Agents and the Semantic Web

James Hendler, *University of Maryland*

At a colloquium I attended recently, a speaker described a "science fiction" vision comprising agents running around the Web performing complex actions for their users. The speaker argued that we are far from the day this vision would become a reality because we don't have the infrastructure to make it happen.

Many challenges of bringing communicating multiagent systems to the Web require ontologies. The integration of agent technology and ontologies could significantly affect the use of Web services and the ability to extend programs to perform tasks for users more efficiently and with less human intervention.

Although I agree with his assessment about infrastructure, his claim that we are "far from the day" is too pessimistic. A crucial component of this infrastructure, a standardized Web ontology language, is emerging. This article offers a few pointers to this emerging area and shows how the ontology languages of the Semantic Web can lead directly to more powerful agent-based approaches—that is, to the realization of my colleague's "science fiction" vision.

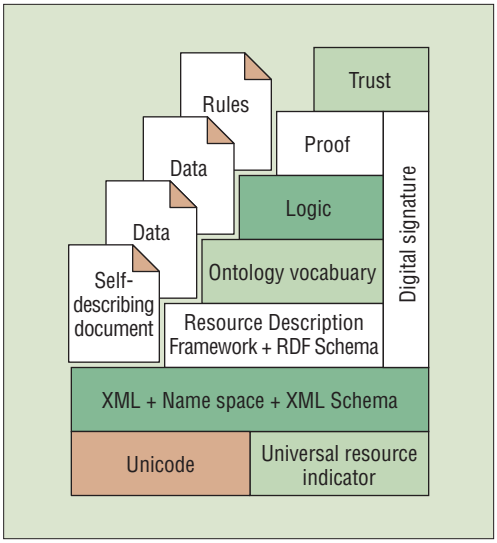

Figure 1. The Semantic Web "layer cake" presented by Tim Berners-Lee at the XML 2000 conference.

What is an ontology, really?

There are a number of terms we sometimes abuse in the AI community. These terms become even more confusing when we interact with other communities, such as Web toolkit developers, who also abuse them. One such term is *ontology*, which the *Oxford English Dictionary* defines as "the science or study of being." In AI, we usually attribute the notion of ontology to, essentially, the specification of a conceptualization—that is, defined terms and relationships between them, usually in some formal and preferably machine-readable manner.[1] Even more complicated is the relationship between ontologies and logics. Some people treat ontology as a subset of logic, some treat logic as a subset of ontological reasoning, and others consider the terms disjoint.

In this article, I employ the term as it is currently being used in Semantic Web circles. I define ontology as a set of knowledge terms, including the vocabulary, the semantic interconnections, and some simple rules of inference and logic for some particular topic. For example, the ontology of cooking and cookbooks includes ingredients, how to stir and combine the ingredients, the difference between simmering and deep-frying, the expectation that the products will be eaten or drunk, that oil is for cooking or consuming and not for lubrication, and so forth.

In practice, it is useful to consider more complex logics and inference systems to be separate from an

1094-7167/01/$10.00 © 2001 IEEE

ontology. Figure 1, derived from a talk given by Tim Berners-Lee at the recent XML 2000 conference, shows the proposed layers of the Semantic Web with higher-level languages using the syntax and semantics of lower levels. This article focuses primarily on the ontology language level and the sort of agent-based computing that ontology languages enable. Higher levels (with complex logics and the exchange of proofs to establish trust relationships) will enable even more interesting functionality, but I've left those to be discussed in other articles.

Semantic Web ontologies

The Semantic Web, as I envision it evolving, will not primarily consist of neat ontologies that expert AI researchers have carefully constructed. I envision a complex Web of semantics ruled by the same sort of anarchy that rules the rest of the Web. Instead of a few large, complex, consistent ontologies that great numbers of users share, I see a great number of small ontological components consisting largely of pointers to each other. Web users will develop these components in much the same way that Web content is created.

In the next few years, almost every company, university, government agency, or ad hoc interest group will want their Web resources linked to ontological content because of the many powerful tools that will be available for using that content. Information will be exchanged between applications, letting programs collect and process Web content and exchange information freely. On top of this infrastructure, agent-based computing will become much more practical. Distributed computer programs interacting with nonlocal Web-based resources might eventually become the dominant way in which computers interact with humans and each other. Such interaction will also be a primary means of computation in the not-so-distant future.

However, for this vision to become a reality, a phenomenon similar to the Web's early days must occur. Web users will not mark up their Web pages unless they perceive value in doing so, and tools to demonstrate this value will not be developed unless Web resources are marked up. To help solve this chicken-and-egg problem, DARPA is funding a set of researchers to both develop freely available tools and provide significant content for these tools to manipulate. This should demonstrate to the government and other parts of society that the Semantic Web can be a reality.

