Ontology-based Web Agents

Sean Luke* seanl@cs.umd.edu

Lee Spector* † lspector@hampshire.edu

David Rager* rager@cs.umd.edu

James Hendler* hendler@cs.umd.edu

*Department of Computer Science
University of Maryland
College Park, MD 20742

†School of Cognitive Science and Cultural Studies
Hampshire College
Amherst, MA 01002

Abstract

This paper describes SHOE, a set of Simple HTML Ontology Extensions which allow World-Wide Web authors to annotate their pages with semantic knowledge such as "I am a graduate student" or "This person is my graduate advisor". These annotations are expressed in terms of ontological knowledge which can be generated by using or extending standard ontologies available on the Web. This makes it possible to ask Web agent queries such as "Find me all graduate students in Maryland who are working on a project funded by DoD initiative 123-4567", instead of simplistic keyword searches enabled by current search engines. We have also developed a web-crawling agent, Exposé, which interns SHOE knowledge from web documents, making these kinds queries a reality.

Introduction

Imagine that you are searching the World-Wide Web for the home pages of a Mr. and Mrs. Cook, whom you met at a conference last year. You don't remember their first names, but you *do* recall that both work for an employer associated with the Department of Defense funding initiative 123-4567. This would certainly be sufficient information to find these people given a reasonably structured knowledge base containing all of the relevant facts. At first this also seems like enough information to find their home pages by searching the World-Wide Web, but you soon discover otherwise.

Using an existing man-made web catalog, you can find the Department of Defense's home page but learn that hundreds of subcontractors and research groups are working on initiative 123-4567. Searching existing web indices for "Cook" yields thousands of pages about cooking, and searching for "DoD" and "123-4567" provides you with hundreds and hundreds of hits about the initiative. Unfortunately, searching for all of them together yields nothing, because neither person lists the initiative on his or her web page. Aimlessly surfing the web is fruitless. What can you do?

This scenario is common to many people on the World-Wide Web. A major problem with searching on the Web today is that data available on the Web has little semantic organization beyond simple structural arrangement of text, declared keywords, titles, and abstracts. As the Web expands exponentially in size, this lack of organization makes it very difficult to efficiently glean knowledge from the Web, even with state-of-the-art natural language processing techniques, index mechanisms, or the assistance of an army of data-entry workers assembling hand-made Web catalogs. In short, there is no effective way use the World-Wide Web to answer a query like:

Find web pages for all *x, y,* and *z* such that
x is a person, *y* is a person, *z* is an organization where
lastName(*x*,"Cook"), lastName(*y*,"Cook"),
employee(*z*,*x*), employee(*z*,*y*),
marriedTo(*x*,*y*), and involvedIn(*z*,"DoD 123-4567")

Searching the Web

The chief intent of HTML and HTTP is to assist user-level presentation and navigation of the Internet; automated search or sophisticated knowledge-gathering has been a much lower priority. Given this emphasis, relatively few mechanisms have been established to mark up documents with useful semantic information beyond document-oriented information like "abstract" or "table of contents". As a result, most common indexing mechanisms and agent robots for the World-Wide Web have generally fallen into one of three categories:

- Text-indexing engines.
- Catalogs painstakingly built by hand.
- Private robots using ad-hoc methods to gather limited semantic information about pages (like "Everyone with links to me" or "All broken page links").

Each approach has disadvantages. Text indices suffer because they associate the semantic meaning of web pages with actual *lexical* or *syntactic content* From our previous example, searching a keyword index under "Cook" yielded tremendous numbers of web pages, almost none of which

[1]To appear in the Proceedings of First International Conference on Autonomous Agents 1997, AA-97.

are about living people named Cook. “Cook” has many uses besides being a last name. Although text indices are improving, the amount of information on the Web is also growing rapidly.

A major disadvantage of hand-built catalogs is the man-hours required to construct them. Given the size of the World-Wide Web, and the rate at which it is growing, cataloging even a modest percentage of web pages is a Herculean task. Additionally, the criteria used in building any catalog may turn out to be orthogonal to those of interest to a user.

Ad-hoc robots that attempt to gather semantic information from the web typically gather only the limited semantic information inferable from existing HTML tags. The current state of natural language processing technology makes it difficult to infer much semantic meaning from the body text itself at a reasonable rate (if at all). In our experience, even limiting a web robot’s natural language understanding to a small topic like Computer Science Web pages still proves surprisingly difficult to implement, and like many ad-hoc methods, such algorithms are extremely brittle.

