UNIT-2

ONTOLOGIES AND THEIR ROLE IN THE SEMANTIC WEB

Ontology based Knowledge Representation:

The ontology and ontology languages may be viewed as one of the important technology in semantic web. Ontology can be described as formal allotment of conceptualization band to domain. This implies it provides the description of concepts and their corresponding relationships. In particular, the ontologies are designed as domain models that broadly appease or satisfy two special characteristics signifying the semantics. They are,

1. Ontologies are expressed in formal languages with a well-defined semantics.

The first point underlines that ontology needs to be modelled using languages with a formal semantics such languages include RDF and OWL. These languages are treated as most frequently used languages in semantic web. These languages contain those models which are prefferred by term ontology.

2. Ontologies build upon a shared understanding within a *community*.

This understanding represents an agreement among members of the community over the concepts and relationships that are present in a domain and their usage.

RDF and OWL, the ontology languages, have standardized syntaxes and logic-based formal semantics. RDF and OWL are the languages most commonly used on the Semantic Web, and in fact when using the term ontology many practitioners refer to domain models described in one of these two languages. The second point reminds as that there is no such thing as a ―personal ontology‖. For example, the schema of a database or a UML class diagram that we have created for the design of our own application is not an ontology. It is a conceptual model of a domain, but it is not shared: there is no commitment toward this schema from anyone else but us.

The simplest structures are **glossaries** or controlled vocabularies, in essence an agreement on the meaning of a set of terms.

Semantic networks are essentially graphs that show also how terms are related to each other**.**

Thesauri are richer structures in that they describe a hierarchy between concepts and typically also allow describing related terms and aliases. Thesauri are also the simplest structures where logic-based reasoning can be applied: the broadernarrower relationships of these hierarchies are transitive, in that an item that belongs to a narrower category also belongs to its direct parent and all of its ancestors.

Folksonomy structures are regarded as weaker models that do not contain any explicit hierarchies, often compries extra corresponding to the social context of tags, i.e. the set of users who have been using them. These structures again success in extracting hierarchies and also relationship among the tags.

The term ***lightweight ontology*** is typically applied to ontologies that make a distinction between classes, instances and properties, but contain minimal descriptions of them.

On the other hand, ***heavyweight ontologies*** allow to describe more precisely how classes are composed of other classes, and provide a richer set of constructs to constrain how properties can be applied to classes. At the far end of the spectrum are complex knowledge bases that use the full expressivity of **first order logic (FOL)** to define to a great detail the kind of instances a concept may have and in which cases two instances may be related using a certain relationship. The more constrained the descriptions of concepts are, the less likely that their meaning will be misinterpreted by human readers.

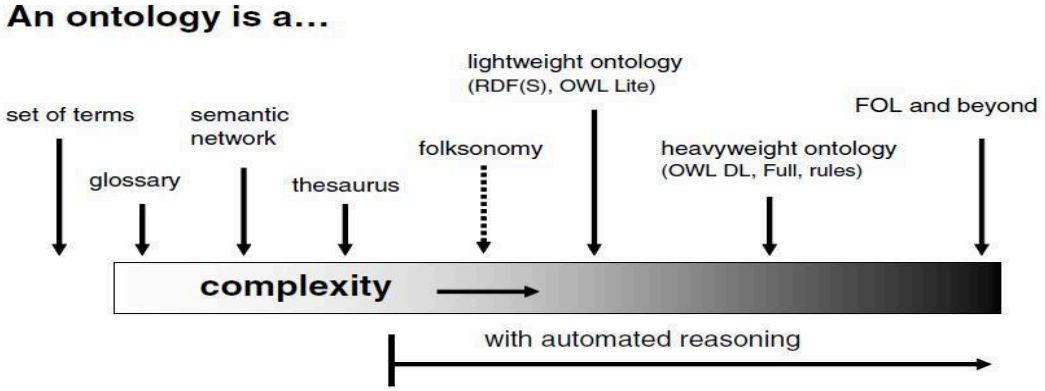


Figure 4.1. Ontologies can be organized according to complexity (informally, the level of semantics).

In practice, the most commonWeb ontologies are all lightweight ontologies due to the need of serving the needs of many applications with divergent goals. Widely shared Web ontologies also tend to be small as they contain only the terms that are agreed on by a broad user base. Large, heavyweight ontologies are more commonly found in targeted expert systems used in focused domains with a tradition of formalized processes and vocabularies such as the area of life sciences and engineering.

Ontologies and ontology languages for the SemanticWeb:

Although the notion of ontologies is independent of the Web, ontologies play a special role in the architecture of the Semantic Web.

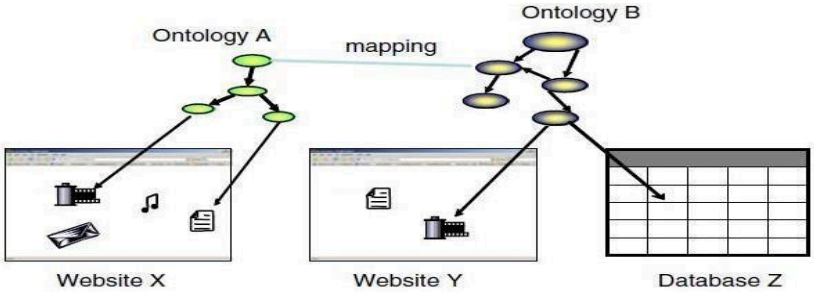


Figure 4.2. The Semantic Web in envisioned as a network of ontologies that adds machine-processable semantics to existing web content, including Web resources and databases.

This architecture provides the main motivation for the design of ontology languages for the SemanticWeb: RDF and OWL are both prepared for the distributed and open environment of the Web.

The Semantic Web will be realized by annotating existing Web resources with ontology-based metadata and by exposing the content of databases by publishing the data and the schema in one of the standard ontology languages.

Ontology languages designed for the SemanticWeb provide themeans to identify concepts in ontologies using globally unique identifiers (URIs). These identifiers can be used in data sources to point to a concept or relationship from an external, public ontology. Similar to creating HTML pages and linking them to existing ones, anyone can create and publish an ontology, which may reference the concepts in remote ontologies. Much like the hyperlinks among web pages, it is expected that these references will form a contiguousweb linking all knowledge sources across theWeb.

