Introduction to Semantic Web

Prepared by:
Besim Verdi Abdullai

Wien, 2010

Content

Table of Figures

I. Introduction to Semantic Web

I.1. RDF- The Resource Description Framework

The Semantic Web is a Web of data — of dates and titles and part numbers and chemical properties and any other data one might conceive of. Various technologies allow you to embed data in documents (RDFa, GRDDL) or expose what you have in SQL databases, or make it available as RDF files.
RDF provides the foundation for publishing and linking your data, in other words: ***“RDF is a language for representing information about resources in the World Wide Web”***.
RDF It is particularly intended for representing metadata about Web resources, such as the title, author, and modification date of a Web page, copyright and licensing information about a Web document, or the availability schedule for some shared resource. However, by generalizing the concept of a "Web resource", RDF can also be used to represent information about things that can be identified on the Web, even when they cannot be directly retrieved on the Web. Examples include information about items available from on-line shopping facilities (e.g., information about specifications, prices, and availability), or the description of a Web user's preferences for information delivery.
RDF is intended for situations in which this information needs to be processed by applications, rather than being only displayed to people. RDF provides a common framework for expressing this information so it can be exchanged between applications without loss of meaning. Since it is a common framework, application designers can leverage the availability of common RDF parsers and processing tools. The ability to exchange information between different applications means that the information may be made available to applications other than those for which it was originally created.
RDF is based on the idea of identifying things using Web identifiers (called **Uniform Resource Identifiers**, or **URI**s), and *describing resources in terms of simple properties and property values*. This enables RDF to represent simple statements about resources as a graph of nodes and arcs representing the resources, and their properties and values.

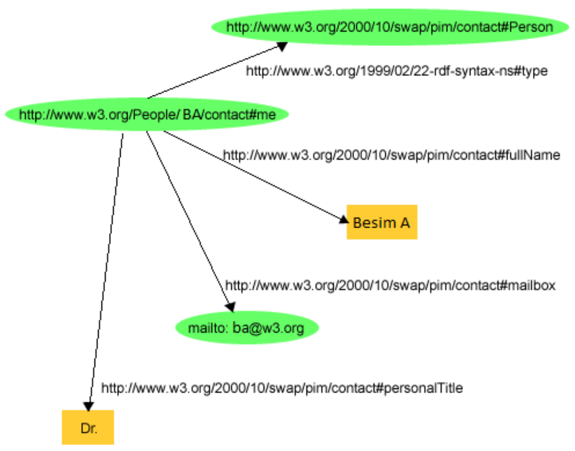
http://www.w3.org/2000/10/swap/pim/contact#Person
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
http://www.w3.org/People/ BA/contact#me
http://www.w3.org/2000/10/swap/pim/contact#fullName
Besim A
http://www.w3.org/2000/10/swap/pim/contact#mailbox
mailto: ba@w3.org
http://www.w3.org/2000/10/swap/pim/contact#personalTitle
Dr.

Figure 1 a RDF graph that identifies the person Besim A. using URIs

In the Figure 1, we have a RDF graph which represents a declaration for the resources as a graph of nodes and arows wihich represent resouces, properties and their values.

Thus the group of words "there is a Person identified by http://www.w3.org/People/BA/ contact#me, which name is Besim A, with an e-mail address ba@w3.org, and the title Dr.", can be presented as a RDF graph shown in the Figure 1. This graph shows that RDF uses URIs in order to identify:

- individuals, for example: Besim A., identified through *http://www.w3.org/People/BA/contact#me*
- types of things, like, persons, identified through *http://www.w3.org/2000/10/swap/pim/contact#Person*
- properties of these things, like: mailbox, identified with *http://www.w3.org/2000/10/swap/pim/contact#mailbox*
- values of these properties, like: mailto: *em@w3.org*, like a value of the properties of the mailbox (RDF also uses strings, like: "Besim A."), and other values of other data types like Integer and Date, as values of these properties.

