How the Semantic Web Works

Status

$Revision: 1.15 $ $Date: 2002/04/22 19:11:29 $

Wishful thinking (also known as a plan, vision, or architecture).

This is to-date all written by Sandro with little feedback from anyone. Some of the ideas are from left field, some are consensus among the RDF community, and some ideas I picked up from others, esp TimBL.

Introduction

The Semantic Web is an evolving collection of knowledge, built to allow anyone on the Internet to add what they know and find answers to their questions. Information on the Semantic web, rather than being in natural language text, is maintained in a structured form which is fairly easy for both computers and people to work with.

Description

The structuring is simple: knowledge is expressed as descriptive statements, saying some relationship exists between one thing and another. "Jane has a mother, Susan" or "Susan is a mother of Jane". An enormous amount of people's knowledge can be expressed in sentences like these. "Part #4521 has a price, $19.95." "George has a city of residence, Washington D.C." "The United States has a president, George W. Bush."

This kind of information structuring was standardized for the Web in 1999 as RDF, but the basic technique goes back decades if not millenia.

Names

Of course names like "Jane" do not work very well to identify a specific person in the whole world, so on the Semantic Web we identify each of the things (and the relationships) using more complicated unambiguous names. The names we use are web addresses (sometimes called URLs or URIs). This can cause a little confusion (is "http://www.ibm.com" a company or the company's web site?), but it also turns out to be very useful for locating information about the thing.

We also have special names for text strings (literals), and we allow temporary names which function like pronouns. This lets us write "The country with the ISO country code 'US' has a president who has the name 'George W. Bush'" without ever using an identifier for the US.

Layering

Our system cannot directly express things like "Every person was once an embryo", but it can at least convey expressions in other languages, so people and systems which understand those languages can benefit from the Semantic Web as a kind of shared database. Some information will remain in natural languages, like english, which in some cases is easier for people to use. Other information may be stored in languages designed primarily for computers to understand.

Logic

Some facts (like "Mary is a mother" and "A mother is a kind of parent"") lead logically to other facts ("Mary is a parent"). That is often intuitive to people, but can be very hard to explain to a computer. When properly programmed, however, computers can be very helpful in figuring out which facts follow logically from other facts.

A precise explanation of one's terms and reasoning in some subject area, which can allow computers to help, is called an *ontology*. Ontologies can be expressed in various languages and carried by the Semantic Web. With them, computers can sometimes act as if they "understand" the information they are carrying. This is where the term "semantic" comes in; on this web, we try to make the meanings so clear that even a computer can understand them.

No logic languages have yet been recommended for the Semantic Web. Some of the experimental languages are RDF Schema, DAML+OIL, and swap/log.

Executable content (computer programs and subroutines) can be considered expressions in a "Turing complete" logic language. Swap/log is both Turing complete and fundamentally modular/reusable; it may turn out to be an essential core in practical Semantic Web clients.

Contributions

The knowledge on the Semantic Web is an aggregate of contributions from many sources, much as the Web is an aggregate of many web sites. The answers to a question you ask, like the results of a web search, will depend to some degree on who you trust, which systems are working at the time you ask, and what kind of search techniques are being used.

As you browse information, a good user interface will make it easy to correct inaccuracies and add your own knowledge as you like. Your additions will be stored in various configurable ways and can be kept private or published as openly as you choose.

Change-Over-Time

The facts stored in the Semantic Web will change over time as the true state of world changes. With each contribution signed and dated, we have a good chance of piecing together a correct and up-to-date picture.

The important point is that people's contributions are factual descriptions asserted to be true at some point in time. Queries are not really "What is the name of the president of the United States?" but "What is the name of the president of the United States according to the latest trustworthy data you have?"

Some more thoughts about handling change.

Services

How do you buy a book over the Semantic Web?

1. You browse/query until you find a suitable offer to sell the book you want.
2. You add information to the Semantic Web saying that you accept the offer and giving details (your name, shipping address, credit card information, etc). Of course you add it (1) with access control so only you and seller can see it, and (2) you store it in a place where the seller can easily get it, perhaps the seller's own server, (3) you notify the seller about it.

3. You wait or query for confirmation that the seller has received your acceptance, and perhaps (later) for shipping information, etc.