But without some killer apps showing the great power of Web semantics, it will still be a long row to hoe. Although I don't claim to have all the answers, perhaps some ideas in the remainder of this article will inspire the creation of exciting Web-agent applications. I will develop this vision one step at a time by describing the creation of pages with ontological information, the definition of services in a machine-readable form, and the use of logics and agents that provide important new capabilities.

Markup for free

A crucial aspect of creating the Semantic Web is to enable users who are not logic experts to create machine-readable Web content. Ideally, most users shouldn't even need to know that Web semantics exist. Lowering markup's cost isn't enough; for many users it must be free. Semantic markup should be a by-product of normal computer use. Much like current Web content, a small number of tool creators and Web ontology designers will need to know the details, but most users will not even know ontologies exist.

The ability to link and browse ontological relations enabled by the Web's use of semantics will be a powerful tool for users who do know what ontologies are and why they should be used.

Consider any of the well-known products for creating online slide shows. Several of these products contain libraries of clippings that you can insert into a presentation. Software developers could mark these clippings with pointers to ontologies. The save-as-HTML feature could include linking these products to their respective ontologies. So, a presentation that had pictures of, for example, a cow and a donkey would be linked to barnyard animals, mammals, animals, and so forth. While doing so would not guarantee appropriate semantics—the cow might be the mascot of some school or the donkey the icon of some political party—retrieval engines could use the markups as clues to what the presentations contain and how they can be linked to other ones. The user simply creates a slide show, but the search tools do a better job of finding results.

An alternative example is a markup tool driven by one or more ontologies. Consider a page-creation tool that represents hierarchical class relations as menus. Properties of the classes could be tied to various types of forms, and these made available through simple Web forms. A user could thus choose from a menu to add information about a person, and then choose a relative (as opposed to a friend, professional acquaintance, and so forth) and then a daughter. The system would use the semantics to retrieve the properties of daughters specified in the ontologies and to display them to the user as a form to be filled out with strings (such as `name`) or numbers (`age`)—or to browse for related links (`homepage`), online images (`photo-of`), and so forth. The system would then lay these out using appropriate Web page design tools while recording the relevant instance information.

Because the tool could be driven by any ontology, libraries of terms could be created (and mixed) in many different ways. Thus, a single easy-to-use tool would allow the creation of homepages (using ontologies on people, hobbies, and so forth), professional pages (using ontologies relating to specific occupations or industries), or agency-specific pages (using ontologies relating to specific functions). In an easy, interactive way the tool would help a user create a page and would provide free markup. Also, mixtures of the various ontologies and forms could be easily created, thus helping to create the Semantic Web of pages linking to many different ontologies, as I mentioned earlier.

Incremental ontology creation

Not only can pages be created with links to numerous ontologies, but the ontologies can also include links between them to reuse or change terms. The notion of creating large ontologies by combining components is not unique to the Semantic Web vision.[2] However, the ability to link and browse ontological relations enabled by the Web's use of semantics will be a powerful tool for users who do know what ontologies are and why they should be used.

How will it all work? Consider Mary, the Webmaster for a new business-to-consumer Web site for an online pet shop. Browsing

Query Processed:

- A satellite image taken yesterday at 10 AM is available on the Web at http://….
- A new satellite image, to be taken today at 10 AM, will be available for $100—click here to authorize transfer of funds and obtain image. (You will need a valid credit card number from one of the following providers.…)
- In an emergency situation, a Coast Guard observer plane can be sent to any location within the area you indicate. Service Note: You will be responsible for cost of flight if the situation does not result in an emergency pickup. Click here for more information.
- A high-altitude observer can be sent to your location in 13 hours. Click here to initiate procedure. (You will need to provide US military authorization, a valid military unit code, and the name of the commanding officer. Abuse of this procedure can result in fine or imprisonment.)
- A service entitled commercial service for providing satellite images is advertised as becoming available in 2004. See http://… for more information.

Figure 2. The results of processing a fictitious agent-based query from a fishing vessel that finds itself in a difficult weather situation.

through a Web ontology repository (such as the one at www.daml.org/ontologies/), she finds that many interesting ontologies are available. Selecting a product ontology, Mary uses a browser to choose the various classes and relations that she wants to include in her ontology. Several of these might need to be further constrained depending on the properties of her particular business. For example, Mary must define some of these properties for the various animals she will sell.

Searching further in the repository, Mary finds a biological taxonomy that contains many classes, such as **feline**, **canine**, **mammal**, and **animal**. She finds that these ontologies contain several properties relevant to her business, so she provides links to them. She adds a new descriptor field to **animal** called **product shipping type** and sets it to default to the value **alive** (not a standard property or default in the product ontology she chose to extend).

