Resource Description Framework

Jeff Z. Pan

University of Aberdeen, Aberdeen, UK, jpan@csd.abdn.ac.uk

Summary. This chapter introduces Resource Description Framework (RDF), the W3C recommendation for semantic annotations in the Semantic Web. It will cover the syntax and semantics of RDF, as well as its relation with the W3C OWL Web Ontology Language. To address the mismatch between RDF and OWL-DL, the most expressive decidable fragment of the OWL standard, we introduce a novel variant of RDF(S), called RDFS-FA, which provides a solid semantic foundation for many of the latest Description Logic-based SW ontology languages, such as OWL-DL and OWL2-DL.

1 Introduction: Heading for the Semantic Web

In *Realising the Full Potential of the Web* [2], Tim Berners-Lee identifies two major objectives that the Web should fulfil. The first goal is to enable people to work together by allowing them to share knowledge. The second goal is to incorporate tools that can help people analyse and manage the information they share in a meaningful way. This vision has become known as the *Semantic Web* (SW) [3].

The Web's provision to allow people to write online content for other people is an appeal that has changed the computer world. This same feature that is responsible for fostering the first goal of the Semantic Web, however, hinders the second objective. Much of the content on the existing Web, the so-called *syntactic Web*, is human but not machine readable. Furthermore, there is great variance in the quality, timeliness and relevance [2] of Web resources (i.e. Web pages as well as a wide range of Web accessible data and services) that makes it difficult for programs to evaluate the worth of a resource.

The vision of the Semantic Web is to augment the syntactic Web so that resources are more easily interpreted by programs (or 'intelligent agents'). The enhancements will be achieved through the *semantic markups* which are machine-understandable annotations associated with Web resources.

Encoding semantic markups will necessitate the Semantic Web adopting an annotation language. To this end, the W3C (World Wide Web Consortium)

S. Staab and R. Studer (eds.), *Handbook on Ontologies,* International Handbooks on Information Systems, DOI 10.1007/978-3-540-92673-3,

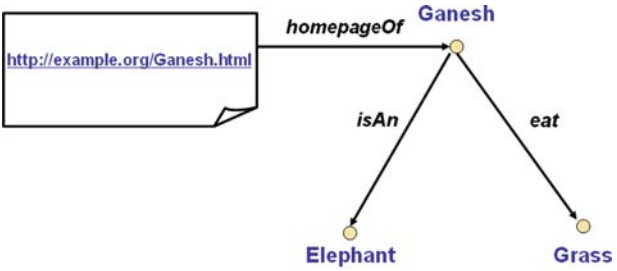

Fig. 1. RDF annotations in a directed labeled graph

community has developed a recommendation called resource description framework (RDF) [13]. The development of RDF is an attempt to support effective creation, exchange and use of annotations on the Web.

Example 1. Annotating Web Resources in RDF

As shown in Fig. 1, we can associate an RDF annotation[1] to `http://example.org/Ganesh.html` and state that it is the homepage of the resource Ganesh, which is an elephant and eats grasses.

We invite the reader to note that the above RDF annotations are different from HTML [27] mark-ups in that they describe the contents of Web resources, instead of the presentation of Web pages.

Annotations alone do not establish the semantics of what is being marked-up. For example, the annotations presented in Fig. 1 do not explain what elephants mean. The rest of the chapter is organised as follows. Section 2 presents RDF and two ways of providing semantics to RDF annotations. Section 3 introduces RDF Schema (or RDFS for short) and its semantics. Section 4 explains the semantic mismatch between RDF(S) and OWL-DL, while Sect. 5 introduces a sub-language of RDF, called RDFS-FA, which on the one hand has a semantics that is compatible with OWL-DL and on the other hand still allows meta-classes and meta-properties. Section 6 concludes the chapter.

2 Annotation and Meaning

The vision of the Semantic Web is to make Web resources (not just HTML pages, but a wide range of Web accessible data and services) more understandable to machines. Machine-understandable annotations are, therefore, introduced to describe the content and functions of Web resources.

[1] See Sect. 2 for precise definitions of RDF syntax.

2.1 RDF

RDF [13] as a W3C recommendation provides a *data model* for annotations in the Semantic Web. It is built upon earlier developments such as the Dublin Core (see Sect. 2.2) and the platform for Internet content selectivity (PICS) [26] content rating initiative.

An RDF statement (or RDF triple) is of the form:

$$\text{subject property object.} \tag{1}$$

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 call an *RDF graph*.

To represent RDF statements in a machine-processable way, RDF defines a specific extensible markup language (XML) syntax, referred to as RDF/XML [14]. RDF-annotated resources (i.e. subjects) are usually named by Uniform Resource Identifier references. Uniform resource identifiers (URIs) are strings that identify Web resources [7]. Uniform resource locators (URLs) are a particular type of URIs, i.e. those have network locations. A *URI reference* (or URIref) is a URI, together with an optional fragment identifier at the end. For example, the URI reference `http://www.example.org/Elephant#Ganesh` consists of the URI `http://www.example.org/Elephant` and (separated by the `#` character) the fragment identifier `Ganesh`. As a convention, *name spaces*, which are sources where multiple resources are from, are (usually) URIs with the `#` character. For example, `http://www.example.org/Elephant#` is a name space. Resources without URIrefs are called *blank nodes*; a blank node indicates the existence of a resource, without explicitly mentioning the URIref of that resource. A *blank node identifier*, which is a local identifier, can be used to allow several RDF statements to reference the same blank node. As RDF/XML is verbose, in this chapter, we use the Notation 3 (or N3) syntax of RDF, where each RDF statement is of the form (1). Figure 2 shows an RDF graph in N3 syntax, where the '@prefix' introduces shorthand identifications (such as 'ex:') of XML namespaces and a semicolon ';' introduces another property of the same subject. In these statements, the

