ABSTRACT

The word semantic stands for the meaning of. The semantic of something is the meaning of something. The Semantic Web is a web that is able to describe things in a way that computers can understand.

- The Beatles was a popular band from Liverpool.
- John Lennon was a member of the Beatles.
- The record "Hey Jude" was recorded by the Beatles.

Sentences like these can be understood by people. But how can they be understood by computers?Statements are built with syntax rules. The syntax of a language defines the rules for building the language statements. But how can syntax become semantic?This is what the Semantic Web is all about. Describing things in a way that computers applications can understand. The Semantic Web is not about links between web pages. The Semantic Web describes the relationships between things (like A is a part of B and Y is a member of Z) and the properties of thing**s** (like size, weight, age, and price)

CONTENTS

1.INTRODUCTION

The Web was designed as an information space, with the goal that it should be useful not only for human-human communication, but also that machines would be able to participate and help. One of the major obstacles to this has been the fact that most information on the Web is designed for human consumption, and even if it was derived from a database with well defined meanings for its columns, that the structure of the data is not evident to a robot browsing the web. Humans are capable of using the Web to carry out tasks such as finding the Finnish word for "car", to reserve a library book, or to search for the cheapest DVD and buy it. However, a computer cannot accomplish the same tasks without human direction because web pages are designed to be read by people, not machines.

The S**emantic Web** is a vision of information that is understandable by computers, so that they can perform more of the tedious works involved in finding, sharing and combining information on the web. For example, a computer might be instructed to list the prices of flat screen HDTVs larger than 40 inches with 1080p resolution at shops in the nearest town that are open until 8pm on Tuesday evenings. Today, this task requires search engines that are individually tailored to every website being searched. The semantic web provides a common standard (RDF) for websites to publish the relevant information in a more readily machine-processable and integratable form.

1.1 What is **Semantic Web**?

The **Semantic Web** is an evolving extension of the **World Wide Web** in which the semantics of information and services on the web is defined, making it possible for the web to understand and satisfy the requests of people and machines to use the Web content. It derives from W3C director Tim Berners -Lee vision of the Web as a universal medium for data ,information and knowledge exchange.

Tim Berners-Lee originally expressed the vision of the semantic web as follows

"I have a dream for the Web [in which computers] become capable of analyzing all the data on the Web – the content, links, and transactions between people and computers. A 'Semantic Web', which should make this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines. The 'intelligent agents' people have touted for ages will finally materialize"

1.2 WWW Vs Semantic Web

Current web contains a hypermedia, a digital library,a library of documents called (web pages) interconnected by a hypermedia of links,a database, an application platform,a common portal to applications accessible through web pages, and presenting their results as web pages,a platform for multimedia,a naming scheme and Unique identity for those documents

Finding information involving background knowledge such as "animals that use sonar but are not either bats or dolphins" is not possible to the current web. Similarly locating information in data repositories such as Travel enquiries, Prices of goods and services, Results of human genome experiments is also not possible

The World Wide Web is based mainly on documents written in HyperText Markup Language(HTML), a markup convention that is used for coding a body of text interspersed with multimedia objects such as images and interactive forms. The semantic web involves publishing the data in a language, Resource Description Framework (RDF) specifically for data, so that it can be manipulated and combined just as can data files on a local computer. The HTML language describes documents and the links between them. RDF, by contrast, describes arbitrary things such as people, meetings, and airplane parts.

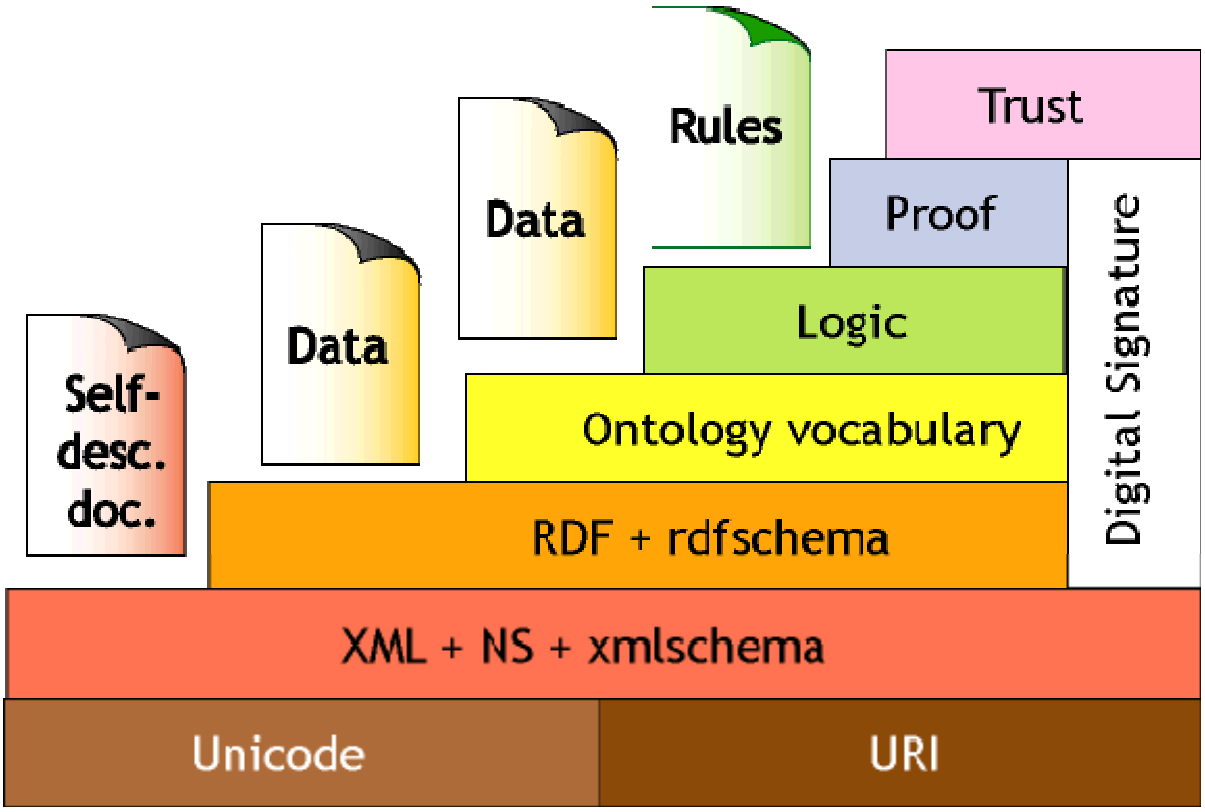
For example, with HTML and a tool to render it (perhaps Web browser software, perhaps another user agent), one can create and present a page that lists items for sale. The HTML of this catalog page can make simple, document-level assertions such as "this document's title is 'Widget Superstore'". But there is no capability within the HTML itself to assert unambiguously that, for example, item number X586172 is an Acme Gizmo with a retail price of €199, or that it is a consumer product. Rather, HTML can only say

