Semantic Web Core Technologies

Muhammad Alsharif, mhalsharif at gmail.com (A paper written under the guidance of Prof. Raj Jain)

Download 

Abstract

This is survey of semantic web primary infrastructure. Semantic web aims to link the data everywhere and make them understandable for machines as well as humans. Implementing the full vision of the semantic web has a long way to go but a number of its necessary building blocks are being standardized. In this paper, the three core technologies for linking web data, defining vocabularies, and querying information are going to be introduced and discussed.

Keywords: web 3.0, semantic web, semantic web stack, resource description framework, RDF schema, web ontology language, SPARQL

Table of Contents

1. Introduction

The main goal of the semantic web is to improve upon the existing â€œweb of documentsâ€ to be a â€œweb of dataâ€. The first version of the web (web 1.0) revolutionized the use of Internet in both academia and industry by introducing the idea of hyperlinking, which has interconnected billions of web pages over the globe and made documents accessible from multiple points. Web 2.0 has expanded over this concept by adding the user interactivity/participation element such as the dynamic creation of data on websites by content management systems (CMS) which has led to the emergence of social media such as blogs, flickr, youtube, twitter, etc. While web 1.0 and 2.0 were designed only for humans, the semantic web aims to include the machines in the process

of creation, reading and understanding data on the webpages.

2. Semantic Web Stack

The following diagram shows the architecture of semantic web, which is often called semantic web stack.