As URLs are also URIs, ontologies can also reference existing Web resources and describe their characteristics.

Semantic Web applications collect or query such data sources, aggregate and reason with the results. Information described according to a single schema that is known by the application developer can be directly consumed: by committing to ontology the parties involved have already implicitly agreed on the meaning of the information.

When annotating Web content or exposing the content of a database, one may choose to create ontology from scratch, or reuse an existing ontology while possibly extending it. However, as there is no coordination of any kind in reusing ontologies, it may happen that two communities develop ontologies that cover the same or overlapping domains.

Ontologies Languages for the Semantic Web:

We introduce the ontology languages RDF and OWL, which have been standardized in recent years by the World Wide Web Consortium.

The Resource Description Framework (RDF) / RDF Schema:

The Resource Description Framework (RDF) was originally created to describe resources on the World Wide Web. In reality, RDF is domain-independent and can be used to model both real world objects and information resources. RDF itself is a very primitive modeling language, but it is the basis of more complex languages such as OWL.

There are two kinds of primitives in RDF: *resources and literals*

The definition of a resource is intentionally vague; in general everything is modelled as a resource that can be (potentially) identified and described. Resources are either identified by a URI or left blank. Blank resources (*blank nodes*) are the existential quantifiers of the language: they are resources with an identity, but whose identifier is not known or irrelevant.

Literals are strings (character literals) with optional language and datatype identifiers.

Expressions are formed bymaking statements (*triples*) of the form (subject, predicate, and object). The subject of a statement must be a resource (blank or with a URI), the predicate must be a URI and the object can be either kind of resource or a literal. Literals are thus only allowed at the end of a statement.

The eXtensible Markup Language (XML) is a universal meta-language for defining markup. It provides a uniform framework for exchanging data between applications. It builds upon the original and most basic layer of the Web, Hypertext Markup Language (HTML). However, XML does not provide a mechanism to deal with the semantics (the meaning) of data.

Resource Description Framework (RDF) was developed by the World Wide Web Consortium (W3C) for Web-based metadata in order to build and extend XML. The goal of RDF is to make work easier for autonomous agents and automated services by supplying a rudimentary semantic capability.

The RDF is a format for data that uses a simple relational model that allows structured and semistructured data to be mixed, exported, and shared across different applications. It is a data model for objects and relationships between them and is constructed with an object-attribute-value triple called a statement. While XML provides interoperability within one application (e.g., producing and exchanging bank statements) using a given schema, RDF provides interoperability *across* applications (e.g., importing bank statements into a tax calculating program).

HTML LANGUAGE

In 1990, when Tim Berners-Lee laid the foundation for the World Wide Web, he included three primary components: HTTP (Hypertext Transfer Protocol), URLs (Universal Resource Locators), and HTML (Hypertext Markup Language).

These three components represented the essential ingredients leading to the explosive growth of the World Wide Web. The original idea behind HTML was a modest one. Browsers, such as Internet Explorer or Netscape Navigator, could view information on Web pages written in HTML. The HTML program can be written to a simple text file that is recognized by a browser application and can also be called embedded script programming.

The following listing of HTML markup tags is a HTML ―Hello World‖ example consisting of root tags (<HTML>), head tags (<HEAD>), and body tags (<BODY>) with the displayed information wedged in between the appropriate tags:

```
<HTML>
<HEAD>
<TITLE>My Title</TITLE>
</HEAD>
<BODY>
Hello World
</BODY>
</HTML>
```

In particular, Web applications, such as Web Services, required a means to explicitly manipulate data. This motivated the development of XML.

XML LANGUAGE

The HTML program is not extensible. That is, it has specifically designed tags that require universal agreement before changes can be made. Although over the years, Microsoft was able to add tags that work only in Internet Explorer, and Netscape was able to add tags that work only in Navigator, Web site developers had no way of adding their own tags. The solution was XML. Proposed in late 1996 by the W3C, it offered developers a way to identify and manipulate their own structured data.

The XML document simplified the process of defining and using metadata.

XML is not a replacement, but rather a complementary technology to HTML. While XML is already widely used across the Web today, it is still a relatively new technology. The XML is a meta language, which means it is a language used to create other languages. It can provide a basic structure and set of rules for developing other markup languages.

The XML document lets you name the tags anything you want, unlike HTML, which limits you to predefined tag names. You can choose element names that make sense in the context of the document. Tag names are case-sensitive, although either case may be used as long as the opening and closing tag names are consistent.

The text between the tags is the content of the document, raw information that may be the body of a message, a title, or a field of data. In its simplest form, an XML document is comprised of one or more named elements organized into a nested hierarchy. An element consists of an opening tag, some data, and a closing tag. For any given element, the name of the opening tag must match that of the closing tag. A closing tag is identical to an opening tag except that the less-than symbol (<) is immediately followed by a forward-slash (/). Keeping this simple view, we can construct the major portions of the XML document to include the following six ingredients: (*1*) XML declaration (required), (*2*) Document Type Definition (or XML Schema), (*3*) elements (required), (*4*) attributes, (*5*) entity, and (*6*) notations.

An example of a well-formed XML declaration is
<?xml version=―1.0‖ encoding=―iso-8859-1‖ standalone="yes"?>

Following the XML declaration is a document type declaration that links to a DTD in a separate file. This is followed by a set of declarations. These parts together comprise the prolog. A simple XML ―Hello World‖ example follows:

```
<?xml version="1.0"?>
<!DOCTYPE message [
<!ELEMENT message (#PCDATA)>
]>
<message>
Hello World!
</message>
```

In addition, XML is both a powerful and essential language for Web Services. It is the open standard that allows data to be exchanged between applications and databases over the Web.

XML does not offer semantics and logic capabilities. The next step up the markup language pyramid is RDF, which begins to establish a basis for semantics on the Web.