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix ex:  <http://example.org/#>
@prefix elp: <http://example.org/Animal#>

elp:Ganesh ex:mytitle "A resource called Ganesh" ;
               ex:mycreator "Pat Gregory" ;
               ex:mypublisher _:b1 .
_:b1 elp:name "Elephant United" .
```

Fig. 2. RDF statements

annotated resource is elp:Ganesh, which is annotated with three properties ex:mytitle, ex:mycreator and ex:mypublisher. Note that _:b1 is a blank node identifier.

Given that RDF alone does not specify the intended meaning for Web resources, how do we provide meaning to Web resources through annotations? The meaning comes either from pre-agreed informal semantics, e.g. from Dublin Core, or from ontologies.

2.2 Dublin Core

One way of giving meaning to annotations is to provide some pre-agreed informal semantics for a set of information properties. For example, the Dublin Core Metadata Element Set [5] provides 15 'core' information properties, such as 'Title', 'Creator', 'Date', with descriptive semantic definitions (in natural language). One can use these information properties in, e.g. RDF or META tags of HTML.

If we replace the properties ex:mytitle, ex:mycreator and ex:mypublisher used in Fig. 2 with dc:title, dc:creator and dc:publisher as shown in Fig. 3, Dublin Core compatible intelligent agents can then understand that the title of the Web resource is 'A resource called Ganesh', and the creator is Pat Gregory. This is not possible for the RDF statements in Fig. 2 because, in general, users may use arbitrary names for the title, creator and publisher properties, etc.

The limitation of the 'pre-agreed informal semantics' approach is its inflexibility, i.e. only a limited range of pre-agreed information properties can be expressed.

2.3 Ontology

An alternative approach is to use ontologies to specify the meaning of Web resources. *Ontology* is a term borrowed from philosophy that refers to the science of describing the kinds of entities in the world and how they are

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix dc:  <http://purl.org/dc/elements/1.1/>
@prefix elp: <http://example.org/Animal#>

elp:Ganesh dc:title "A resource called Ganesh" ;
               dc:creator "Pat Gregory" ;
               dc:publisher _:b1 .
_:b1 elp:name "Elephant United" .
```

Fig. 3. Dublin core properties in RDF statements

related. In computer science, ontology is, in general, a 'representation of a shared conceptualisation' of a specific domain [8, 30]. It provides a shared and common *vocabulary*, including important concepts, properties and their definitions, and *constraints*, sometimes referred to as background assumptions regarding the intended meaning of the vocabulary, used in a domain that can be communicated between people and heterogeneous, distributed application systems.

The ontology approach is more flexible than the pre-agreed informal semantics approach because users can customise vocabulary and constraints in ontologies. For example, applications in different domains can use different ontologies. Typically, ontologies can be used to specify the meaning of Web resources (through annotations) by asserting resources as instances of some important concepts and/or asserting resources relating to resources by some important properties defined in ontologies.

Ontologies can be expressed in Description Logics. An ontology usually corresponds to a TBox in Description Logics (see chapter "Description Logics"). Vocabulary in an ontology can be expressed by named concepts and roles, and concept definitions can be expressed by equivalence introductions. Background assumptions can be represented by general concept and role axioms. Sometimes, an ontology corresponds to a DL knowledge base. For example, in the OWL Web ontology language to be introduced in chapter "Web Ontology Language: OWL," an ontology also contains instances of important concepts and relationships among these instances, which can be represented by DL assertions. In the rest of the chapter, we will introduce RDF Schema (RDFS), an ontological schema language, and a novel modification of RDF(S) as a semantic foundation for many of the latest Description Logics-based SW ontology languages, including OWL-DL and OWL 1.1.

3 RDFS: A Web Ontological Schema Language

Following W3C's 'one small step at a time' strategy, RDFS can be seen as a first try to support expressing simple ontologies with RDF syntax. In RDFS, predefined Web resources rdfs:Class, rdfs:Resource and rdf:Property can be used to define classes (concepts), resources and properties (roles), respectively.

Unlike Dublin Core, RDFS does not predefine information properties but a set of meta-properties that can be used to represent background assumptions in ontologies:

- rdf:type: the instance-of relationship
- rdfs:subClassOf: the property that models the subsumption hierarchy between classes
- rdfs:subPropertyOf: the property that models the subsumption hierarchy between properties

```
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
@prefix elp:  <http://example.org/Animal#>

elp:Animal rdf:type rdfs:Class .
elp:Habitat rdf:type rdfs:Class .
elp:Elephant rdf:type rdfs:Class ; rdfs:subClassOf elp:Animal .
elp:liveIn rdf:type rdf:Property ;
                rdfs:domain elp:Animal ; rdfs:range elp:Habitat .

elp:south-sahara rdf:type elp:Habitat .
elp:Ganesh rdf:type elp:Elephant ; elp:liveIn elp:south-sahara .
```

Fig. 4. An RDFS ontology

- rdfs:domain: the property that constrains all instances of a particular property to describe instances of a particular class
- rdfs:range: the property that constrains all instances of a particular property to have values that are instances of a particular class

RDFS statements are simply RDF triples; viz. RDFS provides no syntactic restrictions on RDF triples. Figure 4 shows an animal ontology in RDFS; it has three classes, i.e. `elp:Animal`, `elp:Habitat` and `elp:Elephant` (which is rdfs:subClassOf `elp:Animal`), and a property `elp:`*liveIn*, the rdfs:domain and rdfs:range of which are `elp:Animal` and `elp:Habitat`, respectively. In addition, it states that the resource `elp:Ganesh` is an instance of `elp:Elephant`, and that it `elp:`*liveIn*s an `elp:Habitat` called `elp:south-sahara`.

