EDITOR'S PROOF

Chapter 3
The Semantic Web Languages

**Fausto Giunchiglia, Feroz Farazi, Letizia Tanca,
and Roberto De Virgilio**

Abstract The Semantic Web is basically an extension of the Web and of the Web-enabling database and Internet technology, and, as a consequence, the Semantic Web methodologies, representation mechanisms and logics strongly rely on those developed in databases. This is the motivation for many attempts to, more or less loosely, merge the two worlds like, for instance, the various proposals to use relational technology for storing web data or the use of ontologies for data integration. This article comes second in this book, after an article on data management, in order to first complete the picture with the description of the languages that can be used to represent information on the Semantic Web, and then highlight a few fundamental differences which make the database and Semantic Web paradigms complementary but somehow difficult to integrate.

3.1 Introduction

The *World Wide Web* (Web from now on) is an enormous collection of data and documents of any kind, mixed and integrated in all possible ways, that keeps growing not monotonically. The Web is an open environment, where users can add or delete documents and data as they prefer, without any restriction. Some documents stay

UNCORRECTED PROOF

F. Giunchiglia (✉)
Dipto. Ingegneria e Scienza dell'Informazione, Università di Trento, Via Sommarive 14,
38100 Povo TN, Italy
e-mail: fausto@disi.unitn.it

F. Farazi
Department of Information Engineering and Computer Science, University of Trento,
Trento, Italy
e-mail: farazi@disi.unitn.it

L. Tanca
Dipto. Elettronica e Informazione (DEI), Politecnico di Milano, Via Ponzio 34/5,
20133 Milan, Italy
e-mail: tanca@elet.polimi.it

R. De Virgilio
Dipto. Informática e Automazione, Università Roma Tre, Via della Vasca Navale 79,
00146 Rome, Italy
e-mail: dvr@dia.uniroma3.it

R. De Virgilio et al. (eds.), *Semantic Web Information Management*,
DOI 10.1007/978-3-642-04329-1_3, © Springer-Verlag Berlin Heidelberg 2010

EDITOR'S PROOF

in time, some change, some appear and disappear and this process is completely unpredictable. And this applies not only to the Web but virtually to any repository of data (e.g., text, media, sensor data), also within company intranets. As a further complication, these data are highly *semantically heterogeneous*, in other words, we have, as a widespread common phenomenon, that the same information is represented in many different ways (e.g., the same amount of amount of money can be represented in dollars, in euros, in pounds).

The *Semantic Web* [5, 6] was originally proposed by its inventor as the way to solve the problem of semantic heterogeneity in the Web. The proposed solution is to add as an extra abstraction layer, a so-called *semantic layer*, to be built on top of the Web, which makes data not only human processable but also machine processable. In the research in data and knowledge management, the word semantics has been used and abused many times. In the Semantic Web, this word assumes a rather precise connotation and it amounts to assuming that the meaning of data and documents is codified as *metadata*, namely, data about data. The key idea is, therefore, to incrementally add new (meta)data whose only purpose is to explicitly codify the intended meaning of Web data. As a trivial example, the fact that a photo contains the face of Fausto can be codified into a data structure (a triple) whose contents can be represented, using a logical notation, as $about(photo1, Fausto)$ where $photo1$ and $Fausto$ are unique identifiers for the involved resources.

The Semantic Web, as clearly shown in Parts I, II of this book, is therefore an extension of the Web and of the Web enabling database and Internet technology, and, as a consequence, the Semantic Web methodologies, representation mechanisms and logics strongly rely on those developed in databases. And, this is the motivation for the many attempts to (more or less loosely) merge the two worlds like, for instance, the various proposals to use relational technology for storing web data (e.g., Chap. 4) or the use of ontologies for data integration (Chap. 17), just to name a few. And, this is also why this article comes second in this book after an article on data management.

At the same time, this is also the place to highlight a few fundamental differences which make the database and Semantic Web paradigms complementary but very different and somehow difficult to integrate. The crucial distinction is between the “closed” nature of the first vs. the “open” nature of the second. For instance, since incompleteness is inherent in the nature of Web data, in the Web no assumption is made about information which has not been explicitly stated, while in the database realm what has not been asserted or inferred is considered as false. In an analogous way, no uniqueness hypothesis is made as for the identifiers of web objects (this is why the Web had to recover this notion via Unique Resource Identifiers (URI)), while one strong requirement of database objects is that they be uniquely identified. Confronting the strengths and weaknesses of both paradigms, in order to be able to build new systems that are able to encompass the strengths of both, is thus worthwhile: the lessons learned from Classical Logic, which is the logical paradigm disciplining the Semantic Web, can be used to extend the expressive power of database query languages and to deal with incomplete information in databases; on the other hand, the introduction of some restrictions to the logics adopted for

UNCORRECTED PROOF

EDITOR'S PROOF

CORRECTED PROOF

the Semantic Web may help retain the good complexity results typical of database querying. This book should be read exactly in this perspective, keeping in mind that each chapter relates research which is ongoing in one of these two general directions.

