The Semantic Web: A Network of Content for the Digital City

Aaron Swartz 1, and James Hendler 2

1 Co-Founder, Semantic Web Agreement Group

2 Director, Semantic Web and Agent Technologies
Maryland Information and Network Dynamics Laboratory
University of Maryland

Publication Note

This paper was submitted to the Second Annual Digital Cities Workshop and may be cited as: Swartz, A. and Hendler, J. The Semantic Web: A Network of Content for the Digital City, Proceedings Second Annual Digital Cities Workshop, Kyoto, Japan, October, 2001.

This version was translated from the original created by James Hendler in Microsoft Word, so the HTML is not as nice as I might like. Apologies. - *Aaron*

Abstract

The "semantic web" is an emerging concept in which a kind of machine-readable content enables a wide variety of new and exciting applications. Helping to power this web will be new languages for ontologies, rules and proofs. In this paper we describe the semantic web and the various components that will make it work.

1 Introduction

Recently, papers have been showing up in publications ranging from *Scientific American* [1] to *Nature* [2] describing a new and emerging vision of a more powerful World Wide Web often referred to as the "Semantic Web." On the Semantic Web, not just web pages, but databases, programs, sensors and even household appliances will be able to present data , multimedia, and status information in ways that powerful computing agents can use to search, filter and prepare information in new and exciting ways to assist the web user. New languages, making significantly more of the information on the web machine-readable, power this vision and will enable the development of a new generation of technologies and toolkits.

To understand the difference in the emerging net, imagine going to a search engine and typing the query "How many train lines are there in Japan?" If you type this as a query to any of the popular search engines, you will get many pages back (at the time of this writing, we found between 122,000 and 99,000,00 answers returned depending on the search engine and the query mode). Despite the many documents found, few, if any, actually contain the answer to the query being asked! As a foundation for browsing the web and learning much about Japanese transportation, this may be okay - but for trying to find the answer to the question asked - worthless!

Why, however, can't this answer be found? There are many resources on the web that could be used. First, among the multitude of documents found there probably are some which do provide this piece of information, but current language processing and search technologies are no where near good enough to find them. Second, there are web-accessible databases and programs that could be accessed to provide the answer, but word-based text matching is not sufficient to pull them out. Third, since each train line in Japan makes its presence known on the web in some way or another, a complicated program could be written to find these, identify them, and count how many there are - but writing such a program is a massive undertaking, and far more effort than users would be willing to make to find answers to a single query.

Thus, it is possible to imagine a day when a query like this could be typed to the web and a very different sort of response could be found. For example, a semantic web query tool could give replies like:

- http://www.transit.co.jp/lines says the number of train lines is over 5000
- There is a database that can provide that number, but you will need to provide an authorization number.
- There is a web service that can compute that number, but it will cost you 500 yen for the answer.
- I can get you an approximate answer by search and filtering, but it will take many hours to compute.

The goal of semantic web research is to develop the languages and tools that will make all this possible.

The foundation on which all this rests will be new web languages that take us from document presentation (as now done in HTML) to metadata languages like XML and RDF, and then beyond that to new languages which allow ontologies, rules, proofs and logics to be realized at a web-wide scale. Figure 1, developed by Tim Berners-Lee (the inventor of the World Wide Web), is affectionately know as "the layer cake" as it shows how these languages are expected to sit, one on top of another. In the remainder of this article, we describe the layers of this layer cake in more detail, to help explain the foundations of this new and emerging web.

2 Identifiers: Uniform Resource Identifier (URI)

Whenever you want to talk to someone about something, you use some sort of identifier. "The North Star" "The strange man at the grocery store" "Those really sour candies Bob always eats" However, when you want to be as exact as possible, you use a name. "Polaris" "Johnathan Roberts" "Mega Warheads"

On the Web, instead of giving something a name, one gives it a URI. Anything that has a URI is said to be "on the Web" and we can give anything a URI: 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 holds the rest of the Web together. You're probably already familiar with one form of URI: the URL or Uniform Resource *Locator*. A URL is the address that lets you visit a webpage, like: `http://www.w3.org/Addressing/`. If you break it down you can see that a URL lets your computer locate a specific resource (in this case, the W3C's Addressing website). In addition to URLs, there are other forms of URIs. Examaples include the familiar "mailto:" URIs which are used to encode email addresses, or the less well-know "mid:" URIs which identify email messages.

URIs are decentralized -- no one person or organization controls who makes them or how they can be used. You don't need any authority or permission to make a URI for something. You can even make URIs for things you don't own, or abstract concepts that don't even physically exist. While this flexibility gives URIs a lot of power, it also brings with it a lot of problems. Since anyone can create a URI, we will inevitably end up with multiple URIs representing the same thing and we have no way to figure out whether two URIs are definitely the same. And we'll never be able to say with certainty exactly what a certain URI means. However, these are the tradeoffs that were made so that something the scale of the Semantic Web could be built.