At a glance, RDFS is a simple ontological schema langauge that supports only class and property hierarchies, as well as domain and range constraints for properties. According to the RDF Model Theory (RDF MT) to be explained in Sect. 3.2, however, it is more complicated than that (see Proposition 1 on page 79).

3.1 RDF(S) Datatyping

RDF(S) provides a specification of datatypes and data values; accordingly, it allows the use of datatypes defined by any external type systems, e.g. the XML Schema type system, which conform to this specification.

Definition 1. (Datatype) *A* datatype d *is characterised by a lexical space,* $L(d)$*, which is a non-empty set of Unicode strings; a value space,* $V(d)$*, which is a non-empty set, and a total mapping* $L2V(d)$ *from the lexical space to the value space.*

For example, *boolean* is a datatype with value space $\{true, false\}$, lexical space {"T", "F","1","0"} and lexical-to-value mapping {"T"$\mapsto$ $true$, "F"$\mapsto$ $false$, "1"$\mapsto$ $true$, "0"$\mapsto$ $false$}.

Definition 2. (Typed and Plain Literals) Typed literals *are of the form "v"^^u, where v is a Unicode string, called the* lexical form *of the typed literal, and u is a URI reference of a datatype.* Plain literals *have a lexical form and optionally a* language tag *as defined by [1], normalised to lowercase.*

The denotation of a typed literal is the value mapped from its enclosed Unicode string by the lexical-to-value mapping of the datatype associated with its enclosed datatype URIref. For example, "1"^^xsd:boolean is a typed literal that represents the boolean value *true*, while "1"^^xsd:integer represents the integer 1. Plain literals, e.g. "1", are considered to denote themselves [9].

The associations between datatype URI references (e.g. xsd:boolean) and datatypes (e.g. *boolean*) can be provided by datatype maps defined as follows.

Definition 3. (Datatype Map) *We consider a datatype map* $\mathbf{M}_d$ *that is a partial mapping from datatype URI references to datatypes.*

Example 2. Datatype Map $\mathbf{M}_{d1} = \{\langle \text{xsd:string}, \textit{string}\rangle, \langle \text{xsd:integer}, \textit{integer}\rangle\}$ is a datatype map, where xsd:string and xsd:integer are datatype URI references, and *string* and *integer* are datatypes. ♢

A datatype map may include some built-in XML Schema datatypes (as seen in Example 2), while other built-in XML Schema datatypes are problematic and thus unsuitable for various reasons. For example, xsd:ENTITIES is a list-value datatype that does not fit the RDF datatype model.[2] Please note that derived XML Schema datatypes are not RDF(S) datatypes, because there is no standard way to access a derived XML Schema datatype through a URI reference. Therefore, there is no way to include a derived XML Schema datatype in a datatype map.

3.2 RDF Model Theory

RDF MT provides semantics not only for RDFS ontologies, but also for RDF triples. RDF MT is built on *simple interpretations*. To simplify presentations, in this chapter we do not cover blank nodes, which are identified by local identifiers instead of URIrefs.

Definition 4. *(***Simple Interpretation***) Given a set of URI references* $\mathbf{V}$*, a* simple interpretation *I of* $\mathbf{V}$ *in the RDF model theory is defined by:*

- *A non-empty set* $\mathbf{IR}$ *of resources, called the* domain *(or* universe*) of I*
- *A set* $\mathbf{IP}$*, called the* set of properties *in I*
- *A mapping IEXT, called the* extension function*, from* $\mathbf{IP}$ *to the powerset of* $\mathbf{IR} \times \mathbf{IR}$
- *A mapping IS from URIrefs in* $\mathbf{V}$ *to* $\mathbf{IR} \cup \mathbf{IP}$

[2] See the RDF semantics document [9] for the complete list of RDF(S) built-in datatypes.

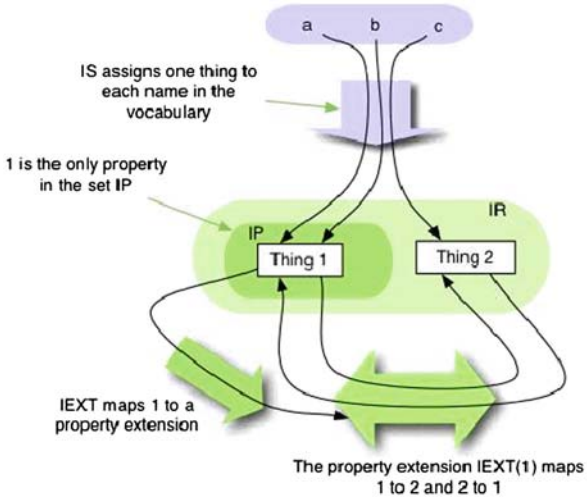

Fig. 5. A simple interpretation of $\mathbf{V} = \{\text{a,b,c}\}$ (from [9])

Given a triple [s p o .], $I([\text{s p o .}]) = true$ *if* $s,p,o \in \mathbf{V}$, $IS(\text{p}) \in \mathbf{IP}$, *and* $\langle IS(\text{s}), IS(\text{o})\rangle \in IEXT(IS(\text{p}))$*; otherwise,* $I([\text{s p o .}]) = false$.