that the span of text "X586172" is something that should be positioned near "Acme Gizmo" and "€ 199", etc. There is no way to say "this is a catalog" or even to establish that "Acme Gizmo" is a kind of title or that "€ 199" is a price. There is also no way to express that these pieces of information are bound together in describing a discrete item, distinct from other items perhaps listed on the page.

The semantic web addresses this shortcoming, using the descriptive technologies Resource Description Framework(RDF) and Web Ontology Language(OWL), and the data-centric, customizable Extensible Markup Language (XML). These technologies are combined in order to provide descriptions that supplement or replace the content of Web documents. Thus, content may manifest as descriptive data stored in Web-accessible databases, or as markup within documents (particularly, in Extensible HTML (XHTML) interspersed with XML, or, more often, purely in XML, with layout/rendering cues stored separately). The machine-readable descriptions enable content managers to add meaning to the content, i.e. to describe the structure of the knowledge we have about that content. In this way, a machine can process knowledge itself, instead of text, using processes similar to human deductive reasoning and inference, thereby obtaining more meaningful results and facilitating automated information gathering and research by computers.

2. COMPONENTS OF SEMANTIC WEB

Several formats and languages form the building blocks of the semantic web. Some of these include Identifiers: Uniform Resource Identifier(URI), Documents : Extensible Markup Language(XML), Statements : Resource Description Framework (RDF), a variety of data interchange formats (e.g. RDF/XML, N3) and notations such as RDF Schemas(RDFS) and the Web Ontology Language (OWL), all of which are intended to provide a formal desription of concepts, terms and relastionships within a given knowledge domain ,Logic,Proof and Trust.



2.1 Identifiers: Uniform Resource Identifier(URI)

To identify items on the Web, we use identifiers. Because we use a uniform system of identifiers, and because each item identified is considered a "resource," we call these identifiers "Uniform Resource Identifiers" or URIs for short. We can give a URI to anything, and anything that has a URI can be said to be "on the Web": you, the book you bought last week, the fly that keeps buzzing in your ear and anything else you can think of -- they all can have a URI.

The URI is the foundation of the Web. While nearly every other part of the Web can be replaced, the URI cannot: it holds the rest of the Web together. One familiar form of URI is the URL or Uniform Resource Locater. A URL is an address that lets you visit a web page, such as: **http://www.w3.org/Addressing/**. If you break it down, you can see that a URL tells your computer where to find a specific resource (in this case, the W3C's Addressing web site) Unlike most other forms of URI's a URL both identifies and locates. Contrast this with a "mid:" URI. A "mid:" URI identifies an email message, but it isn't able to locate a copy of the message for you.

Because the Web is far too large for any one organization to control it, URIs are decentralized. No one person or organization controls who makes them or how they can be used. While some URI schemes (such as http:) depend on centralized systems (such as DNS), other schemes (such as freenet:) are completely decentralized.

This means that we don't need anyone's permission to create a URI. We can even create URIs for things we don't own. While this flexibility makes URIs powerful, it brings with it more than a few problems. Because anyone can create a URI, we will inevitably end up with multiple URIs representing the same thing. Worse, there will be no way to figure out whether two URIs refer to exactly the same resource. Thus, we'll never be able to say with certainty exactly what a given URI means. But these are trade offs that must be made if we are to create something as enormous as the Semantic Web.

A URI is not a set of directions telling your computer how to get to a specific file on the Web (though it may also do this). It is a name for a "resource" (a thing). This resource may or may not be accessible over the Internet. The URI may or may not provide a way for our computer to get more information about that resource. Yes, a URL is a type of URI that does provide a way to get information about a resource, or perhaps to retrieve the resource itself, and other methods for providing information about URIs and the resources they identify are under development. It is also true that the ability to say things about URIs is an important part of the Semantic Web.

2.2 Documents : Extensible Markup Language(XML)

XML was designed to be a simple way to send documents across the Web. It allows anyone to design their own document format and then write a document in that format. These document formats can include markup to enhance the meaning of the document's content. This markup is "machine-readable," that is, programs can read and understand it. By including machine-readable meaning in our documents, we make them much more powerful.

Consider a simple example: if a document contains certain words that are marked as "emphasized," the way those words are rendered can be adapted to the context. A Web browser might simply display them in italics, whereas a voice browser (which reads Web

pages aloud) might indicate the emphasis by changing the tone or the volume of its voice. Each program can respond appropriately to the meaning encoded in the markup. In contrast, if we simply marked the words as "in italics", the computer has no way of knowing *why* those words are in italics. Is it for emphasis or simply for a visual effect? How does the voice browser display this effect?