Finally, she notices that although the biological ontology contains several kinds of felines, it didn't use the categories she wanted (popular pets, exotic pets, and so forth), so she adds these classes as subclasses of the ones in the parent ontology and defines their properties. Saving this ontology on her Web site, she can now use other ontology-based tools to organize and manage her Web site. Mary is motivated to add the semantics to her site by both these tools and the other powerful browsing and search tools that the semantics enable.

The many ontology-based search and browsing tools on the Web, when pointed at her pages, can use this information to distinguish her site from the non-ontology-based sites that her competitors run. This makes it easy for her to extend her site to use various business-to-business e-commerce tools that can exploit Web ontologies for automated business uses. In addition, she might submit her ontology back into one of the repositories so that others in her profession can find it and use it for their own sites. After all, the power of the ontologies is in the sharing; the more people using common terms with her, the better.

Ontologies and services

Web services might be one of the most powerful uses of Web ontologies and will be a key enabler for Web agents. Recently, numerous small businesses, particularly those in supply chain management for business-to-business e-commerce, have been discussing the role of ontologies in managing machine-to-machine interactions. In most cases, however, these approaches assume that computer program constructors primarily use ontologies to ensure that everyone agrees on terms, types, constraints, and so forth. So, the agreement is recorded primarily offline and used in Web management applications. On the Semantic Web, we will go much further than this, creating machine-readable ontologies used by "capable" agents to find these Web services and automate their use.

A well-known problem with the Web is that finding the many available Web services is difficult. For example, when I first started writing this article, I wanted to send a Web greeting card but didn't know the name of any companies offering such a service. Using standard keyword-based searches did not help much. The query "web greeting card" turned up many links to sites displaying greeting cards or using the terms on their pages. In fact, for these three keywords, several of the most common search engines did not turn up the most popular Web greeting card service provider in their top 20 suggestions. A search on "eCards" would have found the most popular site, but I didn't happen to know this particular neologism.

As I'm finalizing this article, the search engines are now actually finding the most popular site with the "web greeting card" keywords. However, if I want something more complex—for example, an anniversary card for my mother-in-law that plays "Hava Nagila"—I'm still pretty much out of luck. As the number of services grows and the specificity of our needs increases, the ability of current search engines to find the most appropriate services is strained to the limit.

Several efforts are underway to improve this situation. Some examples are the Universal Description, Discovery, and Integration specification (www.uddi.org); ebXML (www.ebXML.org); and eSpeak (www.e-speak.hp.com). These efforts focus on *service advertisements*. By creating a controlled vocabulary for service advertisements, search engines could find these Web services. So, Mary's pet site (discussed above) might have an annotation that it provides a "sell" service of object "pet," which would let pet buyers find it more easily. Similarly, a Web greeting card site could register as something such as "personal service, e-mail, communications," and a user could more easily get to it without knowing the term "eCard."

Semantic Web techniques can—and must—go much further. The first use of ontologies on the Web for this purpose is straightforward. By creating the service advertisements in an ontological language, you would be able to use the hierarchy (and property restrictions) to find matches through class and subclass properties or other semantic links. For example, someone looking to buy roses might find florists (who sell flowers) even if no exact match served the purpose. Using description logic (or other inferential means), the user could even find categorizations that weren't explicit. So, for example, specifying a search for animals that were of "size = small" and "type = friendly," the user could end up finding the pet shop Mary is working for, which happens to be overflowing in hamsters and gerbils.

However, by using a combination of Web pointers, Web markup, and ontology languages, we can do even better than just

putting service advertisements into ontologies. By using these techniques we can also include a machine-readable description of a service (as to how it runs) and some explicit logic describing the consequences of using the service. Such service descriptions and service logic will lead us to the integration of agents and ontologies in some exciting ways.

Agents and services

In an earlier article, I described a vision of intelligent Web agents using the analogy of travel agents.[3] Rather than doing everything for a user, the agents would find possible ways to meet user needs and offer the user choices for their achievement. Much as a travel agent might give you a list of several flights to take, or a choice of flying as opposed to taking a train, a Web agent could offer several possible ways to get you what you need on the Web.

Consider a Web-enabled method for saving the doomed crew of *The Perfect Storm*.[4] In this story, now a major motion picture, a crew of fishermen is out at sea when weather conditions conspire to create a storm of epic proportions. For various reasons, the crew is unable to get a detailed weather map, so they miss that the storm is developing right in their way. Instead of avoiding it, they end up at its center, with tragic results.

How could Web agents have helped? As the ship's captain goes to call land, a wave hits and his cell phone is swept overboard. Luckily, he is a savvy Web user and has brought his wireless Web device with him. Checking the weather forecast from a standard weather site, he determines that a storm is coming, but he does not find enough detail for his needs. He goes to an agent-enabled geographical server site and invokes the query "Get me a satellite photo of this region of the Atlantic," and he draws a box on an appropriate map.

