

Semantic Web Services

by Bijan Parsia

Bijan Parsia, a Semantic Web researcher in the MIND Laboratory at the University of Maryland, College Park, can be reached by e-mail at bparsia@isr.umd.edu

The World Wide Web allows people to follow bewildering paths through hundreds of documents on a variety of topics without losing equilibrium. People shop, plan and book travel, check accounts and pay bills, publish articles and artwork, share photos with family and friends and total strangers, build complex information systems and manage a variety of business interactions. Web programmers have done very well in building programs that help people write Web pages, build and maintain Web sites and develop sophisticated Web stores. It is much more challenging to develop programs that can *use* the Web with more human-like flexibility without a human being constantly in the loop. The Semantic Web and Web Services are two visions of how to make the Web more amenable to automated use.

The Semantic Web

“The Semantic Web is the web of connections between different forms of data that allow a machine to do something it wasn’t able to do directly.” (*Weaving the Web*, p. 185)

Programs can do a lot with the current Web, much of it critical to successful and pleasant human interaction with it. Web crawlers and spiders make link checking and site archiving easy and are absolutely essential for search engines. What makes Google *the* search engine for the Web is its scope (achieved by crawling the Web), the “goodness” of its results (achieved, in part, by automated reasoning about the links between pages) and the fact that it doesn’t require massive, disciplined human intervention by the search engine operators or by Web authors. The last point needs a bit of explanation. Google does require a staff of brilliant people who constantly enhance and tune the crawling, indexing, ranking, storage and retrieval programs. This is a significant amount of human effort, but it is dwarfed by the alternative: a team of catalogers to categorize three billion Web pages by hand.

Google also doesn’t require that page authors supply correct metadata for their pages *above* what they do naturally in writing link-rich hypertext pages. There’s no need for an explicit cataloging step, either by page authors or by Google operators. Moreover, it’s not clear that adding that step would produce better results (or even results as good).

A striking feature of this sort of automation is that it depends on the interconnectedness of the Web – its *link structure*. Web links are what make the Web work for human browsers, authors and, it turns out, for some content-sensitive programs. This suggests that a richer link structure could support not just better search, but other activities. Hyperlinking Web pages together for the convenience of human browsers turns out to be useful for building search engines that cater to the needs of those human browsers. What can we build if we have different kinds of links supporting activities besides human browsing? The best way to find out what we can do is to do it. The Semantic Web is driven by a specific vision: to explore what sorts of links between which kinds of representations supply the greatest achievable potential for the Web.

The Semantic Web is rooted in Tim Berners-Lee’s original conception of the Web. Web links were seen not just as providing a navigatory connection for the reader, but also as (partially) constituting the meaning of the linked representations. On this view, the Web is a kind of *semantic net* with Web pages as nodes and hyperlinks as arcs. The machine processable meanings of the nodes are constituted by the patterns of arcs between them. The original Web was a hypertext system instead of a pure semantic net. The nodes were bits of text with quite a bit of meaning all on their own – meaning that is largely inaccessible to current programs, though quite handy for literate people. So, the extra meaning of the Web based on knowledge representation (KR) tends to take a back seat to that based on natural language. Web links only give us

a little semantics, but it turns out that a little semantics goes a long way.

Still, it would be nice to have a little more semantics. If the original Web is a hypermedia system with aspirations toward KR, the Semantic Web seeks to be a KR system deeply integrated with global, distributed hypermedia. More precisely, the Semantic Web is a Web-like – global, distributed, universal, loosely coupled – KR extension of the human/hypermedia Web. There are no content-directed constraints on putting up a Web page or with making a Web link to anything else on the Web. These design desiderata are commonly expressed with the slogan, “Anyone can say anything about anything.” It’s not enough, therefore, to have a KR system with the same scope of subject matter as the Web (that is, any topic you care to write about), but that system must also accept input from practically anyone. This vision is as much a departure from traditional KR as the Web is from traditional hypertext. The Semantic Web requires a blending of KR and Web architecture. The difference between the current Web and the Semantic Web is that the Semantic Web asks people to say their anythings about anything in a way that’s more amenable to significant, content-sensitive machine processing.

If the Semantic Web requires more effort from the vast Web-authoring masses only to make it easier for some programmers to do useful and intelligent things with Web content, it’s not going to fly. Arguably, there’s a fairly firm limit to how much effort can sanely and fruitfully be required of Web authors. If you have to be a knowledge engineer to slap up a Semantic Web page, then most people, even smart, interested people, aren’t going to do it. On the other hand, a lot of websites already require a bit of reasonably heavy modeling, typically in the form of relational databases. Homebrew content management systems are everywhere, and many go beyond modeling the generic aspects of a website (authors, articles, pages, ads, bookmarks, etc.) and add support for various sorts of classification and content modeling.