Here's an example of a document in plain text:

I just got a new pet dog.

As far as our computer is concerned, this is just text. It has no particular meaning to the computer. But now consider this same passage marked up using an XML-based markup language (we'll make one up for this example):

<sentence>
<person href="http://aaronsw.com/"> I </person> just got a new pet <animal> dog </animal>.
</sentence>

Notice that this has the same content, but that parts of that content are labeled. Each label consists of two "tags": an opening tag (e.g., <sentence>) and a closing tag (e.g., </sentence>). The name of the tag ("sentence") is the label for the content enclosed by the tags. We call this collection of tags and content an "element." Thus, the sentence element in the above document contains the sentence, "I just got a new pet dog." This tells the computer that "I just got a new pet dog" is a "sentence," but -- importantly -- it does not tell the computer what a sentence is. Still, the computer now has some information about the document, and we can put this information to use.Similarly, the computer now knows that "I" is a "person" (whatever that is) and that "dog" is an "animal."

Sometimes it is useful to provide more information about the content of an element than we can provide with the name of the element alone. For example, the computer knows that "I" in the above sentence represents a "person," but it does not know which person. We can provide this sort of information by adding *attributes* to our elements. An attribute has both a name and a value. For example, we can rewrite our example thus:

<sentence>
<person href="http://aaronsw.com">I</person> just got a new pet <animal type="dog" href="http://aaronsw.com/myDog">dog</animal>.
</sentence>

A problem with this is that we've used the words "sentence," "person," and "animal" in the markup language. But these are pretty common words. What if others have used these same words in their own markup languages? What if those words have different meanings in those languages? Perhaps "sentence" in another markup language refers to the amount of time that a convicted criminal must serve in a penal institution.

To prevent confusion, we must uniquely identify my markup elements. And what better way to identify them than with a Uniform Resource Identifier? So we assign a URI to each of our elements and attributes. We do this using something called XML namespaces. This way, anyone can create their own tags and mix them with tags made by others. A namespace is just a way of identifying a part of the Web (space) from which we derive the meaning of these names. I create a "namespace" for my markup language by creating a URI for it.

Since everyone's tags have their own URIs, we don't have to worry about tag names conflicting. XML, of course, lets us abbreviate and set default URIs so we don't have to type them out each time.

2.3 Statements : Resource Description Framework (RDF)

The most fundamental building block is Resource Description Framework(RDF), a format for defining information on the web. RDF is a markup language for describing information and resources on the web. Putting information into RDF files, makes it possible for computer programs ("web spiders") to search, discover, pick up, collect, analyze and process information from the web. The Semantic Web uses RDF to describe web resources. RDF provides a model for data, and a syntax so that independent parties can exchange and use it. It is designed to be read and understood by computers. It is not designed for being displayed to people.

RDF is really quite simple. An RDF statement is a lot like a simple sentence,

except that almost all the words are URIs. Each RDF statement has three parts: a subject, a predicate and an object. Let's look at a simple RDF statement:

<http://aaron.com/>

<http://love.example.org/terms/reallyLikes>

<http://www.w3.org/People/Berners-Lee/Weaving/> .

The first URI is the subject. In this instance, the subject is aaron. The second URI is the predicate. It relates the subject to the object. In this instance, the predicate is "reallyLikes." The third URI is the object. Here, the object is Tim Berners-Lee's book "Weaving the Web." So the RDF statement above says that aaron really like "Weaving the Web."

Once information is in RDF form, it becomes easy to process it, since RDF is a generic format, which already has many parsers. XML RDF is quite a verbose specification, and it can take some getting used to (for example, to learn XML RDF properly, you need to understand a little about XML and namespaces beforehand...), but let's take a quick look at an example of XML RDF right now:-

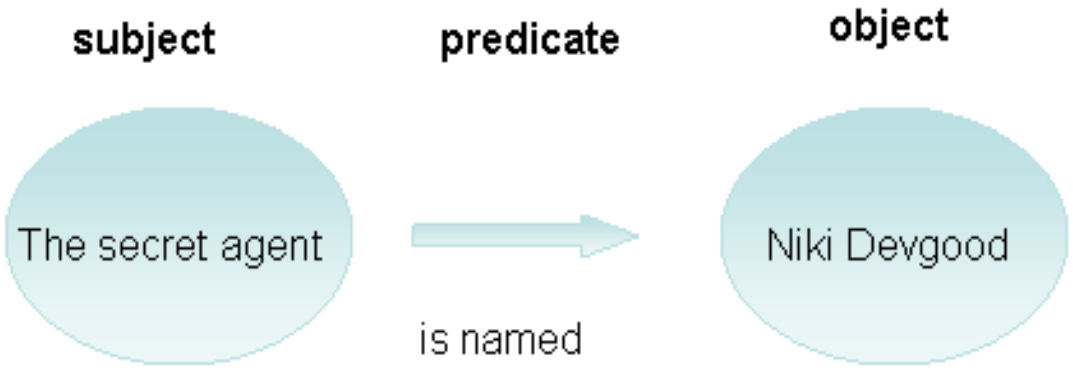
```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:dc="http://purl.org/dc/elements/1.1/"
   xmlns:foaf="http://xmlns.com/0.1/foaf/" >
   <rdf:Description rdf:about="">
      <dc:creator rdf:parseType="Resource">
         <foaf:name>Sean B. Palmer</foaf:name>
      </dc:creator>
      <dc:title>The Semantic Web: An Introduction</dc:title>
   </rdf:Description>
</rdf:RDF>
```

This piece of RDF basically says that this article has the title "The Semantic Web: An Introduction", and was written by someone whose name is "Sean B. Palmer". Here are the triples that this RDF produces:-

<> <http://purl.org/dc/elements/1.1/creator> _:x0 .
this <http://purl.org/dc/elements/1.1/title> "The Semantic Web: An Introduction" .

