Semantic triage for increased Web accessibility

S. Harper
S. Bechhofer

Visually impaired users are hindered in their efforts to access the largest repository of electronic information in the world, namely the World Wide Web. A visually impaired user's information and presentation requirements are different from those of a sighted user, in that they are highly individualized and nonvisual. These requirements can become problems in that the Web is visual-centric with regard to presentation as well as information order and layout. This can and does hinder users who need access to information but cannot take advantage of the visual cues available to sighted users. Our objective is to address these problems by creating usable and appropriately "displayed" Web pages for all users who wish to understand the *meaning* of the information, as opposed to its presentation and order. We assert that the only way to accomplish this is to encode the semantic information of the page directly into the page. In this paper we describe work toward a low-overhead system to enable just this kind of semantic encoding. In particular, our approach allows semantics-based triage, that is, prioritized removal of unnecessary information from the presentation of a Web site, to make the interaction of visually impaired users with that site more productive.

INTRODUCTION

Access to and movement around complex hypermedia environments, of which the Web is the most obvious example, have long been considered important and major issues in the field of Web design and usability.[1,2] The commonly used slang phrase "surfing the Web" implies rapid and free access, pointing to the importance accorded such access by designers and users alike. It has also long been established[3,4] that this potentially complex and difficult access is further complicated, indeed becomes neither rapid nor free, if the user is visually impaired. (The term visually impaired is used here as a general term encompassing the World Health Organization definition of both profoundly blind and partially sighted individuals.[5])

We assert that the preferred way to enhance visually impaired individuals' access to information on Web pages is to encode the meaning of that information into the specific Web page involved. There are, however, problems with this approach. Empirical

evidence suggests that authors and designers will not separately create semantic markup to coexist with standard XHTML (Extensible Hypertext Markup Language) because they see it as an unnecessary overhead.

Recently, we have seen a movement in Web page design toward a separation of presentation, meta-data (XHTML), and information. However, this has not been enough to support unfettered access for visually impaired users. Consider the excellent CSS (Cascading Style Sheet) Zen Garden Web site.[6] This site is a model of the state of the art, including the application of current standards as well as the separation of presentation and content. It is also visually quite stunning. However, it is still relatively inaccessible to visually impaired people, because the information is rendered in an order defined by the designer and not in the order required by the user. Visually impaired users interact with these systems in a *serial* manner, characteristic of audio input, as opposed to the *parallel* manner characteristic of visual input. For the visually impaired, content is read from top left to bottom right; there is no scanning, and progress through information is slow. Given this interaction paradigm, we can see that visually impaired users are at a disadvantage, because they have no idea which items are menus, what the page layout is, what the extent of the content is, and where the focus of the information lies. In effect, the implicit meaning contained in the visual presentation is lost, and any possibility of enhanced meaning is also unavailable.

Even when CSS concepts do look as though they have a meaning with regard to the information presented, there is no way of relating this to the user due to the lack of machine-interpretable semantics. Therefore, the question that we face and that our Low-Cost Lightweight Instance Store (LLIS) research approach is dedicated to answering is specifically: How can semantic information be built into general purpose Web pages, without compromising the page's design vision, such that the information is as accessible to visually impaired users as it is to sighted users?

We based our approach on the following set of beliefs:

1. Visually impaired surfers need access to the meaning of information to assist in their cognition, perception, and movement around that information, and to assist in the formulation of their world-view.[3,7] This is also true for sighted users, but pages are normally created with sighted users in mind, and thus these requirements are typically more often met for sighted users.
2. Based on empirical and anecdotal evidence, in building Web pages authors and designers will not accept a *semantic overhead*, that is, a demand for significant additional effort to encode semantic information when creating these pages.[8]
3. A Web page should itself be thought of as an application, comprised of functional elements, presentation elements, and information elements, within the browser application.

One of the goals of the Semantic Web vision[9] is to make knowledge accessible to automated agents and, at the same time, also provide strong human input and benefit. In this framework, our goal is to make the role of the objects that support visual accessibility through presentation explicitly *interpretable* by humans by means of Web browsers, thereby enhancing the ability of the visually impaired to interact with the Web. Thus, it becomes necessary to associate meta-data and semantics with XHTML objects. In the context of the Semantic Web, this also implies that objects should be machine-understandable rather than simply machine-readable.