The system comes back a little later with the message shown in Figure 2. Options range from a picture available on the Web (possibly out of date) to other services (that might need special resources) and even future options being announced. The captain now chooses an option on the basis of what available resources he has and what criterion he is willing to accept. Recognizing the gravity of his situation, he invokes the Coast Guard option, which creates a scheduled overflight for his GPS location. Seeing the emerging weather, the Coast Guard arranges an emergency pickup at sea, and the sailors can go on to fish again some other day.

Using the tools of the Semantic Web, we can make this sort of thing routine and available to anyone who needs to use a Web service for any purpose. We simply need to make expressive service capability advertisements available to, and usable by, Web agents. Figure 3 depicts a complete instance of a potential *service class*. Each service class has three properties: a pointer to the service advertisement as discussed above, a pointer to a service description, and a declarative service logic. I will discuss the service logic later; I first want to concentrate on service descriptions.

Consider visiting a current business-to-consumer Web site, such as a book vendor. When you are ready to order, you usually have to fill out a form. When you click on the Submit button, you're taken to another form or returned to the same form to provide missing information or to fix an error. When you pass through the first form, you get directed to a new form where the same might happen, until eventually you provide the information necessary to complete the order. Most other Web services require similar interactions, whether to buy an item, get directions to a location, or find a particular image.

The most common way to develop these systems is with the Common Gateway Interface (CGI), in which procedural code is written to invoke various functions of the Web protocols. This code links the set of Web pages to an external resource, which means that the invocation procedure is represented procedurally on the Web. Thus, an agent visiting the page cannot easily determine the set of information that must be provided or analyze other features of the code.

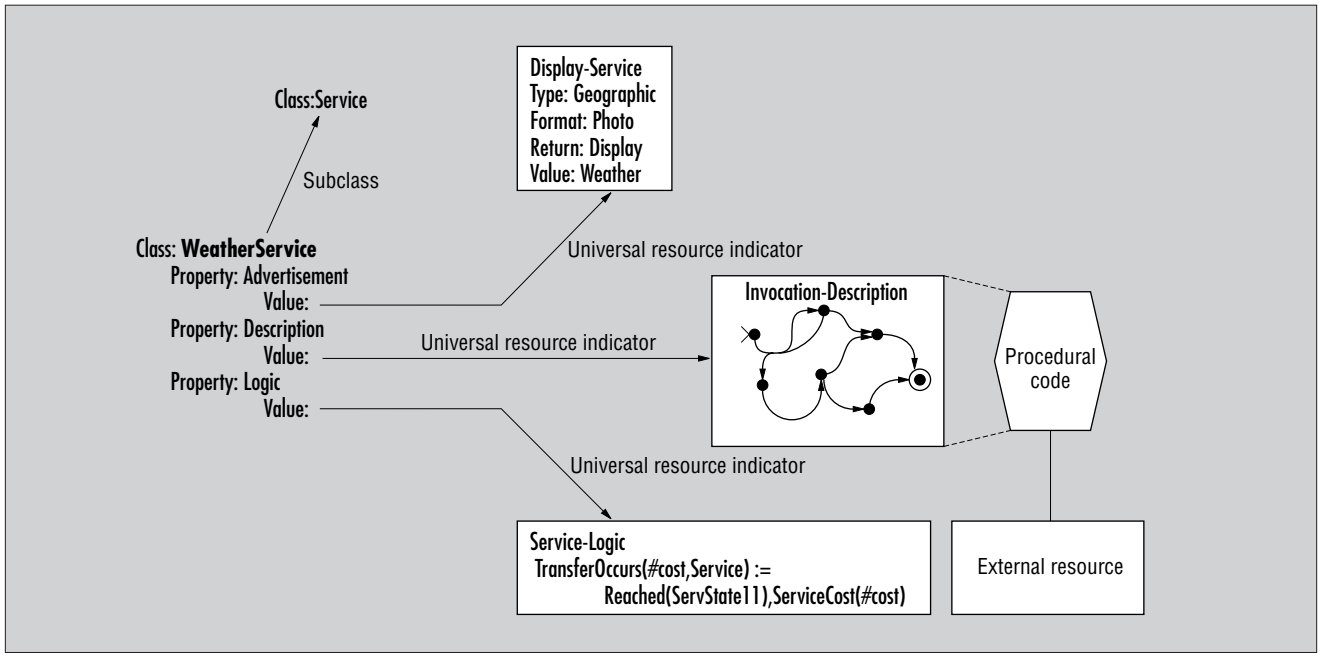

Figure 3. A potential service class and its properties on the Semantic Web.

