SEMANTIC WEB

SUBMITTED BY

TAPAS KUMAR MISHRA

11CS60R32

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

ACKNOWLEDGEMENT

I express my profound gratitude to **Dr. Jayanta Mukhopadhyay** (Professor & HOD), **Dipanwita Roy Chowdhury** (Professor), Department of Computer Science & Engineering for the valuable help and guidance in the preparation of this report "SEMANTIC WEB".

I would like to extend my sincere thanks to **Dr. Rajib Mall**, (Professor) and all other staff members of M.Tech, Computer Science & Engineering.

Finally I would also wish to record my gratefulness to all my friends and classmates for their help.

TAPAS KUMAR MISHRA

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.

- What is birthplace of Sachin Tendulkar?
- On which date, Sachin was born?
- The record "Hey Jude" was recorded by the Beatles.

Sentences like these can be understood by people. But these cannot be understood by the computers in the current representation of data on web. To make computer understand, Statements are built with syntax rules. The syntax of a language defines the rules for building the language statements. This is what the Semantic Web is all about - Describing things in a way that computer 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 things (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 analysing 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 Mark-up 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 mark-up 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.

1.3 Metadata

The first form of semantic data on the Web was metadata :
“information about information”. These basically include:

- Means of creation of the data
- Purpose of the data
- Time and date of creation
- Creator or author of data
- Placement on a computer network where the data was created
- Standards used

Example :

A meta element specifies name and associated content attributes describing aspects of the HTML page.

```
<meta name="keywords"content="wikipedia,encyclopedia">
```

Default charset for plain text is simply set with meta:

```
<meta http-equiv="Content-Type" content="text/html charset=UTF-8" >
```

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 description of concepts, terms and relationships within a given knowledge domain , Logic, Proof and Trust.

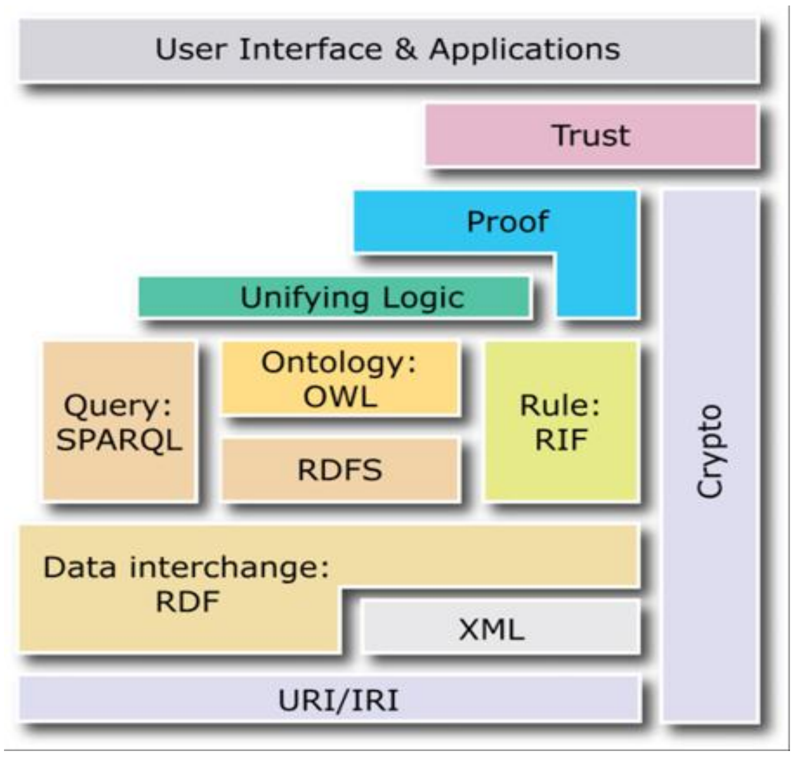


FIGURE 1-SEMANTIC WEB STACK

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.

One can classify URIs as locators (URLs), or as names (URNs), or as both. A Uniform Resource Name (URN) functions like a person's name, while a Uniform Resource Locator (URL) resembles that person's street address. In other words: the URN defines an item's identity, while the URL provides a method for finding it.

The ISBN system for uniquely identifying books provides a typical example of the use of URNs. ISBN 0486275574 (urn: isbn: 0-486-27557-4) cites, unambiguously, a specific edition of Shakespeare's play Romeo and Juliet. To gain access to this object and read the book, one needs its location: a URL address. A typical URL for this book on a Unix-like operating system would be a file path such as file:///home/username/RomeoAndJuliet.pdf, identifying the electronic book saved in a file on a local hard disk. So URNs and URLs have complementary purposes.

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.