The Semantic Web offers common languages and techniques to share and use such models in a naturally Web-like way. Just as HTML expanded from a drop-dead-simple, static hypertext format to a very rich, complex language capable of expressing very dynamic, media rich pages and websites, the Resource Description Framework (RDF) is being enriched and extended into more generally powerful modeling languages such as the forthcoming Web Ontology Language (OWL). OWL is an expressive, declarative language for describing sophisticated terminologies, classification schemes, taxonomies and ontologies in a way that allows

machine-based reasoners to derive significant, and often surprising, results. OWL reasoners can check the consistency of a model or classify resources with greater specificity as you add information or derive new classifications from your data (among other things). OWL also has constructs to express mappings between ontologies, and a hot area of research – and tentative deployment – is automating these mappings.

Web Services and The Problem of Discovery

As described, the Semantic Web emphasizes making Web-based information more friendly for programs (especially reasoning programs). Web Services, by contrast, are rooted in making it more straightforward for programs to do things on the Web. The early conception of Web Services was primarily of remote procedure calls over the Web with both the invocations (the request) and the returned value (the response) encoded in XML (Extensible Markup Language). This makes it easier to write programs that use other programs across the Web by making the remote programs (i.e., the services) feel very similar to code libraries locally provided by the operating system and programming environment.

There are two strong motivations for using Web technologies for services. The first is to reuse existing deployment of and expertise with those technologies. Web servers are everywhere, Web client libraries are even more prevalent and one can scarcely turn around without tripping over an XML parser. This motivation holds even if the services in question are to be used solely on private intranets or, for that matter, for inter-process communication on a single machine.

The second motivation is to deploy the service on the Web. A Web Service is much like a Web page or site: You’ve put it up – now you want people to use it. Often, you want a lot of people to use it. If your service is primarily intended for human beings who have already found your associated website, then the problem of how to get people using your Web Service reduces to getting them to your website and, once they are there, giving them enough information so that they can figure out how to use the associated service. Getting people to come to your website has known solutions, and a programmer can simply read your service’s documentation to learn how to invoke it. Notice that both steps rely heavily on human intervention: A human programmer has to find the service and then figure out how and when to use it. If there are only 10 Web Services, finding the right one isn’t hard. If there are a few hundred, it’s tedious, but feasible. Even these simple cases, however, can defeat automated discovery.

Furthermore, while useful Web Services may be scarce,

The big Web Services hopefuls recognize the importance of the problem of discovery and have moved to address it with Universal Description, Discovery and Integration of Web Services (UDDI).

they aren't that scarce. The hope of a lot of players, big and small, is that we'll have problems of Web Service superabundance rather than scarcity. On the Web perhaps most searches aren't critical, or, at least, aren't time critical. After all, if you're simply looking for a recipe for honey-almond nougat, spending a bit of time using search engines or browsing various recipe sites is quite reasonable. In fact, like wandering the stacks in a library, this sort of surfing has pleasures and rewards. Casual consumer shopping also works well with a leisurely air. However, these examples have a critical feature: low cost of failure. If you pay a dollar more for the book or never find an appealing recipe, the loss of money or opportunity is not very significant. However, if you are trying to secure fairly large supplies of a part (and only if you can get enough of another part), or make reservations for an emergency trip as quickly and cheaply as possible, failure can be a serious and costly problem.

Solving the Problem with Semantics

The big Web Services hopefuls recognize the importance of the problem of discovery and have moved to address it with Universal Description, Discovery and Integration of Web Services (UDDI). The *UDDI Executive White Paper* argues fervently that the current crop of search engines isn't up to the task – in fact, that a service-oriented Web requires, according to the UDDI 3.0 specification, “a ‘meta service’ for locating Web services by enabling robust queries against rich metadata.”

“Enabling robust queries against rich metadata” sounds like a pitch for the Semantic Web. But the UDDI has two additional components: UDDI is a meta-service, and it specifically focuses on locating Web Services. These constrain the task of UDDI considerably, especially in contrast with the Semantic Web's goal of helping machines deal with anyone saying anything about anything. Indeed, making UDDI a Web Service seems to have introduced a very peculiar, centralized

and anti-Web attitude into the effort. For example, the *UDDI Executive White Paper* claims “[u]ntil now, there has been no central way to easily get information about what standards different services support and no single point of access to all markets of opportunity, allowing them to easily connect with all potential service consumers.” (p. 1)

To Web people, the lack of a central way and a single point of access are good things, indeed critical features of the Web. The White Paper goes on to claim

Publishing your URL's [sic] to the UDDI registry is much more of an exact science that [sic] it is trying to get your Web site's URL into an Internet search engine. With UDDI, you have complete control over which of your business and service and service address information is published and when. This is because you are the one who publishes it. (p. 4)

Of course, “being the one who publishes” is what you are for your own website, which is a perfectly sane place to put your business and service and service address information. Being the publisher is also exactly what you are not when you put that information into “public” UDDI repositories, at least in any normal sense of being the publisher. The companies running the repositories, the “UDDI Operators,” are the publishers. You give your data to them, and they publish it. Is it surprising that the examples of UDDI Operators are huge corporations like Hewlett-Packard, IBM and Microsoft?