Given a set of triples S, $I(S) = false$ *if* $I([\text{s p o .}]) = false$ *for some triple* [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.*

Note that Definition 4 does not specify the relationship between **IP** and **IR**, i.e. **IP** may or may not be disjoint with **IR**. Figure 5 presents a simple interpretation I of $\mathbf{V} = \{\text{a,b,c}\}$, where the URIref b is simply interpreted as a property because $IS(\text{b}) = 1 \in \mathbf{IP}$, and $IEXT(IS(\text{b}))$, the extension of $IS(\text{b})$, is a set of pairs of resources that are in **IR**, i.e. $\{\langle 1,2\rangle,\langle 2,1\rangle\}$. Since $\langle IS(\text{a}), IS(\text{c})\rangle \in IEXT(IS(\text{b}))$, $I([\text{a b c .}]) = true$; hence, we can conclude that I satisfies [a b c .].

The semantics of RDF triples is given in terms of RDF-Interpretations.

Definition 5. *(***RDF-Interpretation***) Given a set of URI references* **V** *and the set* **rdfV***, called the* RDF vocabulary*, of URI references in the rdf: namespace, an RDF-interpretation of* **V** *is a simple interpretation I of* $\mathbf{V} \cup \mathbf{rdfV}$ *that satisfies:*

1. *For* $p \in \mathbf{V} \cup \mathbf{rdfV}$, $IS(\text{p}) \in \mathbf{IP}$ *iff* $\langle IS(\text{p}), IS(\text{rdf:Property})\rangle \in IEXT(IS(\text{rdf:type}))$
2. *All the RDF axiomatic statements*[3]

Condition 1 of Definition 5 implies that each member of **IP** is a resource in **IR**, due to the definition of $IEXT$ in Definition 4; in other words, RDF-interpretations require **IP** to be a subset of **IR**. RDF axiomatic statements

[3] Readers are referred to [9] for the list of the RDF axiomatic statements.

mentioned in Condition 2 are RDF statements about RDF built-in vocabularies in **rdfV**; e.g. [rdf:type rdf:type rdf:Property.] is an RDF axiomatic statement. According to Definition 5, any RDF-interpretation I should satisfy [rdf:type rdf:type rdf:Property.], viz. IS(rdf:type) should be in **IP**.

Finally, the semantics of RDFS statements written in RDF triples is given in terms of RDFS-Interpretations.

Definition 6. *(***RDFS-Interpretation***) Given* **rdfV***, a set of URI references* **V** *and the set* **rdfsV***, called the* RDFS vocabulary*, of URI references in the rdfs: namespace, an RDFS-interpretation I of* **V** *is an RDF-interpretation of* $\mathbf{V} \cup \mathbf{rdfV} \cup \mathbf{rdfsV}$ *which introduces:*

- *A set* **IC***, called the set of classes in I*
- *A mapping ICEXT (called the* class extension function*) from* **IC** *to the set of subsets of* **IR**

and satisfies the following conditions (let x,y,u,v be URIrefs in $\mathbf{V} \cup \mathbf{rdfV} \cup \mathbf{rdfsV}$*)*[4]*:*

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

and satisfies all the RDFS axiomatic statements.[5]

Condition 1 indicates that a 'class' is not a strictly necessary but convenient semantic construct [9] because the class extension function $ICEXT$ is simply 'syntactic sugar' and is defined in terms of $IEXT$. Handling classes in this way can be counter-intuitive (cf. Proposition 1). Condition 2 to 8 are about RDFS meta-properties rdfs:domain, rdfs:range, rdfs:subPropertyOf and rdfs:subClassOf. Condition 9 ensures that all classes are sub-classes of rdfs:Resource.

Proposition 1. *The RDFS statements* [rdfs:Resource rdf:type rdfs:Class .] *and* [rdfs:Class rdfs:subClassOf rdfs:Resource.] *are always true in all RDFS-interpretations.*

[4] We only focus on the core RDFS primitives, i.e. the RDFS predefined meta-properties introduced on page 75.

[5] Again, readers are referred to [9] for a list of the RDFS axiomatic statements, which includes, e.g. [rdf:type rdfs:range rdfs:Class.].

Proof. For [rdfs:Resource rdf:type rdfs:Class.]:

1. According to the definition of IS and Definition 5, for any resource x, we have $IS(\text{x}) \in \mathbf{IR}$. Due to $\mathbf{IR} = ICEXT(IS(\text{rdfs:Resource}))$ and Condition 1 in Definition 6, $\langle IS(\text{x}), IS(\text{rdfs:Resource})\rangle \in IEXT(IS(\text{rdf:type}))$. Since rdf:Property is a built-in resource, we have $\langle IS(\text{rdf:Property}), IS(\text{rdfs:Resource})\ \rangle \in IEXT(IS(\text{rdf:type}))$.
2. Due to [rdf:type rdfs:range rdfs:Class.] (an RDFS axiomatic statement), $\langle IS(\text{rdf:Property}), IS(\text{rdfs:Resource})\rangle \in IEXT(IS(\text{rdf:type}))$ and Condition 4 in Definition 6, we have $IS(\text{rdfs:Resource}) \in ICEXT(IS())$ rdfs:Class. Therefore, for any RDFS-interpretation I, we have I $\models$ [rdfs:Resource rdf:type rdfs:Class.].