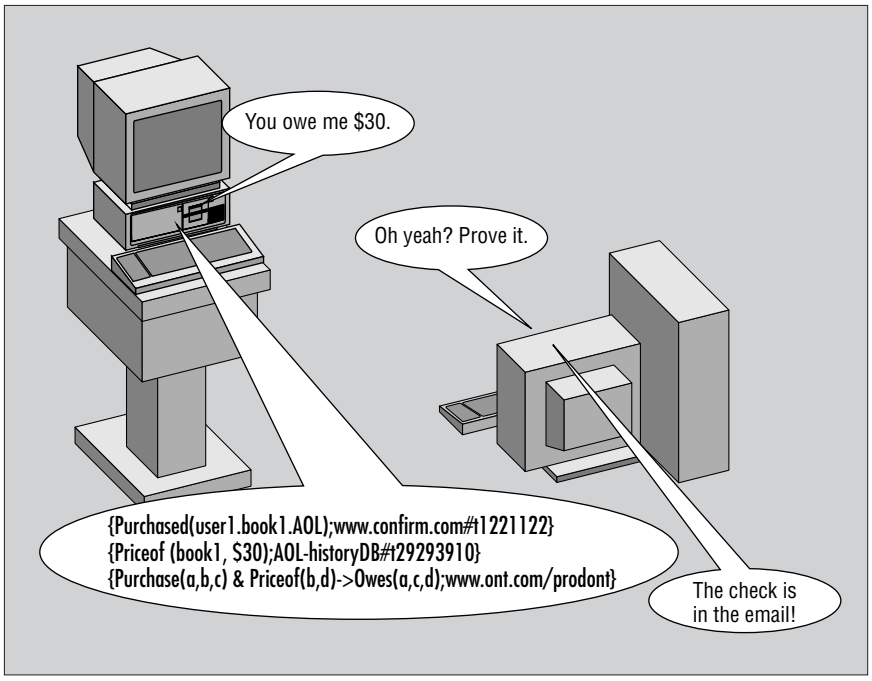

Figure 4. Agents exchanging simple proofs.

On the Semantic Web, solving this problem will be easy by using a declarative framework. Eventually you might wish to use some sort of Web-enabled logic language, but there is a much simpler way to get started. Figure 3 shows the invocation of the procedural code through a simple finite-state automaton. An ontology language such as DAML+OIL (see the sidebar "DAML and Other Languages") could be easily used to define an ontology—not of services but of the terms needed to describe the invocation of services.

Using the example of a finite-state machine (FSM), we can see what this ontology would contain. It would start with classes such as State and Link and have special subclasses such as StartState and EndState. Constraints and properties would be described to give links a head and tail, to give states a list of the links that lead out from them, and to give states a name, URI (universal resource identifier), or other identifying property. This would provide a base ontology that specific types of service providers could extend (much as Mary extended a biological ontology in the earlier example), and in which specialized ontologies could easily describe sets of terms for general use.

For example, a "standard Web sale" could be defined in some service ontology comprising a particular set of states and links. A service provider could then simply say that a particular part of a transaction was a standard Web sale, which would then find the necessary set of links and nodes through a pointer on the Web.

Exciting capabilities arise through creating such ontologies. Because these ontologies are Web-enabled and declarative, agents coming to a page containing a service description could analyze the FSM found there and would be able to determine the particular information needs for invoking the service (and reaching an EndState). An agent that had access to a set of information about a user could analyze the FSM and determine if that information would be sufficient for using this service. If not, the agent could inform the user as to what additional information would be required or other action taken.

While I've described primarily an FSM approach, there is no reason this couldn't be done using any other declarative framework. More expressive logic languages or other declarative frameworks would extend the capabilities of agents to analyze the information needs, resource requirements, and processing burden of the services so described. As these languages are linked to CGI scripts or other procedural techniques, the agents could perform the procedural invocation. This would let them actually run the services (without user intervention), thus allowing a very general form of agent interaction with off-Web resources.

Service logics

By defining languages that let users define structural ontologies, current projects (including the DARPA DAML initiative) are exploring the extension of Web ontologies to allow rules to be expressed within the languages themselves. These efforts vary in the complexity of the rules allowed, and range from description logics (as in the DAML+OIL language mentioned earlier), to SHOE's use of Horn-clause-like rules,[5] and even to first- and higher-order logics in several exploratory efforts.[6–9]

Whatever types of rules you use, they can be particularly effective in connection with the service classes, as Figure 3 shows. The service class contains (in addition to the service advertisement and service description) a pointer to a URI containing associated service logic. This logic can be used to express information that goes beyond the information contained in the service description.