_:x0 <http://xmlns.com/0.1/foaf/name> "Sean B. Palmer" .

This format is actually a plain text serialization of RDF called "Notation3", which we shall be covering later on. Note that some people actually prefer using XML RDF to Notation3, but it is generally accepted that Notation3 is easier to use, and is of course convertable to XML RDF anyway.RDF triples can be written with XML tags, and they are represented graphically as shown below



Why RDF?

When people are confronted with XML RDF for the first time, they usually have two questions: "why use RDF rather than XML?", and "do we use XML Schema in conjunction with RDF?".

The answer to "why use RDF rather than XML?" is quite simple, and is twofold. Firstly, the benefit that one gets from drafting a language in RDF is that the information maps *directly* and *unambiguously* to a model, a model which is decentralized, and for which there are many generic parsers already available. This means that when you have an RDF application, you know which bits of data are the semantics of the application, and which bits are just syntactic fluff. And not only do you know that, *everyone* knows that, often implicitly without even reading a specification because RDF is so well known. The second part of the twofold answer is that we hope that RDF data will become a part of the Semantic Web, so the benefits of drafting your data in RDF now draws parallels with drafting your information in HTML in the early days of the Web.

The answer to "do we use XML Schema in conjunction with RDF?" is almost as brief. XML Schema is a language for restricting the *syntax* of XML applications. RDF

already has a built in BNF that sets out how the language is to be used, so on the face of it the answer is a solid "no". However, using XML Schema in conjunction with RDF *may* be useful for creating datatypes and so on. Therefore the answer is "possibly", with a caveat that it is not really used to control the syntax of RDF. This is a common misunderstanding, perpetuated for too long now.

Screen Scraping, and Forms

For the Semantic Web to reach its full potential, many people need to start publishing data as RDF. Where is this information going to come from? A lot of it can be derived from many data publications that exist today, using a process called "screen scraping". Screen scraping is the act of literally getting the data from a source into a more manageable form (i.e. RDF) using whatever means come to hand. Two useful tools for screen scraping are XSLT (an XML transformations language), and RegExps (in Perl, Python, and so on).

However, screen scraping is often a tedious solution, so another way to approach it is to build proper RDF systems that take input from the user and then store it straight away in RDF. Data such as signing up for a new mail account, buying some CDs online, or searching for a used car can all be stored as RDF and then used on the Semantic Web.

Notation3: RDF Made Easy

XML RDF can be rather difficult, but thankfully, there are simpler teaching forms of RDF. One of these is called "Notation3", and was developed by Tim Berners-Lee. There is some documentation covering N3, including a specification and an excellent Primer.

The design criteria behind Notation3 were fairly simple: design a simple easy to learn scribblable RDF format, that is easy to parse and build larger applications on top of. In Notation3, we can simply write out the URIs in a triple, delimiting them with a "<" and ">" symbols. For example, here's a simple triple consisting of three URI triples:-

<http://xyz.org/#a> <http://xyz.org/#b> <http://xyz.org/#c> .

To use literal values, simply enclose the value in double quote marks, thus:-

<http://xyz.org/#Sean> <http://xyz.org/#name> "Sean" .

Notation3 does have many other little constructs including contexts, DAML lists, and alternative ways of representing anonymous nodes, but we need not concern ourselves with them here.

2.4 Schemas and Ontologies: RDF Schemas, DAML+OIL, and WebOnt

A "schema" (plural "schemata") is simply a document or piece of code that controls a set of terms in another document or piece of code. It's like a master checklist, or definition grammar. A schema and an ontology are ways to describe the meaning and realtionships of terms. This description (in RDF, of course) helps computer systems use terms more easily, and decide how to convert between them.

RDF Schema

RDF Schema was designed to be a simple datatyping model for RDF. Using RDF Schema, we can say that "Fido" is a type of "Dog", and that "Dog" is a sub class of animal. We can also create properties and classes, as well as doing some slightly more "advanced" stuff such as creating ranges and domains for properties.

The first three most important concepts that RDF and RDF Schema give us are the "Resource" (rdfs:Resource), the "Class" (rdfs:Class), and the "Property" (rdf:Property). These are all "classes", in that terms may belong to these classes. For example, all terms in RDF are types of resource. To declare that something is a "type" of something else, we just use the rdf:type property:-

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 simply says 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". These are all true statements.

It is quite easy to make up our own classes. For example, let's create a class called "Dog", which contains all of the dogs in the world:-

```
:Dog rdf:type rdfs:Class .
```

Now we can say that "Fido is a type of Dog":-

```
:Fido rdf:type :Dog .
```

We can also create properties quite easily by saying that a term is a type of rdf:Property, and then use those properties in our RDF:-

```
:name rdf:type rdf:Property .
:Fido :name "Fido" .
```

Why have we said that Fido's name is "Fido"? Because the term ":Fido" is a URI, and we could quite easily have chosen any URI for Fido, including ":Squiggle" or ":n508s0srh". We just happened to use the URI ":Fido" because it's easier to remember. However, we still have to tell machines that his name is Fido, because although people can guess that from the URI (even though they probably shouldn't), machines can't.

RDF Schema also has a few more properties that we can make use of: rdfs:subClassOf and rdfs:subPropertyOf. These allow us to say that one class or property is a sub class or sub property of another. For example, we might want to say that the class "Dog" is a sub class of the class "Animal". To do that, we simply say:-

```
:Dog rdfs:subClassOf :Animal .
```

Hence, when we say that Fido is a Dog, we are also saying that Fido is an Animal. We can also say that there are other sub classes of Animal:-

```
:Human rdfs:subClassOf :Animal .
:Duck rdfs:subClassOf :Animal .
```

And then create new instances of those classes:-

```
:Bob rdf:type :Human .
:Quakcy rdf:type :Duck .
```

And so on. RDF schema allows one to build up knowledge bases of data in RDF very very quickly.

