+$
DatatypeProperty(T super(T_1)...super(T_n)	$T \sqsubseteq T_i$	$T^{\mathcal{I}} \subseteq T_i^{\mathcal{I}}$
domain(C_1)...domain(C_k)	$\geqslant 1T \sqsubseteq C_i$	$T^{\mathcal{I}} \subseteq C_i^{\mathcal{I}} \times \Delta_{\mathbf{D}}$
range(d_1)...range(d_h)	$\top \sqsubseteq \forall T.d_i$	$T^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times d_i^{\mathbf{D}}$
[Functional])	$\mathsf{Func}(T)$	$\forall x \in \Delta^{\mathcal{I}}.\sharp\{t \mid \langle x, t\rangle \in T^{\mathcal{I}}\} \leq 1$
AnnotationProperty(R)		
Individual(o type(C_1) ...type(C_n)	$\mathsf{o} : C_i, 1 \leq i \leq n$	$\mathsf{o}^{\mathcal{I}} \in C_i^{\mathcal{I}}, 1 \leq i \leq n$
value(R_1, o_1) ...value(R_n, o_n)	$\langle \mathsf{o}, \mathsf{o}_i\rangle : R_i, 1 \leq i \leq n$	$\langle \mathsf{o}^{\mathcal{I}}, \mathsf{o}_i^{\mathcal{I}}\rangle \in R_i^{\mathcal{I}}, 1 \leq i \leq n$
SameIndividual($\mathsf{o}_1 \ \ldots \ \mathsf{o}_n$)	$\mathsf{o}_1 = \ldots = \mathsf{o}_n$	$\mathsf{o}_1^{\mathcal{I}} = \ldots = \mathsf{o}_n^{\mathcal{I}}$
DifferentIndividuals($\mathsf{o}_1 \ \ldots \ \mathsf{o}_n$)	$\mathsf{o}_i \neq \mathsf{o}_j, 1 \leq i < j \leq n$	$\mathsf{o}_i^{\mathcal{I}} \neq \mathsf{o}_j^{\mathcal{I}}, 1 \leq i < j \leq n$

TABLE II
OWL DL AXIOMS

as OWL DL provides (cf. Table II). On the other hand, RDF(S) supports axioms about meta-classes and meta-properties, which OWL DL does not support. OWL Full provides all the above constructors and axioms, including the metamodeling of RDF(S). However, OWL Full is not decidable, thanks to its metamodeling [36].[11]

III. MISMATCH BETWEEN RDF(S) AND OWL DL

This section discusses *both* the syntactic and semantic mismatches between RDF(S) and OWL DL.

From the syntax aspect, OWL DL heavily restricts the syntax of RDF(S), viz. some RDF(S) annotations are not recognisable by OWL DL-compatible agents. The RDF/XML syntax form of an OWL DL ontology is *valid*, iff it can be translated (according to the mapping rules provided in [47]) from the abstract syntax form of the ontology. Actually, it is far from an easy task to check if an RDF graph is an OWL DL ontology [27].

[11]There are other reasons why OWL Full is not decidable; e.g., non-simple properties are *not* disallowed in number restrictions. The point here is that even if non-simple properties were disallowed in number restrictions, OWL Full would still be undecidable.

From the semantics aspect, OWL DL has an RDF MT-style semantics, in which (including built-in) classes and properties are treated as objects (or resources) in the domain. In order to make it equivalent to the direct semantics of OWL DL presented in the previous section, the domain of discourse is divided into several disjoint parts. In particular, the interpretations of classes, properties, individuals and OWL/RDF vocabulary are strictly separated. Therefore, classes and properties, unsurprisingly, *cannot* be treated as ordinary resources as they are in RDF MT. In other words, even those RDF(S) statements which are valid OWL DL statements do not share the same meaning in an RDF(S) ontology and an OWL DL ontology.

Although the above disjointness restriction is not required in the RDF MT-style semantics of OWL Full, there exist at least three known issues that the RDF–style semantics for OWL Full needs to solve, and a proven solution has yet to be given. The first issue is about entailment [44]. Consider the following question: does the following individual axiom

```
Individual(ex:John
    type(intersectionOf(ex:Student ex:Employee ex:European)))
```

entail the individual axiom

```
Individual(ex:John
    type(intersectionOf(ex:Student ex:European)))?
```

In OWL DL, the answer is simply ‘yes’, since intersectionOf(ex:Student ex:Employee ex:European) is a sub-class of intersectionOf(ex:Student ex:European). Since in RDF(S) every class is a resource, OWL Full needs to make sure of the existence of the resource intersectionOf(ex:Student ex:European) in every possible interpretation; otherwise, the answer will be ‘no’ which leads to a disagreement between OWL DL and OWL Full. In general, OWL Full introduces so called *comprehension principles* to add all the missing resources into the domain for all the OWL class descriptions. It has yet to be proved that the proper resources are all added into the universe, no more and no less, and that the added resources will not bring any side-effects.

The second issue is about contradiction classes [44], [45], [27]. In OWL Full, it is possible to construct a class the instances of which have no rdf:type relationship linked to:

```
_:c owl:onProperty rdf:type; owl:allValuesFrom _:d .
```

_:d owl:complementOf _:e .

_:e owl:oneOf _:l

_:l rdf:first _:c; rdf:rest rdf:nil.

The above triples require that rdf:type relates members of the class _:c to anything but _:c. It is impossible for one to determine the membership of _:c. If an object is an instance of _:c, then it is not; but if it is not then it is — this is a contradiction class. Note that it is not a valid OWL DL class, as OWL DL disallows using rdf:type as an object property. With naive comprehension principles, resources of contradiction classes would be added to all possible OWL Full interpretations, which thus have ill-defined class memberships. To avoid the issue, the comprehension principles must also consider avoiding contradiction classes. Unsurprisingly, devising such comprehension principles took a considerable amount of effort [27], and no proof has ever shown that all possible contradiction classes are excluded in the comprehension principles of OWL Full.

The third issue is about the size of the universe [26]. Consider the following question: is it possible that there is only one object in an interpretation of the following OWL ontology?

```
Individual(elp:Ganesh type(elp:Elephant))
DisjointClasses(elp:Elephant elp:Plant)
```

In OWL DL, classes are not objects, so the answer is 'yes': The only object in the domain is the interpretation of elp:Ganesh, the elp:Elephant class thus has one instance, i.e., the interpretation of elp:Ganesh, and the elp:Plant class has no instances. In OWL Full, since classes are also objects, besides elp:Ganesh, the classes elp:Elephant and elp:Plant should both be mapped to the only one object in the universe. This is not possible because the interpretation of elp:Ganesh is an instance of elp:Elephant, but not an instance of elp:Plant; hence, elp:Elephant and elp:Plant should be different, i.e., there should be at least two objects in the universe. As the above axioms are valid OWL DL axioms, this example show s that OWL Full disagrees with OWL DL on valid OWL DL ontologies. Furthermore, this example shows that the interpretation of OWL Full has different features than the interpretation of standard First Order Logic (FOL) model theoretic semantics. This raises the question as to whether it is possible to layer FOL languages on top of RDF(S).

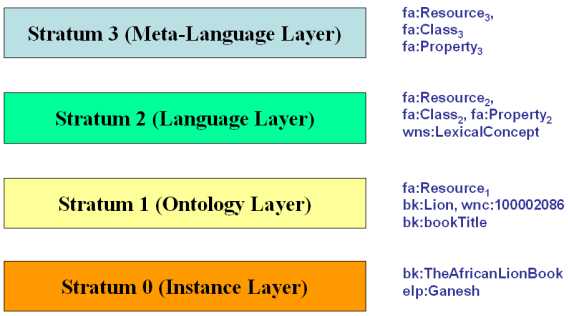

Fig. 2. The UML-like metamodeling architecture (number of strata = 4) of RDFS(FA)

Consequently, there is a serious mismatch between the semantics of OWL DL and OWL Full. Even for two *OWL DL* ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$, $\mathcal{O}_1$ OWL Full-entails $\mathcal{O}_2$ does *not* imply that $\mathcal{O}_1$ OWL DL-entails $\mathcal{O}_2$ [47]. Therefore, the semantic connection (at least in terms of entailment) between OWL DL and OWL Full seems rather weak. Furthermore, [36] shows that the metamodeling of OWL Full contributes to its undecidability too. In short, OWL Full has yet integrated RDF(S) and OWL DL in a satisfactory manner.

IV. RDFS(FA)

In this section, we propose *RDFS(FA)* (RDFS with Fixed layered metamodeling Architecture), as a sub-language of RDF(S), to restore the desired connection between RDF(S) and OWL DL. From the lessons we learnt in previous sections and related works (cf. Section VII), RDFS(FA) should address the following characteristics of RDF(S):

- RDF triples have built-in semantics.
- Classes and properties, including built-in classes and properties of RDF(S) and its subsequent languages such as OWL, are treated as objects (or resources) in the domain.
- There are no restrictions on the use of built-in vocabularies.

Intuitively, RDFS(FA) provides a UML like metamodeling architecture. Let us recall that RDFS has a non-layered metamodeling architecture; resources in RDFS can be classes, objects and properties at the same time, viz. classes and their instances (as well as relationships between the instances) are the same layer. RDFS(FA), instead, divides up the universe of discourse into a series of strata (or layers). The built-in modelling primitives of RDFS are separated into different strata of RDFS(FA), and the semantics of modelling primitives depend

on the stratum they belong to. Theoretically there can be a large number of strata in the metamodeling architecture; in practice, four strata (as shown in Figure 2) are usually enough. The UML-like meta-modeling architecture makes it easier for users who are familiar with UML to understand and use RDFS(FA).