The rest of the chapter is structured as follows. In Sect. 3.2, we describe the hierarchy of the languages that can be used to represent information on the Semantic Web. Section 3.3 presents the data model used in RDF and an example of how simple statements can be represented in RDF. Section 3.4 describes OWL, its sublanguages and an example representing the same statements represented in RDF. In Sect. 3.5, we describe C-OWL (Context OWL) namely OWL extended to take into account context via mappings across multiple ontologies. In Sect. 3.6, after the introduction to the most important Web Languages, we dig a little deeper in the connections between the Semantic Web and databases briefly discussed above. We conclude the chapter in Sect. 3.7.

3.2 The Hierarchy of Languages

We stated above that the Semantic Web is just metadata explicitly encoding the implicit semantics of Web data. But which kinds of metadata? According to the Semantic Web approach, data are organized in (at least) four levels of increased expressibility, each corresponding to a specific representation need, namely: XML [8] and XML Schema [13], RDF [3] and RDF Schema [9], OWL [27] and C-OWL [7]. Notice that, strictly speaking, XML is not a semantic Web language as it codifies no semantics. Its presentation is however very relevant as all the Semantic Web languages are defined as extensions of XML and, anyhow, XML is a first important step, with respect to HTML,[1] towards semantic interoperability as it provides a way to standardize the use of tags, thus enabling syntactic interoperability.

XML: Raw Data—No Semantics XML is designed to represent information by using customized tags. Because of the customizable tag support, it is used to exchange a wide variety of information on the Web and elsewhere. Statements like "GeoNames has coverage of all countries" and "It was modified on April 25, 2009" can be represented in XML using tags 'GeoNames', 'coverage' and 'modified' and a preceding statement saying that the following information is in XML along with the XML version used to represent this information:

```
<?xml version="1.0" ?>
   < GeoNames >
   <coverage>Countries</coverage>
   <modified>April 25, 2009</modified>
   </GeoNames>
```

---

[1] http://www.w3.org/html/.

EDITOR'S PROOF

The purpose of XML Schema is to define a set of rules to which an XML document conforms. An XML Schema is similar to a class in object oriented programming language and an XML document is similar to an instance of that class. XML Schema is used for exchanging information between interested parties who have agreed to a predefined set of rules. But the absence of meaning of the vocabulary terms used in XML Schema makes it difficult for machines to accomplish communication between them when new XML vocabulary terms are used. On one hand machines can not differentiate between polysemous terms, and on the other hand they can not combine the synonymous terms.

RDF(S): Representing Objects and Relations Among Them RDFS is an acronym for RDF Schema. We use RDF(S) meaning both RDF and RDFS. The goal of RDF(S) is to provide meaning to data therefore overcoming the drawback (absence of meaning) of XML. The simplest forms of RDF metadata are tags of single resources, e.g., photo tags in Flickr. One such metadata could state, for instance, that a specific Web page is the homepage of a specific user, or that a photo is about a specific location, or that a document is about a specific topic.

RDF is used to (i) describe information about Web resources and the systems that use these resources; (ii) make information machine processable; (iii) provide internetworking among applications; (iv) provide automated processing of Web information by intelligent agents. It is designed to provide flexibility in representing information. Its specification is given in [3, 9, 19, 21, 24, 26].

RDF Schema is an extension of RDF. It provides a vocabulary for RDF to represent classes of the resources, subclasses of the classes, properties of the classes and relations between properties. The capability of representing classes and subclasses allows users to publish ontologies on the Web. But these ontologies have limited use as RDFS can not represent information containing disjointness and specific cardinality values.

OWL: Ontologies—Representing Classes and Relations Among Them OWL is a quite expressive representation language. It provides the syntax to specify classes (sets of objects, also called concepts), various operations on classes such as, for instance, that two or more classes are disjoint. However, OWL does not have built-in primitives for the (very important) part-whole relations [29]. The simplest metadata expressing properties of classes are tags which encode properties of sets of resources, e.g., del.ic.ious tags. One such metadata could state that a set of web pages is about a specific topic, or that a set of photos is about the same person. In most common uses, however, the OWL metadata are organized in graph structures encoding complex relations among classes, i.e., ontologies [20], where each node is associated to a concept (represented as a natural language label) and where links codify semantic (logical) relations between the labels of the two nodes involved. As a very important example, in the case of lightweight ontologies [14, 18], schematic metadata are organized as trees where the labels of nodes lower in the tree are more specific than the labels of the nodes above.