For example, returning to the agent replies in Figure 2, consider a case in which the service offers an up-to-date picture (to be taken tomorrow) at some particular cost. A rule such as

```
TransferOccurs(#cost,Service) :=
    Reached(ServState11), ServiceCost(#cost)
```

might represent the information that the actual transfer of funds will occur upon reaching a particular point in the service invocation (ServState11 in this case). This information would not be obvious from the state machine itself but could be useful in several kinds of e-commerce transactions. For example, users often leave a site without completing a particular CGI script, and they cannot always know whether they've actually completed a transaction and incurred a credit card charge. Using service logics, such things could be made explicit.

More interesting transactional logics might also be used. Figure 4 shows a potential interaction between two Web agents that can use proof checking to confirm transactions. An agent sends an annotated proof to another agent. The annotations can be pointers to a particular fact on the Web or to an ontology where a particular rule resides. The agent receiving this proof can analyze it, check the pointers (or decide they are trusted

by some previous agreements), and check that the ontology is one it can read and agree with. This lets the agent recognize that a valid transaction has occurred and allow the funds to be transferred.

Such service logics could serve many other purposes as well. For example, *Heterogeneous Agent Systems*[10] discusses the use of deontic logics and agent programs for multiagent systems. These logics, tied to the appropriate service descriptions, can represent what an agent can do and when it can or cannot do so. Logical descriptions of services could also be used for automated matchmaking and brokering, for planning a set of services that together achieve a user's goal, and for other capabilities currently discussed (but not yet implemented) for multiagent systems.

Agent-to-agent communication

Of course, having pages, service descriptions, and agent programs that are linked to many ontologies, which might themselves include links to still other ontologies and so on, introduces some compelling issues. Figure 5 shows a representation of a small piece of this ontological Web. The small boxes represent agents or other Web resources that use the terms in Web ontologies represented by the larger boxes. The arrows represent any mechanism that provides a mapping (full or partial) from one ontology to another. This mapping can be as simple as inclusion of terms or as complex as some sort of ad hoc mapping program that simply reads in terms from one and spits out terms of another. The figure shows one DAG (directed acyclic graph) that could be taken from the much larger Web of ontologies.

Assuming agents are communicating with each other using the terms in these ontologies for the content terms, it is relatively straightforward for them to communicate. By linking to these ontologies, the agents commit to using the terms consistently with the usage mandated in that ontology. If the ontology specifies that a particular class has a particular property and that the property has some restriction, then each agent can assume that the other has legal values for that property maintaining that restriction.

What is more interesting, agents that are not using the same ontologies might still be able to communicate. If all mappings were perfect, then obviously any agent could communicate with any other by finding a common ontology they could both map into. More likely, however, is that the ontologies are only partially or imperfectly mapped. This would happen, for example, with Mary's pet shop site. When Mary defined her site's ontology as linking back to the zoo's animal ontology, she changed some definitions but left others untouched. Those terms that were not modified, or were modified in

DAML and Other Languages

The modern IT world is a dynamically changing environment with an exponentially increasing ability to create and publish data that rapidly swamps human abilities to process that data into information. Agent-based computing can potentially help us recognize complex patterns in this widely distributed, heterogeneous, uncertain information environment. Unfortunately, this potential is hampered by the difficulty agents face in understanding and interacting with data that is either unprocessed or in natural languages. The inability of agents to understand the conceptual aspects of a Web page, their difficulty in handling the semantics inherent in program output, and the complexity of fusing sensor output information—to name but a few problems—truly keep the agent revolution from happening.

One potential solution is for humans to meet the computer halfway. By using tools to provide markup annotations attached to data sources, we can make information available to agents in new and exciting ways. The goal of the DARPA Agent Markup Language (DAML) program is to develop a language aimed at representing semantic relations in machine-readable ways that will be compatible with current and future Internet technologies. The program is currently developing prototype tools to show the potential of such markups to provide revolutionary capabilities that will change the way humans interact with information.

To realize these goals, Internet markup languages must move beyond the implicit semantic agreements inherent in XML and community-specific controlled languages. DARPA is leading the way with DAML, which will be a semantic language that ties the information on a page to machine-readable semantics. The language must allow for communities to extend simple ontologies for their own use, allowing the bottom-up design of meaning while allowing sharing of higher-level concepts. In addition, the language will provide mechanisms for the explicit representation of services, processes, and business models so as to allow nonexplicit information (such as that encapsulated in programs or sensors) to be recognized.

DAML will provide a number of advantages over current markup approaches. It will allow semantic interoperability at the level we currently have syntactic interoperability in XML. Objects in the Web can be marked (manually or automatically) to include descriptions of information they encode, descriptions of functions they provide, and descriptions of data they can produce. Doing so will allow Web pages, databases, programs, models, and sensors all to be linked together by agents that use DAML to recognize the concepts they are looking for. If successful, information fusion from diverse sources will become a reality.