Common practice for giving something a URI is to create a web page that describes the object and explains that the URL of that webpage represents it. For example, we could create the URI http://logicerror.com/myWeavingTheWeb which represents my copy of the book *Weaving the Web* [3]. Now I have said that that specific URI no longer represents the web page you get back when you visit it, but instead the physical book that it describes.

This is an important fact to understand. URIs are not recipes describing to your computer how to get a specific file. Instead, they are names, which may or may not contain one way for your computer to get more information about them. Other ways to find out information about a URI are developing and ways to say things about URIs are an important part of the Semantic Web.

3 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 it. These document formats can include markup to enhance the meaning of the document. If we include enhanced meaning in our documents, they become much more useful. Instead of only being able to be used by one program (a web browser, for example) they can be used by many programs, each only using the markup it understands and ignoring the rest. Even better, each program is free to interpret the markup in the way that's best for it. For example, in a document where words are marked as "emphasized" a web browser might display them in bold. On the other hand, a voice browser (which reads web pages out loud) my read them with extra emphasis. Each program is free to do what it feels is appropriate.

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

```
I just got a new pet dog.
```

And marked-up in XML:

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

The items that I added to the XML version are called tags, because they are like attaching descriptive "tags" to certain portions of the document. A full set of tags (both an opening and closing tag) and their content is called an element and descriptions like `href="http://aaronsw.com/"` are called attributes.

With **XML Namespaces** we give each element and attribute a URI. This way, anyone can create their own tags and mix them with tags made by others. 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:

<sentence

xmlns="http://example.org/xml/documents/"
xmlns:c="http://animals.example.net/xmlns/">

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

</sentence>

4 Statements: Resource Description Framework (RDF)

Now we begin to start getting into the meat of the Semantic Web. It's wonderful that we can create URIs and talk about them with our web pages. However, it'd be even better if we could talk about them in a way that computers could begin to process what we're saying. For example, it's one thing to say "I really like Weaving the Web." on a web discussion forum. However, no computer could process what you said.

RDF gives you a way to make statements that are **machine-processable**. Now the computer (of course) can't actually "understand" what you said, but it can deal with it in a way that seems like it does. For example, someone could search the Web for all book ratings and provide an average rating for each book. Then, they could put that information back on the Web. Another website could take that information (the list of book rating averages) and create a "Top Ten Most Popular Books" page.

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://aaronsw.com/>
  <http://love.example.org/terms/reallyLikes>
<http://www.w3.org/People/Berners-Lee/Weaving/>.
```

As you might have guessed, that says that I really like Weaving the Web. You may notice that RDF statements can say practically anything, and that it doesn't matter who says them. There is no one official website that says everything about Weaving the Web, or about me. Instead, this information is spread across the Web. Two people can even say contradictory things -- Bob can say that Aaron loves Weaving the Web and John can say that Aaron hates it. This is the freedom that the Web provides.

The statement above is written in [Notation3](), a language that allows you to write simple RDF statements. However, the official RDF specification defines an XML representation of RDF:

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    xmlns:love="http://love.example.org/terms/">
    <rdf:Description rdf:about="http://aaronsw.com/">
        <love:reallyLikes
           rdf:resource="http://www.w3.org/People/Berners-Lee/Weaving/" />
    </rdf:Description>
  </rdf:RDF>
```

Now, to write RDF like that is not the easiest thing in the world, and it seems unlikely that everyone will start speaking this strange new language anytime soon. So where do we expect all this RDF information to come from? The most likely source is databases.

In the world there are thousands of databases, most containing interesting machine-processable information. Governments store arrest records in databases; companies store part and inventory information in a database; most computerized address books store people's names and phone numbers in ... you guessed it! ... a database. When information is stored in a database, it's very easy to ask the computer certain questions about the data: "Show me everyone that was arrested in the past 6 months." "Print a list of all parts we're running low on." "Get me the phone numbers of the people whose last name is Jones."

RDF is ideally suited for publishing these databases to the Web. And when we put them on the Web, we give everything in the database a URI, so that other people can talk about it too. Now, intelligent programs can begin to fit the data together. Using the available information, the computer can begin to connect the Bob Jones whose phone number is in your address book with the Bob Jones who was arrested last week and the Bob Jones who just ordered 100,000 widgets. Now, we can ask questions of all these databases at once: "Get me the phone number of everyone who ordered more than 1,000 widgets and was arrested in the last 6 months."

5 Schemas and Ontologies: RDF Schemas and DAML+OIL

All the work on databases assumes that the data is nearly perfect. Few (if any) database systems are ready for the messiness of the Web. Any system that is "hard-coded" to understand certain terms will likely go out of date, or at least have limited usefulness, as new terms are invented and defined. What if someone comes up with a new system that rates books on a scale of 1-10 instead of just saying that someone "reallyLikes" them. Programs built based on the old system won't be able to process the new information.

Worse, there's no way for a computer or human to figure out what a specific term means, or how it should be used. The use of all these URIs is useless if we never describe what they mean. This is where schemas and ontologies come in. 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.