The restriction of the subject matter of searches to Web Services seems to have encouraged the development of less sophisticated metadata techniques – perhaps the constrained domain suggests that you can get away with (much) less. The metadata representation “system” at the core of UDDI looks especially impoverished next to OWL. Version 3.0 of the UDDI specification has begun to take taxonomies seriously, but its way of associating properties with services, combining

them, deriving sets of them from each other is horribly kludgy. It is a warning sign when a large number of your search techniques involve prefix, suffix, substring, approximate or “fuzzy” matching on strings – you are going to need a person driving that search. The lack of a logic for UDDI “tModels” (the key metadata construct) makes it practically impossible to write a system that does anything more than heuristic reasoning with them, and even that is fairly difficult and often highly application or domain specific.

Furthermore, the ontologies are coming. OWL’s predecessor language, DAML+OIL, already enjoys wide interest and use (see the ontology library at www.daml.org for a sampling). As the Web Ontology Working Group heads toward completion, it’s clear that OWL the language and associated documents defining it are very solid. The two years experience with DAML+OIL has produced a mature user base that is not only comfortable with its understanding of the language and how to use it, but of its ability to motivate and explain it to the wider world. It would be silly if the UDDI community didn’t exploit this expertise.

But if they don’t, others will. The DAML Web Services (DAML-S) joint committee is an initiative to develop Semantic Web Services to deal with the challenges of Web Services in the context of the Semantic Web. DAML-S goes far beyond discovery to provide ontologies for the composition, execution and monitoring of Web Services. One key goal is to support rich enough descriptions of how a service works that we can derive facts about what it does, how much it would cost and other features of the service. Thus, the service provider (or broker) can concentrate on describing the service from a fairly narrow, functional perspective and let AI planners figure out if their service can be fruitfully combined with others at a reasonable price and in a timely manner to produce the desired result. In other words, the DAML-S committee is developing a language for describing Web-based services to support programs that can write other programs with little or no human intervention.

Staying on the Web

When caught up with enthusiasm for a technique or technology that seems to promise a significant new bit of coolness for the Web, it’s easy to confuse coolness inherent in the new toy with coolness derived from enhancing (or being enhanced by) the Web. Expert systems are neat, but wrapping one in an HTML user interface doesn’t really change it in any interesting way. Distributed computing is sexy, but using Web servers just because they’re there doesn’t just miss one opportunity, but many.

The Semantic Web and Web Services turn out not to be quite the rivals that they sometimes seem to be. DAML-S makes good use of many Web Service technologies – SOAP (Simple Object Access Protocol) and WSDL (Web Service Definition Language), for example – and UDDI already reflects the need for rich metadata descriptions. So it seems that more Web Semantics are in the cards, assuming that either camp is basically right about the future of the Web. One thing that the history of the Web should teach us is that the unexpected is the norm.

Acknowledgements

This research comes out of work of members of MIND’s Semantic Web Agents Project, especially the work of the director, James Hendler, in collaboration with Fujitsu Laboratories of America, College Park, and also from the DAML-S coalition.

References and Further Reading

Theory of the Web

Weaving The Web: The original design and the ultimate destiny of the World Wide Web by its inventor, Tim Berners-Lee (with Mark Fischetti), HarperBusiness, 2000.

“Information management: A Proposal”, Tim Berners-Lee, www.funet.fi/index/FUNET/history/internet/w3c/proposal.html, (also in *Weaving the Web*).

Architectural Styles and the Design of Network-based Software Architectures, Roy T. Fielding, www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

Architecture of the World Wide Web, W3C Working Draft (work in progress) of the Technical Architecture Group, www.w3.org/TR/Webarch/.

Semantic Networks, John F. Sowa, www.jfsowa.com/pubs/semantic.htm

OWL/DAML-S

Web Ontology Working Group website, www.w3.org/2001/sw/WebOnt/

DAML website, www.daml.org

DAML Web Services (DAML-S), www.daml.org/services/

The True Meaning of Service, Kendall Grant Clark, www.xml.com/pub/a/2002/07/17/daml-s.html?page=1

UDDI

UDDI 3.0 specification, http://uddi.org/pubs/uddi_v3.htm

UDDI Technical White Paper, http://uddi.org/pubs/UDDI_Executive_White_Paper.pdf