Further, none of these approaches (except perhaps the last, for specific domains) allows for inferences about relationships *between* web pages, aside from simple facts about linkage. Sophisticated queries such as our initial “Cook” example are therefore clearly out of reach.

Solution: Adding Semantics to HTML

Instead of trying to glean knowledge from existing HTML, another approach is to give authors the ability to embed knowledge directly into HTML pages, making it simple for user-agents and robots to retrieve and store this knowledge. The straightforward way to do this is to provide authors with a clean superset of HTML that adds a knowledge markup syntax; that is, to enable them to directly classify their web pages and detail their web pages’ relationships and attributes in machine-readable form using HTML.

Using such a language, a document could claim that it is the home page of a graduate student. A link from this page to a research group might declare that the graduate student works for this group as a research assistant. And the page could assert that “Cook” is the graduate student’s last name. These claims are *not* simple keywords; rather they are semantic tags defined in some “official” set of attributes and relationships (an *ontology*). In this example the ontology would include attributes like “lastName”, classifications like “Person”, and relationships like “employee”. Systems that gather claims about these attributes and relationships could use the resulting gathered knowledge to provide answers to sophisticated knowledge-based queries.

Moreover, user-agents or robots could use gathered semantic information to refine their web-crawling process. For example, consider an intelligent agent whose task is to gather web pages about cooking. If this agent were using a thesaurus-lookup or keyword-search mechanism, it might accidentally decide that Helena Cook’s web page, and pages linked from it, are good search candidates for this topic. This could be a bad mistake of course, not only for the obvious reasons, but also because Helena Cook’s links are to the rest of the University of Maryland (where she works). The University of Maryland’s web server network is very, very large, and the robot might waste a great deal of time in fruitless searching. However, if the agent gathered semantic tags from Helena Cook’s web page which indicated that Cook was her last name, then the agent would know better than to search this web page and its links.

Related Work

HTML 2.0 (Berners-Lee and Connolly 1995) includes several weak mechanisms for semantic markup (the REL, REV, and CLASS subtags, and the META tag). HTML 3.0 (Ragget 1995) advances these mechanisms somewhat, though it is not yet an official standard. Unfortunately, the semantic markup elements of HTML have so far been used primarily for document meta-information (such as declared keywords) or for hypertext-oriented relationships (like “abstract” or “table of contents”). Furthermore, relationships can only be established along hypertext links (using <LINK> or <A>). It appears that the intent of HTML’s existing set of semantic markup tags is only to provide semantics for hypertext applications or other document-oriented functions.

To address some of these problems, Dobson and Burrill (1995) have attempted to reconcile HTML with the Entity-Relationship (ER) database model. This is done by adding to HTML a simple set of tags that define “entities” within documents, labelling sections of body text as “attributes” of these entities, and defining relationships from an entity to outside entities. Documents may contain as many entities as necessary. Dobson and Burrill associate with each entity a unique key, and establish relationships not between URL links but between keys.

Although Dobson and Burrill’s ER scheme is a significant improvement over HTML’s existing mechanism, it does not provide for any *ontological* declarations. For example, their scheme does not give any clear mechanism for classification through an “is a” hierarchy of classes. Yet one of the most significant uses for semantics in documents is to categorize them according to some classification scheme or taxonomy. For example, paper documents are often classified using hierarchical schemes like the Library of Congress subject headings, the Dewey Decimal system, or Universal Decimal Classification. Similarly, a good semantics mechanism for World-Wide Web documents needs the ability to do flexible, hierarchical classification. The ability to establish relationships between Web entities is important, but secondary to the ability to classify those entities.

Moreover, the ER model does not allow one to specify inferences that can be drawn from relationships given in web pages. Even simple specifications such as transitive closure inferences can be helpful: if Helena Cook’s home page claims that she works for the PLUS research group, and this research group is part of the Computer Science Department, part of the College of Computer, Mathematical, and Physical Sciences, part of the University of Maryland at College Park, part of the University of Maryland at College Park, part of the State of Maryland, she should not have to declare that she works for *all* of these entities; such a fact should be inferable. Invertable relationships are also useful: if George Cook is known to be married to Helena Cook, the inverse should be automatically inferable, without George or Helena having to say it. Through the addition of more powerful inferential rule capabilities, full knowledge base semantics could be provided.

SHOE

We have developed a small superset of HTML that provides many of these mechanisms. This scheme is called SHOE: Simple HTML Ontology Extensions. SHOE is intended to provide user agents with easy access to machine-readable semantic knowledge on the Web. It does this by adding to HTML a simple knowledge-representation language. SHOE provides the ability to:

- Define ontologies using HTML, which lay out classifications and entity relationship rules.
- Create new ontologies which extend existing ontologies.
- Declare entities for both whole documents and for document subsections.
- Declare relationships between entities.
- Declare entity attributes.
- Classify entities under an "is a" classification scheme.