Two closely related systems, RDF Schemas [4] and the DARPA Agent Markup Language with Ontology Inference Layer (DAML+OIL) [5] have been developed to solve this problem. For example, a schema might state that:

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .

@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .



# A creator is a type of contributor:

dc:creator rdfs:subClassOf dc:contributor .
```

Let's say for example that you build a program to collect the authors and contributors to various documents. It uses this vocabulary to understand the information it finds. One day, a vast influx of novice users from AOL start creating RDF documents. None of them know about dc:creator, so they make up their own term: ed:hasAuthor.

```
# The old way:

<http://aaronsw.com/> is dc:creator of
```

```
        <http://logicerror.com/semanticWeb-long>


  # The new way:

  <http://logicerror.com/semanticWeb-long>

      ed:hasAuthor <http://aaronsw.com/> .
```

Normally, your program would simply ignore these new statements, since it can't understand them. However, one kind soul was smart enough to bridge the gap between these two worlds, by providing information on how to convert between them:

```
  # [X dc:creator Y] is the same as [Y ed:hasAuthor X]

  dc:creator daml:inverse ed:hasAuthor .
```

Since your program understands DAML ontologies, it can now take this information and use it to process all of the hasAuthor statements it couldn't understand before

6 Logic and Proofs

From this point on, we're discussing parts of the Semantic Web that are being explored in research labs, but not yet available on the web. Thus, here we're not discussing specific systems, but instead general concepts that could become the bases for many different systems.

While it's nice to have systems that understand these basic concepts (subclass, inverse, etc.) it would be even better if we could state any logical principles that we wanted to. We make logical statements (rules) that allow the computer to make inferences and deductions.

Here's an example: Let's say one company decides that if someone sells more than 100 of our products, then they are a member of the Super Salesman club. A smart program can now follow this rule to make a simple deduction: "John has sold 102 things, therefore John is a member of the Super Salesman club." More complex rules, and the inference engines to make them available on the web, are also being explored.

Once we begin to build systems that follow logic, it makes sense to use them to prove things. Different people all around the World can write logic statements, then your machine can follow these Semantic "links" to begin to prove facts.

Example: Corporate sales records show that John has sold 55 widgets and 47 sprockets. The inventory system states that widgets and sprockets are both different company products. The built-in math rules state that 55 + 47 = 102 and that 102 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 John is a Super Salesman.

While it's very difficult to create these proofs (it could 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 could merely provide data for others to use. Others would be smarter, and could use this data to build rules. The smartest would be heuristic engines, powering "intelligent agents" which follow all these rules and statements to draw conclusions, and place their results back on the Web as proofs as well as data or query answers like those shown in the introduction.

7 Trust

Now you've probably been thinking that this whole plan is great, but rather useless if anyone can say anything. Who would trust anything from this system if anyone can say whatever they want? So you don't let me into your site? Ok, I just say I'm the King of the World and I have permission. Who's to stop me?

That's where Digital Signatures [6] 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. Aha! So I digitally sign all of my RDF statements. That way, you can be sure that I wrote them (or at least vouch for their authenticity). Now, you simply tell your 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 these measures of trust is to a certain degree (Robert can trust Wendy a whole lot, but Sally only a little).

In addition to trust, levels of distrust can be factored in. If your computer discovers a document which no one explicitly trusts, but no one has said it has totally false either, it will probably trust that information a little more than one which many people have said is false.

The computer takes all these factors into account when deciding how trustworthy a piece of information is. It can combine all this information into a simple display (thumbs-up / thumbs-down) or a more complex explanation (a description of all the various trust factors involved).

8 The Grand Semantic Web Vision

One of the best things about the Web is that it's so many different things to so many different people. Everyone can see something useful to them in the Semantic Web. Perhaps it's the fact that now your PDA, laptop, desktop, server and car can all begin to talk to each other. Perhaps it's the fact that corporate decisions that used to be hand-processed can no be automated. Perhaps it's the fact that it will become easier than ever to find the answers to your questions on the Web. Perhaps it's the fact that you can now discover how trustworthy a document on the Web is.

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 -- and there's no guarantee we'll make it -- but we've made quite a bit of progress so far. The possibilities are endless, and even if we don't ever achieve all of them, the journey will most certainly be its own reward.

References

1. Berners-Lee, T., Hendler, J., and Lassila, O., The Semantic Web, Scientific American, Vol. 284, 5, pp.34-43 (2001)

2. T. Berners-Lee, and James Hendler, Publishing on the semantic web, *Nature* 410, 1023 - 1024 (26 April 2001)

3. T. Berners-Lee, Weaving the Web, Harper, San Francisco, CA, 1999.

4. Brickley, D. and Guha, S.V. (eds) Resource Description Framework (RDF) Schema Specification 1.0, 27 March 2000. **http://www.w3.org/TR/rdf-schema/**

5. F. van Harmelen, P. F. Patel-Schneider and I. Horrocks (eds), DAML+, March, 2001. **http://www.daml.org/2001/03/reference.html**

6. J. Reagle, Digital Signature Initiative, June, 1999. **http://www.w3.org/DSig/**