RDF LANGUAGE

The XML tags can often add meaning to data, however, actually understanding the tags is meaningful only to humans. For example, given the following segment of XML markup tags:

```
<book>
<title>G¨odel, Escher, Bach: An Eternal Golden Braid<title>
</book>
```

A human might infer that: ―The book has the title *G¨odel, Escher, Bach: An Eternal Golden Braid*.‖ This simple grammatical sentence is understood to contain **three basic parts: a subject** [The book], **a predicate** [has title], and **an object** [*G¨odel, Escher, Bach: An Eternal*

Golden Braid].

Regardless, the computer would not take action based upon this string (e.g., checking to see related titles, prices, availability, etc.) without additional explicit programming.

For machines to do more automatically, it is necessary to go beyond the notion of the HTML display model, or XML data model, toward a ―meaning‖ model. This is where RDF and metadata can provide new machine-processing capabilities built upon XML technology.

What is metadata? It is information about other data. Building upon XML, the W3C developed the RDF metadata standard. The goal was to add semantics defined on top of XML.

While RDF is actually built upon a very simple model and it can support very large-scale information processing. An RDF document can delineate precise relationships between vocabulary items by constructing a grammatical representation.

RDF Triple

The RDF model is based on statements made about resources that can be anything with an associated URI (Universal Resource Identifier). The ***basic RDF model produces a triple***, where **a resource** (the subject) is linked through an arc labeled with a property (the predicate) to a value (the object).

The RDF statements can be represented as - ***A resource[subject] has a property[predicate] with a specific value[object].***

This can be reduced to a triple: (subject, predicate, object)
Subject, predicate, and object can be defined in terms of resources, properties, and value as:

Subject: The resource (a person, place, or thing) that the statement describes. A RDF resource can be anything in the data model (document, user, product, etc) and is uniquely identified by a URI. A URI can be a URL (Universal Resource Locator).

Predicate: The property (name, city, title, color, shape, characteristic) of the subject (person, place, or thing) and is uniquely identified by a URI.

Object: The value (Douglas R. Hofstadter, San Jose, ―*G¨odel, Escher,Bach: An Eternal Golden Braid*,‖ blue, circle, strong) can be specified for the property (name, city, title, color, shape, characteristic), which describes the subject (person, place, or thing). This value can be any valid RDF data type. (RDF supports all of the XML data types.)

This simple model of the triple with URIs used by RDF to describe information has many advantages. One of the most important is that any data model can be reduced to a common storage format based on a triple.

This makes RDF ideal for aggregating disparate data models because all the data from all models can be treated the same. This means that information can be combined from many sources and processed as if it came from a single source.

The RDF relationships can be between two resources or between a resource and a literal. These relationships form arcs. The RDF arc can be graphically represented where the subject is shown as an oval, the predicate as a connecting arc or line, and the object as an oval. Graphs are easy to read and the directed arc removes any possibility of confusion over what are the subject and the objects.

Let us examine a very simple statement and identify the components that comprise an RDF model:

Ex: Consider this sentence as an RDF Statement

―The book has the title *G¨odel, Escher, Bach: An Eternal Golden Braid*.‖

―The book [subject] has the title [predicate] *G¨odel, Escher, Bach: An Eternal Golden Braid* [object].‖

This can be represented as the triple:

(The book has the title, *G¨odel, Escher, Bach: An Eternal Golden Braid*).

It is a directed graph with labeled nodes and labeled arcs. The arc is directed from the resource (the subject) to the value (the object), and this kind of graph is recognized in the AI community as a semantic net.

We can think of the triple (*x*,P,*y*) as a logical formula P(*x*,*y*) where the binary predicate P relates the object *x* to the object *y*.

Applying this to our triple:

(The book, has the title, *G¨odel, Escher, Bach: An Eternal Golden Braid*)

Produces a logical formula:

_has the title‘ (The book, *G¨odel, Escher, Bach: An Eternal Golden Braid*)

Where the binary predicate (P): _has the title‘

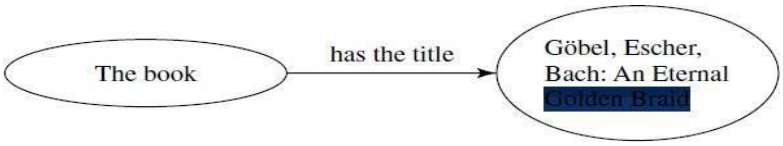


Fig: Graphical representation of the RDF statement

relates the object (*x*): The book

to the object (*y*): *G¨odel, Escher, Bach: An Eternal Golden Braid.*

Think of a collection of interrelated RDF statements represented as a graph of interconnected nodes. The nodes are connected via various relationships. For example, let us say each node represents a person. Each person might be related to another person because they are siblings, parents, spouses, friends, or employees.

Each interconnection is labeled with the relationship name.

The RDF is used in this manner to describe these relationships. It does not actually include the nodes directly, but it does indirectly since the relationships point to the nodes. At any time, we could introduce a new node, such as a newborn child, and all that is needed is for us to add the appropriate relationship for the two parents.

BASIC ELEMENTS

Most of the elements of RDF concern classes, properties, and instances of classes.

Syntax

Both RDF and RDF Schema (RDFS) use XML-based syntax.

The RDF system provides a means of describing the relationships among resources in terms of named properties and values. Since RDF and XML were developed about the same time, RDF was defined as an excellent complement to XML. Encoding RDF triples in XML makes an

object portable across platforms and interoperable among applications. Because RDF data can be expressed using XML syntax, it can be passed over the Web as a document and parsed using existing XML-based software. This combination of RDF and XML enables individuals or programs to locate, retrieve, process, store, or manage the information objects that comprise a Semantic Web site.

Header

An RDF Document looks very much like all XML documents in terms of elements, tags, and namespaces. An RDF document starts with a header including the root element as an ***"rdf:RDF"*** element that also specifies a number of namespaces. It then defines properties and classes.