The next concepts which RDF Schema provides us, which are important to mention, are ranges and domains. Ranges and domains let us say what classes the subject and object of each property must belong to. For example, we might want to say that the property ":bookTitle" must always apply to a book, and have a literal value:-

```
:Book rdf:type rdfs:Class .
:bookTitle rdf:type rdf:Property .
:bookTitle rdfs:domain :Book .
:bookTitle rdfs:range rdfs:Literal .
:MyBook rdf:type :Book .
:MyBook :bookTitle "My Book" .
```

rdfs:domain always says what class the subject of a triple using that property belongs to, and rdfs:range always says what class the object of a triple using that property belongs to.

RDF Schema also contains a set of properties for annotating schemata, providing comments, labels, and the like. The two properties for doing this are rdfs:label and rdfs:comment, and an example of their use is:-

```
:bookTitle rdfs:label "bookTitle";
  rdfs:comment "the title of a book" .
```

DAML+OIL

DAML is a language created by DARPA as an ontology and inference langauge based upon RDF. DAML takes RDF Schema a step further, by giving us more in depth properties and classes. DAML allows one to be even more expressive than with RDF Schema, and brings us back on track with our Semantic Web discussion by providing some simple terms for creating inferences.

DAML provides us a method of saying things such as inverses, unambiguous properties, unique properties, lists, restrictions, cardinalities, pairwise disjoint lists,

datatypes, and so on. We shall run through a couple of these here, but armed with the knowledge that you've already gotten from this introduction (assuming that you haven't skipped any of it!), it should be just as beneficial going through the DAML + OIL walkthru

One DAML construct that we shall run through is the daml:inverseOf property. Using this property, we can say that one property is the inverse of another. The rdfs:range and rdfs:domain values of daml:inverseOf is rdf:Property. Here is an example of daml:inverseOf being used:-

:hasName daml:inverseOf :isNameOf .
:Sean :hasName "Sean" .
"Sean" :isNameOf :Sean .

The second useful DAML construct that we shall go through is the daml:UnambiguousProperty class. Saying that a Property is a daml:UnambiguousProperty means that if the object of the property is the same, then the subjects are equivalent. For example:-

foaf:mbox rdf:type daml:UnambiguousProperty .
:x foaf:mbox .
:y foaf:mbox .

implies that:-

:x daml:equivalentTo :y .

Inference

The principle of "inference" is quite a simple one: being able to derive new data from data that already know. In a mathematical sense, querying is a form of inference (being able to infer some search results from a mass of data, for example). Inference is one of the driving principles of the Semantic Web, because it will allow us to create SW applications quite easily.

To demonstrate the power of inference, we can use some simple examples. Let's take the simple car example: we can say that:-

:MyCar de:macht "160KW" .

Now, to a German Semantic Web processor, the term ":macht" may well be built into it, and although an English processor may have an equivalent term built into it somewhere, it will not understand the code with the term in it that it doesn't understand. Here, then, is a piece of inference data that makes things clearer to the processor:-

de:macht daml:equivalentTo en:power .

We have used the DAML "equivalentTo" property to say that "macht" in the German system is equivalent to "power" in the English system. Now, using an inference engine, a Semantic Web client could successfully determine that:-

:MyCar en:power "160KW" .

2.5 Logic

For the Semantic Web to become expressive enough to help us in a wide range of situations, it will become necessary to construct a powerful logical language for making inferences. There is a raging debate as to how and even whether this can be accomplished, with people pointing out that RDF lacks the power to quantify, and that the scope of quantification is not well defined. Predicate logic is better discussed in John Sowa's excellent Mathematical Background(Predicate Logic)

At any rate, we already have a great range of tools with which to build the Semantic Web: assertions (i.e. "and"), and quoting (reification) in RDF, classes, properties, ranges and documentation in RDF Schema, disjoint classes, unambiguous and unique properties, data types, inverses, equivalencies, lists, and much more in DAML+OIL.

Note that Notation3 introduces a "context" construct, enabling one to group statements together and quantify over them using a specially designed logic vocabulary. Using this vocabulary, for example, one can express "or", using NANDs:-

{ { :Joe :loves :TheSimpsons } a log:Falsehood .

{ :Joe :is :Nuts } a log:Falsehood .

} a log:Falsehood .

Which can be read as "it is not true that Joe does not love The Simpsons and is not nuts". I resisted the temptation to make Joe a universally quantified variable.

Note that the above example does not serialize "properly" into XML RDF, because XML RDF does not have the context construct as denoted by the curly brackets in the example above. However a similar effect can be achieved using reification and containers.

Eg:

Jack is an engineer of Scandinavian Airline (SAS).

Engineers are permanent employees.

All permanent employees of SAS will get 50% discount for all Radison hotels.

THEREFORE, Jack will get 50% discount for all Radison hotels.

2.6 Proof

Once we begin to build systems that follow logic, it makes sense to use them to prove things. People all around the world could write logic statements. Then your machine could follow these Semantic "links" to construct proofs.

Example: Corporate sales records show that Jane has sold 55 widgets and 66 sprockets. The inventory system states that widgets and sprockets are both different company products. The built-in math rules state that 55 + 66 = 121 and that 121 is more than 100. And, as we know, someone who sells more than 100 products is a member of the Super Salesman club. The computer puts all these logical rules together into a proof that Jane is a Super Salesman.

While it's very difficult to create these proofs (it can require following thousands, or perhaps millions of the links in the Semantic Web), it's very easy to check them. In this way, we begin to build a Web of information processors. Some of them merely provide data for others to use. Others are smarter, and can use this data to build rules. The smartest are "heuristic engines" which follow all these rules and statements to draw conclusions, and kindly place their results back on the Web as proofs, as well as plain old data.