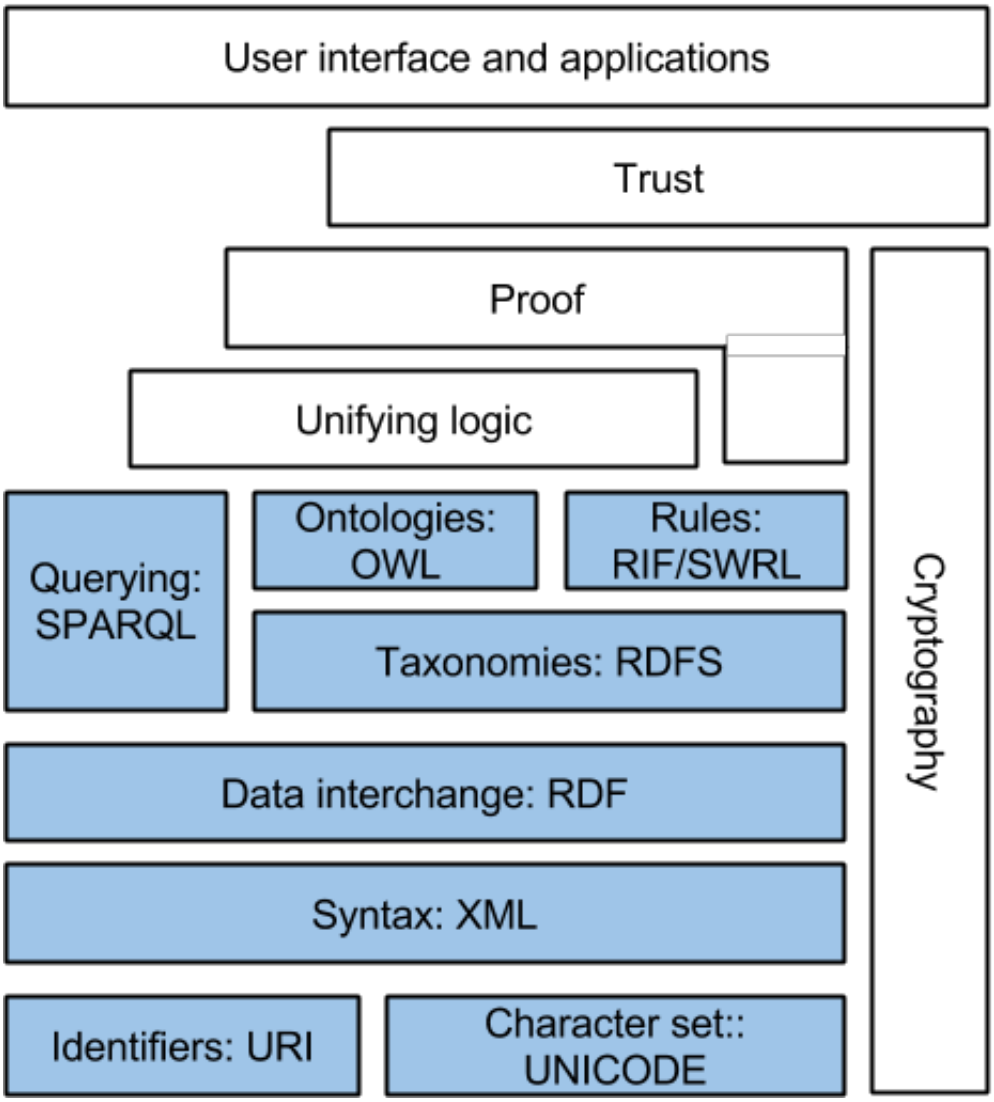

Figure 1: Semantic Web Stack, source: Wikipedia01

As shown in figure 1, there are multiple layers and each layer uses the capabilities of the layer below. The bottom layers of the stack (in blue) indicate the parts of the semantic web that have been standardized. It is still unclear for the upper layers (not colored) how to be implemented which are necessary to have the full working semantic web [Wikipedia01].

In the following sections, the main technologies of the semantic web will be described. This includes: RDF, RDFS, OWL and SPARQL.

3. Linked Data

To achieve the semantic web vision of evolving the web into a web of data, the means of how to create linked data and represent them have to be established. In this section we will explore the concept of Resource

Description Framework and then compare it to XML.

3.1. Resource Description Framework (RDF)

RDF (Resource Description Framework) is a data model that is used to represent all data in the semantic web and the relationship between them. In computer science terms, RDF is a labeled directed graph [Gonzalez1]. This means that all nodes and edges of the graph are labeled. The following diagram shows an example of two RDF statements.

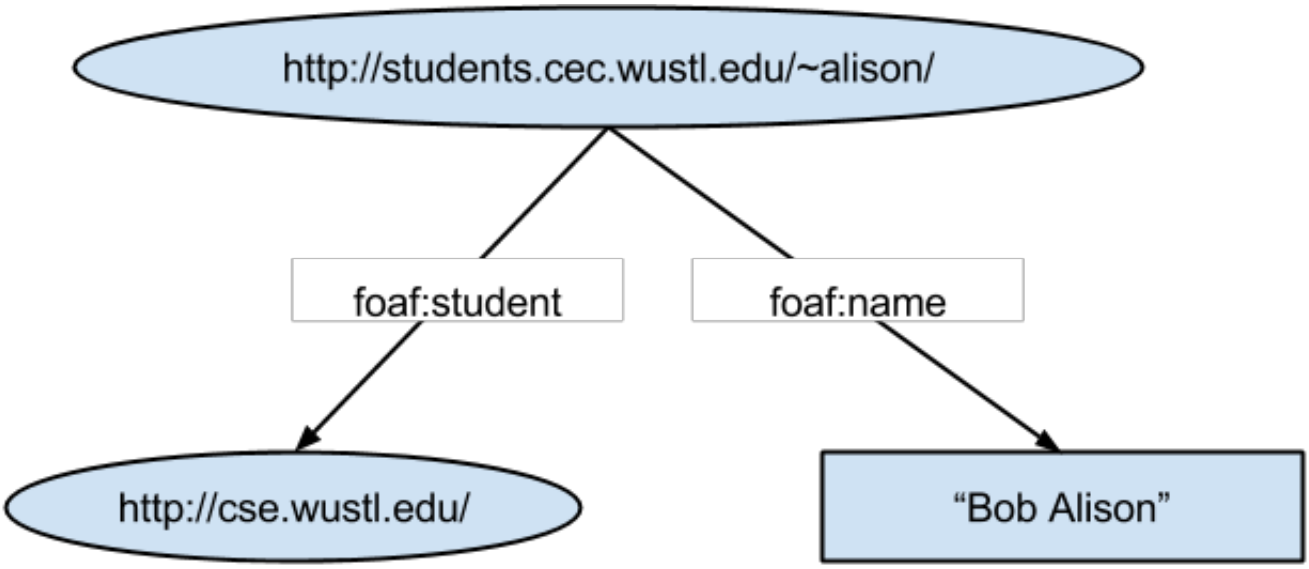

Figure 2: Graph representation of two RDF triples

Figure 2 shows two types of nodes, resource nodes (oval) and literal node (square). The resource node represent an object in which there is something to say about. Resource nodes always use URI (Unique Resource Identifier) as the label of the node. The URI does not necessarily have to refer to a webpage but could be a web address of an entity such as a person, an institution or any world object. The literal node is used to indicate a value of an object property such as a name, a date, a number, etc. Unlike resource nodes, where the label of the node is a URI, the label of a literal node is always a value [Gonzalez1].

