INTRODUCTION: ONTOLOGIES AND SEMANTIC WEB

The current state of Web technology—the “first generation” or “syntactic” Web—gives rise to well-known serious problems when trying to accomplish, in a non-trivial way, essential tasks like indexing, searching, extracting, maintaining, and generating information. These tasks would, in fact, require some sort of “deep understanding” of the information dealt with. In a “syntactic” Web context, on the contrary, computers are only used as tools for posting and rendering information by brute force. Faced with this situation, Tim Berners-Lee first proposed a sort of “Semantic Web,” where the access to information is based mainly on the processing of the semantic properties of this information: “The Semantic Web is an extension of the current Web in which information is given well-defined meaning [italics added], better enabling computers and people to work in cooperation” (Berners-Lee, Hendler, & Lassila, 2001, p. 35). The Semantic Web’s challenge consists then in being able to access and retrieve information on the Web by “understanding” its proper semantic content (its meaning) and not simply by matching some keywords.

From a technical point of view, the Semantic Web vision is deeply rooted into an “ontological” approach, with some proper characteristics that differentiate it from the “classical” approach to the construction of ontologies that has been described in companion article of this encyclopedia title “Ontologies and Their Practical Implementation.” We will describe these characteristics in the following sections.

BACKGROUND INFORMATION

Berners-Lee’s Architectural Proposal for the Semantic Web

To support his vision, Berners-Lee has proposed in several talks an “architecture” for the Semantic Web like that reproduced in Figure 1. Making abstraction now from all the discussions and criticisms that this proposal has brought up, what is relevant for the topic of this article is the central position that “ontologies” occupy in the architecture—and that nobody wishes to challenge. A first important difference with respect to what is expounded in the article “Ontologies and Their Practical Implementation” is, however, that ontologies are no more considered “in isolation”: They are now supported by lower-level tools like XML and RDF and must also implement an additional logic level.

In the embedded architecture of Figure 1, Unicode and URI make up the basis of the hierarchy. The Unicode standard provides a unique numerical code for every character that can be found in documents produced according to any possible language, no matter what are the hardware and software used to deal with such documents. It is supported in many operating systems and all the modern browsers, and it enables a single software product or a single Web site to be targeted across multiple platforms, languages, and countries without reengineering. URI (Uniform Resource Identifier) represents a generalization of the well-known URL (Uniform Resource Locator), which is used to identify a “Web resource” (e.g., a particular page) by denoting its primary access mechanism (essentially, its “location” on the network). URI has been created to allow recording information about all those “notions” that, unlike Web pages, do not have network locations or URLs but that need to be referred to in an RDF statement. These notions include network-accessible things, such as an electronic document or an image, and things that are not
Using Semantic Web Tools for Ontologies Construction

network-accessible, such as human beings, corporations, and bound books in a library, or abstract concepts like the concept of a “creator.”