In RDFS(FA), classes cannot be objects and vice versa;[12] in RDFS, Web resources can be classes, properties, objects or even datatypes all at once. We argue that RDFS(FA) is more intuitive than RDFS based on the following observation: when users design their ontologies, a common concern is to decide whether to model something in the domain as a class or as an object. This concern suggests that users intuitively tend to assume that classes and objects should be different from each other. Therefore, layered meta-models seems to be more intuitive than non-layered meta-models. As the HCI (Human Computer Interaction) aspects of ontology engineering are relatively unexplored and pretty challenging, further investigation of this aspect will be interesting and necessary.

In the rest of this section, we will give *formal* semantics of RDFS(FA) and ontologies written in RDFS(FA). We will discuss a strong connection between RDFS(FA) and OWL DL in Section V. Further discussions of the role RDFS(FA) plays in the Semantic Web, illustrated by some examples, will be presented in Section VI.

A. RDFS(FA) Semantics

Let us introduce the design philosophy of RDFS(FA), before moving on to the formal semantics of RDFS(FA).

1) Design Philosophy: The design of RDFS(FA) embodies two main principles. Firstly, in RDFS(FA), RDF is used (only) as standard *syntax* for annotations, i.e., the built-in semantics for RDF triples are disregarded, and new semantics is given to RDFS(FA) triples, or RDFS(FA) axioms (cf. Section IV-B). Secondly, RDFS(FA) provides various Web resources with Description Logic-style semantics.

2) Interpretations: The semantics of RDFS(FA) starts with the notation of vocabulary. Instead of having a mixed vocabulary like that of RDF(S), RDFS(FA) provides a separated vocabulary as follows. For ease of presentation, this paper does not cover blank nodes, which

[12]Classes can be regarded as mega-objects in upper strata of the metamodeling architecture.

can be handled similar to the way that URI references are handled.

Definition 4 (RDFS(FA) Vocabulary) An *RDFS(FA) vocabulary* $\mathbf{V}$ consists of a set of literals $\mathbf{V}_{\mathrm{L}}$, and seven sets of pairwise disjoint URI references, which are $\mathbf{V}_{\mathrm{C}}$ (class URIrefs), $\mathbf{V}_{\mathrm{D}}$ (datatype URIrefs), $\mathbf{V}_{\mathrm{AP}}$ (abstract property URIrefs), $\mathbf{V}_{\mathrm{DP}}$ (datatype property URIrefs), $\mathbf{V}_{\mathrm{ANP}}$ (annotation property URIrefs), $\mathbf{V}_{\mathrm{I}}$ (individual URIrefs) and $\mathbf{V}_{\mathrm{S}}$ ={$\mathrm{fa{:}Literal}$, $\mathrm{fa{:}type_1}$, $\mathrm{fa{:}type_2}$, ...}. $\mathbf{V}_{\mathrm{C}}$ ($\mathbf{V}_{\mathrm{AP}}$) is divided into disjoint *stratified subsets* $\mathbf{V}_{\mathrm{C_1}}$, $\mathbf{V}_{\mathrm{C_2}}$,..., ($\mathbf{V}_{\mathrm{AP1}}$, $\mathbf{V}_{\mathrm{AP2}}$,...) of class (abstract property) URIrefs in strata 1,2, ..., where we use a subscript i to indicate URI references in the stratum i. The built-in class URIrefs of RDFS(FA) are $\mathrm{fa{:}Resource_{i+1}}$, $\mathrm{fa{:}Class_{i+2}}$, $\mathrm{fa{:}Property_{i+2}}$, $\mathrm{fa{:}AbstractProperty_{i+2}}$, $\mathrm{fa{:}DatatypeProperty}$ and $\mathrm{fa{:}AnnotationProperty}$; the built-in abstract property URIrefs of RDFS(FA) are $\mathrm{fa{:}subClassOf_{i+2}}$, $\mathrm{fa{:}subPropertyOf_{i+2}}$, $\mathrm{fa{:}domain_{i+2}}$ and $\mathrm{fa{:}range_{i+2}}$; the built-in annotation property URIrefs of RDFS(FA) are $\mathrm{fa{:}label}$, $\mathrm{fa{:}comment}$, $\mathrm{fa{:}seeAlso}$ and $\mathrm{fa{:}isDefinedBy}$; other built-in URIrefs of RDFS(FA) are those in $\mathbf{V}_{\mathrm{S}}$. We use a superscript b (u) together with $\mathbf{V}_{\mathrm{C}}$, $\mathbf{V}_{\mathrm{AP}}$ and their stratified subsets, to indicate the corresponding subsets of built-in (user-defined) URI references. ◇

Formally, the semantics of RDFS(FA) individuals, classes, datatypes, abstract properties, datatype properties and typed literals is defined in terms of an interpretation as follows. A datatype map $\mathbf{M}_d$ is a partial mapping from datatype URIrefs to datatypes [19].

Definition 5 (RDFS(FA) Interpretation) Given an RDFS(FA) vocabulary $\mathbf{V}$, an *RDFS(FA) interpretation* w.r.t. a datatype map $\mathbf{M}_d$ is a tuple of the form $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$, where $\Delta^{\mathcal{J}}$ is the domain (a non-empty set) and $\cdot^{\mathcal{J}}$ is the interpretation function. Let $\Delta_A^{\mathcal{J}}$ be the abstract domain (a non-empty set), i a non-negative integer, $\Delta_{A\mathrm{i}}^{\mathcal{J}}$ the abstract domain in stratum i and $\Delta_{\mathbf{D}}$ the domain (a non-empty set) for datatypes in a datatype map $\mathbf{M}_d$, $\mathcal{J}$ satisfies the following conditions:

1) $\Delta_A^{\mathcal{J}} = \bigcup_{\mathrm{i}\geq 0} \Delta_{A\mathrm{i}}^{\mathcal{J}}$,
2) $\Delta_{A\mathrm{i+1}}^{\mathcal{J}} = 2^{\Delta_{A\mathrm{i}}^{\mathcal{J}}} \cup 2^{\Delta_{A\mathrm{i}}^{\mathcal{J}} \times \Delta_{A\mathrm{i}}^{\mathcal{J}}}$
3) $\Delta_{\mathbf{D}} \cap \Delta_A^{\mathcal{J}} = \emptyset$,
4) $\bigcup_{\forall d = \mathbf{M}_d(u)} V(d) \subseteq \Delta_{\mathbf{D}}$, where $V(d)$ is the value space of d,

5) $\Delta^{\mathcal{J}} = \Delta_A^{\mathcal{J}} \cup \Delta_{\mathbf{D}}$,

6) $\forall \mathsf{a} \in \mathbf{V}_{\mathrm{I}} :\ \mathsf{a}^{\mathcal{J}} \in \Delta_{A0}^{\mathcal{J}}$,

7) $\forall \mathsf{C} \in \mathbf{V}_{\mathrm{C_{i+1}}} :\ \mathsf{C}^{\mathcal{J}} \subseteq \Delta_{A\mathrm{i}}^{\mathcal{J}}$,

8) $\forall p \in \mathbf{V}_{\mathrm{APi+1}} :\ p^{\mathcal{J}} \subseteq \Delta_{A\mathrm{i}}^{\mathcal{J}} \times \Delta_{A\mathrm{i}}^{\mathcal{J}}$,

9) $\forall n \in \mathbf{V}_{\mathrm{ANP}} :\ \langle x, y\rangle \in n^{\mathcal{J}} \rightarrow y \in \Delta_{\mathbf{D}}$,

10) $\forall r \in \mathbf{V}_{\mathrm{DP}} :\ r^{\mathcal{J}} \subseteq \Delta_{A0}^{\mathcal{J}} \times \Delta_{\mathbf{D}}$,

11) $\mathrm{fa{:}type_{i+1}}^{\mathcal{J}} \subseteq \Delta_{A\mathrm{i}}^{\mathcal{J}} \times \mathrm{fa{:}Class_{i+2}}^{\mathcal{J}}$,

12) $\mathrm{fa{:}Literal}^{\mathcal{J}} = \Delta_{\mathbf{D}}$,

13) $\mathrm{fa{:}Resource_{i+1}}^{\mathcal{J}} = \Delta_{A\mathrm{i}}^{\mathcal{J}}$,

14) $\forall \mathsf{C} \in \mathbf{V}_{\mathrm{C_{i+1}}} :\ \mathsf{C}^{\mathcal{J}} \in \mathrm{fa{:}Class_{i+2}}^{\mathcal{J}}$,

15) $\forall p \in \mathbf{V}_{\mathrm{APi+1}} :\ p^{\mathcal{J}} \in \mathrm{fa{:}AbstractProperty_{i+2}}^{\mathcal{J}}$,

16) $\forall r \in \mathbf{V}_{\mathrm{DP}} :\ r^{\mathcal{J}} \in \mathrm{fa{:}DatatypeProperty}^{\mathcal{J}}$,

17) $\forall n \in \mathbf{V}_{\mathrm{ANP}} :\ n^{\mathcal{J}} \in \mathrm{fa{:}AnnotationProperty}^{\mathcal{J}}$,