The directed edges in figure 2 are used to indicate the relationship between different nodes, and are labeled with URIs, as well. In the example above â€œfoafâ€ was used as shorthand of the full URIs:

- foaf:student full URI is: http://xmlns.com/foaf/0.1/student
- foaf:name full URI is: http://xmlns.com/foaf/0.1/name

Generally speaking, the edges used in RDF could be one of two cases: resource-to-resource or resource-to-value. [Gonzalez1]. In the example above, we have two edges that represent both cases. From the graph, we can write two statements (also called triples) in the form [**subject, predicate, object**] as following:

- [wustl-student:alison, foaf:name, "Bob Alison"]
- [wustl-student:alison, foaf:student, http://cse.wustl.edu/]

The use of URIs in labeling the resource nodes and edges is of central importance in RDF because it removes the possible ambiguity in terminologies [W3C 01]. For example, without a URI, the â€œnameâ€ relation could

be interpreted as a nickname, a first name, a name of a street, a place, or an animal. When a URI is used, however, a detailed explanation of what â€œnameâ€ specifically means here is referred by the URI, which could be accessed by both humans and machines [W3C 01].

Triplestores are used to store and retrieve billions of triples. They could be thought of as databases for RDF statements. There are many varieties of triplestores available on the web such as AllegroGraph, ARC2, BigData, Jena, OntoBroker, Pointrel, RDF-3X, etc. In order to build well-defined resources and relationships for our own triplestore, we need to define RDF vocabularies, which is the topic of the following section.

3.2. RDF vs XML

Although XML (Extensible Markup Language) is a widely used language for encoding documents and representing data structures, it cannot be a real substitute for RDF for building the semantic web. XML is meant to be a *serialization format* so that information could be parsed appropriately when they are transferred between machines [Gonzalez2]. On the other hand, RDF is a *data model*, which aims to represent information and the relationship in an abstract level. This is why there is no unique RDF serialization format but many in use such as: Turtl, RDFa, Notation 3, and RDF/XML.

Semantic web needs a way to convey semantics and thus comes RDF. In contrast, an XML document is a data format, which has no built-in meaning or semantic. A useful analogy used by [Gonzalez2] to illustrate the difference is the formats of a book and the information inside it. A book could be published in different formats such as a paperback, a hardcopy, an audiobook or, an electronic book. However, the same content/information are conveyed in all these formats. Similarly, a data on website could be serialized in different formats such as XML, JSON, etc. However, the semantic it conveys would be the same.

4. Vocabularies

RDF defines the framework in which the data and their relationship are modeled. To reduce the ambiguity in RDF, especially when integrating triples from different RDf triplestores, a set of vocabularies that defines common relationships between entities are defined and used. In this section, the RDF Schema will be introduced first and then compared with the more sophisticated Web Ontology Language (OWL). The words â€˜vocabularyâ€™ and â€˜ontologyâ€™ here are used interchangeably.

4.1. RDFS

Resource Description Framework Schema (or RDFS) is a lightweight, easy to use language for defining RDF vocabularies (i.e. ontology). RDFS is expressed in RDF and used to define object-oriented concepts such as classes and properties [Tester1]. For example, in order to define the class â€œStudentâ€ in a hypothetical wustl triplestore, we write:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix wustl: <http://www.wustl.edu/> .
wustl:Student   rdf:type   rdf:Class
```

In the example above, we first defined rdf, rdfs, xsd and wustl as shorthands for the full URIs so we donâ€™t need to type their full URIs every time we write a triple. We then, linked wustl:Student to the resource rdf:Class using rdf:type. This defined the resource wustl:Student as a class. Now we can create the instances of the student class as following:

```
wustl:Bob    rdf:type   wustl:Student
wustl:Alice  rdf:type   wustl:Student
```

In order to create properties for our Student class, we could use rdf:Property. For instance, we can define the following two properties:

```
wustl:ID     rdf:type   rdfs:Property
wustl:DoB    rdf:type   rdfs:Property
```

However, these two properties are generic and do not belong to the Student class yet. In order to do define them as properties for that class, we simply add rdfs:Domain and rdfs:Range triples as following:

```
wustl:ID     rdfs:Domain    wustl:Student
wustl:ID     rdfs:Range     xsd:int
wustl:DoB    rdfs:Domain    wustl:Student
wustl:DoB    rdfs:Range     xsd:Date
```

By defining the domain of the ID property as the Student class and the range as an integer, we simply are saying that wustl:ID can only link a resource of type wustl:Student to a resource of type xsd:int (i.e. integer). Similarly, wustl:DoB can only link a resource of type wustl:Student to a resource of type xsd:Date (i.e. date). Using the previous example of Bob and Alice (who are instances of Student class), we can write:

```
wustl:Bob    wustl:ID     123456
wustl:Alice  wustl:DoB    01/01/1993
```

RDFS also allows inheritance using constructs such as rdfs:subClassOf and rdfs:subPropertyOf. This is useful when we want to represent hierarchies among classes such as â€œa square is a rectangleâ€ and â€œa rectangle is a polygonâ€. In the example of our Student class, instead of defining Student as a base class, we could define it as a subclass of foaf:Person by writing:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
wustl:Student    rdfs:subClassOf    foaf:Person
```

This would entail that all instances of wustl:Student are also instances of foaf:Person and all properties of foaf:Person are properties of wustl:Student [Tester1]. Using inheritance, therefore, could significantly reduce the amount of entries used in a triplestore by avoiding the repetition whenever possible.

4.2. RDFS vs Web Ontology Language (OWL):

Like RDFS, OWL is a data modeling language used to describe RDF vocabularies. All constructs that were defined in RDFS such as rdfs:subClassOf, rdfs:Domain, rdfs:Range, etc, could all be used in OWL. However, OWL provides much larger set of vocabularies that could be used to say a lot of things about the relationship

between data [Tester2]. For example, we can use owl:intersection and owl:unionOf to describe Father, and Parent respectively as following [Tester3]:

@prefix foaf: <http://xmlns.com/foaf/0.1/> .
foaf:Father owl:intersectionOf (foaf:Male, foaf:Parent)
foaf:Parent owl:unionOf (foaf:Father, foaf:Mother)

Another major OWL distinction from RDFS is its rigidity. While RDFS tells you how you can use vocabularies, OWL could also tell you how you **cannot** use them [Tester2]. For example, the non-rigidity of RDFS could define a cat as an instance of animal as well as a subclass of animal at the same time. By contrast, this may not be permitted in some flavors of OWL.

The graph example in figure 3 was slightly adopted from [Obitko1]. It shows how complex relations could be build with the use of OWL. The example basically states that a â€œNon-vegetarian Pizza is equivalent to the intersection of Pizza and the complement of Vegetarian Pizza; Pizza has PizzaBase as its base; Pizza is disjoint with PizzaBaseâ€. Blank ovals (resource) are used to here to indicate results of operation.