2.7 Trust: Digital Signatures and Web of Trust

Now we can say that this whole plan is great, but rather useless if anyone can say anything. Who would trust such as system? That's where Digital Signature come in. Based on work in mathematics and cryptography, digital signatures provide proof that a certain person wrote (or agrees with) a document or statement. So one digitally sign all of their RDF statements. That way, we can be sure that he wrote them (or at least vouch for their authenticity). Now, we can simply tell our program whose signatures to trust and whose not to. Each can set their own levels or trust (or paranoia) the computer can decide how much of what it reads to believe.

Now it's highly unlikely that you'll trust enough people to make use of most of the things on the Web. That's where the "Web of Trust" comes in. You tell your computer that you trust your best friend, Robert. Robert happens to be a rather popular guy on the Net, and trusts quite a number of people. And of course, all the people he trusts, trust another set of people. Each of those people trust another set of people, and so on. As these trust relationships fan out from you, they form a "Web of Trust." And each of these relationships has a degree of trust (or distrust) associated with it.

Note that distrust can be as useful as trust. Suppose that computer discovers a document that no one explicitly trusts, but that no one explicitly distrusts either. Most likely, computer will trust this document more than it trusts one that has been explicitly labeled as untrustworthy.

The computer takes all these factors into account when deciding how trustworthy a piece of information is. It can also make this process as transparent or opaque as you desire.

3. PROJECTS

FOAF

A popular application of the semantic web is Friend of a Friend(or FOAF), which describes relationships among people and other agents in terms of RDF. FOAF project is about creating a Web of machine-readable homepages describing people, the links

between them and the things they create and do.

SIOC

The SIOC Project - Semantically-Interlinked Online Communities provides a vocabulary of terms and relationships that model web data spaces. Examples of such data spaces include, among others: discussion forums, weblogs, blogrolls / feed subscriptions, mailing lists, shared bookmarks, image galleries.

SIMILE

Semantic **I**nteroperability of **M**etadata and **I**nformation in un**L**ike **E**nvironments Massachusetts Institute of Technologies.SIMILE is a joint project, conducted by the MIT Libraries and MIT CSALE which seeks to enhance interoperability among digital assets, schemata/vocabularies/ontologies, meta data, and services.

Linking Open Data

The Linking Open Data Project is a community lead effort to create openly accessible, and interlinked, RDF Data on the Web. The data in question takes the form of RDF Data Sets drawn from a broad collection of data sources. The project is one of several sponsored by the W3C's **S**emantic **W**eb **E**ducation & **O**utreach Interest Group (SWEO)

4. BROWSERS

A semantic web Browser is a form of Web User Agent that expressly requests RDF data from Web Servers using the best practice known as "Content Negotiation". These tools provide a user interface that enables data-link oriented navigation of RDF data by dereferencing the data links (URIs) in the RDF Data Sets returned by Web Servers.

Examples of semantic web browsers include: Tabulator, DISCO, OpenLink DF Browser OntoWiki Browser Crowbar - SIMILE

5. CASE STUDY

Semantic-based Search and Query System for the Traditional Chinese Medicine Community

General Description

The long standing curation effort in the Chinese medicine community has been accumulating huge amounts of data, which are typically stored in relational data management systems such as Oracle, and published as HTML pages for public presentation. The China Academy of Chinese Medicine Sciences(CACMS) hosts much of the data. However, it has become increasingly difficult and time-consuming to manage the data, and the links to data sources from other institutions. Although they could be physically put together, but the logical links among the data are usually implicit or even lost at all. Moreover, the randomness of choosing names for relational tables, table columns, and record values make the data only understandable to the original database designer and data curator and exclusively controlled by ad hoc applications. This has caused a huge hindrance in sharing, and reusing data across databases, and organizational boundaries.

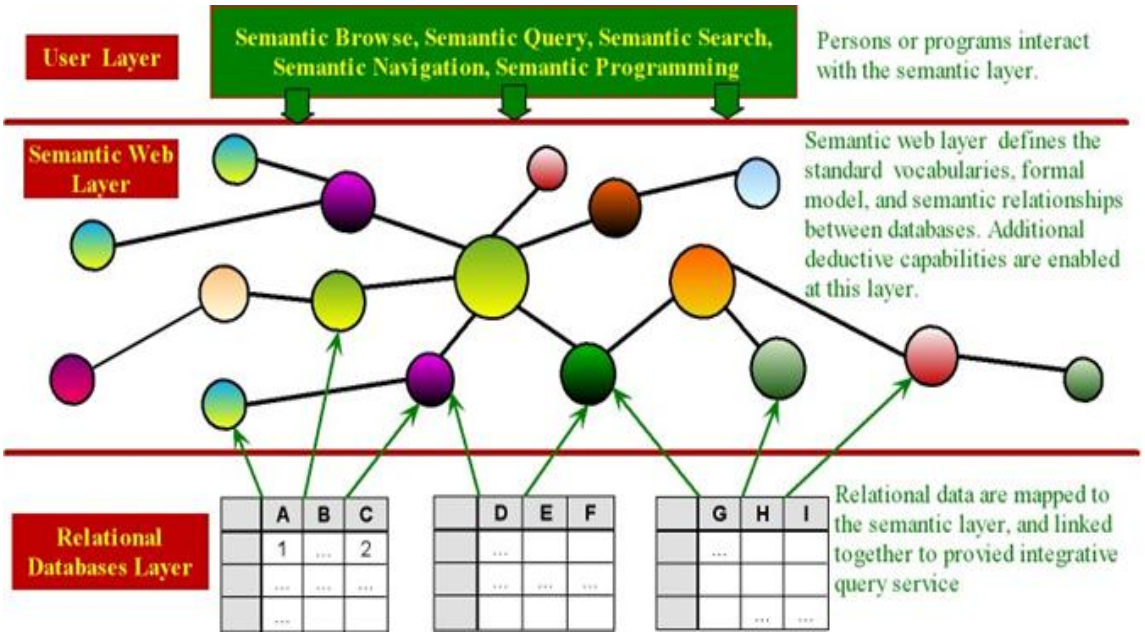


Figure 1: This figure shows the architecture of the Semantic Web layer and its role in unifying and linking heterogeneous relational data.