DARPA funds work in the development of DAML to help the US military in areas of command and control and for use in military intelligence. For example, one use of DAML is to improve the organization and retrieval of large military information stores such as those at the US Center for Army Lessons Learned. With respect to intelligence, DAML is aimed at improving the integration of information from many sources to provide specific indications and warnings aimed at preventing terrorist attacks on military targets such as last year's attack on the USS Cole in Yemen.

Recently, an ad hoc group of researchers formed the Joint US–EU committee on Agent Markup Languages and released a new version of DAML called DAML+OIL. This language is based on the Resource Description Framework (www.w3.org/rdf); you can find discussion of RDF's features on an open mailing list archived at http://lists.w3.org/Archives/Public/www-rdf-logic. For details of the language, a repository of numerous ontologies and annotated Web pages, and a full description of DAML and related projects see www.daml.org.

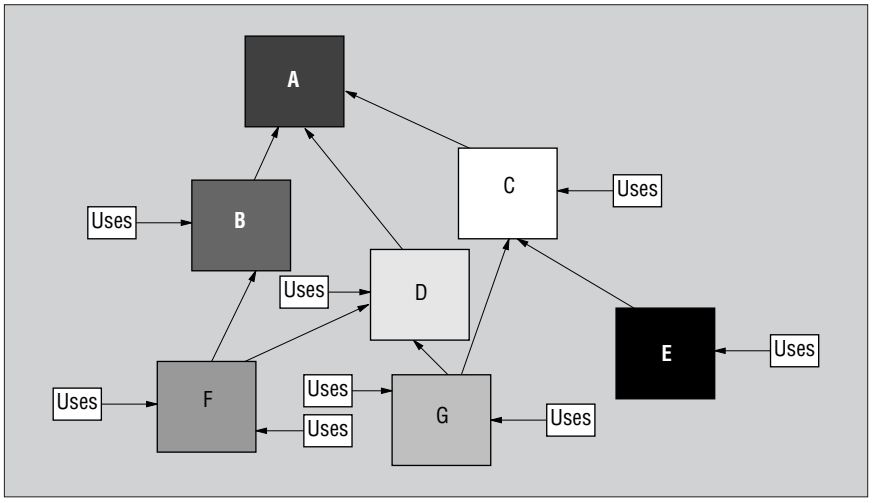

Figure 5. Mappings between agents and the ontologies they use.

certain restricted ways, could be mapped even if others couldn't. So, those ontologies made by combination and extension of others could, in principle, be partially mapped without too much trouble.

With this in mind, let's reconsider the DAG in Figure 5. Clearly, many of these agents could be able to find at least some terms that they could share with others. For agents such as those pointing at ontologies C and E, the terms they share might be some sort of subset. In this case the agent at E might be able to use only some of the terms in C (those that were not significantly changed when E was defined). Other agents, such as the ones pointing at F and G, might share partial terms from another ontology that they both changed (D in this case). In fact, all of the agents might share some terms with all the others, although this might take several mappings (and thus there might be very few common terms, if any, in some cases).

The previous discussion is purposely vague regarding what these mappings are and how they work. For certain kinds of restricted mappings, we might be able to obtain some interesting formal results. For example, if all mappings are inclusion links—that is, the lower ontology includes all the terms from the upper one in Figure 5—and we can find a rooted DAG among a set of agents, then we could guarantee that all those agents will share some terms with all others (although, in the worst case, some might only share the terms from the uppermost ontology). If the mappings are more ad hoc—they might, for example, be some sort of procedural maps defined by hand—we might lose provable properties but gain power or efficiency.

The research issues inherent in such ontology mappings are quite interesting and challenging. Two agents that communicate often might want to have maximal mappings or even a merged ontology. Two agents that are simply sending a single message (such as the invocation of an online service) might want some sort of quick on-the-fly translation limited to the terms in a particular message. Another approach might be to use very large ontologies, such as CYC,[11] to infer mapping terms between agents in other ontologies. The possibilities are endless and are another exciting challenge for researchers interested in bringing agents to the Semantic Web.

I did not intend this article to be a comprehensive technical tome. Rather, I hope that I have convinced you that several strands of research in AI, Web languages, and multiagent systems can be brought together in exciting and interesting ways.

Many of the challenges inherent in bringing communicating multiagent systems to the Web require ontologies of the type being developed in DARPA's DAML program and elsewhere. What is more important, the integration of agent technology and ontologies might significantly affect the use of Web services and the ability to extend programs to perform tasks for users more efficiently and with less human intervention.