Document Parts	RDF Document
Header–XML Syntax declaration	`<?xml version="1.0"?>`
Root element tag	`<rdf:RDF`
XML namespaces for rdf and dc, as well as, the URLs where they are defined.	`xmlns:rdf=http://www.w3.org/1999/02/ 22-rdf-syntax-ns#xmlns:dc= "http://purl.org/dc/elements/1.1/">`
Inserting the Triple (subject, predicate, object) within the code.	`<rdf:Description rdf:about="SUBJECT"> <dc:PREDICATE>"OBJECT"</dc:PREDICATE> </rdf:Description>`
End of root element indicates end of RDF document.	`</rdf:RDF>`

Table: RDF document parts (header, XML syntax, root element, namespace, the RDF triple, and the end element)

Namespaces

The namespace mechanism of XML is also used in RDF. However, in XML, namespaces are only used to remove ambiguities. In RDF, external namespaces are expected to be RDF documents defining resources, which are used to import RDF documents.

To add a namespace to an RDF document, a namespace attribute can be added anywhere in the document, but is usually added to the RDF tag itself. The namespace declaration for RDF vocabularies usually points to a URI of the RDF Schema document for the vocabulary. We can add a namespace as:

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

Note, the prefix for the RDF Syntax is given as ―rdf,‖ the RDF Schema is given as ―rdfs,‖ and the Dublin Core schema (a special publication ontology) is given as ―dc.‖ DC is a well- established RDF vocabulary for publications.

Description

The **"rdf:about"** attribute of the element ―rdf:Description‖ is equivalent to that of an ID attribute, but is often used to suggest the object may be defined elsewhere. A set of RDF statements form a large graph relating things to other things through their properties. The content of **"rdf:Description"** elements are called property elements. The **"rdf:resource"** attribute and the **"rdf:type"** element introduces structure to the rdf document.

While RDF is required to be well formed, it does not require XML-style validation. The RDF parsers do not use Document Type Definitions (DTDs) or XML Schema to ensure that the RDF is valid.

Data Types

Sometimes it is useful to be able to identify what kind of thing a resource is, much like how object-oriented systems use classes. The RDF system uses a type for this purpose. While there are two very general types, ***a resource and a literal***, every resource may be given a precise type.

For example, the resource ―John‖ might be given a type of ―Person.‖ The value of the type
should be another resource that would mean that more information could be associated with the type itself.

As with other properties, types can be specified with a triple:

<http://www.web-iq.com/people/John>,
rdf:type, http://xmlns.com/wordnet/1.6/Person

EX: ―The book has the title
G¨odel, Escher, Bach: An Eternal Golden Braid,‖ as:

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#
xmlns:dc="http://purl.org/dc/elements/1.1/">
<rdf:Description rdf:about=―http://www.amazon.com/books‖>
<dc:title>G¨odel, Escher, Bach: An Eternal Golden Braid</dc:title>
</rdf:Description>
</rdf:RDF>
```

Vocabularies

Any kind of business resource vocabularies can be used to model business resources using the syntax of RDF.

Because RDF creates domain-specific vocabularies that are then used to model resources, we can use RDF to model business-specific resources. The only limitation is the need for industry cooperation in developing an interoperable vocabulary. We can consider RDF as a way of recording information about resources.

The RDF recorded in XML is a powerful tool. By using XML we have access to a great number of existing XML applications, such as parsers and APIs.

Classes and Properties

The RDF and RDF Schema (RDFS) classes and properties can be found at: ***RDF W3C specifications***

RDF Model and Syntax Specification:
http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/

RDFS Specification:
http://www.w3.org/TR/2003/WD-rdf-schema-20030123/

Collections

A collection is considered to be a finite grouping of items with a given terminator.

Within RDF, a collection is defined through the use of **rdf:parseType** = "Collection" and through listing the collected resources within the collection block.

Reification

The RDF allows us to make statements about statements using a reification mechanism. This is particularly useful to describe belief or trust in other statements.

The following example discusses the interpretation of multiple statements in relationship to RDF statements.

Interpreting Multiple Sentences as RDF Statement:

Let us start with five simple facts that we wish to represent as RDF triplets.

1. The name of this URI (mailto: Hofstadter@yahoo.com) is Douglas R. Hofstadter. (It is the name)
2. The type of this URI (mailto: Hofstadter@yahoo.com) is a type of person.
3. The author of this URI (mailto: Hofstadter@yahoo.com) is an author of isbn:0465026567.
4. The id of this URI (isbn:0465026567) is the identity of a book.
5. The title of this URI (isbn:0465026567) has the title of *G¨odel, Escher, Bach: An Eternal Golden Braid*.

Subject	Predicate	Object
mailto:Hofstadter@yahoo.com	name	Douglas R. Hofstadter
mailto:Hofstadter@yahoo.com	type	Person
mailto:Hofstadter@yahoo.com	author-of	isbn: 0465026567
isbn:0465026567	type	book
isbn:0465026567	title	*Gödel, Escher, Bach: An Eternal Golden Braid*

RDF Triplet Data Table

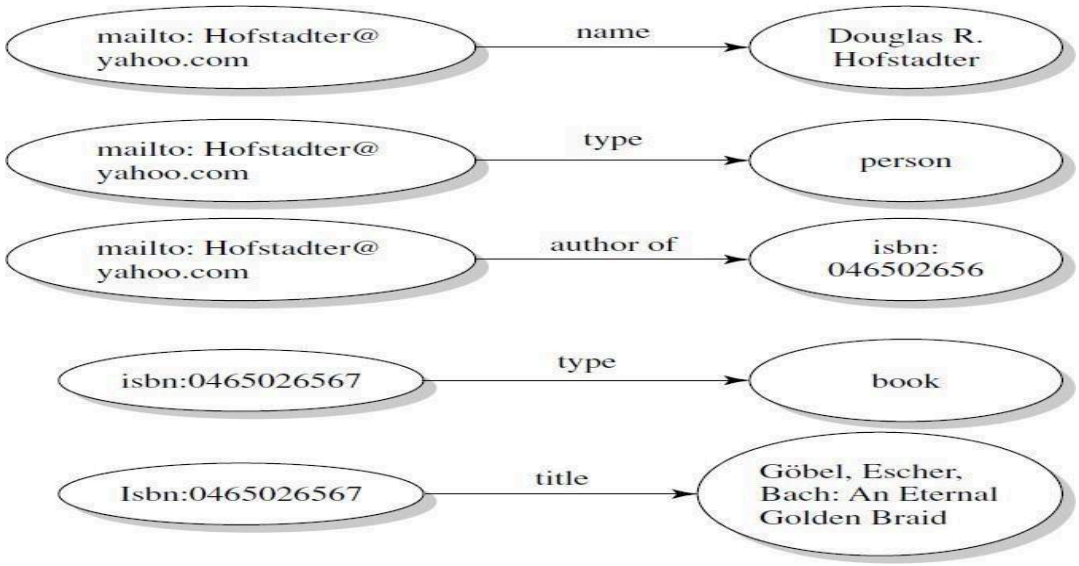
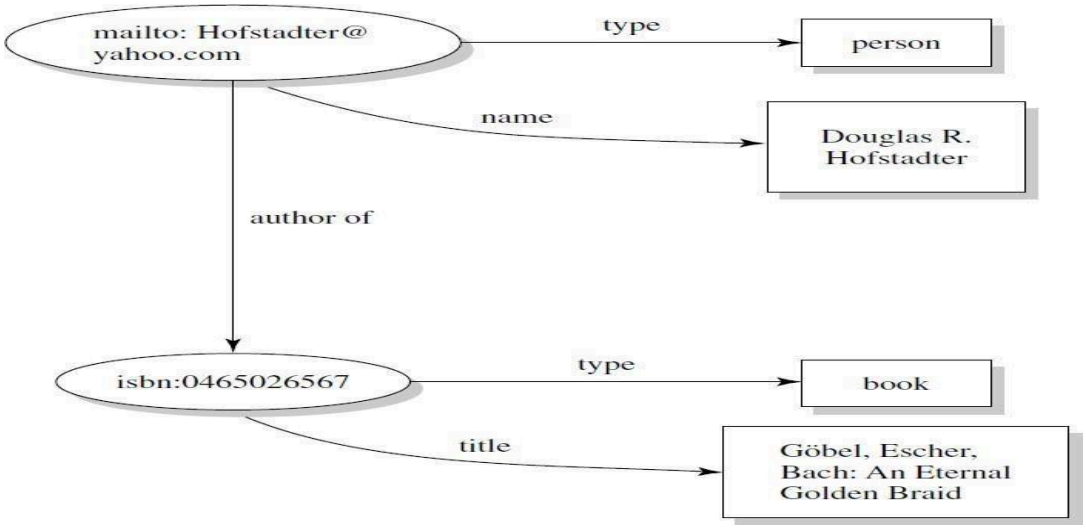


Fig: Individual graphs for each triplet statement of above Example



Merged RDF graph

The serialized form of the RDF document for this example can be written as:

Serialization of RDF Statement as
<?xml version="1.0"?>
<Class rdf:ID="book"
xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns# xmlns="uri">
<title>G¨odel, Escher, Bach: An Eternal Golden Braid</title>
. . .
</Class>

In any RDF graph, a subgraph would be a subset of the triples contained in the graph. Each triple is its own unique RDF graph. The union of two or more graphs is a new graph called a merged graph.

RDF SCHEMA

The RDF provides a simple yet powerful model for describing information including a basic directed graph, but the semantics (meaning of the information) is described using RDFS. The purpose of RDFS is to provide an XML vocabulary that can express classes and their (subclass) relationships, as well as to define properties associated with classes. This Schema is actually a primitive ontology language.

Classes and Properties

To describe a specific domain, we specify the ―things‖ we want to talk about. We can talk about either individual objects (resources) or classes that define types of objects.

A class can be considered as a set of elements. Individual objects that belong to a class are instances of the class. The relationship between instances and classes in RDF is expressed by **"rdf:type."**

The three most important RDF concepts are "Resource" (rdfs:Resource), "Class" (rdfs:Class), and "Property" (rdf:Property).

These are all ―classes.‖ Class is in the rdfs namespace. Property is in the rdf namespace.

We just use the rdf:type property, to declare that something is a ―type‖ of something else as following:

rdfs:Resource rdf:type
rdfs:Class. rdfs:Class rdf:type
rdfs:Class. rdf:Property rdf:type
rdfs:Class. rdf:type rdf:type
rdf:Property.

This means that ―Resource is a type of Class, Class is a type of Class, Property is a type of Class, and type is a type of Property.‖

For example, the ***rdf:ID*** provides a name for the class while the conjunction (AND) of two subClassOf statements is a subset of the intersection of the classes:

<rdfs:Class rdf:ID="Set1 AND Set 2">
<rdfs:subClassOf rdf:resource="#Set1"/>
<rdfs:subClassOf rdf:resource="#Set2"/>
</rdfs:Class>

RDF and RDFS Classes

Class Name	Comment
rdfs:Resource	Class of all resources
rdfs:Literal	Class of all literals (strings)
rdfs:XMLLiteral	The class of XML literals
rdfs:Class	Class of all classes
rdf:Property	Class of all properties
rdfs:Datatype	Class of datatypes
rdf:Statement	Class of all reified RDF statements
rdf:Bag	An unordered collection
rdf:Seq	An ordered collection
rdf:Alt	A collection of alternatives
rdfs:Container	This represents the set Containers
rdfs:ContainerMembershipProperty	The container membership properties, rdf:1, rdf:2, . . ., all of which are subproperties of 'member'
rdf:List	The class of RDF Lists

RDF and RDFS Properties

Property Name	Comment
rdf:type	Related a resource to its class
rdfs:subClassOf	Indicates membership of a class
rdfs:subPropertyOf	Indicates specialization of properties
rdfs:domain	A domain class for a property type
rdfs:range	A range class for a property type
rdfs:label	Provides a human-readable version of a resource name.
rdfs:comment	Use this for descriptions.
rdfs:member	A member of a container.
rdf:first	The first item in an RDF list. Also often called the head.
rdf:rest	The rest of an RDF list after the first item, called the tail.
rdfs:seeAlso	A resource that provides information about the subject resource
rdfs:isDefinedBy	Indicates the namespace of a resource.
rdf:value	Identifies the principal value (usually a string) of a property when the property value is a structured resource.
rdf:subject	The subject of an RDF statement.
rdf:predicate	The predicate of an RDF statement.
rdf:object	The object of an RDF statement.

1. quadrilaterals(X)→ polygons(X)
2. polygons(X) → shapes(X)
3. quadrilaterals (squares)

And now from this knowledge the following conclusions can be deduced:

1. polygons (squares)
2. shapes (squares)
3. quadrilateral(X)→ shapes(X)

The hierarchy relationship of classes is shown in Figure

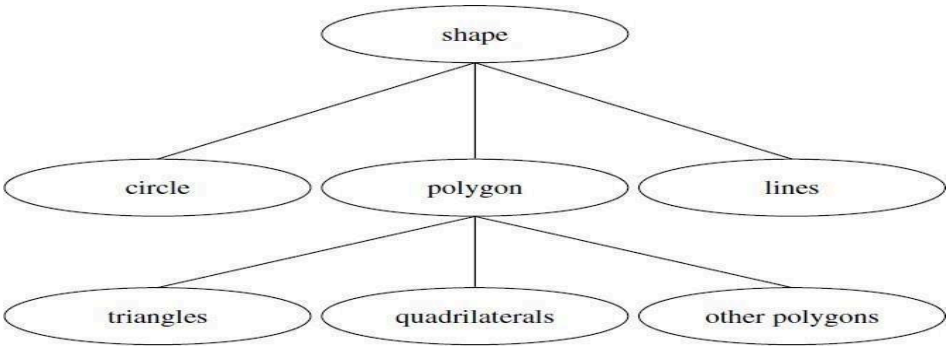


Fig: Hierarchy of Classes

Ontology Web Language(OWL)

For machines to perform useful automatic reasoning tasks on Web documents, the language machines use must go beyond the basic semantics of XML Schema and RDF Schema. They will require a more expressive and reasoning ontology language; as a result, the World Wide Web Consortium (W3C) has defined Web Ontology Language (called OWL).

Web Ontology Language enhances RDF with more vocabulary for describing properties and classes, including relations between classes (e.g., disjointedness), cardinality (e.g., exactly one), equality, richer typing of properties, characteristics of properties (e.g., symmetry), and enumerated classes.

Ontologies are usually expressed in a logic-based language, so that accurate, consistent, and meaningful distinctions can be made among the classes, properties, and relations. Some ontology tools can perform automated reasoning using the ontologies, and thus provide advanced services to intelligent applications, such as conceptual (semantic) search and retrieval, software agents, speech understanding, knowledge management, intelligent databases, and e-commerce.

OWL was developed in 2003, when the W3C began final unification of the disparate international ontology efforts into a standardized ontology. Web Ontology Language is designed to express a wide variety of knowledge, as well as provide for an efficient means to reason with it in order to express the most important kinds of knowledge. Using an ontology with a rule-based system, we can reach logic inferences about Web information.

OWL can be used to describe the classes and relations between classes that are inherent in Web documents and applications.

A set of XML statements by itself does not allow us to reach a conclusion about any other XML statements. To employ XML to generate new data, we need knowledge embedded in some

proprietary procedural code that exists as a server page on a remote server. However, a set of OWL statements by itself can allow us to reach a conclusion about another OWL statement.

OWL ontology documents are designed to be modular and independent. They can be combined dynamically to provide additional meaning if required.

Web Ontology Language ontology documents have a logical consistency to them. They provide machine-based systems with the ability to interpret the declared relationships within them. More importantly, they also allow mathematical techniques to be applied that can interpret and calculate the relationships that are implied within the logical formulations. These inferences make the use of OWL ontologies tractable and realistic for organizations, drastically reducing the amount of information that has to be modeled, encoded, or worked around by systems engineers and integrators.

COMPATIBILITY OF OWL AND RDF/RDFS

The layered architecture of the Semantic Web would suggest that one way to develop the necessary ontology language is to extend RDF Schema by using the RDF meaning of classes and properties (rdfs:classes, etc.) and adding primitives to support richer expressiveness.

The W3C has defined OWL to include three different sublanguages (OWL Full, OWL DL, OWL Lite) in order to offer different balances of expressive power and efficient reasoning.

OWL Full

The entire language is called OWL Full and it uses all the primitives and allows their combination with RDF and RDFS. The OWL Full supports maximum expressiveness and the syntactic freedom of RDF, but has 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.

The advantage of OWL Full is that it is fully compatible with RDF syntax and semantics. Any legal RDF document is also a legal OWL Full document. Any valid RDF–RDFS conclusion is also a valid OWL Full conclusion. The disadvantage of OWL Full is that the language is undecidable, and therefore cannot provide complete (or efficient) reasoning support.

OWL DL

Web Ontology Language DL (Descriptive Logic) is a sublanguage of OWL Full that restricts how the constructors from OWL and RDF can be used. This ensures that the language is related to description logic. Description Logics are a decidable fragment of First-Order Logic (FOL).

The OWL DL supports strong expressiveness while retaining computational completeness and decidability. It is therefore possible to automatically compute the classification hierarchy and check for inconsistencies in an ontology that conforms to OWL DL.

The advantage of this sublanguage is efficient reasoning support. The disadvantage is the loss of full compatibility with RDF. However, every legal OWL DL document is a legal RDF document.

OWL Lite

Further restricting OWL DL produces a subset of the language called OWL Lite, which excludes enumerated classes, disjointness statements, and arbitrary cardinality. The OWL Lite supports a classification hierarchy and simple constraints.

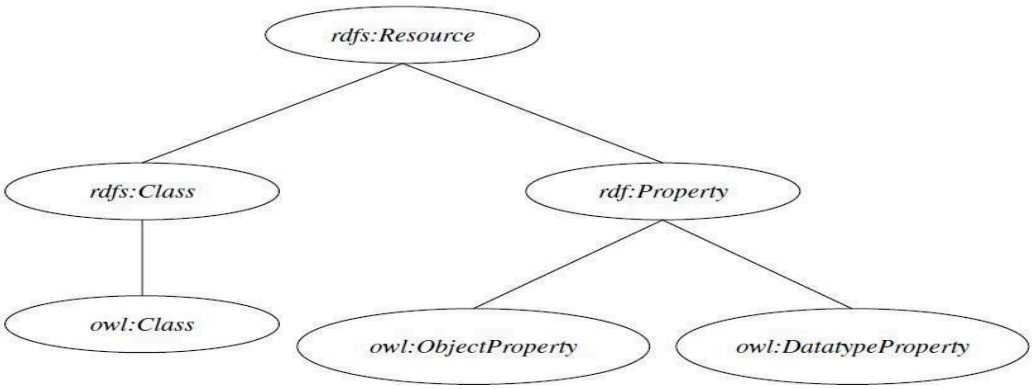


Fig: The OWL and RDF-RDFS subclass relationships

An OWL document identifies:

• **Class hierarchy:** Defines class–subclass relationships.

• **Synonym:** Identifies equivalent classes and equivalent properties.

• **Class association:** Maps one or more classes to one or more classes, through the use of a property (i.e., domain/range).

• **Property metadata:** Contains metadata for properties.

• **Class definition:** Specifies the composition of classes.

The OWL Syntax Specification http://www.w3.org/TR/owl-features/

Web Ontology Language defines the classes and properties, as well as their relationship to each other in the document; consequently, they are extremely similar to RDF Schema.

Unlike RDF, the OWL vocabulary is quite large. Like RDF, OWL makes use of elements from RDFS. However, OWL has several concepts unique to it, such as Boolean combination of class expressions and property restrictions, which add a layer of reasoning to applications. Both the RDFS and OWL are compatible.

BASIC ELEMENTS

Most of the elements of an OWL ontology concern classes, properties, instances of classes, and relationships between these instances.

Syntax

The OWL builds on RDF and RDFS and uses RDFs XML-based syntax.

OWL Document Parts

Document Parts	OWL Document
Header	
XML Syntax	<?xml version="1.0" encoding="UTF-8"?>
Root element	<rdf:RDF
Namespace	xmlns:iq = “http://www.web-iq.com”> xmlns:owl = “http://www.w3.org/2002/07/owl#” xmlns:rdf = “http://www.w3.org/1999/02/22-rdf-syntax-ns#” xmlns:rdfs = “http://www.w3.org/1999/02/22-rdf-schema#” xmlns:dc = “http://purl.org/dc/elements/1.1/” xmlns:xsd = “http://www.w3.org/2000/1/XMLSchema#”>
OWL properties and classes	<owl:Ontology rdf:about = “http://www.amazon.com”> <owl:versionInfo> $ID: Overview.html </owl:versionInfo> <dc:creator> Douglas R. Hofstadter </dc:creator> <dc:title> Gödel, Escher, Bach: An Eternal Golden Braid </dc:title>
End of OWL	</owl:Ontology>
End of RDF	</rdf:RDF>

Header

An OWL document contains an OWL ontology and is an RDF document with elements, tags, and namespaces. An OWL document starts with a header that identifies the root element as an ***rdf:RDF element***, which also specifies a number of namespaces.

Class Elements

Classes are defined using an owl:Class element. An example of an OWL class ―computer‖ is defined with a subclass ―laptop‖ as