The details of the OWL specification are described in [4, 10, 22, 27, 28, 32].

UNCORRECTED PROOF

EDITOR'S PROOF

UNCORRECTED PROOF

C-OWL: Contextual Ontologies—Representing Context Mappings OWL allows to represent one ontology at a time. In practice, the Semantic Web is plenty of ontologies developed independently, each modeling a specific subdomain. OWL has an import operation which allows to import an ontology as a part of the specification of a more complex ontology. However, in most cases, the import operation is far too strong. One would simply like to relate the concept in one ontology with the concept of another ontology. Furthermore, OWL cannot natively deal with the fact that the meaning of certain words (class names) is context dependent [7], in other words, that the same word in different ontologies may represent a different concept. One trivial example of context dependency is that the meaning of the word *car* as codified in the FIAT database means, e.g., the set of FIAT cars, and is therefore different from the meaning of this same word inside the BMW database. Context OWL (C-OWL) [7] is a proposed extension of OWL (but not a Web Standard) which allows to represent multiple OWL ontologies and the mappings (relations) between these ontologies, where each ontology represents a localized view of a domain.

Two of the papers in Part II describe how reasoning about ontologies can be exploited in order to automatically compute context mappings.

The step from XML to RDF is key as the encoding of semantics is the basis for *achieving semantic interoperability*. Once the semantics are explicitly represented, the meaning of a given data can be normalized with respect to all its syntactic variations. Or, viceversa, the multiple meanings (also called senses) of a word can be made explicit. For instance, it is possible to distinguish between the three possible meanings of the word *Java* (a kind of coffee bean, a programming language, and an island name) and, dually, it is possible to say that *automobile* and *car*, which are synonyms, mean actually the same thing. The step from RDF to OWL is key for allowing complex *reasoning about documents*, sets of documents and their relations. Of course, it is also possible to perform reasoning with RDF only. Reasoning about instances amounts to propositional reasoning. At this level, it is possible to reason about single instances (documents), for instance to derive, given the proper background knowledge [15, 17] that the content of a document which talks about *animal*s is more general than the content of another document which talks about *cats*. Reasoning in OWL is much more powerful and it allows to reason about complex properties of sets of instances. It allows, for instance, to derive, given the proper background knowledge, that any *professor* in a given university *teaches at least one course*.

3.3 RDF(S)

RDF is a language for representing data in the Semantic Web. RDF is designed (i) to provide a simple data model so that users can make statements about Web resources; (ii) to provide the capability to perform inference on the statements represented by users.

The data model in RDF is a graph data model. The graph used in RDF is a directed graph. A graph consists of nodes and edges. Statements about resources can

EDITOR'S PROOF

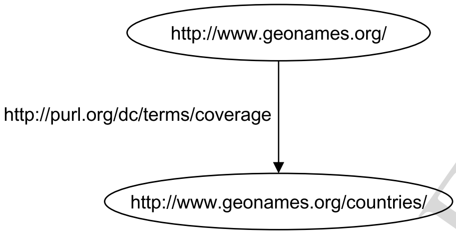

Fig. 3.1 Graph data model of a statement representing subject, object and predicate as URIs

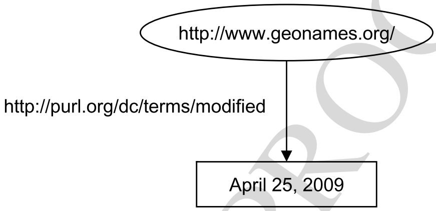

Fig. 3.2 Graph data model of a statement representing subject and predicate as URIs and the object as a literal

be represented by using graph nodes and edges. Edges in RDF graphs are labeled. An edge with two connecting nodes form a triple. Among two nodes a node represents subject, another node represents object and the edge represents predicate of the statements. As the graph is a directed graph, the edge is directed edge and the direction of the edge is from subject to object. The predicate is also called as property of the subject or relationship between subject and object.

RDF uses URI references to identify subjects, objects and predicates. The statement "GeoNames has coverage of all countries" can be represented in RDF, where 'GeoNames' is a subject, 'countries' is an object and 'coverage' is a predicate. The URIs of the subject 'GeoNames', object 'countries' and predicate 'coverage' are "http://www.geonames.org", "http://www.geonames.org/countries" and "http://purl.org/dc/terms/cove-rage", respectively. Figure 3.1 provides a graphical representation of this RDF statement.