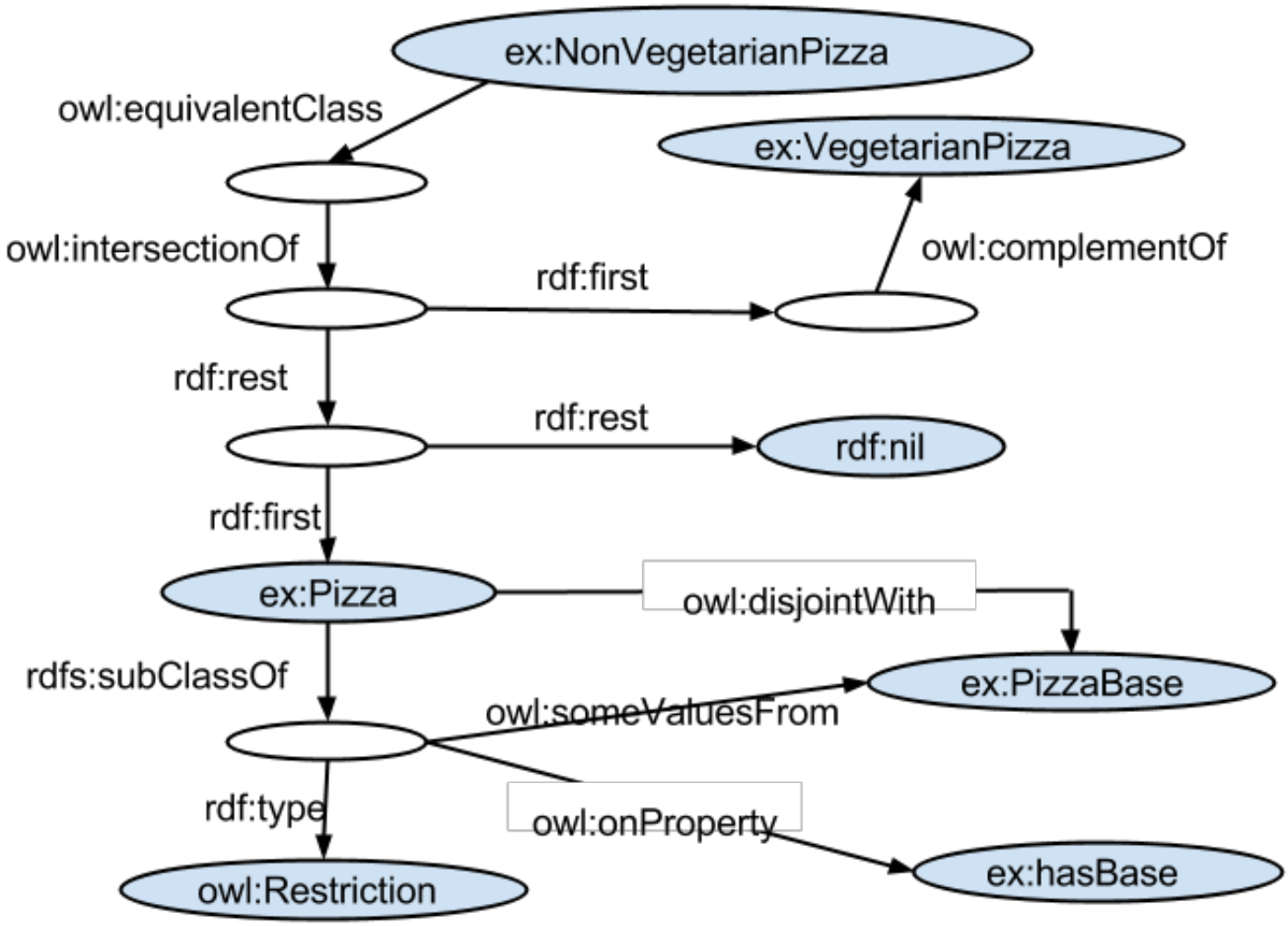

Figure 3: OWL example of Pizza ontology, source [Obitko1]

Building large set of triples and vocabularies would be of no use without the ability to query them and extract useful information. This is discussed in the next section.

5. Query

Defining a query language that can retrieve information effectively and integrate them from different RDF triplestores is an essential functional block for a working semantic web. In this section, we will review SPARQL (pronounced sparkle) and how it is compared with SQL.

5.1. SPARQL

SPARQL is an acronym of a recursive nature, which stands for **S**PARQL **P**rotocol and **R**DF **Q**uery **L**anguage [Wikipedia02]. As the definition indicates, it consists of two main parts: the query language and the protocol.

SPARQL query is the query language for the semantic web. SPARQL queries are based on graph pattern matching [Sequeda1]. A typical SPARQL query is an RDF triple with some variables, and the returned result from the query is the triples that match the pattern of the query. There are four forms of return clauses [Sequeda2]:

- SELECT: similar to SQLâ€™s SELECT, it is used to extract values of triples that match a certain pattern.
- ASK: Is used to check whether there is at least one result that matches a matching pattern.
- DESCRIBE: is used to obtain an RDF graph, which describes a resource.
- CONSTRUCT: is used to obtain an RDF graph that matches a pattern and transform it into a provided template.

The last version of SPARQL is 1.1, which is in the process of standardization, and it introduces many more needed concepts such as: Aggregates, Projected expressions, Negation, Update, Sub-queries [W3C02].

Unlike relational databases where SQL queries require establishing a connection to the database first, SPARQL protocol enables sending SPARQL queries over HTTP requests. A service that implements the SPARQL protocol over HTTP is called SPARQL endpoint [Sequeda1]. DBpedia has a popular end point, which is useful for testing SPARQL queries [DBpedia01].

5.2. SPARQL vs SQL

While SQL is the query used for relational databases, SPARQL could be used for both RDF databases as well as relational database [Prud'hommeaux1]. In the following example we will show how the same question (query) could be asked using both SQL and SPAQL.