For [rdfs:Class rdfs:subClassOf rdfs:Resource .]: According to the definition of **IC**, every class is its member, including IS(rdfs:Class), viz.$IS()$ rdfs:Class $\in$ **IC**. Due to Condition 9 of Definition 6, $\langle IS(\text{rdfs:Class}), IS(\text{rdfs:Resource})\rangle \in IEXT\ (IS(\text{rdfs:subClassOf}))$; hence, for any RDFS-interpretation I, we have I $\models$ [rdfs:Class rdfs:subClassOf rdfs:Resource.] □

The two RDFS statements in Proposition 1 suggest a strange situation for rdfs:Class and rdfs:Resource as discussed in [18]: 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? Users may find this counter-intuitive and thus hard to understand – this is why we say that RDF(S) is more complicated than it appears. We will address this issue in Sect. 5.

Now we define RDFS-interpretations w.r.t. a datatype map $\mathbf{M}_d$.

Definition 7. (RDFS $\mathbf{M}_d$-Interpretation) *Given a datatype map* $\mathbf{M}_d$*, an* RDFS $\mathbf{M}_d$-interpretation *I of a vocabulary* $\mathbf{V}$ *is any RDFS-interpretation of* $\mathbf{V} \cup \{u \mid \exists\ d.\langle u, d\rangle \in \mathbf{M}_d\}$ *which introduces*

- *A distinguished subset* **LV** *of* **IR***, called the* set of literal values*, which contains all the plain literals in* $\mathbf{V}$
- *A mapping IL from typed literals in* $\mathbf{V}$ *into* **IR**

and satisfies the following extra conditions:

1. $\mathbf{LV} = ICEXT(IS(\text{rdfs:Literal}))$
2. *For each pair* $\langle u, d\rangle \in \mathbf{M}_d$
   (a) $ICEXT(d) = V(d) \subseteq \mathbf{LV}$
   (b) *There exist* $d \in \mathbf{IR}$ *s.t.* $IS(\text{u}) = d$
   (c) $IS(\text{u}) \in ICEXT(IS(\text{rdfs:Datatype}))$
   (d) *For* "s" ^^u' $\in \mathbf{V}$, $IS(\text{u}') = d$, *if* $s \in L(d)$, *then* $IL(\text{"}s\text{"}\ \hat{}\hat{}\ u') = L2S(d)(s)$, *otherwise,* $IL(\text{"}s\text{"}\ \hat{}\hat{}\ u') \notin \mathbf{LV}$,
3. *If* $d \in ICEXT(IS(\text{rdfs:Datatype}))$, *then* $\langle d, IS(\text{rdfs:Literal})\rangle \in IEXT(rdfs:subClassOf)$.

According to Definition 7, **LV** is a subset of **IR**; i.e. literal values are resources. Condition 1 ensures that the class extension of rdfs:Literal is **LV**. Condition 2) asserts that RDF(S) datatypes are classes, condition 2) ensures that there is a resource d for datatype d in $\mathbf{M}_d$, condition 2) ensures that the class rdfs:Datatype contains the datatypes used in any satisfying $\mathbf{M}_d$-interpretation, and condition 2) explains why the range of IL is **IR** rather than **LV** (because, for "s"^^u, if $s \notin L(IS(\mathrm{u}))$, then $IL(\text{"}s\text{"}\hat{}\hat{}u) \notin \mathbf{LV}$). Condition 3 requires that RDF(S) datatypes are sub-classes of rdfs:Literal.

If the datatypes in the datatype map $\mathbf{M}_d$ impose disjointness conditions on their value spaces, it is possible for an RDF graph to have no RDFS $\mathbf{M}_d$-interpretation which satisfies it, i.e. there exists a *datatype clash*. For example,

```
_:x rdf:type xsd:string.
_:x rdf:type xsd:decimal.
```

would constitute a datatype clash because the value spaces of xsd:string and xsd:decimal are disjoint. In RDF(S), an ill-typed literal does not in itself constitute a datatype clash, cf. Condition 2) in Definition 7, but a graph which entails that an ill-typed literal has rdf:type rdfs:Literal would be inconsistent.

Having described the semantics, we now briefly discuss reasoning in RDF(S). Entailment is the key inference problem in RDF(S), which can be defined on the basis of interpretations. Indeed, cRDF is impossible to express contradictions if we do not consider datatypes.

Definition 8. *(***RDF Entailments***)* *Given two sets of RDF statements S_1 and S_2, and a datamap $\mathbf{M}_d$, S_1* simply entails *(*RDF-entails, RDFS-entails, RDFS-$\mathbf{M}_d$-entails*) S_2 if all the simple interpretations (RDF-interpretations, RDFS-interpretations, RDFS $\mathbf{M}_d$-interpretation, resp.) of S_1 also satisfy S_2.*

4 Mismatch between RDF(S) and OWL-DL

This section describes the relation between RDF(S) and OWL-DL, which is a key sub-language of the standard (W3C recommendation) Web Ontology Langauge. One key question is whether it is possible to use an RDF(S) inference engine to do OWL-DL reasoning, or vice versa. The short answer is *no*, and this section explains why.

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 basically very expressive description logics (DLs). 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). OWL-Full is undecidable, because it combines the OWL expressivity with the meta-modelling architecture of RDF(S) [15].[6] Accordingly, OWL-DL is the most expressive decidable

[6] Another reason that OWL-Full is undecidable is that it does not impose restrictions on the use of transitive properties [12].

sub-language of OWL. More details of the OWL language can be found in chapter "Web Ontology Language: OWL."