```
<owl:Class rdf:ID="Computer">
<rdfs:subClassOf rdf:resource="#laptop"/>
</owl:Class>
```

Equivalence of classes is defined with **owl:equivelentClass**.

Property

A property in RDF provides information about the entity it is describing. Property characteristics increase our ability to understand the inferred information within the data.

The following special identifiers can be used to provide information concerning properties and their values:

- **inverseOf:** One property may be stated to be the inverse of another property.
- **TransitiveProperty:** Properties may be stated to be transitive.
- **SymmetricProperty:** Properties may be stated to be symmetric.
- **FunctionalProperty:** Properties may be stated to have a unique value.
- **InverseFunctionalProperty:** Properties may be stated to be inverse functional.

The OWL Lite allows restrictions to be placed on how properties can be used by instances of a class.

- **allValuesFrom:** The restriction allValuesFrom is stated on a property with respect to a class.
- **someValuesFrom:** The restriction someValuesFrom is stated on a property with respect to a class. A particular class may have a restriction on a property that at least one value for that property is of a certain type.
- **minCardinality:** Cardinality is stated on a property with respect to a particular class. If a minCardinality of 1 is stated on a property with respect to a class, then any instance of that class will be related to at least one individual by that property.
- **maxCardinality:** Cardinality is stated on a property with respect to a particular class. If a maxCardinality of 1 is stated on a property with respect to a class, then any instance of that class will be related to at most one.
- **cardinality:** Cardinality is provided as a convenience when it is useful to state that a property on a class has both minCardinality 0 and maxCardinality 0 or both minCardinality 1 and maxCardinality 1.
- **intersectionOf:** OWL Lite allows intersections of named classes and restrictions.

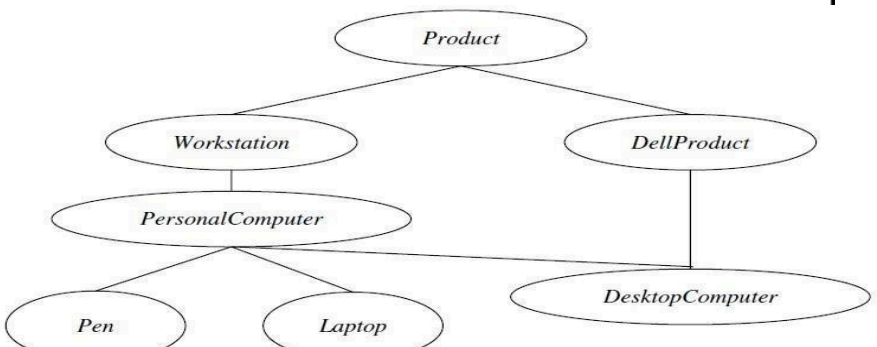


Fig: Classes and subclasses of the computer ontology

OWL EXAMPLE: COMPUTE ONTOLOGY

The serialization for the computer ontology is