SHOE intentionally does not have the semantic completeness of more powerful knowledge-representation schemes; instead, it attempts to strike a balance between full knowledge representation semantics and the unique needs of World-Wide Web agents which must rapidly collect and query very large bodies of information in a distributed and often chaotic domain. The formal specification provides "is a" classification and simple ground relationships.

A proposed SHOE specification adds inferential rules in the form of horn clauses without negation; we are adding semantics conservatively. An abridged specification of this language is located in the Appendix at the end of this paper. The most current specification, including inferential rules, can be found at *http://www.cs.umd.edu/projects/plus/SHOE/spec.html*

Exposé

To demonstrate the use of SHOE, we are also developing a web-crawling agent, Exposé, which parses SHOE-enabled HTML documents and adds SHOE knowledge to its internal knowledge-base. Exposé runs on Macintosh Common Lisp or C, using PARKA (Evett, Anderson, and Hendler 1993), University of Maryland's massively-parallel semantic network system, for its knowledge representation.

Exposé is really two agents. The first agent fetches ontologies from the web as directed, loading them into its PARKA knowledge-base. When fetching a SHOE ontology whose parent ontologies are unknown to Exposé, the agent will first fetch the parents in order to properly intern the ontology in Exposé's knowledge-base.

The second agent wanders the web in search of SHOE-enabled HTML documents, adding SHOE knowledge from these documents to Exposé's knowledge-base. When it discovers a SHOE document, Exposé uses its knowledge-base to interpret the SHOE data in the context of known ontologies, determine errors and redundant claims, and find new web sites to visit. If Exposé decides that this SHOE data is useful, it then adds the data to its knowledge-base. If the second agent does not have the necessary ontologies to interpret a SHOE document, it may ask the first agent to load and intern these ontologies before proceeding.

Exposé needs a knowledge-base to store not only newly-discovered SHOE knowledge but also a set of ontologies with which to interpret SHOE knowledge as it is parsed. This is a very large amount of information: optimistically, the Web could yield hundreds of commonly-used ontologies and hundreds of thousands of SHOE-enabled documents. Exposé must query this knowledge several times for each document. PARKA provides the horsepower to help Exposé do this; PARKA has been streamlined to provide real-time queries on knowledge-bases with millions of data entities and assertions.

Once Exposé has gathered knowledge from the Web, we can then use this knowledge to answer sophisticated queries about these entities and their relationships. For example, after Exposé has gathered claims from Helena Cook's web page, we can query PARKA to find her and her husband. Figure 1 shows the query we introduced in the beginning of this paper, as laid out using PARKA's Graphical Query mechanism. This is equivalent to querying PARKA with:

```
(query! '(:and
  (#!instanceOf ?X #!Person) (#!instanceOf ?Y #!Person)
  (#!instanceOf ?Z #!Organization)
  (#!lastName ?X "Cook") (#!lastName ?Y "Cook")
  (#!employee ?Z ?X) (#!employee ?Z ?Y)
  (#!marriedTo ?X ?Y) (#!involvedIn ?Z "DoD 123-4567")))
```

In conjunction with Exposé, we are designing Java applications (graphical page annotators, query mechanisms, etc.) to help users to annotate web pages with semantic knowledge and to query robot servers using SHOE. In conjunction with this effort, we are investigating the use of knowledge-representation standards like KQML (Finin et al. 1994) and KIF (Genesereth and Fikes 1992) to facilitate communication between clients and servers in retrieving results or in building up results from a number of sources.

Web Knowledge Representation Issues

In using the World-Wide Web to provide knowledge representation (KR) for agents, SHOE must address a number of issues not normally present in more common KR domains. For example, the body of knowledge on the Web is in constant flux. Ontologies may be obsolete as soon as they become popular. SHOE addresses this in two ways. First, SHOE's ontology versioning allows ontologies to be updated with newer versions while retaining integrity. Second, SHOE lets ontologies extend existing ontologies to reflect more specialized information. In SHOE, one can develop a hierarchy of ontologies: "parent" ontologies high in the hierarchy may reflect abstract, standardized classification