Our goals and set of beliefs led us to a simple and lightweight solution. The approach is basically to create an *ontology* (a collection of shared terms that can be communicated to both people and applications) to represent the meaning of data within XHTML metatags and then to encode this meaning into the data by leveraging the `class` and `ID` attributes common to most XHTML elements. CSS presentation is unaffected, but semantics is then an implicit part of the data. By using this method to encode semantic information, we can also deal with legacy sites and make these compatible with our scheme. Throughout this paper we use an example site called `blogger.com` to show how this can be done.

The outline of the paper is as follows: In the next section we discuss in more detail the concept of semantics and its application in our work. In particular, the Semantic Web aims at making Web resources more accessible to automated processes

by adding *semantic annotations*, meta-data that describes information content. It is envisaged that the semantics in these semantic annotations will be given by ontologies, which in turn will provide a source of precisely defined terms (vocabularies) that are amenable to automated reasoning. (In this context, an important concept is that of a *foundation ontology*, a core glossary in whose terms everything else must be described.) We use this automated reasoning to assist with our triage activity. (The term *triage* is used in this paper to describe the sorting and allocation of information on the basis of need or likely benefit.)

Adding semantics to an XHTML document is not a new concept. Work dates from the late 1990s, and concrete solutions were proposed as early as 2002. Likewise, *transcoding*, a technology used to adapt incomplete or badly written hypertext Web content, has been in use since the mid-to-late 1990s. Both technology domains are important to our work, as they place our contribution in context. In the next section of the paper, we describe the problems associated with this prior work and give an overview of why our system is both different and unique. We then describe the concepts, rationale, and techniques behind our system, focusing on XHTML. We show how these features are referenced through XHTML pages and demonstrate how our lightweight LLIS system can contribute to the accessibility of information by means of low-cost semantics in the form of an *instance store* (described later). We again consider the legacy site `blogger.com`, for which we have created an ontology, and show how our Mozilla**-based LLIS application can transform its pages into more accessible forms.

Thus far we have demonstrated the first stage in a more elaborate system to enable semantic information to be freely accessible by all users. By knowing the meaning of the information that is being encountered, visually impaired users are able to perform their own triage on that information. In the final section of the paper we discuss our conclusions from the work done to date and our plans for future work.

ON SEMANTICS

As articulated by Tim Berners-Lee,[9] the Semantic Web vision describes a Web in which resources are accessible not only to humans but also to automated processes, for example, automated *agents* roaming the Web performing useful tasks such as improved (in terms of precision) search and resource discovery, information brokering, and information filtering. The automation of tasks depends on elevating the status of the Web from machine-readable to something we might call machine-understandable. The key idea is to have data on the Web defined and linked in such a way that its meaning is explicitly interpretable by software processes rather than just being implicitly interpretable by humans.

To realize this vision, it is necessary to annotate Web resources with meta-data (i.e., data describing content and functionality). Such annotations and meta-data are, however, of limited value to automated processes unless these processes share a common understanding as to their meaning. This sharing of meaning will be achieved partly through the use of ontologies.[10]

There has been substantial effort toward this goal. For example, the Semantic Web Language Stack consists of a number of languages and specifications intended to deliver meta-data and ontologies.[11] The Resource Description Framework (RDF) provides a simple data model based on triples (so-called because each RDF triple consists of a subject, a predicate, and an object).[12] This gives a common data model, but does not supply any ontological primitives. The RDF Schema (RDFS) approach provides basic machinery allowing the definition of classes and properties along with simple class hierarchies.[13] However, RDFS is a relatively inexpressive language, and realistic domain ontologies require the ability to describe concepts and classes in a rich way. This capability is provided by OWL, the recently standardized W3C** (World Wide Web Consortium) Web Ontology Language.[14] OWL is an expressive language, in which rich, explicit descriptions of terms or concepts can be given. Moreover, OWL is a *formal* language—it has a well-defined semantics that describes how such rich expressions should be interpreted. The provision of the semantics means that the information contained in the ontology, for example, concept descriptions, is then amenable to automated reasoning. This latter process draws, in particular, on work from the knowledge representation community.[15] A reasoning engine, or *reasoner*, can then make inferences about the relationships among classes and support queries over collections of instances.[16,17]

Overall, this approach supports the use of compositional or property-based modeling. Rather than explicitly building a *subsumption* (or *kind-of*) hierarchy explicitly, concepts can be described, and the hierarchical relationships between those descriptions can then be inferred. The process of constructing such hierarchies is often referred to as *classification*. OWL supports the description of concepts in terms of both necessary and sufficient conditions. This is in contrast to traditional frame-based systems, in which, in general, necessary conditions can be described but sufficient conditions are not. For example, we can now define the `removableCSSComponent` concept as being one for which the `isRemovable` property is set to `true`. Any concepts which are described as having this property and value combination will then be classified as being concepts of the kind `removableCSSComponent`. This is a simple example, but more complex situations involving Boolean combinations of concepts and quantification of relationships (e.g., the statement that a menu-only page is one which contains only elements that are kinds of menus) can be supported. The semantics ensures that the interpretations of these combinations, and thus the inferences that can be drawn, are consistent.