RDF also provides an XML-based syntax (called RDF/XML) for recording and exchanging these graphs. Example 1 is a small chunk of RDF in RDF/XML corresponding to the graph in Figure 1:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"

xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#">

  <contact:Person rdf:about="http://www.w3.org/People/BA/contact#me">
    <contact:fullName>Besim A</contact: fullName>
    <contact:mailbox rdf:resource="mailto:ba@w3.org"/>
    <contact:personalTitle>Dr.</contact:personalTitle>
  </contact:Person>

</rdf:RDF>
```

Example 1: RDF/XML which describes Besim A.

Note that this RDF/XML also contains URIs, as well as properties like *mailbox* and *fullName* (in an abbreviated form), and their respective values *ba@w3.org*, and *Besim A*.

Like HTML, this RDF/XML is machine processable and, using URIs, can link pieces of information across the Web. However, unlike conventional hypertext, RDF URIs can refer to any identifiable thing, including things that may not be directly retrievable on the Web (such as the person Eric Miller). The result is that in addition to describing such things as Web pages, RDF can also describe cars, businesses, people, news events, etc. In addition, RDF properties themselves have URIs, to precisely identify the relationships that exist between the linked items.
More detailed knowledge over RDF Framework and specifications you can find in the following tutorials:

- RDF Concepts and Abstract Syntax [RDF-CONCEPTS]
- RDF/XML Syntax Specification [RDF-SYNTAX]
- RDF Vocabulary Description Language 1.0: RDF Schema [RDF-VOCABULARY]
- RDF Semantics [RDF-SEMANTICS]
- RDF Test Cases [RDF-TESTS]
- RDF Primer

I.2. OWL – Web Ontology Language

The Semantic Web is a vision for the future of the Web in which information is given explicit meaning, making it easier for machines to automatically process and integrate information available on the Web. The Semantic Web will build on XML's ability to define customized tagging schemes and RDF's flexible approach to representing data. The first level above RDF required for the Semantic Web is an ontology language what can formally describe the meaning of terminology used in Web documents. If machines are expected to perform useful reasoning tasks on these documents, the language must go beyond the basic semantics of RDF Schema
Using OWL (to build vocabularies, or “ontologies”) and SKOS (for designing knowledge organization systems) it is possible to enrich data with additional meaning, which allows more people (and more machines) to do more with the data.
The **OWL**-Web Ontology Language is intended to provide a language that can be used to describe the classes and relations between them that are inherent in Web documents and applications, i.e. a language for defining and instantiating Web ontologies.
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 related. An OWL ontology may include descriptions of classes, properties and their instances. Given such an ontology, the OWL *formal semantics* specifies how to derive its logical consequences, i.e. facts not literally present in the ontology, but entailed by the semantics. These entailments may be based on a single document or multiple distributed documents that have been combined using defined OWL mechanisms.

One question that comes up when describing yet another XML/Web standard is "What does this by me that XML and XML Schema don't?" There are two answers to this question:

1. **An ontology** differs from an XML schema in that, that **it is a knowledge representation**, <u>not a message format</u>. Most industry based Web standards consist of a combination of message formats and protocol specifications. These formats have been given an operational semantics, such as, "Upon receipt of this *PurchaseOrder* message, transfer *Amount* dollars from *AccountFrom* to *AccountTo* and ship Product." But, the specification is not designed to support reasoning outside the transaction context. For example, we won't in general have a mechanism to conclude that because the Product is a type of Chardonnay it must also be a white wine.
2. **One advantage of OWL ontologies will be the availability of tools that can reason about them**. Tools will provide generic support that is not specific to the particular subject domain, which would be the case if one were to build a system to reason about a specific industry-standard XML schema. Building a sound and useful reasoning system is not a simple effort. Constructing an ontology is much more tractable.

Example:

A general example may help to understand the role of OWL Ontology. A bookseller may want to integrate data coming from different publishers. The data can be imported into a common RDF model, eg, by using converters to the publishers' databases. However, one database may use the term “author”, whereas the other may use the term “creator”. To make the integration complete and extra definition should be added to the RDF data, describing the fact that the relationship described as “author” is the same as “creator”. This extra piece of information is, in fact, a vocabulary (or an ontology), albeit an extremely simple one.
In a more complex case the application may need a more detailed ontology as part of the extra information. This may include formal description on how authors are to be uniquely identified (eg, in a US setting, by referring to a unique social security number), how the terms used in this particular application relate to other datasets on the Web (eg, Wikipedia or geographic information), how the term “author” (or “creator”) can be related to terms like “editors”, etc.

I.2.1. The Species of OWL

The OWL language provides three increasingly expressive sublanguages designed for use by specific communities of implementers and users:

- **OWL Lite** supports those users primarily needing a classification hierarchy and simple constraint features. For example, while OWL Lite supports cardinality constraints, it only permits cardinality values of 0 or 1. It should be simpler to provide tool support for OWL Lite than its more expressive relatives, and provide a quick migration path for thesauri and other taxonomies.
- **OWL DL** supports those users who want the maximum expressiveness without losing computational completeness (all entailments are guaranteed to be computed) and decidability (all computations will finish in finite time) of