Letâ€™s say that we want to retrieve the id and date of birth of all male students whose home address are in St. Louis. For an SQL query, we write:

SELECT Student.id, Student.dob
FROM Student, Address
WHERE Student.gender = â€œmaleâ€
AND Student.haddress =Address.id
AND Address.city = â€œSt. Louisâ€

Assuming that there are Student table and Address table, the query will use id matching to join the Student table

to the Address. In contrast, writing SPARQL query to perform the same operation could look like:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX wustl: <http://www.wustl.edu/>
SELECT ?id ?dob
WHERE {
?student   rdf:type     wustl:Student.
?student   wustl:gender    â€œmaleâ€.
?student   wustl:address  ?addr.
?addr       wustl:city         â€St. Louisâ€.
?student   wustl:id           ?id.
?student   wustl:dob         ?dob.
}
```

In the SPARQL query, words that were preceded by â€œ?â€ are called variables. Variables could represent a resource (i.e. URL) or a value. The query would return all IDs and DOBs which â€“when substituted- would satisfy all the conditions inside the WHERE clause. Unlike querying relational databases by SQL, SPARQL uses variable binding to resolve the query. Another difference in SPARQLâ€™s SELECT is that we have to mention the variables id and dob inside the WHERE clause if we want to return them in the final result. This is not necessarily the case with SQL queries.

6. Summary:

In this paper, I surveyed the core technologies of the next generation web. First, the semantic web is a web of data so it uses the Resource Description Framework (RDF) to link resources and form triplestores. RDF Schema (RDFS) is a small, lightweight language for structuring RDF resources using object-oriented concepts such as a class, instance, property, subclass etc. Web Ontology Language (OWL) expands over RDFS by adding far more constructs to describe complex relations such as union, complement of class, equivalent class etc. Finally, in order to extract useful information from RDF triplestores, SPARQL is the query language to use. It also provides a protocol that allows sending queries by using HTTP.

7. Acronyms:

- CMS: Content Management System
- HTTP: Hypertext Transfer Protocol
- JSON: JavaScript Object Notation
- OWL: Web Ontology Language
- RDF: Resource Description Framework
- RDFS: Resource Description Framework Schema
- SPARQL: SPARQL Protocol And RDF Query Language
- SQL: Structured Query Language
- URI: Uniform Resource Identifier
- URL: Uniform Resource Locator
- XML: Extensible Markup Language

8. References:

[W3C 01]:	W3C Working Group, "RDFa 1.1 Primer - Second Edition", 22 August 2013, http://www.w3.org/TR/rdfa-primer/
[W3C 02]:	W3C Working Group, "SPARQL 1.1 Query Language", 21 March 2013, http://www.w3.org/TR/2013/REC-sparql11-query-20130321/
[W3C 03]:	W3C Working Group, "OWL 2 Web Ontology Language Primer (Second Edition)", 11 December 2012, http://www.w3.org/TR/2012/REC-owl2-primer-20121211/
[Gonzalez1]:	Rob Gonzalez, "RDF 101", http://www.cambridgesemantics.com/semantic-university/rdf-101
[Gonzalez2]:	Rob Gonzalez, "RDF vs XML", http://www.cambridgesemantics.com/semantic-university/rdf-vs-xml
[Tester1]:	David Tester, "RDFS Introduction", http://www.cambridgesemantics.com/semantic-university/rdfs-introduction
[Tester2]:	David Tester, "RDFS vs. OWL", http://www.cambridgesemantics.com/semantic-university/rdfs-vs.-owl
[Tester3]:	David Tester, "OWL 101", http://www.cambridgesemantics.com/semantic-university/owl-101
[Sequeda1]:	Juan Sequeda, "SPARQL 101", http://www.cambridgesemantics.com/semantic-university/sparql-101
[Sequeda2]:	Juan Sequeda, "SPARQL Nuts & Bolts", http://www.cambridgesemantics.com/semantic-university/sparql-nuts-and-bolts
[Prud'hommeaux1]:	Eric Prud'hommeaux, "SPARQL vs. SQL - Intro", http://www.cambridgesemantics.com/semantic-university/sparql-vs-sql-intro
[Obitko1]:	Marek Obitko, "OWL Example with RDF Graph", http://www.obitko.com/tutorials/ontologies-semantic-web/owl-example-with-rdf-graph.html
[DBpedia01]:	DBpedia, Retrieved 18 November 2013, http://dbpedia.org/sparql/
[Wikipedia01]:	â€œSemantic Web Stack â€“ Wikipedia, the free encyclopediaâ€, Retrieved 18 November 2013, from http://en.wikipedia.org/wiki/Semantic_Web_Stack
[Wikipedia02]:	â€œSPARQL â€“ Wikipedia, the free encyclopediaâ€, Retrieved 18 November 2013, from http://en.wikipedia.org/wiki/SPARQL

Last Modified: December 10, 2013
This and other papers on latest advances in computer networking are available on line at
http://www.cse.wustl.edu/~jain/cse570-13/index.html
Back to Raj Jain's Home Page