Objects in RDF statements can be literals. In the statement "GeoNames was modified on April 25, 2009", 'GeoNames' is a subject, 'modified' is an object and 'April 25, 2009' is a predicate, which is a literal. The URIs of the subject 'GeoNames' and predicate 'modified' are "http://www.geonames.org" and "http://purl.org/dc/terms/modified" respectively and the object 'April 25, 2009' can be represented as is without a URI. Figure 3.2 provides a graphical representation of this RDF statement.

Statements about GeoNames can be described in RDF using constructs rdf:Description, rdf:resource, rdf:about and rdfs:label as follows:

```
<?xml version="1.0"?>
    <rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xmlns:dc="http://purl.org/dc/terms#">
```

UNCORRECTED PROOF

EDITOR'S PROOF

```
  <rdf:Description rdf:about=“http://www.geonames.org”>
    <rdfs:label>GeoNames</rdfs:label>
    <dc:coverage rdf:resource=“http://www.geonames.org/countries”/>
    <dc:modified>April 25, 2009</dc:modified>
  </rdf:Description>
</rdf:RDF>
```

3.4 OWL

Similarly, to what happens with RDF, OWL data are represented as triples subject, object and predicate. As it turns out, there are (at least) three OWL languages of increasing logical expressivity, namely: *OWL Lite*, *OWL DL*, *OWL Full*. As a matter of fact, there are many variants and extensions of OWL each corresponding to a Logic and associated expressivity levels. C-OWL itself is an extension of OWL, one of the papers in Part II in this book describes an extension of OWL to account for time, and similarly for the work on modular ontologies again in Part II of this book.

Concentrating on the three basic OWL languages, the most important is OWL DL, where DL stands for *Description Logic* and owns its name to the fact that it is a notational variant, tuned to Web use, of Description Logics [2]. The key feature is that reasoning in OWL DL can be implemented by exploiting the many state-of-the-art DL reasoners, e.g., Pellet [31].

More detailed descriptions of all three sub-languages of OWL—OWL Lite, OWL DL and OWL Full, are provided below.

OWL Lite OWL Lite allows the use of a subset of the OWL and RDF(S) vocabulary. The main goal is to trade expressivity for efficiency (and guaranteed termination) of reasoning. In particular, it is possible to use thirty-five out of forty OWL constructs and eleven out of thirty-three RDF(S) constructs (not including the subproperties of the property `rdfs:member`). The lists of the thirty-three RDF(S) constructs, of the forty OWL constructs and of the eleven RDF(S) constructs that can be used in OWL are provided in Appendixes A and B at the end of this chapter.

In OWL Lite to define a class, one must use the OWL construct `owl:Class` rather than the RDF(S) construct `rdfs:Class` which is not allowed. Other five OWL constructs, namely: `complementOf`, `disjointWith`, `hasValue`, `oneOf` and `unionOf` are not allowed in OWL Lite. Other OWL Constructs are allowed to use in OWL Lite but their use is limited. Thus, all three cardinality constructs—`cardinality`, `maxCardinality` and `minCardinality`, can only have 0 or 1 in their value fields. Furthermore, `equivalentClass` and `intersectionOf` cannot be used in a triple if the subject or object represents an anonymous class.

UNCORRECTED PROOF

EDITOR'S PROOF

OWL DL OWL DL can use all eleven RDF(S) constructs used by OWL Lite. Similarly, to OWL Lite, it uses only the `owl:Class` construct to define a class. OWL DL allows to use all forty OWL constructs. However, some of these constructs have restricted use. In particular, classes cannot be used as individuals, and vice versa. Each individual must be an extension of a class. Even if an individual cannot be classified under any user defined class, it must be classified under the general `owl:Thing` class. Individuals can not be used as properties, and vice versa. Moreover, properties can not be used as classes, and vice versa.

Properties in OWL DL are differentiated into data type properties and object properties. Object properties connect class instances and data type properties connect instances to literals. OWL DL allows the use of the `intersectionOf` construct with any number of classes and of any non negative integer in the cardinality restrictions value fields.

The restrictions provided in OWL DL allow to maintain a balance between expressivity and computational completeness. Even though its computational complexity is higher than that of OWL Lite, reasoning in OWL DL remains decidable (of the same complexity of the corresponding Description Logic).

OWL Full OWL Full can use all forty OWL constructs and eleven RDF(S) constructs without any of the OWL DL restrictions that imposed on OWL. Moreover, the constructs `rdfs:Class` as well as `owl:Class` can be used to define a class. The key difference from OWL DL is that properties can be assigned to classes, a class can be represented as an individual or a property, and vice versa. The price for this increased expressivity is that reasoning in OWL Full is undecidable, i.e., it may not terminate on certain inputs.