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 agents, since they are syntactically ill formed. The RDF/XML syntax form of an OWL-DL ontology is *valid*, iff it can be translated (according to the mapping rules provided in [25]) 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 [11], since no inverse mapping is defined in the OWL specification.

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 [25], 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. Strictly speaking, 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.

OWL-Full seems to be a bridge between RDF(S) and OWL-DL; however, 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 [23]. 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 [11, 23, 24]. 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 [11], 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 [10]. 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 shows that OWL-Full disagrees with OWL-DL on valid OWL-DL ontologies. To partially address this issue, the OWL specification weakens the relations between OWL-DL and OWL-Full by claiming (with a sketched proof) that, given two OWL-DL ontologies O1 and O2, O1 entails O2 w.r.t. the OWL-DL semantics implies that O1 entails O2 w.r.t. the OWL-Full semantics. 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).

It should be noted that *for some* the above presentation of the three issues might be a little too negative about the situation w.r.t. OWL-Full and OWL-DL: the first two issues are difficulties that have, in theory, been claimed to be solved by the use of comprehension principles and restrictions on the syntactic form of OWL-DL's RDF serialisation. From this perspective, the main side effect of comprehension principles is that all OWL-Full models have infinite domains; hence, any OWL-DL ontologies that have only finite models are necessarily inconsistent when treated as OWL-Full ontologies. This leads to the third issue and demonstrates why, in the OWL specification, the relations between OWL-Full and OWL-DL is weakened.

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

In this section, we introduce *RDFS-FA* (RDFS with Fixed layered meta-modelling architecture), as a sub-language of RDF(S), to restore the desired connection between RDF(S) and OWL-DL. RDFS-FA addresses 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 meta-modelling architecture. Let us recall that RDFS has a non-layered meta-modelling 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 meta-modelling architecture; in practice, four strata (as shown in Fig. 6) are usually enough. The UML-like meta-modelling 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;[7] 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

[7] Classes can be regarded as mega-objects in upper strata of the meta-modelling architecture.