reasoning systems. OWL DL includes all OWL language constructs with restrictions such as type separation (a class can not also be an individual or property; a property can not also be an individual or class). OWL DL is so named due to its correspondence with *Description Logics*, a field of research that has studied a particular decidable fragment of first order logic. OWL DL was designed to support the existing *Description Logic* business segment and has desirable computational properties for reasoning systems.

- **OWL Full** is meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. For example, in OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right. Another significant difference from OWL DL is that a *owl:DatatypeProperty* can be marked as an *owl:InverseFunctionalProperty*. OWL Full allows an ontology to augment the meaning of the pre-defined (RDF or OWL) vocabulary. It is unlikely that any reasoning software will be able to support every feature of OWL Full.

Each of these sublanguages is an extension of its simpler predecessor, both in what can be legally expressed and in what can be validly concluded. The following set of relations hold. Their inverses do not.

- Every legal OWL Lite ontology is a legal OWL DL ontology.
- Every legal OWL DL ontology is a legal OWL Full ontology.
- Every valid OWL Lite conclusion is a valid OWL DL conclusion.
- Every valid OWL DL conclusion is a valid OWL Full conclusion.

Ontology developers adopting OWL should consider which species best suits their needs. The choice between OWL Lite and OWL DL depends on the extent to which users require the more expressive restriction constructs provided by OWL DL. Reasoners for OWL Lite will have desirable computational properties. Reasoners for OWL DL, while dealing with a decidable sublanguage, will be subject to higher worst-case complexity. The choice between OWL DL and OWL Full mainly depends on the extent to which users require the meta-modelling facilities of RDF Schema (i.e. defining classes of classes). When using OWL Full as compared to OWL DL, reasoning support is less predictable. For more information about this issue see the OWL semantics document.

Users migrating from RDF to OWL DL or OWL Lite need to take care to ensure that the original RDF document complies with the constraints imposed by OWL DL and OWL Lite.

I.2.2. Ontology Mapping

In order for ontologies to have the maximum impact, they need to be widely shared. In order to minimize the intellectual effort involved in developing an ontology they need to be re-used. In the best of all possible worlds they need to be composed. For example, you might adopt a date ontology from one source and a physical location ontology from another and then extend the notion of location to include the time period during which it holds.

It is important to realize that much of the effort of developing an ontology is devoted to hooking together classes and properties in ways that maximize implications. We

want simple assertions about class membership to have broad and useful implications. This is the hardest part of ontology development. If you can find an existing ontology that has already undergone extensive use and refinement, it makes sense to adopt it.
It will be challenging to merge a collection of ontologies. Tool support will almost certainly be required to maintain consistency.

I.2.3. Further readings:

The OWL Language is described by a set of documents, each fulfilling a different purpose, and catering to a different audience. The following provides a brief roadmap for navigating through this set of documents:

- This OWL Overview gives a simple introduction to OWL by providing a language feature listing with very brief feature descriptions;
- The OWL Guide demonstrates the use of the OWL language by providing an extended example. It also provides a glossary of the terminology used in these documents;
- The OWL Reference gives a systematic and compact (but still informally stated) description of all the modelling primitives of OWL;
- The OWL Semantics and Abstract Syntax document is the final and formally stated normative definition of the language;
- The OWL Web Ontology Language Test Cases document contains a large set of test cases for the language;
- The OWL Use Cases and Requirements document contains a set of use cases for a web ontology language and compiles a set of requirements for OWL.

I.3. SPARQL (SPARQL Protocol and RDF Query Language)

Query languages go hand-in-hand with databases. If the Semantic Web is viewed as a global database, then it is easy to understand why one would need a query language for that data. SPARQL is the query language for the Semantic Web.
“Query”, in the Semantic Web context means technologies and protocols that can programmatically retrieve information from the Web of Data.
The Semantic Web is a Web of data — of dates and titles and part numbers and chemical properties and any other data one might conceive of. RDF provides the foundation for publishing and linking your data. Various technologies allow you to embed data in documents (RDFa, GRDDL) or expose what you have in SQL databases, or make it available as RDF files.
However, just as relational databases or XML need specific query languages (SQL and XQuery, respectively), the Web of Data, typically represented using RDF as a data format, needs its own, RDF-specific query language and facilities. This is provided by the SPARQLquery language and the accompanying protocols. SPARQL makes it possible to send queries and receive results, e.g., through HTTP or SOAP.
Technically, SPARQL queries are based on *(triple) patterns*. RDF can be seen as a set of relationships among resources (i.e., RDF triples); SPARQL queries provide one or more patterns against such relationships. These triple patterns are similar to RDF triples, except that one or more of the constituent resource references are variables. A SPARQL engine would returns the resources for all triples that match these patterns.

Using SPARQL consumers of the Web of Data can extract possibly complex information (i.e., existing resource references and their relationships) which are returned, for example, in a table format. This table can be incorporated into another Web page; using this approach SPARQL provides a powerful tool to build, for example, complex mash-up sites or search engines that include data stemming from the Semantic Web.

SPARQL has several query forms. The SELECT query form returns variable bindings. The CONSTRUCT query form returns an RDF graph. The graph is built based on a template which is used to generate RDF triples based on the results of matching the graph pattern of the query.

Data:

```
@prefix org:    <http://example.com/ns#> .

_:a  org:employeeName   "Alice" .
_:a  org:employeeId     12345 .

_:b  org:employeeName   "Bob" .
_:b  org:employeeId     67890 .
```

Query:

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
PREFIX org:    <http://example.com/ns#>
CONSTRUCT { ?x foaf:name ?name }
WHERE  { ?x org:employeeName ?name }
```

Results:

```
@prefix org: <http://example.com/ns#> .

_:x foaf:name "Alice" .
_:y foaf:name "Bob" .
```

which can be serialized in RDF/XML as:

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:foaf="http://xmlns.com/foaf/0.1/" >
  <rdf:Description>
    <foaf:name>Alice</foaf:name>
  </rdf:Description>
  <rdf:Description>
    <foaf:name>Bob</foaf:name>
  </rdf:Description>
</rdf:RDF>
```

I.3.1. SPARQL Query Results XML Format

RDF is used to represent, among other things, personal information, social networks, metadata about digital artifacts like music and images, as well as provide a means of integration over disparate sources of information. A standardized query language for RDF data with multiple implementations offers developers and end users a way to write and to consume the results of queries across this wide range of information.
The SPARQL language includes IRI's, a subset of RDF URI References that omits spaces. Note that all IRI's in SPARQL queries are absolute. They may or may not include a fragment identifier. IRI's include URI's and URL's. The abbreviated forms (relative IRIs and prefixed names) in the SPARQL syntax are resolved to produce absolute IRIs.

A **SPARQL Results Document** is an XML document that is valid with respect to either the RELAX NG XML Schema or the W3C XML Schema.
The SPARQL Results Document begins with sparql document element in the http://www.w3.org/2005/sparql-results# namespace, written as follows:

```
<?xml version="1.0"?>

<sparql xmlns="http://www.w3.org/2005/sparql-
results#">

 ...

</sparql>
```

Inside the **sparql** element are two sub-elements, **head** and a results element (either **results** or **boolean**) which must appear in that order.

The **head** element is the first child element of the **sparql** element. For a variable binding query result, **head** must contain a sequence of elements describing the set of Query Variable names in the Solution Sequence (here called query results).

The order of the variable names in the sequence is the order of the variable names given to the argument of the **SELECT** statement in the SPARQL query. If **SELECT *** is used, the order of the names is undefined. Inside the **head** element, the ordered sequences of variable names chosen are used to create empty child elements **variable** with the variable name as the value of an attribute **name** giving a document like this:

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-
results#">

  <head>
    <variable name="x"/>
    <variable name="hpage"/>
    <variable name="name"/>
    <variable name="mbox"/>
    <variable name="blurb"/>
  </head>
...
</sparql>
```