```
<[DOCTYPE owl [
<!ENTITY xsd —http://www.w3.org/2001/XMLSchema#‖>
]>
<rdf:RDF
xmlns:rdf=‖http://www.w3.org/1999/02/22-rdf-syntax-ns#‖
xmlns:rdfs=‖http://www.w3.org/200/01/rdf-schema#‖
xmlns:xsd=‖http://www.w3.org/2001/XMLSchema#‖
xmlns:owl=‖http://www.w3.org/2002/07/owl#‖
xmlns=‖http://www.web-iq.com/computer.owl#‖>
<owl:Ontology rdf:about=‖‖>
<owl:versionInfo>
</owl:versionInfo>
</owl:Ontology>
<owl:Class rdf:ID=—Product‖>
</owl:Class>
<owl:Class rdf:ID=—Workstation‖>
<rdfs:label>Device</rdfs:label>
<rdfs:subClassOf rdf:resource=—#product‖/>
</owl:Class>
<owl:Class rdf:ID=—DellProducts‖>
<rdfs:label>Dell Devices</rdfs:label>
<owl:intersectionOf rdf:parseType=—Collection‖>
<owl:Class rdf:about —#product‖/>
<owl:Restriction>
<owl:onProperty rdf:resource=—#manufactured by‖/>
<owl:hasValue
rdf:datatype=—&xsd;string‖> DELL
</owl:hasValue>
</owl:Restriction>
</owl:Intersection>
</owl:Class>
<owl:Class rdf:ID=—PersonalComputer‖>
<rdfs:subClassOf rdf:resource=—#workstation‖/>
</owl:Class>
<owl:Class rdf:ID=—Laptop‖>
<rdfs:subClassOf rdf:resource=—#personalcomputer‖/>
</owl:Class>
<owl:Class rdf:ID=—DesktopComputer‖>
<rdfs:subClassOf rdf:resource=—#personalcomputer‖/>
<rdfs:subClassOf rdf:resource=—#dellproduct‖/>
```