18) $\mathrm{fa{:}Class_{i+2}}^{\mathcal{J}} \subseteq \mathrm{fa{:}Resource_{i+2}}^{\mathcal{J}}$ and $\mathrm{fa{:}Property_{i+2}}^{\mathcal{J}} \subseteq \mathrm{fa{:}Resource_{i+2}}^{\mathcal{J}}$,

19) $\mathrm{fa{:}AbstractProperty_{i+2}}^{\mathcal{J}} \subseteq \mathrm{fa{:}Property_{i+2}}^{\mathcal{J}}$ and $\mathrm{fa{:}DatatypeProperty}^{\mathcal{J}} \subseteq \mathrm{fa{:}Property_2}^{\mathcal{J}}$,

20) $\forall u \in \mathbf{V}_{\mathrm{D}}$, if $\mathbf{M}_d(u) = d$, then

   a) $u^{\mathcal{J}} = V(d)$, where $V(d)$ is the value space of d,

   b) if $v \in L(d)$, then $(\text{“}v\text{”}\hat{}\hat{}u)^{\mathcal{J}} = L2V(d)(v)$, where $L(d)$ is lexical space of d and $L2V(d)$ is the lexical-to-value mapping of d,

   c) if $v \notin L(d)$, then $(\text{“}v\text{”}\hat{}\hat{}u)^{\mathcal{J}}$ is undefined;[13]

   otherwise, $u^{\mathcal{J}} \subseteq \Delta_{\mathbf{D}}$ and $\text{“}v\text{”}\hat{}\hat{}u \in \Delta_{\mathbf{D}}$. ◇

There are some remark on Definition 5. Firstly, the domain (of universe) $\Delta^{\mathcal{J}}$ in RDFS(FA) is disjointly divided into the abstract domain $\Delta_A^{\mathcal{J}}$ and the datatype domain $\Delta_{\mathbf{D}}$ (cf. Figure 3), where $\Delta_A^{\mathcal{J}}$ is further disjointly divided into sub-abstract domains $\Delta_{A\mathrm{i}}^{\mathcal{J}}$ in different strata (layers) and $\Delta_{\mathbf{D}}$ is a super-set of the union of the value spaces of all the datatypes in $\mathbf{M}_d$. Secondly, Conditions 12-19 are extra semantic constraints on the built-in URIrefs in $\mathbf{V}_{\mathrm{S}}$ and $\mathbf{V}_{\mathrm{C}}$. Condition 12 ensures that $\mathrm{fa{:}Literal}$ is interpreted as the datatype domain $\Delta_{\mathbf{D}}$,

[13]The reader is invited to note that there is a tiny difference between OWL and RDF datatyping in handling typed literals with invalid lexical forms. Like RDFS(FA), OWL datatyping treats them as contradictions; RDF datatyping does not, but interprets them as some non-data-valued objects.

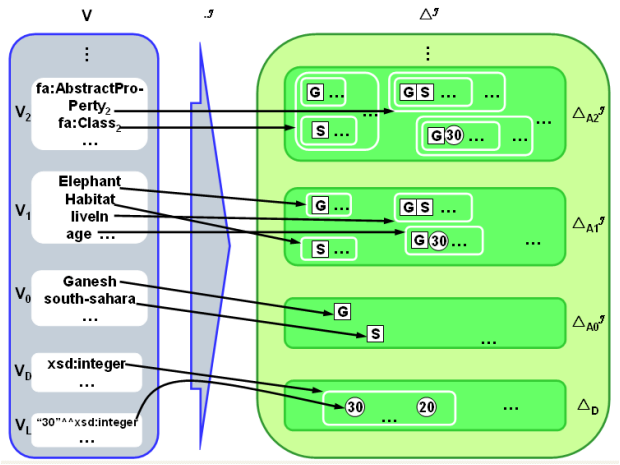

Fig. 3. RDFS(FA) interpretation

while condition 13 ensures that $\mathrm{fa{:}Resource_{i+1}}$ is interpreted as the abstract domain $\Delta_{A_i}^{\mathcal{J}}$. Conditions 14-17 ensures that the interpretations of $\mathrm{fa{:}Class_{i+2}}$, $\mathrm{fa{:}AbstractProperty_{i+2}}$, $\mathrm{fa{:}DatatypeProperty_{i+2}}$ and $\mathrm{fa{:}AnnotationProperty}$ should contain the interpretations of corresponding URI references. Condition 18 ensure that classes and properties are resources in corresponding strata; condition 19 ensures that abstract properties and properties in corresponding strata, and that datatype properties are in stratum 2.

Figure 3 illustrates the interpretation of RDFS(FA). Typed literals (such as $\mathrm{“30”\hat{}\hat{}xsd{:}integer}$) are interpreted as values in the value space corresponding datatypes (such as $V(integer)$). All value spaces of datatypes in $\mathbf{M}_d$ are subset of $\Delta_{\mathbf{D}}$. The datatype domain is disjoint with the abstract domain, which is stratified into sub-abstract domains ($\Delta_{A_0}^{\mathcal{J}}$, $\Delta_{A_1}^{\mathcal{J}}$, etc.). In stratum 0 (the Instance Layer), object URIrefs (e.g., `elp:Ganesh` and `elp: south-sahara`) are interpreted as objects (i.e., resources in $\Delta_{A_0}^{\mathcal{J}}$). In stratum 1 (the Ontology Layer), class URIrefs (such as `elp:`Elephant and `elp:`Habitat) are interpreted as sets of objects. Abstract property URIrefs (such as `elp:`*liveIn*) are interpreted as sets of pairs of objects. Datatype property URIrefs (such as `elp:`*age*) are interpreted as a set of pairs where the first resource (e.g., `elp:Ganesh`) is an object, and the second resource is a datatyped value (e.g., the integer 30). In stratum 2 (the Language Layer), fa:Class$_2$ is interpreted as a set of sets of objects, and $\mathrm{fa{:}AbstractProperty_2}$ is interpreted as a set of sets of pairs of objects.

B. RDFS(FA) Ontologies

Informally speaking, an RDFS(FA) ontology is a set of RDFS(FA) axioms, which are basically RDF triples (in N3 syntax)[14] with extra syntactic rules, which (1) disallow arbitrary use of its built-in vocabulary and (2) enable the use of meta-classes and meta-properties in specified layers as well as the use of annotation properties.

Definition 6 (RDFS(FA) Ontologies) Given an RDFS(FA) vocabulary $\mathbf{V}$, let i be a non-negative integer, $\mathtt{a}, \mathtt{b} \in \mathbf{V}_{\mathrm{I}}$, $\mathsf{D}_1 \in \mathbf{V}_{\mathrm{C}_1}$, $\mathsf{C} \in \mathbf{V}^u_{\mathrm{C}_{i+1}}$, $\mathsf{D} \in \mathbf{V}_{\mathrm{C}_{i+1}}$, $\mathsf{H} \in \mathbf{V}_{\mathrm{C}_{i+2}}$, $p_1 \in \mathbf{V}^u_{\mathrm{AP}_1}$, $p \in \mathbf{V}^u_{\mathrm{AP}_{i+1}}$, $q \in \mathbf{V}_{\mathrm{APi+1}}$ $r, s \in \mathbf{V}_{\mathrm{DP}}$, $q' \in \mathbf{V}^u_{\mathrm{AP}_{i+2}}$, $u \in \mathbf{V}_{\mathrm{D}}$, $X, Y \in \mathbf{V}^u_{\mathrm{C}_{i+1}} \cup \mathbf{V}^u_{\mathrm{AP}_{i+1}}$, $n \in \mathbf{V}_{\mathrm{ANP}}$ and $w \in \mathbf{V} \setminus \mathbf{V}_{\mathrm{L}}$.

An RDFS(FA) ontology is a finite, possibly empty, set of axioms of the form:

1) [C $\mathrm{fa{:}subClassOf}_{i+2}$ D .], called *class inclusions*,
2) [p $\mathrm{fa{:}subPropertyOf}_{i+2}$ q .], called *abstract property inclusions*;
3) [r $\mathrm{fa{:}subPropertyOf}_2$ s .], called *datatype property inclusions*;
4) [p $\mathrm{fa{:}domain}_{i+2}$ D .], called *abstract property domain restrictions*;
5) [r $\mathrm{fa{:}domain}_2$ D_1 .], called *datatype property domain restrictions*;
6) [p $\mathrm{fa{:}range}_{i+2}$ D .], called *abstract property range restrictions*;
7) [r $\mathrm{fa{:}range}_2$ u .], called *datatype property range restrictions*;
8) [$\mathtt{a}$ $\mathrm{fa{:}type}_1$ D_1 .], called *class assertions*,
9) [$\mathtt{a}$ p_1 $\mathtt{b}$.], called *abstract property assertions*,
10) [$\mathtt{a}$ r “v”^^u .], called *datatype property assertions*,
11) [X $\mathrm{fa{:}type}_{i+2}$ H .], called *meta class assertions*,
12) [X q' Y .], called *meta abstract property assertions*,
13) [w n “v”^^u .], called *annotation property assertions*,
14) [n $\mathrm{rdf{:}type}$ $\mathrm{fa{:}AnnotationProperty}$.], called *annotation property declarations*.

Axioms of the form of 1) to 7) are called *conceptual* axioms;, those of the forms of 8) to 12) are called *assertive* axioms; those of the forms of 13) and 14) are called *annotation* axioms. We say an axiom [s p o .] is in stratum m if m = min (i,j,k), where i, j and k are the strata numbers of s, p and o, respectively. An interpretation $\mathcal{J}$ *satisfies* an RDFS(FA) axiom

[14]Here we use the N3 syntax, instead of the RDF/XML syntax, as it is more compact.