Unifying these research areas and bringing to fruition a Web teeming with complex, intelligent agents is both possible and practical, although a number of research challenges still remain. The pieces are coming together, and thus the Semantic Web of agents is no longer a science fiction future. It is a practical application on which to focus current efforts. ❐

Acknowledgments

This paper benefited from reviews by a wide number of early readers. I especially thank Oliver Selfridge, who offered a comprehensive review, including the addition of several paragraphs. I also thank David Ackley, Tim Berners-Lee, Dan Brickley, Dan Connolly, Jeff Heflin, George Cybenko, Ora Lassila, Deborah McGuinness, Sheila McIlraith, Frank van Harmelen, Dieter Fensel, and

For Further Reading

Web sites

W3C Semantic Web Activity: www.w3.org/2001/sw
The DAML project: www.daml.org
The SHOE project: www.cs.umd.edu/projects/plus/SHOE
The Semantic Web Community Portal: www.semanticweb.org

Articles

J. Heflin, J. Hendler, and S. Luke, "Reading between the Lines: Using SHOE to Discover Implicit Knowledge from the Web," *Proc. AAAI-98 Workshop AI and Information Integration*, AAAI Press, Menlo Park, Calif., 1998, www.cs.umd.edu/projects/plus/SHOE/pubs/shoe-aaai98.ps.

S. McIlraith, "Modeling and Programming Devices and Web Agents," to be published in *Proc. NASA Goddard Workshop Formal Approaches to Agent-Based Systems*, Springer-Verlag, New York.

F. Zini and L. Sterling, "Designing Ontologies for Agents," *Proc. Appia-Gulp-Prode 99: Joint Conf. Declarative Programming*, 1999, pp. 29–42.

many participants in the CoABS, DAML, and TASK DARPA initiatives who offered comments on earlier drafts. Finally, I am indebted to an anonymous reviewer who, shall we say, wasn't impressed by an earlier version of this article and demanded copious changes. I made many of these changes, which improved the article greatly.

References

1. T.R. Gruber, "A Translation Approach to Portable Ontologies," *Knowledge Acquisition*, vol. 5, no. 2, 1993, pp. 199–220.
2. P. Clark and B. Porter, "Building Concept Representations from Reusable Components," *Proc. 14th Nat'l Conf. Artificial Intelligence* (AAAI-97), MIT Press, Cambridge, Mass., 1997, pp. 369–376.
3. J. Hendler, "Is There an Intelligent Agent in Your Future?" *Nature*, 11 Mar. 1999, www.nature.com/nature/webmatters/agents/agents.html (current 19 Mar. 2001).
4. S. Junger, *The Perfect Storm: A True Story of Men against the Sea*, W.W. Norton and Co., London, 1997.
5. J. Heflin and J. Hendler, "Dynamic Ontologies on the Web," *Proc. 17th Nat'l Conf. Artificial Intelligence* (AAAI 2000), MIT Press, Cambridge, Mass., 2000, pp. 443–449.
6. A.W. Appel and E.W. Felten, "Proof-Carrying Authentication," *Proc. 6th ACM Conf. Computer and Communications Security*, ACM Press, New York, 1999; www.cs.princeton.edu/~appel/papers/fpcc.pdf.
7. D. Fensel et al., "The Component Model of UPML in a Nutshell," *Proc. 1st Working IFIP Conf. Software Architecture* (WICSA 1), Kluwer Academic Publishers, 1999, ftp.aifb.uni-karlsruhe.de/pub/mike/dfe/paper/upml.ifip.pdf.
8. D. Fensel et al., "OIL in a Nutshell," *Proc. 12th European Workshop Knowledge Acquisition, Modeling, and Management* (EKAW-00), Springer-Verlag, New York, 2000; www.few.vu.nl/~frankh/postscript/EKAW00.pdf.
9. M. Genesereth et al., *Knowledge Interchange Format Version 3.0 Reference Manual*, http://logic.stanford.edu/kif/Hypertext/kif-manual.html.
10. V.S. Subrahmanian et al., *Heterogeneous Agent Systems*, MIT Press, Cambridge, Mass., 2000.
11. D. Lenat and R. Guha, *Building Large Knowledge-Based Systems: Representation and Inference in the CYC Project*, Addison-Wesley, Reading, Mass., 1990.

The Author

James Hendler is the Chief Scientist of DARPA's Information Systems Office and the program manager responsible for agent-based computing. He is on leave from the University of Maryland where he is a professor and head of both the Autonomous Mobile Robots Laboratory and the Advanced Information Technology Laboratory. He has joint appointments in the Department of Computer Science, the Institute for Advanced Computer Studies, the Institute for Systems Research, and is also an affiliate of the Electrical Engineering Department. He received a PhD in artificial intelligence from Brown University. Hendler received a 1995 Fulbright Foundation Fellowship, is a Fellow of the American Association for Artificial Intelligence, and is a member of the US Air Force Science Advisory Board. Contact him at jhendler@darpa.mil; www.cs.umd.edu/~hendler.