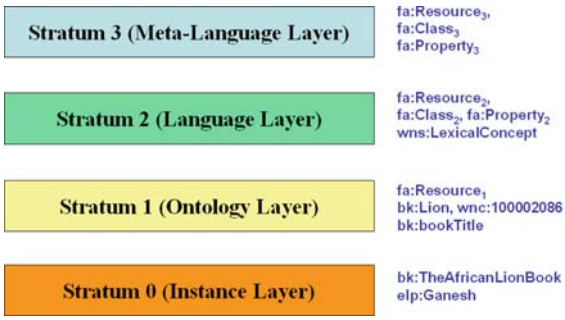

Fig. 6. The UML-like meta-modelling architecture (number of strata = 4) of RDFS-FA

```
@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. 7. An RDFS-FA ontology

whether to model something in the domain as a class or as an object (see also [17]). This concern suggests that users intuitively tend to assume that classes and objects should be different from each other. Therefore, layered meta-models could be more intuitive than non-layered meta-models.

Readers are referred to [21] for a formal introduction of RDFS-FA ontologies and their semantics. Informally speaking, an RDFS-FA ontology is a set of RDFS-FA axioms, which are basically RDF triples (in N3 syntax)[8] 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.

Figure 7 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

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

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.

We conclude this section by showing the interoperability between RDFS-FA and OWL-DL. It is much easier to layer OWL-DL, syntactically *and* semantically, on top of RDFS-FA than on top of RDF(S). In particular, there is a one-to-one bidirectional mapping (see [21] for details) 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 [25], 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 SubClassOf (C_1 D_1) to the RDF(S) statement [C_1 rdfs:subClassOf D_1.], 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). In the semantic level, it can be shown that the above bidirectional mapping is a semantics-preserving mapping [21].

It has be shown [22] that we can extend OWL DL with the meta-modelling architecture of RDFS-FA into OWL-FA, and that OWL-FA is also decidable.

6 Related Work

As earlier works [4, 16] pointed out, RDFS has a non-standard and non-fixed layer meta-modelling architecture, which makes some elements in the model have multiple roles in the RDFS specification. Therefore, it makes even the RDFS specification itself somehow confusing and difficult to understand for users. To clear up any confusion, Pan and Horrocks [18] proposed a Fixed layer meta-modelling Architecture for RDFS, reducing the multiple roles of RDFS built-in primitives by stratifying them into different layers of the meta-modelling architecture. Subsequently, the RDF Model Theory (RDF MT) [9] gave an official semantics for RDF and RDFS, justifying the dual roles by treating both classes and properties as objects in the universe. Pan

and Horrocks [19] suggested that RDFS could have two kinds of semantics, i.e. RDF MT and the stratified semantics of RDFS(FA).

Horst [29] extends RDF MT to cover some OWL constructors and axioms by proposing the so-called pD* semantics. Interestingly, the pD* semantics is in line with the 'if-semantics' of RDFS and weaker than the 'iff-semantics' that is used in the RDF-compatible semantic for OWL DL and OWL Full. One of the motivations of having the iff-semantics in the RDF-compatible semantic for OWL is to solve the 'too few entailment' problem [19]. Note that the iff-semantics is *not* relevant to the direct semantics of OWL DL. Among the 15 OWL URIs, the pD* interprets `owl:FunctionalProperty`, `owl:InverseFunctionalProperty`, `owl:SymmetricProperty` and `owl:Transi- tiveProperty` as the if conditions of the standard mathematical definitions. The `owl:inverseOf` is interpreted as that if two properties are owl:inverseOf-related, then their extensions are each other's inverse as binary relations. The pD* semantics requires that two classes are equivalent if and only if they are both subclasses of each other. `owl:equivelantProperty` is treated in a similar way to `owl:equivalentClass`. The pD* semantics interprets `owl:sameAs` as an equivalence relation.In particular, the pD* semantics includes the iff condition for `owl:hasValue`. But for `owl:someValueFrom` and `owl:allValueFrom`, the pD* semantics still includes half of OWL's iff conditions. If two classes are owl:disjointWith-related the pD* semantics requires their extensions are disjoint. The pD* semantics requires that the extensions of `owl:sameAs` and `owl:differentForm` are disjoint. Based on the pD* semantics discussed above, the corresponding pD* entailment rules are also given in [29]. It consists of 23 rules to illustrate that what conclusion can be deduced from some given premises. These rules are proved to be sound and complete with respect to the pD* semantics.

Patel-Shneider et al. [25] extended RDFS with OWL constructors to OWL Full, which keeps the meta-modelling architecture of RDFS. Motik [15] shows that the meta-modelling architecture of OWL Full contributes to its undecidability. Motik [15] also provides two alternative meta-modelling approaches for OWL DL, i.e. the contextual approach and the HiLog approach.

- In the context approach, the names for classes, properties and individuals are not distinct and are interpreted depending on the context; i.e. they are interpreted by class interpretation functions, property interpretation functions and individual interpretation functions, respectively. Intuitively speaking, this approach provides a 'two-layered' meta-modelling architecture, i.e. the instance layer and class layer. OWL FA provides a 'multi-layered' meta-modelling architecture. At a quick glance, the 'two-layered' and the 'multi-layered' meta-modelling architectures should be similar; however, the example we show later in this section indicates that they are quite different.

- The HiLog approach is closer to the spirit of OWL Full meta-modelling. It has a 'two-step' interpretation function for classes, which first maps symbols to resources in the domain and then maps these resources to a set of resources in the domain. Intuitively speaking, this approach provides a 'one-layered' meta-modelling architecture, in the sense that classes and individuals are both interpreted as resources in the domain. Note that it is dificult/impossible to map classes in the 'one-layered' meta-modelling architecture to the 'multi-layered' meta-modelling architectures such as that of MOF.

We now use an example in [15] to illustrate some of the differences among the above two approaches and our approach. Let us consider the following knowledge base[9] $\Sigma = \{ \texttt{Harry} :_1 \mathsf{Eagle}, \texttt{Harry} :_1 \neg\mathsf{Aquila}, \mathsf{Eagle} =_1 \mathsf{Aquila}\}$. In the contextual approach, since Eagle and Aquila as concepts and as individuals are independent, Σ is satisfiable. In the HiLog approach, it is not satisfiable because Eagle and Aquila are interpreted as the same object, let us call it a, and Harry cannot be both in and not in the concept extension of a. In OWL FA, Σ is unsatisfiable because the meta-individual equality axiom $\mathsf{Eagle} =_1 \mathsf{Aquila}$ indicates two concepts Eagle and Aquila are equivalent, and $\texttt{Harry}^{\mathcal{I}}$ cannot be both in and not in $\mathsf{Eagle}^{\mathcal{I}}$. This example indicates the contextual semantics (at least sometimes) is not as intuitive as the Hilog semantics and the FA semantics.