φ, written as $\mathcal{J} \models \varphi$, if $\mathcal{J}$ meets certain semantic condition:

1) $\mathcal{J} \models [\mathsf{C}\ \mathrm{fa{:}subClassOf_{i+2}}\ \mathsf{D}\ .]$ if $\mathsf{C}^{\mathcal{J}} \subseteq \mathsf{D}^{\mathcal{J}}$;
2) $\mathcal{J} \models [p\ \mathrm{fa{:}subPropertyOf_{i+2}}\ q\ .]$ if $p^{\mathcal{J}} \subseteq q^{\mathcal{J}}$;
3) $\mathcal{J} \models [r\ \mathrm{fa{:}subPropertyOf_{2}}\ s\ .]$ if $r^{\mathcal{J}} \subseteq s^{\mathcal{J}}$;
4) $\mathcal{J} \models [p\ \mathrm{fa{:}domain_{i+2}}\ \mathsf{D}]$ if $\forall x.\langle x, y\rangle \in p^{\mathcal{J}} \rightarrow x^{\mathcal{J}} \in \mathsf{D}^{\mathcal{J}}$;
5) $\mathcal{J} \models [r\ \mathrm{fa{:}domain_{2}}\ \mathsf{D_1}\ .]$ if $\forall x.\langle x, t\rangle \in r^{\mathcal{J}} \rightarrow x^{\mathcal{J}} \in \mathsf{D}_1^{\mathcal{J}}$;
6) $\mathcal{J} \models [\mathrm{p}\ \mathrm{fa{:}range_{i+2}}\ \mathsf{D}\ .]$ if $\forall y.\langle x, y\rangle \in p^{\mathcal{J}} \rightarrow y^{\mathcal{J}} \in \mathsf{D}^{\mathcal{J}}$;
7) $\mathcal{J} \models [r\ \mathrm{fa{:}range_{2}}\ \mathsf{u}\ .]$ if $\forall t.\langle x, t\rangle \in r^{\mathcal{J}} \rightarrow t^{\mathcal{J}} \in u^{\mathcal{J}}$;
8) $\mathcal{J} \models [\mathsf{a}\ \mathrm{fa{:}type_1}\ \mathsf{C_1}\ .]$ if $\mathsf{a}^{\mathcal{J}} \in \mathsf{C_1}^{\mathcal{J}}$;
9) $\mathcal{J} \models [\mathsf{a}\ p_1\ \mathsf{b}\ .]$ if $\langle \mathsf{a}^{\mathcal{J}}, \mathsf{b}^{\mathcal{J}}\rangle \in {p_1}^{\mathcal{J}}$;
10) $\mathcal{J} \models [\mathsf{a}\ r\ \text{“}v\text{”}\hat{}\hat{}u\ .]$ if $\langle \mathsf{a}^{\mathcal{J}}, (\text{“}v\text{”}\hat{}\hat{}u)^{\mathcal{J}}\rangle \in r^{\mathcal{J}}$;
11) $\mathcal{J} \models [X\ \mathrm{fa{:}type_{i+2}}\ \mathsf{H}\ .]$ if $X^{\mathcal{J}} \in \mathsf{H}^{\mathcal{J}}$;
12) $\mathcal{J} \models [X\ q'\ Y\ .]$ if $\langle X^{\mathcal{J}}, Y^{\mathcal{J}}\rangle \in {q'}^{\mathcal{J}}$;
13) $\mathcal{J} \models [w\ n\ \text{“}v\text{”}\hat{}\hat{}u\ .]$ if $(\text{“}v\text{”}\hat{}\hat{}u)^{\mathcal{J}} \in \Delta_{\mathbf{D}}$,
14) $\mathcal{J} \models [\ n\ \mathrm{rdf{:}type\ fa{:}AnnotationProperty.}]$ if $n^{\mathcal{J}} \in \mathrm{fa{:}AnnotationProperty}^{\mathcal{J}}$.

An interpretation $\mathcal{J}$ *satisfies* an ontology $\mathcal{O}$, written as $\mathcal{J} \models \mathcal{O}$, iff it satisfies all the axioms in $\mathcal{O}$; $\mathcal{O}$ is *satisfiable* (*unsatisfiable*), written as $\mathcal{O} \not\models \bot$ ($\mathcal{O} \models \bot$), iff there exists (does not exist) such an interpretation $\mathcal{J}$.

Given an RDFS(FA) axiom φ, $\mathcal{O}$ *entails* φ, written as $\mathcal{O} \models \varphi$, iff for all models $\mathcal{J}$ of $\mathcal{O}$ we have $\mathcal{J} \models \varphi$. An ontology $\mathcal{O}$ *entails* an ontology $\mathcal{O}'$, written as $\mathcal{O} \models \mathcal{O}'$, iff for all models $\mathcal{J}$ of $\mathcal{O}$ we have $\mathcal{J} \models \mathcal{O}'$. Two ontologies $\mathcal{O}$ and $\mathcal{O}'$ are *equivalent*, written as $\mathcal{O} \equiv \mathcal{O}'$, iff $\mathcal{O} \models \mathcal{O}'$ and $\mathcal{O}' \models \mathcal{O}$. $\diamond$

We invite the reader to note that RDFS(FA) axioms of the form 1-8 and 11 are RDFS statements with extra (subscript) information specifying the strata that the related resources belong to. For example, [C $\mathrm{fa{:}subClassOf_{i+2}}$ D .] requires that the classes C and D should be on stratum i+1. Furthermore, RDFS(FA) provides the use of three kinds of properties: abstract properties, datatype properties and annotation properties (cf. RDFS(FA) axioms of the form 9,12, 10 and 13). Last but not least, let us point out that $\mathrm{rdf{:}type}$ is used in annotation property declarations because annotation property are not bound to any stratum.

```
@prefix fa:  <http://dl-web.man.ac.uk/rdfsfa/ns#>
@prefix elp:  <http://example.org/Animal#>

elp:Animal fa:type2 fa:Class2 .
elp:Habitat fa:type2 fa:Class2 .
elp:Elephant fa:type2 fa:Class2 ; fa:subClassOf2 elp:Animal .
elp:liveIn fa:type2 fa:AbstractProperty2 ;
           fa:domain2 elp:Animal ; fa:range2 elp:Habitat .

elp:south-sahara fa:type1 elp:Habitat .
elp:Ganesh fa:type1 elp:Elephant ; elp:liveIn elp:south-sahara .
```

Fig. 4. An RDFS(FA) ontology

The interpretation of class inclusions, property inclusions in stratum 1 as well as class assertions and property assertions are exactly the same as the corresponding OWL DL axioms (cf. Section V). RDFS(FA) meta-axioms are very similar to the above, except that they apply on classes and properties in strata that are higher than stratum 1. RDFS(FA) annotation property assertions require that values of annotation properties should be data values in the datatype domain.

Figure 4 shows an example RDFS(FA) ontology. Firstly, the layering structure is clear. elp:Animal, elp:Habitat, elp:Elephant and elp:*liveIn* are in stratum 1 (the Ontology layer), while elp:Ganesh and elp:south- sahara are in stratum 0 (the Instance Layer). Secondly, RDFS(FA) disallows arbitrary use of its built-in vocabulary. For example, in class inclusion axioms, the subjects can only be only user-defined class URIrefs (such as elp:Animal), which could disallow triples like

fa:Resource$_1$ fa:subClassOf$_2$ elp:Animal .

Furthermore, RDFS(FA) allows users to specify classes and properties in specified strata. For example, the class inclusion axiom

elp:Elephant fa:subClassOf$_2$ elp:Animal .

requires that both elp:Elephant and elp:Animal are class URIrefs in stratum 1.

C. Rules of Thumb on Strata Numbers

Writing an RDFS(FA) ontology should be an enjoyable task. Although the numbers of strata can/should be encapsulated by tools, in this section, we are going to present some *rules*

of thumb to help authors of RDFS(FA) ontologies quickly get these numbers of strata right. We will use RDFS(FA) axioms in Figure 4 to illustrate these rules of thumb.

1) The *first* rule of thumb is that the subscripts of built-in RDFS(FA) vocabulary represent exactly the stratum that they are in. For example, $\mathrm{fa{:}Resource}_1$ is in stratum 1 and $\mathrm{fa{:}Class}_2$ is in stratum 2.
2) Let [s p o .] be an RDFS(FA) axiom. The *second* rule of thumb is that if p *is* an instance-of relationship, then o is one stratum higher than s, and p is in the same stratum as o. For example, in the axiom [`elp:Ganesh` $\mathrm{fa{:}type}_1$ `elp:Elephant` .], `elp:Ganesh` is an object in stratum 0 (Instance Layer), `elp:Elephant` and $\mathrm{fa{:}type}_1$ are one stratum higher, i.e. in stratum 1; in the axiom [`elp:Elephant` $\mathrm{fa{:}type}_2$ $\mathrm{fa{:}Class}_2$.], `elp:Elephant` is in stratum 1 and both $\mathrm{fa{:}type}_2$ and $\mathrm{fa{:}Class}_2$ are in stratum 2.
3) Let [s p o .] be an RDFS(FA) axiom. The *third* rule of thumb is that if p is *not* an instance-of relationship, then s and o should be in the same stratum, and p should be one stratum higher than s and o. For example, in the axiom [`elp:Elephant` $\mathrm{fa{:}subClassOf}_2$ `elp:Animal` .], `elp:Elephant` and `elp:Animal` are in stratum 1 and $\mathrm{fa{:}subClassOf}_2$ is in stratum 2; in the axiom [`elp:Ganesh elp:`*liveIn* `elp: south-sahara`.], `elp:Ganesh` and `elp:south-sahara` are objects in stratum 0 (Instance Layer), while `elp:`*liveIn* is in stratum 1.
4) Let [s p o .] be an RDFS(FA) axiom, and s, p and o in strata i, j and k, respectively. The stratum number of the axiom [s p o .] is min(i,j,k), i.e., the smallest stratum number among those of s, p and o. For example, the axiom [`elp:Ganesh` $\mathrm{fa{:}type}_1$ `elp:Elephant` .] is in stratum 0 and the axiom [`elp:Elephant` $\mathrm{fa{:}subClassOf}_2$ `elp:Animal` .] is in stratum 1.