They have applied Semantic Web technologies to relational data to make it more sharable and machine-processable. They have also developed a semantic-based search and query system for the traditional Chinese medicine community in China (TCM Search), which has been deployed for real life usage since fall 2005. For the TCM system, a TCM ontology and the semantic layer has been constructed to unify and link the legacy databases, which typically have heterogeneous logic structures and physical properties. Users and applications now only need to interact with the semantic layer, and the semantic interconnections allow for searching, querying, navigating around an extensible set of databases without the awareness of the boundaries (Figure 1). Additional deductive capabilities can then be implemented at the semantic layer to increase the usability and re-usability of data. Besides, a visualized mapping tool has been developed to facilitate the mapping from relational data to the TCM ontology, and an ontology-based query and search portal has also been implemented to assist the semantic interaction with the system.

Mapping from relational data to semantic web ontologies

The informal approach taken for the selection of names and values within relational databases makes the data only understandable by specific applications. The mapping from relational data to the Semantic Web layer makes the semantics of the data more formal, explicit, and prepared for sharing and reusing by other applications. However, because of the inherent model difference between relational data model and the Semantic Web languages, mapping is always a complicated task and can be time-consuming and error-prone. We have therefore developed a visualize mapping tool to simplify the work as much as possible, as Figure 2 displays. The tool generates mapping rules that are used when a SPARQL query is rewritten into a set of SQL queries.

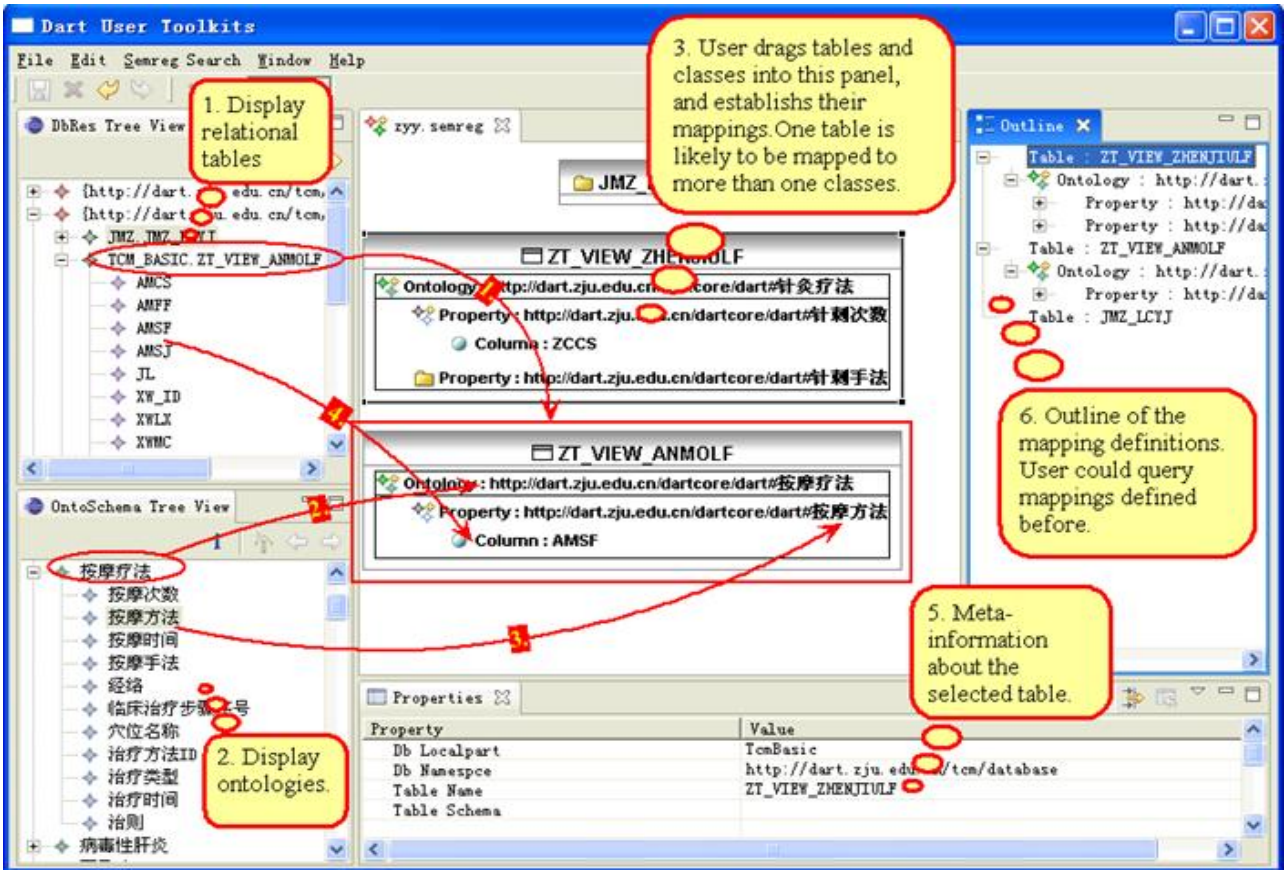


Figure 2: This figure shows a visualized mapping from a TCM relational database to the TCM ontology.