Let us conclude this section by briefly comparing the three approaches. In terms of syntax, the contextual and Hilog approaches seem to be more elegant in that they do not have to change the syntax of OWL DL, while the FA approach introduces strata numbers to facilitate the 'multi-layered' meta-modelling architecture. In terms of semantics, it seems that the FA approach is closer to the Hilog approach (according to the above example). It is an interesting peace of future work to investigate more detailed differences between the Hilog approach and the FA approach. In terms of computability, the FA approach is closer to the contextual approach in that we can reduce the reasoning services (such as knowledge base satisfiability) to existing DL reasoning services. Finally, the contextual approach and the Hilog approach have not covered datatypes yet, while the FA approach covers datatypes. In order to support datatypes in the contextual approach, some extra syntax may be needed for OWL DL, otherwise it is difficult to distinguish the contexts. For example, in $\exists R.E$, E can be either a class or a datatype. It is not clear how to support datatypes in the Hilog approach yet.

Other existing approaches either limit the extension of RDF(S) to only a property-related subset of OWL with a weaker semantics proposed by ter Horst ([28, 29]), or weaken the semantic connection between the individual interpretation and class interpretation of a given URI [6], hence failing to propagate important inferences from meta-classes to classes (see [21]).

[9] In [15], the subscripts are not used.

7 Conclusion

In this chapter, we have presented RDF. RDF is a standard syntax for Semantic Web annotations and languages. RDF Schema is an ontological schema language that supports only class and property hierarchies, as well as domain and range constraints for properties. 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 shown that there exist syntactic and semantic mismatch between RDF(S) and OWL-DL. Accordingly, this chapter includes a novel modification of RDF(S), called RDFS-FA, 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, OWL2-DL and OWL-Eu [20]).

In chapter "RDF Storage and Retrieval Systems," we will further describe entailment and querying over RDF(S) ontologies. As for RDFS-FA, reasoning in RDFS-FA and its OWL extension, OWL-FA, is discussed in [22]; such reasoning can be performed by reduction to OWL-DL reasoning.

References

1. H. Alvestrand. Rfc 3066 – tags for the identification of languages. Technical report, IETF, Jan 2001. `http://www.isi.edu/in-notes/rfc3066.txt.`
2. Tim Berners-Lee. Realising the Full Potential of the Web. W3C Document, URL `http://www.w3.org/1998/02/Potential.html`, Dec 1997.
3. T. Berners-lee. Semantic Web Road Map. W3C Design Issues. URL http://www.w3.org/DesignIssues/Semantic.html, Oct. 1998.
4. J. Broekstra, M. Klein, S. Decker, D. Fensel, F. van Harmelen, and I. Horrocks. Enabling knowledge representation on the web by extending rdf schema. In *Proc. of the International World Wide Web Conference*, 2001.
5. DCMI. Dublin Core Metadata Element Set, Version 1.1: Reference Description. DCMI Recommendation, URL `http://dublincore.org/documents/dces/`, June 2003.
6. J. de Bruijn, E. Franconi, and S. Tessaris. Logical reconstruction of normative RDF. In *OWL: Experiences and Directions Workshop*, 2005.
7. Joint W3C/IETF URI Planning Interest Group. URIs, URLs, and URNs: Clarifications and Recommendations 1.0. URL `http://www.w3.org/TR/uri-clarification/`, 2001. W3C Note.
8. T. R. Gruber. *Towards Principles for the Design of Ontologies Used for Knowledge Sharing*, chapter of Formal Ontology in Conceptual Analysis and Knowledge Representation. Kluwer Academic, New York, 1993.
9. P. Hayes. RDF Semantics. Technical report, W3C, Feb 2004. W3C recommendation, `http://www.w3.org/TR/rdf-mt/.`
10. I. Horrocks and P. F. Patel-Schneider. Three Theses of Representation in the Semantic Web. In *Proc. of the 12th International World Wide Web Conference*, pages 39–47. ACM, New York, 2003.

11. I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From SHIQ and RDF to OWL: The Making of a Web Ontology Language. *Journal of Web Semantics*, 1(1):7–26, 2003.
12. I. Horrocks, U. Sattler, and S. Tobies. Practical Reasoning for Expressive Description Logics. In *Proc. of the 6th International Conference on Logic for Programming and Automated Reasoning*, pages 161–180, 1999.
13. G. Klyne and J. J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. URL `http://www.w3.org/TR/rdf-concepts/`, Feb 2004. Series Editor: Brian McBride.
14. F. Manola and E. Miller. RDF Primer, W3C Recommendation. URL `http://www.w3.org/TR/rdf-primer/`, Feb 2004. Series Editor: Brian McBride.
15. B. Motik. On the Properties of Metamodeling in OWL. In *Proc. of the 4th International Semantic Web Conference*, 2005.
16. W. Nejdl, M. Wolpers, and C. Capella. The RDF Schema Specification Revisited. In *Modelle und Modellierungssprachen in Informatik und Wirtschaftsinformatik, Modellierung 2000*, 2000.
17. N. Noy. Representing Classes as Property Values on the Semantic Web, Jul. 2004.
18. J. Z. Pan and I. Horrocks. Metamodeling Architecture of Web Ontology Languages. In *Proceeding of the Semantic Web Working Symposium (SWWS)*, 2001.
19. J. Z. Pan and I. Horrocks. RDFS(FA) and RDF MT: Two Semantics for RDFS. In *Proc. of the 2nd International Semantic Web Conference*, 2003.
20. J. Z. Pan and I. Horrocks. OWL-Eu: Adding Customised Datatypes into OWL. *Journal of Web Semantics*, 4(1):29–48, 2006.
21. J. Z. Pan and I. Horrocks. RDFS(FA): Connecting RDF(S) and OWL DL. In *IEEE Transactions on Knowledge and Data Engineering*, pages 192–206, 2007.
22. J. Z. Pan, I. Horrocks, and G. Schreiber. OWL FA: A Metamodeling Extension of OWL DL. In *Proc. of OWLED2005*, 2005.
23. P. F. Patel-Schneider. Layering the Semantic Web: Problems and Directions. In *Proc. of the 1st International Semantic Web Conference*, 2002.
24. P. F. Patel-Schneider. Two Proposals for a Semantic Web Ontology Language. In *Proc. of the International Description Logic Workshop*, 2002.
25. P. F. Patel-Schneider, P. Hayes, and I. Horrocks. OWL Web Ontology Language Semantics and Abstract Syntax. Technical report, W3C, Feb. 2004. W3C Recommendation.
26. PICS. Platform for Internet Content Selectivity. `http://www.w3.org/PICS/`, 1997.
27. D. Raggett, A. Le Hors, and I. Jacobs. HTML 4.01 Specification. W3C Recommendation, `http://www.w3.org/TR/html4/`, dEC. 1999.
28. H. J. ter Horst. Extending the RDFS Entailment Lemma. In *Proc. of the 3rd International Semantic Web Conference*, pages 77–91, 2004.
29. H. J. ter Horst. Completeness, Decidability and Complexity of Entailment for RDF Schema and a Semantic Extension Involving the OWL Vocabulary. *Journal of Web Semantic*, 3(2), 2005.
30. M. Uschold and M. Gruninger. Ontologies: Principles, Methods and Applications. *The Knowledge Engineering Review*, 1996.