In practice, although users will use some ontology editor to edit their RDFS(FA) ontologies, keeping these rules of thumb in mind could help them have a better understanding of the ontologies.

V. RDFS(FA) AND OWL DL

In this section, we show that the interoperability between RDFS(FA) and OWL DL.

It is much easier to layer OWL DL, syntactically *and* semantically, on top of RDFS(FA)

RDFS(FA) Axioms	OWL Axioms (Abstract Syntax)	OWL Axioms (RDF Syntax)
[C_1 fa:subClassOf$_2$ D_1 .]	SubClassOf(C_1 D_1)	[C_1 rdfs:subClassOf D_1 .]
[p_1 fa:subPropertyOf$_2$ q_1 .] [r_1 fa:subPropertyOf$_2$ s_1 .] [p_1 fa:domain$_2$ D_1 .] [r_1 fa:domain$_2$ D_1 .] [p_1 fa:range$_2$ D_1 .] [r_1 fa:range$_2$ u .]	SubPropertyOf(p_1 q_1) SubPropertyOf(r_1 s_1) ObjectProperty(p_1 domain(D_1)) DatatypeProperty(r_1 domain(D_1)) ObjectProperty(p_1 range(D_1)) DatatypeProperty(r_1 range(u))	[p_1 rdfs:subPropertyOf q_1 .] [r_1 rdfs:subPropertyOf s_1 .] [p_1 rdfs:domain D_1 .] [r_1 rdfs:domain D_1 .] [p_1 rdfs:range D_1 .] [r_1 rdfs:range u .]
[a fa:type$_1$ C_1 .] [a p_1 b .] [a r_1 "v"^^u .]	Individual(a type(C_1)) Individual(a value(p_1 b)) Individual(a value(r_1 "v"^^u))	[a rdf:type C_1 .] [a p_1 b .] [a r_1 "v"^^u .]
[a fa:type$_1$ fa:Resource$_1$.] [C_1 fa:type$_2$ fa:Class$_2$.] [p_1 fa:type$_2$ fa:AbstractProperty$_2$.] [r_1 fa:type$_2$ fa:DatatypeProperty .]	Individual(a) Class(C_1) ObjectProperty(p_1) DatatypeProperty(r_1)	[a rdf:type rdfs:Resource.] [C_1 rdf:type owl:Class .] [p_1 rdf:type owl:ObjectProperty .] [r_1 rdf:type owl:DatatypeProperty .]

TABLE III

THE MAPPING BETWEEN THE RDFS(FA) AXIOMS IN STRATA 0-1 AND OWL DL AXIOMS

than on top of RDF(S). In particular, there is a one-to-one bidirectional mapping (as shown in Table III) between the RDFS(FA) axioms in strata 0-1 and OWL DL axioms in OWL abstract syntax. For example, the RDFS(FA) class inclusion axiom [C_1 fa:subClassOf$_2$ D_1 .] can be mapped to the OWL class axiom (SubClassOf C_1 D_1) and vice versa.

In the syntactic level, it is easier to layer OWL DL on top of RDFS(FA) than on top of RDF(S), due to the above bidirectional mapping. Let us recall that, according to the OWL Semantics and Abstract Syntax document [48], the mapping between OWL DL axioms, or *OWL axioms* for short, and RDF(S) statements is *only* unidirectional, i.e., from OWL axioms to RDF(S) statements. For example, we can map the following OWL axiom

$$(\text{SubClassOf } \mathsf{C}_1 \ \mathsf{D}_1)$$

to the RDF(S) statement

$$[\mathsf{C}_1 \text{ rdfs:subClassOf } \mathsf{D}_1 \text{ .}],$$

with an implicit OWL constraint, viz., C_1 and D_1 can only be class URIrefs, but not URIrefs for properties or individuals, etc. However, the above RDF(S) statement without such (implicit) constraint cannot be correctly mapped to the OWL axiom (SubClassOf C_1 D_1). Interestingly, in the corresponding RDFS(FA) axioms these kinds of implicit constraints are made explicit via the syntactic constraints of the RDFS(FA) class axioms (cf. Definition 6). For example, the RDFS(FA) class inclusion axiom [C_1 fa:subClassOf$_2$ D_1 .] (in place of

$[\mathsf{C}_1\ \mathrm{rdfs{:}subClassOf}\ \mathsf{D}_1\ .]$) requires that both C_1 and D_1 are class URIrefs in stratum 1. This explains why the above bidirectional mapping (listed in Table III) is possible.

In the semantic level, it can be shown (by the following theorem) that the above bidirectional mapping is a semantics-preserving mapping.

Theorem 7 *The bidirectional mapping, shown in Table III, between the RDFS(FA) axioms in strata 0-1 and the corresponding OWL axioms in the OWL abstract syntax is a satisfiability-preserving mapping.*

Proof: Given a datatype map $\mathbf{M}_d$, we only need to show that there exists an interpretation $\mathcal{J}$ satisfying all the listed RDFS(FA) axioms iff there exists an interpretation $\mathcal{I}$ satisfying all the corresponding OWL DL axioms.

For the *only-if* direction, given an RDFS(FA) interpretation $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$ for $\mathbf{V}$ w.r.t. $\mathbf{M}_d$, we can construct an OWL DL interpretation $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ as follows: $\Delta^{\mathcal{I}} = \Delta_{A_0}^{\mathcal{J}}$ and $\Delta_{\mathbf{D}owl} = \Delta_{\mathbf{D}fa}$; for each class URIref (in stratum 1) C, $\mathsf{C}^{\mathcal{I}} = \mathsf{C}^{\mathcal{J}}$; for each datatype URIref (in stratum 1) u, $u^{\mathcal{I}} = u^{\mathcal{J}}$; for each abstract (object) property URIref p (in stratum 1), $p^{\mathcal{I}} = p^{\mathcal{J}}$; for each datatype property URIref r, $r^{\mathcal{I}} = r^{\mathcal{J}}$.

Now we only need to show that if $\mathcal{J}$ satisfies an RDFS(FA) axiom ϕ_1 in the first column of Table III, we have I satisfies the corresponding OWL DL axiom ϕ_2 in the second column of Table III. According to the semantics of RDFS(FA) (Definition 5 on page 16) and RDFS(FA) axioms (Definition 6 on page 19), the semantics of OWL axioms (Tables II), this is trivially true. Therefore, we only give the proof for the class inclusion axiom to illustrate the proofs for the rest: if $\mathcal{J} \models [\mathsf{C}_1\ \mathrm{fa{:}subClassOf}_2\ \mathsf{D}_1\ .]$, according to Definition 6, we have ${\mathsf{C}_1}^{\mathcal{J}} \subseteq {\mathsf{D}_1}^{\mathcal{J}}$, hence ${\mathsf{C}_1}^{\mathcal{I}} \subseteq {\mathsf{D}_1}^{\mathcal{I}}$. Thus, $\mathcal{I} \models \mathrm{SubClassOf}(\mathsf{C}_1\ \mathsf{D}_1)$.