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 labelled. 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>
```

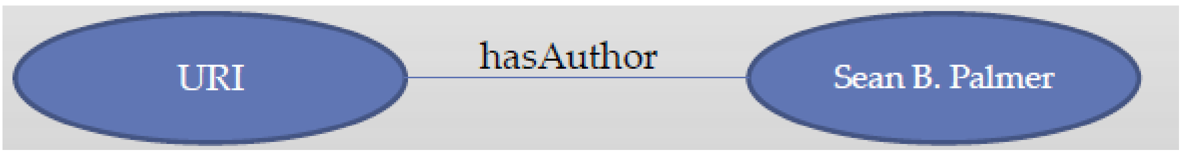
This piece of RDF basically says that this article has the title "The Semantic Web: An Introduction", and was written by someone whose name is "Sean B. Palmer". Here are the triples that this RDF produces:-

<> <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:



The simple RDF assertion triple looks like:

triple (hasAuthor, URI, Sean B. Palmer)

Advantage of RDF over XML

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 data-types 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.

2.4 Schemas of RDF: RDFS

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 is a way to describe the meaning and relationships 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" .

The triples of RDF form webs of information about related things. Because RDF uses URIs to encode this information in a document, the URIs ensure that concepts are not just words in a document but are tied to a unique definition that everyone can find on the Web. For example, imagine that we have access to a variety of databases with information about people, including their addresses. If we want to find people living in a specific zip code, we need to know which fields in each database represent names and which represent zip codes. RDF can specify that "(field 5 in database A) (is a field of type) (zip code)," using URIs rather than phrases for each term.

Problem with RDFS

The main problem with use of RDFS is the “SYNONYM problem”. For Example - two databases may use different identifiers for what is in fact the same concept, such as *zip code*. A program that wants to compare or combine information across the two databases has to know that these two terms are being used to mean the same thing. Ideally, the program must have a way to discover such common meanings for whatever databases it encounters. But there is no way of finding such concepts with RDFS.

2.5 Ontology

A solution to this problem is provided by the third basic component of the Semantic Web, collections of information called ontologies. In philosophy, an ontology is a theory about the nature of existence, of what types of things exist; ontology as a discipline studies such theories. Artificial-intelligence and Web researchers have co-opted the term for their own jargon, and for them an ontology is a document or file that formally defines the relations among terms. An ontology is an explicit description of a domain. It includes

- Concepts
- properties and attributes of concepts
- constraints on properties and attributes
- individuals (often, but not always)

An ontology defines a common vocabulary, a shared understanding.

The most typical kind of ontology for the Web has a taxonomy and a set of inference rules.

Taxonomy

The taxonomy defines classes of objects and relations among them. For example, an address may be defined as a type of location, and city codes may be defined to apply only to locations, and so on. Classes, subclasses and relations among entities are a very powerful tool for Web use. We can express a large number of relations among entities by assigning properties to classes and allowing subclasses to inherit such properties. If city codes must be of type city and cities generally have Web sites, we can discuss the Web site associated with a city code even if no database links a city code directly to a Web site.

Inference Rule

Inference rules allows to infer conclusions based on rules and facts available in the knowledge base. So this increases the power of Semantic web even more. For example: An ontology may express the rule "If a city code is associated with a state code, and an address uses that city code, then that address has the associated state code." A program could then readily deduce, for instance, that a Cornell University address, being in Ithaca, must be in New York State, which is in the U.S., and therefore should be formatted to U.S. standards. The computer doesn't truly "understand" any of this information, but it can now manipulate the terms much more effectively in ways that are useful and meaningful to the human user.

With ontology pages on the Web, solutions to terminology (and other) problems begin to emerge. The meaning of terms or XML codes used on a Web page can be defined by pointers from the page to an ontology. Of course, the same problems as before now arise if I point to an ontology that defines addresses as containing a zip code and you point to one that uses postal code. This kind of confusion can be resolved if ontologies (or other Web services) provide equivalence relations: one or both of our ontologies may contain the information that my zip code is equivalent to your postal code.

Our scheme for sending in the clowns to entertain my customers is partially solved when the two databases point to different definitions of address. The program, using distinct URIs for different concepts of address, will not confuse them and in fact will need to discover that the concepts are related at all. The program could then use a service that takes a list of postal addresses (defined in the first ontology) and converts it into a list of physical addresses (the second ontology) by recognizing and removing post office boxes and other unsuitable addresses. The structure and semantics provided by ontologies make it easier for an entrepreneur to provide such a service and can make its use completely transparent.

Ontology can be used in a simple fashion to improve the accuracy of Web searches—the search program can look for only those pages that refer to a precise concept instead of all the ones using ambiguous keywords. More advanced applications will use ontologies to relate the information on a page to the associated knowledge structures and inference rules.

In addition, this markup makes it much easier to develop programs that can tackle complicated questions whose answers do not reside on a single Web page. Suppose you wish to find the Ms. Cook you met at a trade conference last year. You don't remember her first name, but you remember that she worked for one of your clients and that her son was a student at your alma mater. An intelligent search program can sift through all the pages of people whose name is "Cook" (sidestepping all the pages relating to cooks, cooking, the Cook Islands and so forth), find the ones that mention working for a company that's on your list of clients and follow links to Web pages of their children to track down if any are in school at the right place.

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 is generally not required as the information on web does not required to be proved.

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 labelled as untrustworthy.

This part of semantic web is yet to be implemented.

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

It stands for “Semantic Interoperability of Metadata and Information in unLike Environments”, 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 Semantic Web Education & Outreach Interest Group (SWEO).

4. Conclusion

The searches on web as we see today are based on word for word matching, which sometimes is not the best strategy. The coming Semantic Web will multiply this versatility a thousand-fold. For some, the defining feature of the Semantic Web will be the ease with which one’s PDA, laptop, desktop, server, and 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 implementation of the “Trust” and “Crypto” layers of the Semantic stack along with refinement of the ontology techniques is still to be done. 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

[1]. Berners-Lee, Tim; James Hendler and Ora Lassila (May 17, 2001). "The Semantic Web". Scientific American Magazine

[2].www.en.wikipedia.org/wiki/Semantic_Web

[3].Tim Berners-Lee, with Mark Fischetti. Harper San Francisco, 1999.”Weaving the Web”

[4].http://www.w3.org/2001/sw/

[5].www.SemanticWeb.org/

[6].James Farrugia,University of Maine, Orono, ME. ” Model-theoretic semantics for the web”. ACM New York, NY, USA ©2003