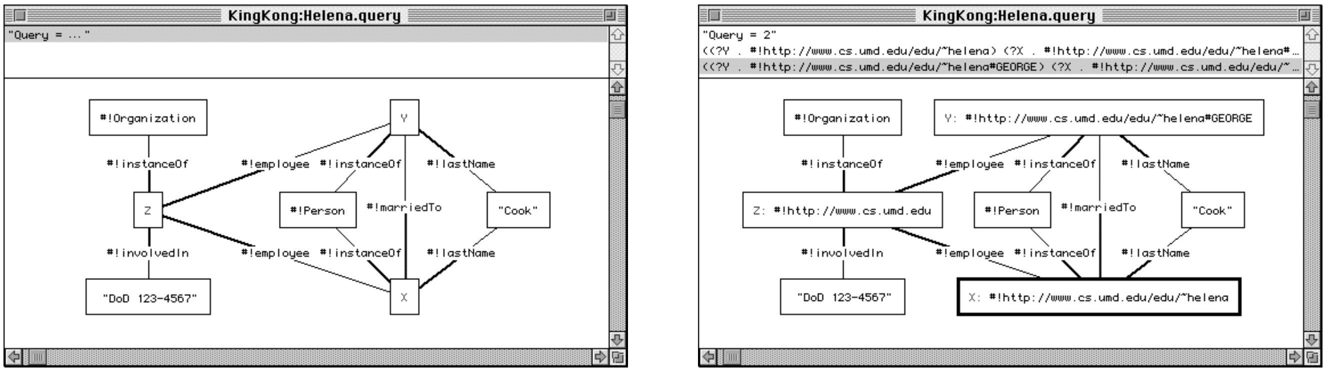

Figure 1: A Knowledge-based World Wide Web Query and Its Result in PARKA

and relationships, while "child" ontologies that extend their parents may reflect current, rapidly changing information in narrow, specialized fields.

Another important issue is the complete lack of control over the knowledge on the Web. Unlike more traditional KR domains, agents on the Web must deal with incomplete, incorrect, or unavailable information from unknown sources and of unknown value. SHOE attempts to addresses this concern in several ways. First, SHOE includes a protocol for generating unique keys for data entities which ensures that no entity can counterfeit another entity in a different document. Second, assertions made with SHOE are assumed to be *claims,* not facts. That is, if `x` claims that predicate `r` is true about `y` and `z`, this relation is best expressed as `r(x,y,z)`, not the traditional `r(y,z)`. This adds the claim's *source* as an additional variable when considering a claim's validity. Third, SHOE only allows assertions, not negations or retractions. Further, all relations are multivalued. This ensures that one entity cannot retract or replace another entity's claim. Last, SHOE attempts to bring some sensibility to what claims may be made by specifying the legal entity classifications and data types for the domain and range of relationship claims.

A final issue is the sheer size of potential Web data. Knowledge-representation languages and systems such as KIF provide a very rich semantics at the cost of computational complexity — NP-complete or worse algorithms are not uncommon. Ordinarily the costs associated with this semantic expressiveness are reasonable because the size of common KR domains is relatively small. However, such systems may not scale well: in very large domains, the cost of comprehensive semantic expressiveness becomes overwhelming. The Web can easily become such a domain, conceivably yielding millions of instances and assertions. SHOE intentionally has less semantic expressivity to help it handle the very large amounts of data that agents on the Web might have to work with.

An Introductory Example

To illustrate how SHOE works, we'll annotate the home page of George Cook (Helena Cook's husband). This example does not describe all the capabilities of our specification, but gives a taste of much of it. Before we can annotate George's home page, we need an ontology that provides:

- "Person" classification.
- "Organization" classification.
- "marriedTo" relationship between people.
- "firstName" and "lastName" attributes for people.
- "employee" relationship from organizations to people.

For the sake of this example we'll build a new ontology that provides these classifications and relationships. Ordinarily we wouldn't have to do this; instead, we'd rely on existing ontologies from common libraries on the web. Such ontologies will offer a unified structure for sharing knowledge on the World-Wide Web.

Let's assume that there already exists an ontology on the Web called organization-ontology version 2.1, which defines the classifications Organization and Thing, and that this particular ontology is available at the URL *http://www.ont.org/orgont.html.* We'll *extend* the organization-ontology ontology to include our other needed classifications and relationships. Namely, we'll borrow Organization directly, and when we define Person we'll claim that Person "is a" Thing. Let's call our extension the our-ontology ontology, version 1.0. We write our new ontology as a piece of HTML:

```
<ONTOLOGY "our-ontology" VERSION="1.0">
<ONTOLOGY-EXTENDS "organization-ontology"
   VERSION="2.1" PREFIX="org"
   URL="http://www.ont.org/orgont.html">
<ONTDEF CATEGORY="Person" ISA="org.Thing">
<ONTDEF RELATION="lastName"
   ARGS="Person STRING">
<ONTDEF RELATION="firstName"
   ARGS="Person STRING">
<ONTDEF RELATION="marriedTo"
   ARGS="Person Person">
<ONTDEF RELATION="employee"
   ARGS="org.Organization Person">
</ONTOLOGY>
```