To provide an example of OWL full the GeoNames statement, can be represented on OWL using the constructs owl:Ontology, owl:Thing, rdfs:labels and rdf:resource as follows:

```
<?xml version="1.0"?>
    <rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xmlns:owl="http://www.w3.org/2002/07/owl#"
        xmlns:dc="http://purl.org/dc/terms#">
     <owl:Ontology rdf:about=""/>
     <owl:Thing rdf:about="http://www.geonames.org">
       <rdfs:label>GeoNames</rdfs:label>
       <dc:coverage rdf:resource="http://www.geonames.org/countries"/>
       <dc:modified>April 25, 2009</dc:modified>
     </owl:Thing>
    </rdf:RDF>
```

UNCORRECTED PROOF

EDITOR'S PROOF

3.5 C-OWL

The key addition that C-OWL provides on top of OWL is the possibility to represent multiple ontologies and context mappings, namely triples subject relation object between two concepts, or between two instances or between two properties in two different ontologies. The mapping relations in the triple can be one of more specific, more general, equivalent, disjoint and compatible. C-OWL allows for the use of any of the OWL sub-languages but the two ontologies involved in a mapping must belong to the same sub-language.

C-OWL mappings are also called bridge rules. An ontology plus the set of bridge rules where the subject concept belongs to the ontology itself is called a contextual ontology. To provide an example of contextual ontology, we provide below the simple Wine ontology originally described in [7]. In this contextual ontology, two ontologies Wine and Vino are mapped. For the detailed description, we refer to the C-OWL paper.

PROOF

```
<?xml version=“1.0”?>
<rdf:RDF
    xmlns:rdf=“http://www.w3.org/1999/02/22-rdf-syntax-ns#”
    xmlns:rdfs=“http://www.w3.org/2000/01/rdf-schema#”
    xmlns:cowl=“http://www.example.org/wine-to-vino.map#”>
<cowl:mapping>
  <rdfs:comment>Example of a mapping of wine into vino</rdfs:comment>
  <cowl:srcOntology rdf:resource=“http://www.ex.org/wine.owl”/>
  <cowl:tgtOntology rdf:resource=“http://www.ex.org/vino.owl”/>
 <cowl:bridgRule cowl:br-type=“equiv”>
   <cowl:srcC rdf:resource=“http://www.ex.org/wine.owl#wine”/>
   <cowl:tgtC rdf:resource=“http://www.ex.org/vino.owl#vino”/>
 </cowl:bridgRule>
 <cowl:bridgRule cowl:br-type=“onto”>
   <cowl:srcC rdf:resource=“http://www.ex.org/wine.owl#RedWine”/>
   <cowl:tgtC rdf:resource=“http://www.ex.org/vino.owl#VinoRosso”/>
 </cowl:bridgRule>
 <cowl:bridgRule cowl:br-type=“into”>
   <cowl:srcC rdf:resource=“http://www.ex.org/wine.owl#Teroldego”/>
   <cowl:tgtC rdf:resource=“http://www.ex.org/vino.owl#VinoRosso”/>
 </cowl:bridgRule>
 <cowl:bridgRule cowl:br-type=“compat”>
   <cowl:srcC rdf:resource=“http://www.ex.org/wine.owl#WhiteWine”/>
   <cowl:tgtC rdf:resource=“http://www.ex.org/vino.owl#Passito”/>
 </cowl:bridgRule>
 <cowl:bridgRule cowl:br-type=“incompat”>
   <cowl:srcC rdf:resource=“http://www.ex.org/wine.owl#WhiteWine”/>
   <cowl:tgtC rdf:resource=“http://www.ex.org/vino.owl#VinoNero”/>
 </cowl:bridgRule>
</cowl:mapping> </rdf:RDF>
```

EDITOR'S PROOF

As it can be noticed, a mapping is defined by source and target ontology and a set of bridge rules, where each bridge rule is defined by the source and target concepts selected from the respective ontologies, and the semantic relation which holds between the two concepts.

3.6 Semantic Web and Databases

As announced by the book subtitle, we are analyzing data management in the Semantic Web in a *model-based perspective*. Indeed, both in databases and in the web, good modeling is crucial, since good modeling is key of having efficient representation and reasoning [1]. Thus, many of the most interesting efforts of the two research communities have been devoted to finding and refining appropriate representation formalisms, each with the aim to capture the distinguishing characters of the context they wish to model. This paper and the previous one in this book try to present these efforts from the two communities. However, since the goal of this book is to bridge the two worlds, and since the appropriate management of data in the Semantic Web is crucial, some brief considerations on the differences between the basic modeling assumptions of the two areas are in order.

As will also be seen in Chap. 7, the most famous approach to deduction and reasoning in databases is based on Datalog [11]. Thus, when referring to the differences between inference in the Semantic Web and inference in the database domain we will mostly refer to the underlying deduction frameworks, namely Classical Logic (mainly Description Logic and its variations) and Datalog.