I.3.2. SPARQL Protocol

SPARQL Protocol is a mean of conveying SPARQL queries from query clients to query processors. SPARQL Protocol has been designed for compatibility with the SPARQL Query Language for RDF. SPARQL Protocol is described in two ways: first, as an abstract interface independent of any concrete realization, implementation, or binding to another protocol; second, as HTTP and SOAP bindings of this interface.
SPARQL Protocol contains one interface, SparqlQuery, which in turn contains one operation, query. SPARQL Protocol is described abstractly with WSDL 2.0 in terms of a web service that implements its interface, types, faults, and operations, as well as by HTTP and SOAP bindings. Note that while this document uses WSDL 2.0 to describe SPARQL Protocol, there is no obligation on the part of any implementation to use any particular implementation strategy, including the use of any WSDL library or programming language framework.
This set of documents comprises the specification of the SPARQL Protocol:

- **SPARQL Protocol for RDF** - The current document which normatively specifies the SPARQL Protocol in human-readable language.
- **SPARQL Protocol WSDL 2.0 Description** - The normative description of the SPARQL Protocol using WSDL 2.0.
- **SPARQL Protocol Types** - The XML Schema document that normatively defines the types used in SPARQL Protocol.

Further readings:

http://www.w3.org/TR/rdf-sparql-query/
http://www.w3.org/TR/rdf-sparql-XMLres/
http://openjena.org/ARQ/Tutorial/index.html
http://www.w3.org/TR/rdf-sparql-protocol/
http://www.w3.org/TR/wsdl20/

I.4. Summary

We saw that the essence of Web Semantics is focused on three basic frameworks or “components”, RDF, OWL and SPARQL.
RDF converts the content of files of different types into meanings described with a code using URI’s, which code is similar to XML. More closely this RDF code is described or explained from the Ontology. The ontology for RDF is called OWL. OWL ontology contains more precise and detailed explanation about the RDF code. In order to query and select the requested things from the Ontology and RDF files, we need a query language. The query language that understands the RDF and OWL is called SPARQL, which is a query language similar to SQL but specialized in semantic web issues. SPARQL drivel meaning from the RDF files, but also from

OWL ontologies. SPARQL is contained within the Jena Framework which is written in JAVA programming language.
All this connections are presented in the following figure:

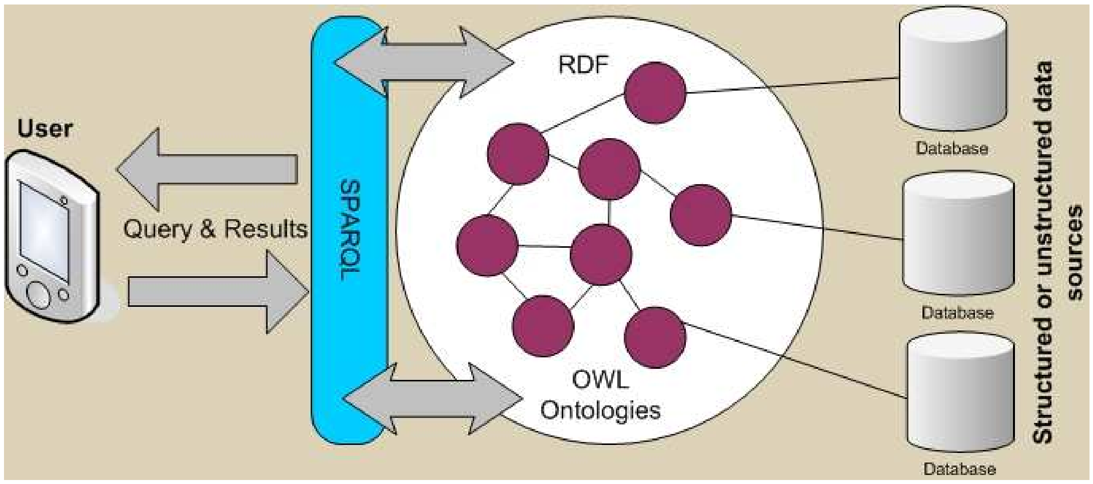

Figure 2 The interconnection between RDF, OWL and SPARQL