This indicates that Person is a subcategory of Thing as defined in the organization-ontology ontology, that people have first and last names which are strings, that people can be married to other people, and that people can be employees of organizations. These tags are embedded in an HTML document, which in turn might be promulgated as an "official" person-relationships ontology.

The "official" location of our ontology is the HTML document at *http://ont.org/our-ont.html*. George Cook can now use this ontology to describe his home page. Assume that, using this ontology, Helena Cook's page has already been classified as a Person, and that its unique key is the same as its official URL: *http://www.cs.umd.edu/˜helena*. Furthermore, the place Helena and George work for, the University of Maryland's Computer Science Department, has its home page classified as an Organization, and that its unique key is the same as its official URL: *http://www.cs.umd.edu.*

To annotate George's home page, we begin by assigning his home page a key that is the same as its official URL: *http://www.cs.umd.edu/˜george* In the HEAD section of George's web page, we add:

```
<META HTTP-EQUIV="Instance-Key"
  CONTENT="http://www.cs.umd.edu/˜george">
<USE-ONTOLOGY "our-ontology" VERSION="1.0"
  PREFIX="our" URL="http://ont.org/our-ont.html">
```

This declares George's web page to be a data entity with a unique key, and indicates that it will use the ontology our-ontology to describe itself. Furthermore, every time elements from our-ontology are used, they will be labelled with the prefix our.

In the BODY section we now declare facts about George's home page, namely George's name, that George is a person, that he is married to Helena, and that he works for the University of Maryland's Computer Science Department:

```
<CATEGORY "our.Person">
<RELATION "our.firstName" TO="George">
<RELATION "our.lastName" TO="Cook">
<RELATION "our.marriedTo"
  TO="http://www.cs.umd.edu/˜helena">
<RELATION "our.employee"
  FROM="http://www.cs.umd.edu">
```

The category declaration says that George is a Person. The first two relations declare that George's name is "George Cook". The next relation declares that George is married to Helena. The last relation declares the relationship employee *from* George's employer *to* George.

If George didn't have his own web page but instead resided on a small part of his wife's web page, it would still be possible to provide George with his own unique identity and describe these relationships. In this case, we'll use *http://www.cs.umd.edu/˜helena#GEORGE* as George's unique key. We add to the HEAD section of his wife's web page (if it's not already there):

```
<USE-ONTOLOGY "our-ontology" VERSION="1.0"
  PREFIX="our" URL="http://ont.org/our-ont.html">
```

And in the BODY section we declare George to be an entity instance by adding (near the section on Helena's page that deals with George):

```
<INSTANCE "http://www.cs.umd.edu/˜helena#GEORGE">
<CATEGORY "our.Person">
<RELATION "our.firstName" TO="George">
<RELATION "our.lastName" TO="Cook">
<RELATION "our.marriedTo"
  TO="http://www.cs.umd.edu/˜helena">
<RELATION "our.employee"
  FROM="http://www.cs.umd.edu">
</INSTANCE>
```

Future Work

Although we feel our current specification provides much of the expressiveness needed for more advanced World-Wide Web agents, it still lacks important features found in sophisticated knowledge-representation systems. We are adding such features conservatively, seeking a compromise that provides some of the power of sophisticated knowledge representation tools while keeping the system simple, efficient, and understandable to the lay HTML community.

For example, the formal specification does not yet provide for annotations that describe inferences such as transitive closure or invertable relations. In a proposed addition to the specification, we have refined a small set of tags that will provide conservative inferential capabilities. The knowledge representation literature provides many insights into the design of such tags, but the unique demands of the World-Wide Web (such as the distribution of knowledge and the varying authority of authors) require that this literature be examined in a new light.

Conclusion

The Web is a disorganized place, and it is growing more disorganized every day. Even with state-of-the-art indexing systems, web catalogs, and intelligent agents, World-Wide Web users are finding it increasingly difficult to gather information relevant to their interests without considerable and often fruitless searching. Much of this is directly attributable to the lack of a coherent way to provide useful semantic knowledge on the Web in a machine-readable form.