Similarly, the *if* direction is trivially true, we only need to show that, in an RDFS(FA) interpretation $\mathcal{J}$, we can construct abstract domains for strata higher than stratum 0. Let $\mathrm{i} \geq 0$. According to the semantics conditions 7, 8, 13 to 19 in Definition 5, we have ${\mathrm{fa{:}Class}_{\mathrm{i}+2}}^{\mathcal{J}} = 2^{\Delta_{A_{\mathrm{i}}}^{\mathcal{J}}}$, ${\mathrm{fa{:}Property}_2}^{\mathcal{J}} = 2^{\Delta_{A_0}^{\mathcal{J}} \times \Delta_{A_0}^{\mathcal{J}}} \cup 2^{\Delta_{A_0}^{\mathcal{J}} \times \Delta_{\mathbf{D}}}$, ${\mathrm{fa{:}Property}_{\mathrm{i}+3}}^{\mathcal{J}} = 2^{\Delta_{A_{\mathrm{i}+1}}^{\mathcal{J}} \times \Delta_{A_{\mathrm{i}+1}}^{\mathcal{J}}}$ and $\Delta_{A_{\mathrm{i}+1}}^{\mathcal{J}} = \mathrm{fa{:}Resource}_{\mathrm{i}+2} = {\mathrm{fa{:}Class}_{\mathrm{i}+2}}^{\mathcal{J}} \cup {\mathrm{fa{:}Property}_{\mathrm{i}+2}}^{\mathcal{J}}$. Hence we have $\Delta_{A_1}^{\mathcal{J}} = 2^{\Delta_{A_0}^{\mathcal{J}}} \cup 2^{\Delta_{A_0}^{\mathcal{J}} \times \Delta_{A_0}^{\mathcal{J}}} \cup 2^{\Delta_{A_1}^{\mathcal{J}} \times \Delta_{\mathbf{D}}}$ and $\Delta_{A_{\mathrm{i}+2}}^{\mathcal{J}} = 2^{\Delta_{A_{\mathrm{i}+1}}^{\mathcal{J}}} \cup 2^{\Delta_{A_{\mathrm{i}+1}}^{\mathcal{J}} \times \Delta_{A_{\mathrm{i}+1}}^{\mathcal{J}}}$.□

We claim that OWL DL can be semantically layered on top of RDFS(FA). Firstly, [41]

OWL Modelling Primitives	RDFS(FA) Modelling Primitives
$\mathrm{owl{:}Thing}$	$\mathrm{fa{:}Resource_1}$
$\mathrm{owl{:}Class}$	$\mathrm{fa{:}Class_2}$
$\mathrm{owl{:}ObjectProperty}$	$\mathrm{fa{:}AbstractProperty_2}$
$\mathrm{owl{:}DatatypeProperty}$	$\mathrm{fa{:}DatatypeProperty}$

TABLE IV
OWL DL PRESERVES THE SEMANTICS OF BUILT-IN RDFS(FA) PRIMITIVES

RDFS Modelling Primitives	RDFS(FA) Modelling Primitives
$\mathrm{rdfs{:}subClassOf}$	$\mathrm{fa{:}subClassOf_2}$
$\mathrm{rdfs{:}subPropertyOf}$	$\mathrm{fa{:}subPropertyOf_2}$
$\mathrm{rdfs{:}domain}$	$\mathrm{fa{:}domain_2}$
$\mathrm{rdfs{:}range}$	$\mathrm{fa{:}range_2}$

TABLE V
OWL DL USES RDFS PRIMITIVES WITH RDFS(FA) SEMANTICS

shows that RDFS(FA) does not have the semantic issues [41], [44], [45], [25] that RDF(S) has, when we layer OWL on top of it. Secondly, OWL DL reserves the semantics of RDFS(FA) built-in primitives; e.g., Table IV shows that owl:Thing is equivalent to fa:$\mathrm{Resource_1}$, Table V shows that OWL DL uses some RDFS modelling primitives with RDFS(FA) semantics, instead of RDFS semantics. Furthermore, OWL DL extends RDFS(FA) in strata 0-1 by introducing new class descriptions (such as class intersections), new property descriptions (such as inverse properties) and new axioms (such as functional axioms for properties). Most importantly, Theorem 7 shows that OWL DL preserves the meaning of the RDFS(FA) axioms in strata 0-1 shown in Table III.

To sum up, RDFS(FA) is syntactically and semantically compatible with OWL DL.

VI. A CLARIFIED VISION OF THE SEMANTIC WEB

In the previous sections, we have presented RDFS(FA), an alternative to RDFS with a DL-style semantics, so as to repair the broken link between RDF(S) and OWL.

RDFS(FA), consequently, provides a clarified vision of the Semantic Web: RDF is *only* a standard syntax for SW annotations and languages (i.e., the built-in semantics of RDF triples is disregarded), and the meaning of annotations comes from either external agreements (such as Dublin Core) or ontologies (which are more flexible), both of which are supported by

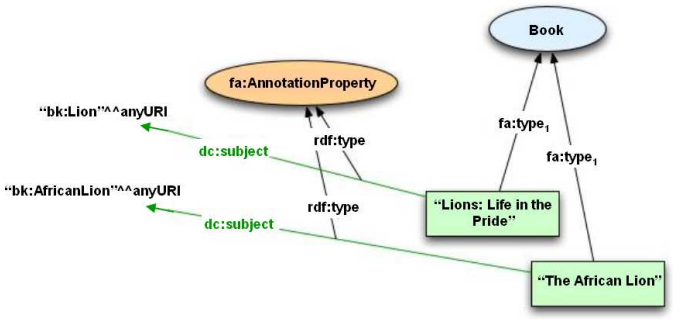

Fig. 5. RDFS(FA): class URIrefs as annotation property values

RDFS(FA).

On the one hand, RDFS(FA) allows the use of Dublin Core information properties as annotation properties. In RDFS(FA), all resources can have annotation properties, such that 'anyone can say anything about anything'. Typed literals can be used to precisely represent values of annotation properties, such as "1999-05-31"^^xsd:date for the dc:date property and "bk:Lion"^^xsd:anyURI for the dc:subject property. In particular, the use of URIrefs as values of annotation properties can enable SW applications to make use of URIrefs of ontology elements, such as classes, in the results of various ontology inferences.

Example 1 *RDFS(FA): Class URIrefs as Values of Annotation Properties*

This example is from [38]. Suppose we have a set of Books *about* Animals and want to annotate each Book with its $subject$, which is a particular species or class of Animals that it talks about. Furthermore, when retrieving all Books *about* Lions from a repository, we want Books that are annotated as books *about* AfricanLions to be included in the results.

We now use the information property dc:*subject* as an annotation property, so as to refer to class URIrefs (cf. Figure 5). The approach we present here is slightly different from Approach 5 in [38] in that annotations are class URIrefs instead of classes.

```
@prefix bk:  <http://protege.stanford.edu/
              swbp/books#>
bk:bookTitle rdf:type fa:AnnotationProperty.
dc:subject rdf:type fa:AnnotationProperty.
bk:AfricanLion fa:type2 fa:Class2; fa:subClassOf2 bk:Lion .
bk:LionsLifeInThePrideBook fa:type1 bk:Book ;
```

bk:*bookTitle* "Lions: Life in the Pride" ;

dc:*subject* "bk:Lion"^^xsd:anyURI .

bk:LionsLifeInThePrideBook fa:type$_1$ bk:Book ;

bk:*bookTitle* "The African Lion" ;

dc:*subject* "bk:AfricanLion"^^xsd:anyURI .

Here the values of dc:subject are interpretations of typed literals "bk:Lion"^^xsd:any URI and "bk:AfricanLion"^^xsd:anyURI, viz. class URIrefs bk:Lion and bk:African Lion, respectively. Since the result of classification of such an RDFS(FA) ontology can be represented as partial orderings of class URIrefs (such as bk:AfricanLion $<$ bk:Lion $<$ bk:Animal), we can make use of such result when retrieving all books about bk:Lion from a repository, i.e., by retrieving books that are annotated (through dc:subject) with bk:Lion *and* books annotated with bk:AfricanLion.

Note that it is not proper to use the information properties defined in Dublin Core as abstract properties (or object properties) in ontologies. Otherwise, there can be unexpected restrictions or implications on the information properties. For example, if one uses dc:*author* as an abstract property in an ontology and there is a (range) constraint in the ontology that an author should be a person, then it disallows anything but persons, such as organisations, to be authors. This is against the intended usage of dc:*author* in Dublin Core.

On the other hand, RDFS(FA) is an ontology language that provides a UML-like layered style for using RDFS. It provides a more intuitive way to use meta-classes and meta-properties, and it is very easy to understand and use by users who are familiar with UML.

Example 2 *RDFS(FA): Meta-classes and Meta-properties*

Applications using WordNet [35] to annotate resources, such as images [58], require the use of meta-classes (such as wns:LexicalConcept) and meta-properties (such as wns:*hyponymOf*).

wns:LexicalConcept fa:subClassOf$_3$ fa:Class$_2$.

wns:*hyponumOf* fa:type$_3$ fa:AbstractProperty$_3$;

fa:subPropertyOf$_3$ fa:subClassOf$_2$;

fa:domain$_3$ wns:LexicalConcept ;

```
    fa:range3 wns:LexicalConcept .
wnc:100002086 fa:type2 fa:Class2 ;
    wns:hyponymOf wnc:100001740 .
```

where wnc:100002086 and wnc:100001740 are WordNet synsets (i.e., classes like 'Elephant' and 'Animal'). The first statement specifies that the class LexicalConcept is a subclass of the built-in RDFS(FA) meta-class $\mathrm{fa{:}Class}_2$, the instances of which are classes in stratum 1. This means that now all instances of LexicalConcept are also classes. In a similar vein, the second statement defines that the WordNet property $hyponymOf$ is a sub-property of the built-in RDFS(FA) meta-property $\mathrm{fa{:}subClassOf}_2$. This enables us to interpret the instances of $hyponymOf$ as subclass links. Based on the rules of thumb presented in Section IV-C, it is easy to see that wnc:100002086 and wnc:100001740 are in stratum 1, wns:LexicalConcept and wns:*hyponymOf* are in stratum 2.

We invite the reader to note the difference between the support of meta-classes and meta-properties in RDFS and RDFS(FA). In RDFS, it is valid to add RDF triples such as

```
rdfs:Class rdf:type wnc:100002086 .
```

which makes the relationship between wnc:100002086 and wns:LexicalConcept rather confusing. Indeed, wnc:100002086 is an instance of wns:LexicalConcept, which is an instance of an instance ($\mathrm{rdfs{:}Class}$) of wnc:100002086; nevertheless, wnc:100002086 and wns:LexicalConcept are not necessarily equivalent to each other.