Ontology-based query and search across database boundaries

As Figure 3 displays, we have developed a semantic-based query and search portal to assist in user interaction with the system. Basically, this system consists of two components. The search component enables users to perform full-text searching through all of the integrated data sources using keywords that is similar to common Internet search engines, while the query component supplies with a means for handling more complex semantic queries posed against the semantic web ontology.

The ontology plays an important role in the mediation of the query, search and navigation. At first, it serves as a logic layer for users in constructing semantic queries. The form-based query interface is automatically generated based on the ontological

structure. The constructed semantic query will then be translated into SQL queries based on the mapping rules generated by the mapping tool. At second, it enables semantic navigation across database boundaries during query and search. At third, it also serves as a control vocabulary to facilitate search by making semantic suggestions such as synonyms, and related concepts.

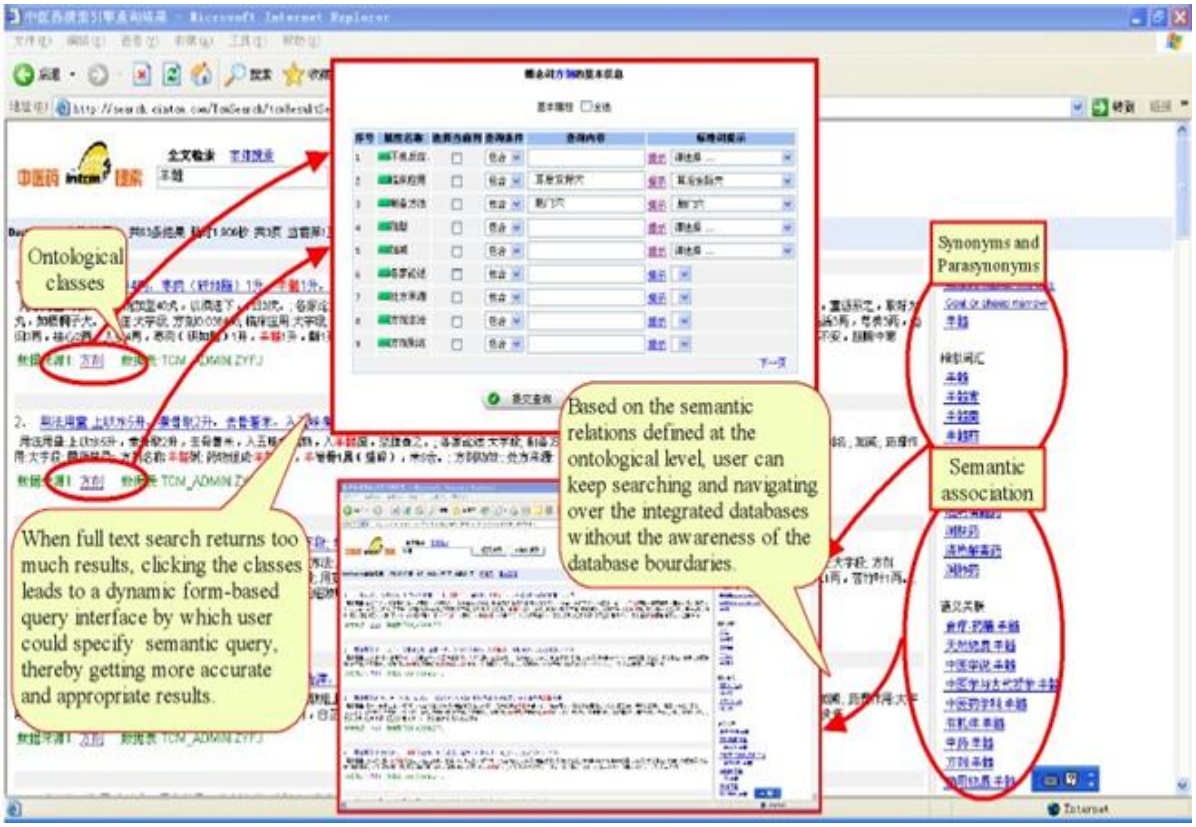


Figure 3: This figure shows the ontology-based query and search portal of the TCM search system.

Key Benefits of Using Semantic Web Technology

- Exposing of legacy data through a semantic layer so that it can be more easily reused and recombined.
- Linking data across database boundaries so as to enabling more intuitive query, search, and navigation without the awareness of the boundaries.
- The ontology serves as the control vocabulary to make semantic suggestions such as synonyms, related concepts to facilitate query and search.

- Reasoning capability such as sub-classing, transitive property can then be implemented at the semantic layer to increase the query expressiveness so as to retrieve more complete answers.

6. CONCLUSION

One of the best things about the Web is that it's so many different things to so many different people. The coming Semantic Web will multiply this versatility a thousandfold. For some, the defining feature of the Semantic Web will be the ease with which your PDA, your laptop, your desktop, your server, and your car will communicate with each other. For others, it will be the automation of corporate decisions that previously had to be laboriously hand-processed. For still others, it will be the ability to assess the trustworthiness of documents on the Web and the remarkable ease with which we'll be able to find the answers to our questions -- a process that is currently fraught with frustration.

Whatever the cause, almost everyone can find a reason to support this grand vision of the Semantic Web. Sure, it's a long way from here to there . The possibilities are endless, and even if we don't ever achieve all of them, the journey will most certainly be its own reward.

7. REFERENCES

http://www.w3.org/2001/sw/

http://www.w3schools.com/rdf/

http://www.w3.org/RDF/FAQ

http://en.wikipedia.org/wiki/Semantic_Web

http://www.hpl.hp.com/semweb/sw-technology.htm

http://www.w3.org/2001/sw/sweo/public/UseCases/UniZheijang/

Article on ***The Semantic Web in Action*** , by Lee Feigenbaum, Ivan Herman, Tonya Hongsermeier, Eric Neumann and Susie Stephens published on ***Scientific American Magazine, Dec 2007***