SHOE gives HTML authors an easy but powerful way to encode useful knowledge in web documents, and it offers intelligent agents a much more sophisticated mechanism for knowledge discovery than is currently available on the World-Wide Web. If used widely, SHOE could greatly expand the speed and usefulness of intelligent agents on the web by removing the single most significant barrier to their effectiveness: a need to comprehend text and graphical presentation as people do. Given the web's explosive growth and its predominance among Internet information services, the ability to directly read semantic information from HTML pages may soon be not only useful but necessary in order to gather information of interest any reasonable amount of time.

Acknowledgements

This research was supported in part by grants from NSF(IRI-9306580), ONR (N00014-J-91-1451), AFOSR (F49620-93-1-0065), the ARPA/Rome Laboratory Planning Initiative (F30602-93-C-0039), the ARPA I3 Initiative (N00014-94-10907) and ARPA contract DAST-95-C0037.

References

Dobson, S.A. and V.A. Burrill. 1995. Lightweight Databases. In *Proceedings of the Third International World-wide Web Conference (special issue of Computer and ISDN Systems)*. v. 27-6. Amsterdam: Elsevier Science. URL: http://www.igd.fhg.de/www/www95/papers/54/darm.html

Evett, M.P., W.A. Andersen, and J.A. Hendler. 1993. Providing Computational Effective Knowledge Representation via Massive Parallelism. In *Parallel Processing for Artificial Intelligence*. L. Kanal, V. Kumar, H. Kitano, and C. Suttner, Eds. Amsterdam: Elsevier Science Publishers. URL: http://www.cs.umd.edu/projects/plus/Parka/parka-kanal.ps

Finin, T., D. McKay, R. Fritzson, and R. McEntire. 1994. KQML: An Information and Knowledge Exchange Protocol. In *Knowledge Building and Knowledge Sharing*. K. Fuchi and T. Yokoi, Eds. Ohmsha and IOS Press. URL: http://www.cs.umbc.edu/kqml/papers/kbks.ps

Genesereth, M. R., and R. E. Fikes, Eds. 1992. *Knowledge Interchange Format, Version 3.0 Reference Manual*. Technical Report Logic-92-1. Computer Science Department, Stanford University. URL: http://www-ksl.stanford.edu/knowledge-sharing/papers/kif.ps

Berners-Lee, T. and D. Connolly. 1995. *Hypertext Markup Language - 2.0*. IETF HTML Working Group. URL: http://www.cs.tu-berlin.de/ ˜jutta/ht/draft-ietf-html-spec-01.html

Ragget, D. 1995. *HyperText Markup Language Specification Version 3.0*. W3C (World-Wide Web Consortium). URL: http://www.w3.org/pub/WWW/MarkUp/html3/CoverPage.html

Appendix: An Abridged SHOE Specification

This specification describes basic features of SHOE, an extension to HTML which provides a way to incorporate machine-readable semantic knowledge in HTML or other World-Wide Web documents. This specification does not describe all of the SHOE declarations, including proposed inferential declarations. A current version of this spec is at *http://www.cs.umd.edu/projects/plus/SHOE/spec.html*

Terms

Terms not described here may be found in the HTML 2.0 specification.

Category An element under which HTML page instances or subinstances can be classified. Category names are element names, and may be prefixed. Categories may have parent categories. Categories define inheritance: if an instance is classified under a category, it is eligible to be in the domain or range of relations defined for that category or any of its parent (or ancestor) categories.

Data Non-instance data in the domain or range of a relationship. This includes:

Strings (STRING) HTML 2.0 String Literals.

Numbers (NUMBER) Floating-point numerical constants like `2, 2.0, -1.432e+4`, etc.

Dates (DATE) Date/Timestamps following RFC 1123, the HTTP/1.0 specification, section 3.3.1.

Booleans (TRUTH) HTML String Literals of the form YES or NO, case-sensitive.

Categories (CATEGORY) Category names.

Relationships (RELATION) Relation names.

Element A category or relationship name, or one of the following reserved keywords (all caps): STRING, NUMBER, DATE, TRUTH, CATEGORY, or RELATION. Element names are case-sensitive, and may contain only letters, digits, or hyphens.

Instance An element which may be classified under categories, and included in the domain or range of relationships (along with other data). *Page instances* are World-Wide Web documents. Page instances are automatically of the category Page. *Subinstances* are subsections of HTML page instance documents. Subinstances are automatically of the category PageSubinstance, and have a parentPage relationship with their parent page instance. A document which has not declared itself as an instance may still be used as one: see *Key*.