This approach allows automation of the process, detailed record-keeping, and excellent process abstraction.

Security

Access Control (three different kinds)

Performance

Protocols

[@@ out of date]The key to building large computer systems is having simple, well-defined communication protocols. For the Semantic Web, our key protocol is one for querying-for-facts, followed closely by one for updating-the-facts. Both are greatly complicated by the presence of logic languages and the distributed nature of the system. Additionally, political and commercial forces can shape a protocol.

Our model of information as descriptions can help, however. If we think of all communication acts (protocol events) in terms of information being conveyed between the parties, we see that moving descriptions around forms a general but precise protocol. (I've been calling this SWAMP).

The simplest transport would be TODL over TCP, but in practice we'll probably need to allow XML and identified-text over various protocols. The nice thing is they all look the same from above a message-passing API. See @@@ blindfold manifesto-type-stuff, with evolution path. Content of RDF-Language-Identifier.

Query Protocol

If we require query-answering-agents to implement a sufficent logic (like simple conditionals), then a query protocol is trivial; it just requires an class of objects which, when described, cause specific (results) information to be sent along an imlicit return path.

KB Update Protocol

Updating a specific knowledge base, held by an agent, can be done much like Query answering, with a logic. In this case, we can send Change objects (conditional, if necessary) to a repository, with the semantics that it should add them to its change log or, equivalently (if the effective-times and permissions are right), just perform them. This brings up the duality again - do you query (and modify) the current-state or the change-log of a repository. Probably either. Potentially, you would want to access the change-log of the change-log!

Interfaces

Semantic Web Browsers

An HTTP URI with no fragment (#whatever) should generally identify a contribution to the web, a collection of knowledge. It probably has an HTML rendering, which is that knowledge put in a form that's easy for humans to understand. It should also have a structural (RDF) form, which is easy for humans and machines to understand. On good sites, with good browsers, the HTML will become more and more useless.

A URI with a fragment identifer is taken to denote an object described in the knowledge base identified by the base URI. This is a slight stretch from its conventional usage, but fragment identifiers are in fact usually used to name something discussed on the page. (Sometimes they just name a section, but in that case usually the page has a table of contents -- information discussing sections! -- and a corresponding knowledge base might similarly identify contained collections of knowledge. But perhaps that's overreaching.)

So "http://www.w3.org/" denotes a collection of information. Typically it would include information about the consortium, perhaps using an identifier like "http://www.w3.org/#theConsortium".

A Semantic Web browser needs (1) an identifier of the thing you want information about, or the collection of things, or the collection of information, and (2) a configuration of what sources to use and what algorithms to use to find more sources, and (3) info about how to store changes you make, (4) appearance preferences.

Legacy Browser

Because both kinds of browsers use the same addresses for similar things, operation of legacy browsers is possible but tricky.

The obvious approach is to say that every SemWebID should get the same info in human-readable form when Accepting what legacy browsers accept.

That's can be difficult.

Another approach is to have the HTML generated by something like SemWalker, a server-side semantic web user interface. Or have the HTML view redirected/proxied to SemWalker.

Applications Programming Interfaces (API)

A SemWeb API looks basically like an RDF API with one (large, amorphous) RDF Graph. You also need an identifier for that client graph so you can configure it. Also, removing from the RDF Graph is much more complicated -- what if the triple being removed was generated by a rule? (do you remove the rule, remove some of its antecedents, just give an error, ... No, you don't remove anything, you just let clients know you think it should be removed at some given time.)

An object-oriented API might look something like this:

Class: (SemWeb) Object, stands in for anything you want to have information about.

Class: (SemWeb) Client, control access to Semantic Web by configuring this.

Class: Describer, Recognizer, Proxier, [yourlang]View: converts between objects in the programming system and described objects on the semweb. The proxier does it as you use the object, probably with some configurable lag to allow some caching. Maybe Object has methods giving you the access you want, but this can be faster.

In modifying data, you have to be careful about what it's supposed to mean. Programming languages have a simpler model of change than we do.

Applications

The sky is the limit. What application wouldn't work better with a global knowledge-base under the hood? Well, some might be slower. :-)

Sandro Hawke
First: 2002-03-29; This: $Id: Overview.html,v 1.15 2002/04/22 19:11:29 sandro Exp $