One of the most important differences between the two worlds is the "open" nature of the Web, vs. the "closed" nature of databases. In Classical Logic, unstated information does not assume a truth value: that is, when an assertion is not found as a known fact, nothing can be said about its truth value. On the other hand, in the database realm the facts that have neither been asserted nor inferred are considered as false. The first attitude is known as the *Open World Assumption* (*OWA*), while the second is the *Closed World Assumption* (*CWA*), and each of them is perfectly coherent with the framework in which it is assumed.

The CWA [30] can be seen as an inference rule that, given a set of sentences S and an atom A, if A does not follow from S (i.e., cannot be inferred from S), derives $\neg A$. The CWA accounts for the way database people see the database as a mirror of the real world. Indeed, though we can reasonably allow for a database to be incomplete, that is, not to contain *all* the facts which are true in the world, most database applications can perfectly accommodate the much more restrictive hypothesis that *what is not recorded must be considered as false*. Indeed, in information systems—where databases are most used—it is reasonable to assume that all relevant information is actually available. The result of this assumption allows for a much simpler treatment of negation, in that not only what is explicitly asserted as false is so.

An important consequence of the CWA is the so-called *minimal model semantics* of databases. Since, from a proof-theoretic point of view, the CWA implies that facts

UNCORRECTED PROOF

EDITOR'S PROOF

that cannot be proven must be considered as false, then *the* model of the database consists of all the facts that are true in *all* the worlds satisfying S, that is, a minimal model.

On the other hand, in the Semantic Web, there is no need to assume that a certain (although ample) collection of information sources should contain all information which is true; thus the Classical paradigm is more appropriate for web modeling since, when a fact F is not inferrable from S, it does not exclude interpretations of S which contain F. This allows for the possibility that, coming into play another information source which entails F, we should not fall into contradiction.

Some sort of reconciliation is possible between the two attitudes by taking an *epistemic* view of the database content: in [25], the epistemic operators provide a clean way to express the difference between the description of the external world, and that of the database itself, that is, *what the database knows*. Thus, of a certain fact we can ask whether it is *known to the database*, mimicking the semantics of a database query. Within this view, a clear model-theoretic semantics can be given to databases which is no longer incompatible with the classical paradigm underlying the semantic web. Including these operators in the various adopted logics may increase their computational complexity, and various researchers have engaged in solving this problem [12].

The "closed" view adopted in the database world also has two more aspects, namely the *unique name assumption*, which states that individuals with different names are different, and the *domain closure assumption*, which comes in different flavors but basically states that there are no other individuals than those in the database. Both assumptions do not favor the richness of expressivity needed for the web, and thus are to be rejected in that context. By contrast, they prove to be very practical in the database domain, where unambiguous answers to "for all" queries and queries involving negation can be provided, based on the three assumptions above.

The above problems are part of the wider question of *incomplete information*: for instance, in the open perspective of the web we would like to be able to assert that an employee belongs to a department, without being obliged to name this department explicitly. One way to (partially) solve the problem in relational databases is the introduction of null values, whose treatment still produces a lot of research because as yet considered unsatisfactory; using different models, like the object-oriented one or semistructured data models helps a little in this direction, though introducing new problems related to a lower efficiency as for data manipulation.

Another example of incomplete information is given by disjunction: we might want to state that John has gone out either with Jane or with Sara, but asserting such disjunctive information is impossible in the relational database model, and requires appropriate extensions. Disjunctive information management is also a difficult task in relation to negation and the CWA. Indeed, suppose that a disjunctive sentence $P \vee Q$ holds in a database: then by the CWA we will be able to derive $\neg P$ and also $\neg Q$, which obviously leads to inconsistency.

Among other important differences of the two approaches, we mention the question of infinity, which in its turn is strictly related to the meaning of database instances. In the traditional context of relational databases, a database (instance) is a

UNCORRECTED PROOF

EDITOR'S PROOF

finite set of finite relations, i.e., the totality of all tuples that can appear in a database is finite. Thus, since a database instance can be viewed as an interpretation of the first-order theory defined by the database schema (plus possibly a deductive program) and the integrity constraints, only finite models for the database schema are admissible. In the Classical paradigm, no assumption is made as to the interpretations that are acceptable for a theory, thus infinite models are not ruled out. Moreover, the idea that an instance is an interpretation leads to identification between information and interpretation (which is the basis of the so-called Herbrand model semantics of datalog), whereas an ontology is seen as a theory which admits many possible interpretations.

More differences between the two paradigms reside in the use and treatment of constraints and restrictions. An interesting and detailed discussion on these topics can be found in [23].