Key A string which uniquely defines a page instance or a subinstance. It is up to you to decide on the keys for your documents. For page instances of SHOE-conformant documents, the proper method is to use a single absolute URL for the document. For example, http://www.cs.umd.edu is a valid key for the document located at that URL. To create keys for subinstances, add to the page instance's unique key a pound-suffix such as #MyDog. For example, http://www.cs.umd.edu#MyDog is a valid key for a subinstance located at http://www.cs.umd.edu. It's good style for this unique key to correspond with an actual anchor in the document. The unique key of a non-SHOE-conformant document is defined to be one particular absolute URL of the document, chosen for the document by a SHOE-conformant document which references it.

Ontology As defined in this specification, a description of valid classifications for HTML page instances and subinstances, and valid relationships between instances and elements.

Prefix A small string attached with a period at the beginning of an instance, category, or relation name. For example, cs is a prefix in cs.junk. Prefixes may also be attached to already-prefixed elements, forming a *prefix chain*. For example, foo.bar.cs is a prefix chain for foo.bar.cs.junk. A prefix indicates the ontology from which the element (or prefixed element) following it is defined.

Relation (Relationship) An element which defines a relationship between two other elements. Relation names are element names, and may be prefixed. Relations may be between two instances, or between an instance and data. In this specification, all relations are binary relations. Relations have a *domain* (the element the relation is "from") and a *range* (the element the relation is "to").

Rule A formal rule in an ontology defining valid classifications (categories) or valid relationships that can be asserted.

Unique Name A string which uniquely defines an ontology. Unique names are different from *keys* in that they do not uniquely define *instances* but rather the ontologies which the instances may use. Different versions of an ontology may have the same unique name so long as they have different version numbers.

Version (Version Number) A string which describes the version of an ontology. Versions are case-sensitive, and may contain only letters, digits, or hyphens.

Declaring Ontologies

Except as specified, all declarations must be made in the BODY section of an HTML document.

Declaring An Ontology Definition An HTML document may contain any number of ontology definitions. Each ontology definition should use a unique name. Ontology definitions are accompanied with a version number. If an ontology completely subsumes previous versions of the same ontology (it contains all the rules defined in those versions), it may declare itself to be backward-compatible with those versions. To begin an ontology definition, use:

<ONTOLOGY "*ontology-unique-name*"
VERSION="*Version*"
[BACKWARD-COMPATIBLE-WITH="*Version List*"]>

"ontology-unique-name" (mandatory) The ontology's unique name.

VERSION (mandatory) The ontology's version.

BACKWARD-COMPATIBLE-WITH A whitespace-delimited list of previous versions which this ontology subsumes.

To end an ontology definition, use:

</ONTOLOGY>

All rules and extensions in an ontology must appear between the beginning and ending declarations. Ontologies may not be nested or overlap.

Extending An Existing Ontology An ontology may be declared to *extend* one or more existing ontologies. This means that it will use elements in those ontologies in its own rules. To distinguish between those elements and its own elements, an ontology must provide a unique prefix for each ontology it extends. This will be prefixed to elements borrowed from each particular ontology whenever they are referred to. To declare that an ontology is extending another ontology, use:

<ONTOLOGY-EXTENDS "*ontology-unique-name*"
VERSION="*Version*" PREFIX="*Prefix*" [URL="*URL*"]>

"ontology-unique-name" (mandatory) The extended ontology's unique name.

VERSION (mandatory) The extended ontology's version.

PREFIX (mandatory) The prefix you are assigning the extended ontology. All categories and relations from the extended ontology which are used in your ontology must be prefixed with this prefix. Within an HTML document, a prefix must be different from all prefixes declared with either <USE-ONTOLOGY ...> or <ONTOLOGY-EXTENDS ...> tags.

URL A URL that points to a document which contains the extended ontology.

Declaring Classification Rules Inside an ontology definition, an ontology may declare various new categories which instances can belong to. Categories should descend from one or more parent categories. To declare a new category, or to add new parent categories for a category, use:

<ONTDEF CATEGORY="*category-name*"
[ISA="*parent-category-list*"]>

CATEGORY (mandatory) The newly declared category, or the one being given more parent categories. Newly declared categories should be distinct from all other categories and relationships declared in the ontology.

ISA A whitespace-delimited list of categories to define as parent categories of this category.

A particular category should not be defined more than once within an ontology's declaration.

Declaring Relationship Rules Inside an ontology definition, an ontology may declare various new valid relationships between category instances or between category instances and data. To declare a relationship, use:

<ONTDEF RELATION="*relation-name*"
ARGS="*element-list*">

RELATION (mandatory) The newly declared relationship name. This should be distinct from all other categories and relationships declared in the ontology.