XML (Extensible Markup Language; see Bray, Paoli, Sperberg-McQueen, Maler, & Yergeau, 2004), has been created to overcome some difficulties proper to HTML (Hypertext Markup Language), developed in 1989 by Tim Berners-Lee as a means for sharing information from any location. An HTML file is a text file characterized by the presence of a small set of “tags”—like <Head>, <Body>, <Input>, <Applet>, <Font>, etc.—that instruct the Web browsers how to display a given Web page. HTML is, then, a “presentation-oriented” markup tool. In spite of its evident utility, HTML suffers from a number of limitations, from its lack of efficiency in handling the complex client/server communication of today’s applications to (mainly) the impossibility of defining new tags to customize exactly the user’s needs. XML is called “extensible” because, at the difference of HTML, it is not characterized by a fixed format but lets the user design his own customized markup languages (a specific DTD, Document Type Description, see below) for limitless different types of documents; XML is a “content-oriented” markup tool. Basically, the syntactic structure of XML is very simple. Its markup elements are normally identified by an opening and a closing tag, like <employees> and </employees> and may contain other elements or text. The elements must be properly nested, and every XML document must have exactly one root element. Markup elements can be characterized by adding attribute/value pairs inside the opening tag of the element, like <person name="Mary">. Taking into account the nesting constraint, a very simple fragment of XML document could then be represented as: <employees> <person name="Mary"> <id>99276</id> </person> </employees>. To allow a computer to interpret correctly a fragment like this, it is necessary, however, to specify the semantics of the markup elements and tags used to make up it; a simple way of doing this is to make use of a DTD. A DTD is a formal description in XML declaration syntax of a particular type of document. It begins with a <!DOCTYPE keyword and sets out what names are to be used for the different types of markup elements, where they may occur, the elements’ possible attributes, and how they all fit together. For example, a DTD may specify that every person markup element must have a name attribute, and that it can have an offspring element called id whose content must be text. Before reading an XML document, the validating parsers and the application programs (editors, search engines, navigators, databases) read the corresponding DTD so that they can identify where every element type ought to come and how each relates to the other. There are many sorts of DTDs ready to be used in all kinds of areas (see, e.g., http://www.w3.org/QA/2002/04/valid-ddl-list.html#full) that can be downloaded and used freely. Some of them are MathML, for mathematical expressions; SMIL, Sync Multimedia Integration Language; CML, Chemical Markup Language; CML, Chemical Markup Language; PICS, Platform for Internet Content Selection; etc. A more complete way of specifying the semantics of a set of XML markup elements is to make use of XML Schema (as mentioned in Figure 1). XML Schema (see Biron & Malhotra, 2001; Thompson, Beech, Maloney, & Mendelsohn, 2001) supplies a more complete grammar for specifying the structure of the elements allowing one, e.g., to define the cardinality of the offspring elements, default values, etc.

RDF, Resource Description Framework

Moving up in the structure of Figure 1, we find now RDF (Resource Description Framework), an example of “metadata” language (metadata = data about data) used to describe generic “things” (“resources,” according to the RDF jargon) on the Web. An RDF document is, basically, a list of statements under the form of triples having the classical format: <object, property, value>, where the elements of the triples can be URIs (Universal Resource Identifiers, see above), literals (mainly, free text), and variables. To follow a well-known RDF example—reproduced, e.g., in the 2004 edition of the “official” W3C RDF Primer (Manola & Miller)—let us suppose we want to represent a situation where someone named John Smith has created a particular Web page. We will then make use of the RDF triple: <http://www.example.org/index.html(object), creator(property), john_smith(value)>.<http://www.example.org/index.html(object), creator(property), john_smith(value)> . Adding additional information about the situation, by stating, e.g., that the Web page was created May 15, 2004, and that the language in which the page is written is English, amounts to adding two statements: <http://www.example.org/index.html(object), creation_date (property), May 15, 2004 (value)> and <http://www.example.org/index.html(object), language (property), English (value)> . Note that, unfortunately, RDF uses a particular terminology for denoting the three elements of the triples, calling them subject, predicate, and object, respectively, the object, property, and value elements of the triples. This decision is unfortunate because it introduces an undue confusion with well-defined and totally different linguistic categories.

RDF triples are very easily represented as directed labeled graphs, by denoting resources as ovals, properties (predicates) as arrows, and literal values like May 15, 2004 or English within boxes. Figure 2a represents then in graph form the original statement: “John Smith has
Related Content

Database Management Issues in the Web Environment
www.igi-global.com/chapter/database-management-issues-web-environment/9208?camid=4v1a

Integrity Constraints Checking in a Distributed Database
www.igi-global.com/chapter/integrity-constraints-checking-distributed-database/44387?camid=4v1a

Is There a Difference Between the Theoretical and Practical Complexity of UML?
www.igi-global.com/chapter/there-difference-between-theoretical-practical/7026?camid=4v1a

Objectification of Relationships
www.igi-global.com/chapter/objectification-relationships/4389?camid=4v1a