RDFS(FA) disallows asserting that $\mathrm{fa{:}Class}_2$ is an instance of wnc:100002086 because $\mathrm{fa{:}Class}_2$ is a built-in class (cf. Definition 6), so there is no confusion here.

Most importantly, OWL DL can be syntactically and semantically layered on top of RDFS(FA). In general, introducing RDFS(FA) as a sub-language of RDF(S) makes it more flexible to layer languages on top of RDF(S). With all these distinguished features, RDFS(FA) surely solidifies RDF(S)'s proposed role as the base of the Semantic Web; accordingly, the Semantic Web tower will become clearer, easier to understand and formalise.

VII. Related Work

Initially RDF and RDFS had no formal model theory, nor any formal meaning at all. This made them unlikely foundations for the Semantic Web. As earlier works [37], [7] pointed out, RDFS has a non-standard and non-fixed layer metamodeling architecture, which makes some elements in the model have multiple roles in the RDFS specification. Therefore, it makes even the RDFS specification itself rather confusing and difficult to understand for users. One of the consequences is that when DAML+OIL is layering on top of RDFS, it uses the syntax of RDFS only, but defines its own semantics [56] for the ontological primitives of RDFS. To clear up any confusion, Pan and Horrocks [40] proposed a Fixed layer metamodeling Architecture for RDFS, reducing the multiple roles of RDFS built-in primitives by stratifying them into different layers of the metamodeling architecture.

Subsequently RDF Model Theory (RDF MT) [19] gave an official semantics for RDF and RDFS, justifying the dual roles by treating both classes and properties as objects in the universe. As RDF(S) is expected to be the foundation of the Semantic Web, solving its own problems is only the first step of standardising RDF(S). RDF(S) should also be easily extendable; i.e., other Semantic Web languages should be easily layered on top of RDF(S). Further research ([44], [45], [27], [41]) pointed out that there are at least three potential issues if one extends the RDF MT with OWL constructors. Accordingly, Pan and Horrocks [41] suggested that RDFS could have two kinds of semantics, i.e., RDF MT and the stratified semantics of RDFS(FA). Now both RDF(S) and OWL become W3C recommendations. However, as we pointed out in Section III, there exist syntactic and semantic mismatch between RDF(S) and OWL DL. Although OWL Full is believed to be serving as a connection between RDF and OWL DL, Motik [36] shows that the metamodeling of OWL Full contributes to its undecidability too. The main purpose of this paper, accordingly, is to find a strong connection for them. In particular, this paper extends [41] by providing strong connections between RDFS(FA) and OWL DL; specifically, Theorem 7 shows that there is a semantic-preserving mapping between them (Section V). Furthermore, this paper provides some rules of thumb to help authors of RDFS(FA) ontologies to get the strata numbers right (Section IV-C) and further illustrates in details how RDFS(FA) solidifies RDF(S)'s proposed role as the base of the Semantic Web (Section VI).

There are some interesting research on handling the issue of extending RDF(S) with OWL constructors. ter Horst [53], [54] shows that RDFS extended with a property-related subset of OWL, namely, FunctionalProperty, InverseFunctionalProperty, sameAs, SymmetricProperty, TransitiveProperty, and inverseOf. To obtain a complete set of simple entailment rules, a weaker semantics ('if-semantics') is used, rather than the RDF-MT style 'iff-semantics' semantics of OWL. In our approach, RDFS(FA) does not impose any restriction on its extensibility to more expressive Description Logics such as OWL DL and OWL-Eu.

de Bruijn et al. [12] replaces RDF MT with one based on Herbrand and canonical models, and shows that OWL DL can be built on top of RDF (in terms of the above modified semantics) if one weakens the semantics connection between individual interpretations and class interpretations of URIs. This approach is very similar to the π-semantics approach proposed in [36]; we call this kind of approach the contextual approach. An advantage of the kind of contextual approach is that, although it modifies the semantics of RDF, it does not change its syntax. A disadvantage of this kind of approach is that the modification of RDF semantics causes some lose of inference, which we now use an example in [36] to illustrate. Let us consider the following ontology $\mathcal{O}_1$:

$\mathtt{Harry}\ \mathrm{rdf{:}type}\ \mathsf{Eagle}\ ;\ \mathrm{rdf{:}type}\ \neg\mathsf{Aquila}\ .$

$\mathsf{Eagle}\ \mathrm{owl{:}sameAs}\ \mathsf{Aquila}.$

In the contextual approach, since Eagle and Aquila as concepts and as individuals are independent, $\mathcal{O}_1$ is satisfiable. In the RDFS(FA) approach,[15] ontology $\mathcal{O}_1$ looks like:

$\mathtt{Harry}\ \mathrm{fa{:}type}_1\ \mathsf{Eagle}\ ;\ \mathrm{fa{:}type}_1\ \neg\mathsf{Aquila}\ .$

$\mathsf{Eagle}\ \mathrm{owl{:}sameAs}\ \mathsf{Aquila}.$

$\mathcal{O}_1$ is unsatisfiable because the meta-individual equality axiom [$\mathsf{Eagle}\ \mathrm{owl{:}sameAs}\ \mathsf{Aquila}$.] implies two classes Eagle and Aquila are equivalent, and $\mathtt{Harry}^{\mathcal{I}}$ cannot be both in and not in $\mathsf{Eagle}^{\mathcal{I}}$. In other words, given the following ontology $\mathcal{O}_2$:

$\mathtt{Harry}\ \mathrm{fa{:}type}_1\ \mathsf{Eagle}\ .$

$\mathsf{Eagle}\ \mathrm{owl{:}sameAs}\ \mathsf{Aquila}.$

In the RDFS(FA) approach, $\mathcal{O}_2$ entails the RDF triple [$\mathtt{Harry}\ \mathrm{fa{:}type}_1\ \mathsf{Aquila}$.]; in the

[15]To be more precise, we need OWL FA [43] to represent $\mathcal{O}_1$ in the RDFS(FA) approach.

contextual approach, the triple [Harry $\mathrm{rdf:type}$ Aquila .] is not entailed by $\mathcal{O}_2$.

It is also worth noting that there exist some languages, including HiLOG ([8], [59]), SKIF [20], Lbase [17] and Common Logic [14], which have a non-standard model theory, with predicates (such as classes and properties) elements in the domain. They differ from RDF(S) in that classes are treated as unary predicates, with their extensions being subsets of the domain, and reflection on language syntax is not supported [26]. Motik [36] proposes two alternative metamodeling approaches for OWL DL, i.e., the contextual approach (discussed above) and the HiLog approach. Details of the differences between these two metamodeling architectures and the metamodeling architecture of RDFS(FA) are summarised in [43].

VIII. Conclusion and Outlook

Semantic interoperability among SW languages is an important feature in knowledge engineering in the Semantic Web era. After showing in detail the syntactic and semantic mismatches between RDF(S) and OWL DL, we have proposed the RDFS(FA) ontology language as a sub-language of RDF(S), specifying both its semantics (including both datatypes and annotation properties) and the kinds of axioms that it provides.

As we have shown in previous sections, RDFS(FA) satisfies the four requirements we presented at the beginning of the paper. It covers many useful features of RDF(S), and is compatible with OWL DL (cf. Theorem 7). The aim of the Semantic Web is to provide a common framework that allows data to be shared and reused across applications and enterprises. As a strong connection between RDF(S) and OWL DL, RDFS(FA) can play a useful role in the Semantic Web. It has been proved that it is *impossible* to extend RDF(S) to first order logic if we want to have a coherent semantics based on RDF MT [46]; having RDFS(FA) as a sub-language of RDF(S), therefore, will surely solidify RDF(S)'s proposed role as the foundation of the Semantic Web. This establishes two strong connections between RDF and OWL; i.e., RDFS to OWL Full, and RDFS(FA) to OWL DL. One possible way forward would be to keep both connections, allowing users to decide if they are willing to use the layering style of RDFS(FA) in return for the benefits of remaining within a decidable sub-language of OWL.

Future work could include many interesting applications in knowledge engineering. From

the knowledge representation perspective, it would be possible to have a new sub-language of OWL — OWL FA [43], which would add useful features of RDFS(FA), such as meta-classes and meta-properties, into OWL DL. In [43], we also proposed a reason technique for use with RDFS(FA) and OWL FA. From the knowledge maintenance perspective, one could implement a plug-in for an ontology editor so as to allow users to switch their ontologies between the RDFS and RDFS(FA) modes. Extensions of this work could facilitate communication between RDFS(FA)-agents and OWL DL-agents. Last but not least, from the knowledge access perspective, query answering in RDFS(FA) and OWL FA remains an open problem.

REFERENCES

[1] F. Baader and W. Nutt. Basic description logics. In Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors, *The Description Logic Handbook: Theory, Implementation, and Applications*, pages 43–95. Cambridge University Press, 2003.

[2] Sean Bechhofer, Frank van Harmelen, James Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein eds. OWL Web Ontology Language Reference. http://www.w3.org/TR/owl-ref/, Feb 2004.

[3] R. Benjamins and D. Fensel. The Ontological Engineering Initiative $(KA)^2$. 1998.

[4] S. Bergamaschi, S. Castano, S. De Capitani di Vimercati, S. Montanari, and M. Vincici. An Intelligent Approach to Information Integration. 1998.

[5] Dan Brickley. OWL CR feedback: owl:Class 'vs' rdfs:Class causing pain. Is owl:Class really needed? URL `http://lists.w3.org/Archives/Public/public-webont-comments/2003Sep/0009.html`, Sept 2003. Discussion in the `public-webont-comments@w3.org` mailing list.