ARGS (mandatory) The arguments of the relation. This should be a whitespace-delimited list of exactly two elements (this specification currently supports only binary relations). The first element defines the domain of the relationship, and the second element defines the range of the relationship. Elements can be either declared categories, or the following keywords (all caps): STRING, NUMBER, DATE, TRUTH, CATEGORY, RELATION.

CATEGORY establishes a relationship not with category *instances* but with categories *themselves.* RELATION establishes a relationship not with instances but with *other relationships.* These last two elements are rare and should only be used in special circumstances.

A particular named relationship should not be defined more than once within an ontology's declaration.

Renaming Rules To reduce the number of prefixes, an ontology may rename a category or relation (plus its prefix chain) to a simpler name, so long as this name is not used in any other category or relation in the ontology. For example, an ontology could rename the category cs.junk.foo.person to simply person, so long as person is not defined elsewhere in the ontology.

Ontologies are not permitted to rename (or rename elements to) the following keywords: STRING, NUMBER, DATE, TRUTH, CATEGORY, or RELATION. To rename a category or relation, use:

<ONTDEF RENAME="*element-name*"
TO="*new-element-name*">

RENAME (mandatory) The element's old name.

TO (mandatory) The element's new name.

Marking Up HTML Documents Using Ontologies

Except as specified, all declarations must be made in the BODY section of an HTML document.

Declaring a Page Instance SHOE-conformant HTML documents must declare themselves page instances and provide a unique key for themselves. To declare an HTML document to be a page instance, add the following text to the HEAD section of the document:

<META HTTP-EQUIV="*Instance-Key*" CONTENT="*Key*">

"Key" The page instance's unique key.

Declaring a Subinstance A document may declare zero or more subinstances. Subinstances may not overlap or be nested in each other. To declare the start of a subinstance, use:

<INSTANCE "*Key*">

"Key" (mandatory) The unique key for the instance.

To mark the end of the section of a subinstance, use:

</INSTANCE>

All relationship and category declarations made within a subinstance belong to that subinstance. All relationship and category declarations made outside a subinstance belong to the page instance.

Declaring Ontology Usage Before you can classify documents or establish relationships between them, you'll need to define exactly which ontologies these classifications and relations are derived from, and associate with each of these ontologies some prefix unique to that ontology. An HTML document may declare that is using as many ontologies as it likes, as long as each ontology has a unique prefix in the document. To declare that a page instance and all its subinstances use a particular ontology, use:

<USE-ONTOLOGY "*ontology-unique-name*"
VERSION="*Version*" PREFIX="*Prefix*"
[URL="*URL*"]>

"ontology-unique-name" (mandatory) The ontology's unique name.

VERSION (mandatory) The ontology's version.

PREFIX (mandatory) The prefix you are assigning the ontology. All categories and relations from this ontology which are used in this document must be prefixed with this prefix. Within this document, the prefix *must* be different from all prefixes declared with either <USE-ONTOLOGY ...> or <ONTOLOGY-EXTENDS ...> tags.

URL A URL that points to a document which contains the used ontology.

Declaring Categories Instances may be *classified*, that is, they may be declared to belong to one or more categories in an ontology, using the CATEGORY tag:

<CATEGORY "*prefixed.category*"
[FOR="*Key*"]>

"prefixed.category" (mandatory) A category with full prefix chains showing a path through used and extended ontologies back to the ontology in which it was defined. This is the category the instance is declared to belong to.

FOR Contains the key of the instance which is being declared to belong to this category. If FOR is not declared, then the key is assumed to be that of the enclosing subinstance, or (if there is no enclosing subinstance) the page instance. If FOR is declared, then it provides the key.

Declaring Relationships Instances may declare relationships with elements:

<RELATION "*prefixed.relationship*"
[FROM="*Key*"] [TO="*Key*"]>

"prefixed.relationship" (mandatory) A relationship with full prefix chains showing a path through used and extended ontologies back to the ontology in which it was defined. This is the relationship declared between *from* the FROM element and *to* the TO element.

FROM Declares the element in the domain of the relationship. This element must be of the type declared as the domain of the relationship.

TO Declares the element in the range of the relationship. This element must be of the type declared as the range of the relationship.

If a tag (FROM or TO) is not declared, the element type for that tag must be INSTANCE, and the key for the instance is assumed to be that of the enclosing subinstance, or (if there is no enclosing subinstance) the page instance. If the tag *is* declared, and the type of the tag's argument is an instance, then it provides the key.