3.7 Conclusion

In this chapter, we have presented a short introduction to the Semantic Web, to its underlying motivations and ideas and to the main languages used to implement it. The main goal of this chapter is to integrate the contents of the previous chapter on database technology and to provide the necessary basic notions needed in order to properly read the contents of the rest of the book.

Appendix A: RDF(S) Constructs

This appendix provides a list of the thirty-three RDF(S)constructs excluding the sub-properties of rdfs:member.

The RDF(S) constructs are rdf:about, rdf:Alt, rdf:Bag, rdf:Description, rdf:first, rdf:ID, rdf:List, rdf:nil, rdf:Object, rdf:predicate, rdf:Property, rdf:resource, rdf:rest, rdf:Seq, rdf:Statement, rdf:subject, rdf:type, rdf:value, rdf:XMLLiter-al, rdfs:Class, rdfs:comment, rdfs:Container, rdfs:ContainerMembershipProp-erty, rdfs:Datatype, rdfs:domain, rdfs:isDefinedBy, rdfs:label, rdfs:Literal, rdfs:member, rdfs:range, rdfs:seeAlso, rdfs:subClassOf, and rdfs:subProperty-Of.

Details of the meaning of the above constructs can be found in the RDF(S) manuals. To provide a few examples, rdfs:Class allows to represent a concept, rdfs:subClassOf to state that a concept is more specific than another, rdf:resource to represent a resource (an instance of a concept), rdfs:label to represent a human readable label (for a concept or resource or property), rdfs:comment to provide a human readable description of a concept or resource or property.

Appendix B: OWL Constructs

This appendix provides the lists of the forty OWL constructs and eleven RDF(S) constructs that can be used in an OWL representation.

UNCORRECTED PROOF

EDITOR'S PROOF

The OWL constructs are owl:AllDifferent, owl:allValuesFrom, owl:Annotation Property, owl:backwardCompatibleWith, owl:cardinality, owl:Class, owl:complementOf, owl:DataRange, owl:DatatypeProperty, owl:DeprecatedClass, owl:DeprecatedProperty, owl:differentFrom, owl:disjointWith, owl:distinctMembers, owl:equivalentClass, owl:equivalentProperty, owl:FunctionalProperty, owl:hasValue, owl:imports, owl:incompatibleWith, owl:intersectionOf, owl: InverseFunctionalProperty, owl:inverseOf, owl:maxCardinality, owl:minCardinality, owl:Nothing, owl:ObjectProperty, owl:oneOf, owl:onProperty, owl:Ontology, owl:OntologyProperty, owl:priorVersion, owl:Restriction, owl:sameAs, owl:someValuesFrom, owl:SymmetricProperty, owl:Thing, owl:TransitiveProperty, owl:unionOf, and owl:versionInfo.

The RDF(S) constructs are rdf:about, rdf:ID, rdf:resource, rdf:type, rdfs:comment, rdfs:domain, rdfs:label, rdfs:Literal, rdfs:range, rdfs:subClassOf, and rdfs:subPropertyOf.

To provide a few examples of the meaning of the constructs above, owl:Class can be used to represent a concept, owl:equivalentClass to state that a concept is equivalent to another, owl:Thing to represent an instance of a concept, owl:sameAs to state that two instances refer the same thing.

References

1. Amarel, S.: On representations of problems of reasoning about actions. In: Machine Intelligence 3, pp. 131–171. Elsevier, Amsterdam (1968)
2. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, Cambridge (2003)
3. Beckett, D.: RDF/XML Syntax Specification (Revised). Technical Report, World Wide Web Consortium (W3C), Recommendation, February 10 (2004)
4. Bechhofer, S., Harmelen, F.V., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., Stein, L.A.: OWL Web Ontology Language Reference. Technical Report, World Wide Web Consortium (W3C), Recommendation, February 10 (2004)
5. Berners-Lee, T.: Weaving the Web. Orion Business Books (1999)
6. Berners-Lee, T., Hendler, J.A., Lassila, O.: The Semantic Web. Sci Am. J. **284**(5), 34–43 (2001)
7. Bouquet, P., Giunchiglia, F., Harmelen, F.V., Serafini, L., Stuckenschmidt, H.: C-OWL: Contextualizing ontologies. In: International Semantic Web Conference, pp. 164–179 (2003)
8. Bray, T., Paoli, J., Sperberg-McQueen, C.M., Maler, E., Yergeau, F.: Extensible Markup Language (XML) 1.0 (Fourth Edition). Technical Report, World Wide Web Consortium (W3C), Recommendation, February 10 (2004)
9. Brickley, D., Guha, R.V.: RDF Vocabulary Description Language 1.0: RDF Schema. Technical Report, World WideWeb Consortium (W3C), Recommendation, February 10 (2004)
10. Carroll, J.J., Roo, J.D.: OWL Web Ontology Language Test Cases. Technical Report, World Wide Web Consortium (W3C), Recommendation, February 10 (2004)
11. Ceri, S., Gottlob, G., Tanca, L.: Logic Programming and Databases. Springer, Berlin (1990)
12. Donini, F.M., Lenzerini, M., Nardi, D., Schaerf, A., Nutt, W.: Adding epistemic operators to concept languages. In: 3th International Conference on Principles of Knowledge Representation and Reasoning (KR'92), Cambridge, MA (1992)
13. Fallside, D.C., Walmsley, P.: XML Schema Part 0: Primer Second Edition. Technical Report, World Wide Web Consortium (W3C), Recommendation, October 28 (2004)