[6] Dan Brickley and R.V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. URL `http://www.w3.org/TR/rdf-schema/`, Feb 2004. Series Editor: Brian McBride.

[7] J. Broekstra, M. Klein, S. Decker, D. Fensel, F. van Harmelen, and I. Horrocks. Enabling knowledge representation on the web by extending rdf schema. Hong Kong, May 2001.

[8] Weidong Chen, Michael Kifer, and David S. Warren. Hilog: a foundation for higher-order logic programming. *J. Log. Program.*, 15(3):187–230, 1993.

[9] Alain Coucho. Improving Web Searching Using Descriptive Graphs. In *Natural Language Processing and Information Systems: 9th International Conference on Applications of Natural Language to Information Systems, NLDB 2004, Salford, UK, June 23-25, 2004.*, 2004.

[10] S. Cranefield and M. Purvis. UML as an ontology modelling language. In *IJCAI-99 Workshop on Intelligent Information Integration*, 1999.

[11] DCMI. Dublin Core Metadata Element Set, Version 1.1: Reference Description. DCMI Recommendation, `URL http://dublincore.org/documents/dces/`, June 2003.

[12] Jos de Bruijn, Enrico Franconi, and Sergio Tessaris. Logical reconstruction of normative RDF. In *OWL: Experiences and Directions Workshop (OWLED-2005)*, 2005.

[13] FaCT++. `http://owl.man.ac.uk/factplusplus/`, 2003.

[14] Common Logic Working Group. Common logic standard.

[15] T. R. Gruber. Towards Principles for the Design of Ontologies Used for Knowledge Sharing. In N. Guarino and R. Poli, editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation*, Deventer, The Netherlands, 1993. Kluwer Academic Publishers.

[16] Nicola Guarino. Semantic matching: Formal ontological distinctions for information organization, extraction, and integration. In Maria Teresa Pazienza, editor, *Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology, International Summer School, SCIE-97, Frascati, Italy*, pages 139–170, 1997.

[17] Ramanathan V. Guha and Pat Hayes. LBase: Semantics for Languages of the Semantic Web. Technical report, W3C Note, Jan 2003. http://www.w3.org/TR/2003/NOTE-lbase-20030123/.

[18] Volker Haarslev and Ralf Möller. RACER System Description. volume 2083, 2001.

[19] Patrick Hayes. RDF Semantics. Technical report, W3C, Feb 2004. W3C recommendation, http://www.w3.org/TR/rdf-mt/.

[20] Patrick Hayes and Chris Menzel. A semantics for the knowledge interchange format. Aug 2001.

[21] I. Horocks, D.Fensel, J.Broestra, S.Decker, M.Erdmann, C.Goble, F.van Harmelen, M.Klein, S.Staab, R.Studer, and E.Motta. The Ontology Inference Layer OIL. Technical report, OIL technical report, Aug. 2000.

[22] I. Horrocks. Using an Expressive Description Logic: FaCT or Fiction? pages 636–647, 1998.

[23] I. Horrocks, U. Sattler, and S. Tobies. Practical Reasoning for Expressive Description Logics. In H. Ganzinger, D. McAllester, and A. Voronkov, editors, *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, number 1705 in Lecture Notes in Artificial Intelligence, pages 161–180. Springer-Verlag, 1999.

[24] Ian Horrocks and Peter F. Patel-Schneider. The generation of DAML+OIL. pages 30–35. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-49/`, 2001.

[25] Ian Horrocks and Peter F. Patel-Schneider. Reducing OWL entailment to description logic satisfiability. In Dieter Fensel, Katia Sycara, and John Mylopoulos, editors, *Proc. of the 2003 International Semantic Web Conference (ISWC 2003)*, number 2870 in Lecture Notes in Computer Science, pages 17–29. Springer, 2003.

[26] Ian Horrocks and Peter F. Patel-Schneider. Three theses of representation in the semantic web. In *Proc. of the Twelfth International World Wide Web Conference (WWW 2003)*, pages 39–47. ACM, 2003.

[27] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.

[28] KAON2. `http://kaon2.semanticweb.org/`, 2005.

[29] Atanas Kiryakov, Borislav Popov, Damyan Ognyanoff, Dimitar Manov, Angel Kirilov, and Miroslav Goranov. Semantic Annotation, Indexing, and Retrieval. 2003.

[30] Graham Klyne and Jeremy J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. URL `http://www.w3.org/TR/rdf-concepts/`, Feb 2004. Series Editor: Brian McBride.

[31] Andreas Maier, Hans-Peter Schnurr, and York Sure. Ontology-based information integration in the automotive industryy. 2003.

[32] Frank Manola and Eric Miller. RDF Primer, W3C Recommendation. URL `http://www.w3.org/TR/rdf-primer/`, Feb 2004. Series Editor: Brian McBride.

[33] D.L. McGuinness. Ontological Issues for Knowledge-Enhanced Search. 1998.

[34] E. Mena, V. Kashyap, A. Illarramendi, and A. Sheth. Domain Specific Ontologies for Semantic Information Brokering on the Global Information Infrastructure. 1998.

[35] G. Miller. WordNet: A Lexical Database for English. *Comm. ACM*, 38(11), November 1995.

[36] Boris Motik. On the Properties of Metamodeling in OWL. In *Proc. of the Fourth International Semantic Web Conference (ISWC2005)*, 2005.

[37] W. Nejdl, M. Wolpers, and C. Capella. The RDF Schema Specification Revisited. In *Modelle und Modellierungssprachen in Informatik und Wirtschaftsinformatik, Modellierung 2000*, Apr. 2000.

[38] Natasha Noy. Representing Classes As Property Values on the Semantic Web. W3C Working Draft, URL `http://www.w3.org/TR/2004/WD-swbp-classes-as-values-20040721/`, Jul. 2004.

[39] Jeff Z. Pan. *Description Logics: Reasoning Support for the Semantic Web*. PhD thesis, School of Computer Science, The University of Manchester, 2004.

[40] Jeff Z. Pan and Ian Horrocks. Metamodeling Architecture of Web Ontology Languages. In *Proceeding of the Semantic Web Working Symposium (SWWS)*, July 2001.

[41] Jeff Z. Pan and Ian Horrocks. RDFS(FA) and RDF MT: Two Semantics for RDFS. In *Proc. of the 2nd International Semantic Web Conference (ISWC2003)*, 2003.

[42] Jeff Z. Pan and Ian Horrocks. OWL-Eu: Adding Customised Datatypes into OWL. In *Proc. of Second European Semantic Web Conference (ESWC 2005)*, 2005.

[43] Jeff Z. Pan, Ian Horrocks, and Guus Schreiber. OWL FA: A Metamodeling Extension of OWL DL. In *Proc. of the First International OWL Experience and Directions Workshop (OWLED-2005)*, 2005.

[44] Peter F. Patel-Schneider. Layering the Semantic Web: Problems and Directions. . In *2002 International Semantic Web Conference*, Jun 2002.

[45] Peter F. Patel-Schneider. Two Proposals for a Semantic Web Ontology Language. In *2002 International Description Logic Workshop*, Apr 2002.

[46] Peter F. Patel-Schneider. Building the Semantic Web Tower from RDF Straw. In *In Proc. of. the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, 2005. To appear.

[47] Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks. OWL Web Ontology Language Semantics and Abstract Syntax. Technical report, W3C working draft, Mar. 2003.

[48] Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks. OWL Web Ontology Language Semantics and Abstract Syntax. Technical report, W3C, Feb. 2004. W3C Recommendation.

[49] Pellet. `http://www.mindswap.org/2003/pellet/`, 2003.

[50] PICS. Platform for Internet Content Selectivity. `http://www.w3.org/PICS/`, 1997.

[51] A. Rector. Re: [UNITS, OEP] FAQ : Constraints on data values range, Apr. 2004. http://lists.w3.org/Archives/Public/public-swbp-wg/2004Apr/0216.html.

[52] Nenad Stojanovic. On the Query Refinement in the Ontology-Based Searching for Information. In *WM 2003: Professionelles Wissesmanagement - Erfahrungen und Visionen, Beiträge der 2. Konferenz Professionelles Wissensmanagement, April 2003 in Luzern*, 2003.

[53] Herman J. ter Horst. Extending the rdfs entailment lemma. In *International Semantic Web Conference*, pages 77–91, 2004.

[54] Herman J. ter Horst. Completeness, decidability and complexity of entailment for rdf schema and a semantic extension involving the owl vocabulary. 3(2), 2005.

[55] M. Uschold and M. Gruninger. Ontologies: Principles, Methods and Applications. *The Knowledge Engineering Review*, 1996.

[56] Frank van Harmelen, Peter F. Patel-Schneider, and Ian Horrocks. A Model-Theortis Semantics of DAML+OIL(March 2001). Mar. 2001.

[57] Christopher A. Welty. The Ontological Nature of Subject Taxonomies. 1998.

[58] Jan Wielemaker, Guus Schreiber, and Bob J. Wielinga. Prolog-based Infrastructure for RDF: Scalability and Performance. In *Proc. of the 2nd International Semantic Web Conference (ISWC2003)*, 2003.

[59] Guizhen Yang and Michael Kifer. Reasoning about anonymous resources and meta statements on the semantic web. *Journal on Data Semantics*, 1:69–97, 2003.