```
</owl:Class>
<owl:Class rdf:ID=—Pen‖>
<rdfs:subClassOf rdf:resource=—#personalcomputer‖/>
</owl:Class>
<owl:DatatypeProperty rdf:ID= —manufactured by‖>
<rdf:domain rdf:resource= —#product‖/>
<rdf:range rdf:resource =—&xsd:string‖/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID= —price‖>
<rdf:domain rdf:resource= —#product‖/>
<rdf:range rdf:resource =‖&xsd:string‖/>
</owl:DatatypeProperty>
</rdf:RDF>
```

This onotology demonstrates siblings in a hierarchy may not be disjoint.

Owl Capabilities and Limitations

The OWL language offers the following features: less chance of misinterpretation, understanding each other‘s data‘s semantics, and OWL uses existing XML syntax to express semantics. The OWL document can be extensible, reusable, and avoids misinterpretation.

Additional OWL problems include no ad hoc discovery and exploitation, thus an application may not be able to effectively process new data when it is encountered.

Comparison to the Unified Modelling Language (UML)

UML is most commonly used in the requirements specification and design of objectoriented software in the middle tier of enterprise applications.

The chief difference between UML and RDF(S)/OWL is their modelling scope: UML contains modelling primitives specific for a special kind of information resource, namely objects in an information system characterized by their static attributes and associations, but also their dynamic behavior. Many of the modelling primitives of UML are thus specific to objects and their role in OO systems; interfaces, functions etc.

Unique features of RDF/OWL

– In general, the modelling of RDF is less constrained than that of UML, which means that many RDF models have no equivalent in UML. OWL DL also provides more primitives than UML such as the disjointness, union, intersection and equivalence of classes.

– OWL allows to describe defined classes, i.e. definitions that give necessary and sufficient conditions for an instance to be considered as a member of the class.

– RDF/OWL gives high priority to its properties. They traet the Properties are global: they do not belong to any class, while UML attributes and associations are defined as part of the description of a certain class. In other words, the same property can be used with multiple classes.

– Properties can be defined as subproperties of other properties.

– Classes can be treated as instances, allowing for meta-modelling.

– RDF reification is more flexible than the association class mechanism of UML. For example, statements concerning literal values can also be reified in RDF

– All non-blank RDF resources are identified with a URI, UML classes, instances, attributes etc.

– Instances can and usually have multiple types.

Unique features of UML

– UML has the notion of relationship roles, which is not present in RDF/OWL.

– UML allows n-ary relations, which are not part of RDF, although they can bere-represented in a number of ways.

– Two common types of part-whole relations are available in UML (aggregation and composition). These can be remodelled in OWL to some extent.

– UML makes a clear differentiation between attributes and associations. This is also different from the distinction between datatype and object-properties in OWL. On the one hand, attributes can have instances as values, while datatype properties can only have literal values.

Comparison to the Extensible Markup Language (XML) and XML Schema

Up to date XML is the most commonly used technology for the exchange of structured information between systems and services. From all languages discussed the role of XML is thus the most similar to RDF in its purpose.

The most commonly observed similarity between XML and RDF is a similarity between the data models: a directed graph for RDF, and a directed, ordered tree for XML. In XML, the tree is defined by the nesting of elements starting with a single root node. This model originates from the predecessor of XML called SGML which was primarily used for marking up large text documents. Text documents follow the tree structure themselves as paragraphs are nested in subsections, subsections are nested in sections, sections are nested chapters etc. The ordering of the children of an element matters, which is again inherited from the text processing tradition.

Namely, schemas written in XML schema languages not only define the types of elements and their attributes but also prescribe syntax i.e. the way elements are allowed to be nested in the tree. XML documents can be validated against a schema on a purely syntactic level.

RDF models are based on arbitrary directed graphs. They are developed from single edges between the nodes of classes or instances.

XML has a variety of schema languages like XMLSchema and Relax NG. Schemas in XML schema language define the elements type, their attributes and the syntax validation of XML document against a schema is done syntactically. RDF Schema language does not introduce constraints directly on the graph model rather they effect the interpretations of data.

RDF for web based data exchanges has an advantage that agreement on shared XML format needs a stronger commitment than the agreement mode by using RDF. This agreement of exchanging RDF documents considering only the individual statements like a simple subject, predicate and object model.