UNCORRECTED PROOF

EDITOR'S PROOF

14. Giunchiglia, F., Marchese, M., Zaihrayeu, I.: Encoding classifications into lightweight ontologies. In: Journal on Data Semantics VIII. LNCS, vol. 4380, pp. 57–81. Springer, Berlin (2007)
15. Giunchiglia, F., Shvaiko, P., Yatskevich, M.: Discovering missing background knowledge in onology matching. In: Proceedings: 17th European Conference on Artificial Intelligence (ECAI), pp. 382–386 (2006)
16. Giunchiglia, F., Walsh, T.: Abstract theorem proving. In: Proceedings of the 11th International Joint Conference on Artificial Intelligence (IJCAI'89), pp. 372–377 (1989) <uncited>
17. Giunchiglia, F., Dutta, B., Maltese, V.: Faceted lightweight ontologies. In: Borgida, A., Chaudhri, V., Giorgini, P., Yu, E. (eds.) Conceptual Modeling: Foundations and Applications. LNCS, vol. 5600. Springer, Berlin (2009)
18. Giunchiglia, F., Zaihrayeu, I.: Lightweight ontologies. In: The Encyclopedia of Database Systems. Springer, Berlin (2008, to appear)
19. Grant, J., Beckett, D.: RDF Test Cases. Technical Report, World Wide Web Consortium (W3C), Recommendation, February 10 (2004)
20. Guarino, N., Giaretta, P.: Ontologies and knowledge bases: towards a terminological clarification. In: Mars, N. (ed.) Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing. IOS Press, Amsterdam (1995)
21. Hayes, P.: RDF Semantics. Technical Report, World Wide Web Consortium (W3C), Recommendation, February 10 (2004)
22. Hefflin, J.: OWL Web Ontology Language Use Cases and Requirements. Technical Report, World Wide Web Consortium (W3C), Recommendation, February 10 (2004)
23. Patel-Schneider, P.F., Horrocks, I.: A comparison of two modelling paradigms in the Semantic Web. J. Web Semant. **5**(4), 240–250 (2007)
24. Klyne, G., Carroll, J.J.: Resource Description Framework (RDF): Concepts and Abstract Syntax. Technical Report, World Wide Web Consortium (W3C), Recommendation, February 10 (2004)
25. Lifschitz, V.: Nonmonotonic databases and epistemic queries. In: 12th International Joint Conference on Artificial Intelligence (IJCAI'91), Sydney, Australia (1991)
26. Manola, F., Miller, M.: RDF Primer. Technical Report, World Wide Web Consortium (W3C), Recommendation, February 10 (2004)
27. McGuinness, D.L., Harmelen, F.V.: OWL Web Ontology Language Overview. Technical Report, World Wide Web Consortium (W3C), Recommendation, February 10 (2004)
28. Patel-Schneider, P.F., Hayes, P., Horrocks, I.: OWL Web Ontology Language Guide Semantics and Abstract Syntax. Technical Report, World Wide Web Consortium (W3C), Recommendation, February 10 (2004)
29. Rector, A., Welty, C.: Simple part-whole relations in OWL Ontologies, W3C Working Draft, August 11 (2005)
30. Reiter, R.: On closed world data bases. In: Symposium on Logic and Data Bases (1977)
31. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: a practical owl-dl reasoner. J. Web Semant. (2003)
32. Smith, M.K., Welty, C., McGuinness, D.L.: OWL Web Ontology Language Guide. Technical Report, World Wide Web Consortium (W3C), Recommendation, February 10 (2004)
33. Zaihrayeu, I., Su, L., Giunchiglia, F., Pan, L., Qi, J., Chi, M., Huang, X.: From web directories to ontologies: natural language processing challenges. In: 6th International Semantic Web Conference and the 2nd Asian Semantic Web Conference (ISWC + ASWC), Busan, Korea (2007) <uncited>

UNCORRECTED PROOF