OWL thus provides a mechanism that supports interoperation of applications not just at a syntactic level (through the adoption of common formats based on XML (Extensible Markup Language)), but also at a semantic level. It is this provision of well-behaved and well-specified inference procedures based on shared semantics that creates the machine understandability of the Semantic Web.

RELATED WORK

As we have seen, semantics (in the form of ontologies) provides a source of precisely defined terms (vocabularies) that are amenable to automated reasoning. We use semantics to drive our triage. There have been other prior solutions based on qualitatively similar approaches. These solutions either annotate or encode terms within XHTML, and then use transcoding and page-clipping techniques to reformat the information. Here we describe these technologies as they relate to our solution.

Encoding semantic information into XHTML

As noted previously, adding semantics to an XHTML document is not a new concept. For example, Berners-Lee proposed embedding XML RDF in HTML (Hypertext Markup Language) documents as part of the tag project.[18] However these documents did not pass XHTML validation checking and thus did not find favor within the community.[19] A subsequent version was created that did pass validation when a small DTD (Document Type Definition) using XHTML Modularization was included. However, this was not considered to be a good solution, as it involved the creation of unique extensions more or less arbitrarily. In fact, this work concluded that the RDF specification describes how to understand the semantics (in terms of RDF triples) in an RDF document that contains *only* RDF, but does not explain how and when one can extract semantics from documents in other namespaces that contain embedded RDF. (A namespace is a set of names defined according to some naming convention.) Similarly, the XHTML specification explains how to process XHTML namespace content, but gives no indication about how to process embedded RDF information.[18]

Other methods have been proposed in which object or script elements are used. However, in these cases the code becomes unreadable and therefore less workable, although this can be partially remedied by linking to the RDF in an external file.[20] Use of the XHTML `link` element has also been proposed. However the main problem with this method is that the RDF is not actually embedded in the HTML source, but rather in a separate file.[20] If the original information should change, this latter file would need to be synchronized with the now changed original information, and the amount of work needed to create the resource is the same as creating two separate and disjoint files. Time and effort are not saved.

As an alternative, Connolly proposed the HyperRDF system, in which HTML is used as the conduit to allow XSLT (Extensible Style Language Transformations) to transform information into RDF. However, HyperRDF cannot be validated, because the `head` element does not allow an `ID` attribute.[21]

Augmented meta-data for XHTML is an implementation that allows so-called Dublin Core meta-data to be incorporated in Web pages in a way that is compatible with today's Web browsers.[22] The basic premise is that one can take the `profile` attribute to be a global namespace prefix for all of the `rel/meta`

and `name` attributes throughout the document. This approach is mainly useful for authors who themselves want to use a simple mechanism for producing RDF from their XHTML. It is ineffective from the point of view of anyone who wants to randomly extract RDF from XHTML because one cannot tell whether the author actually wanted the assertions to be converted into the specific triples produced by the algorithm.

GRDDL

The most recent thinking on semantic encoding comes in the form of GRDDL (Gleaning Resource Descriptions from Dialects of Languages).[23] This work is being undertaken by the W3C Web Coordination Group and is a mechanism for encoding RDF statements in XHTML and XML. GRDDL shares some common features with HyperRDF and works on the principle that the HTML specification provides a mechanism for authors to use particular meta-data vocabularies, and thereby indicate the author's intent to use those terms in accordance with the conventions of the community that originated them. Authors may wish to define additional link types not described in this specification. If they do so, they should use a *profile* to cite the conventions used to define the link types. GRDDL is one of these profiles and uses XSLT to transform a page into an RDF description.

Why GRDDL does not work for us

Our research centers around both the designer and the user. We wish to support the designer because, in doing so, we make sure our target user group is supported by the designer's creation. However, in our conversations with designers the resounding message we have received is:

> If there is any kind of overhead above the normal concept creation then we are less likely to implement it. If our design is compromised in any way we will not implement. We create beautiful and effective sites; we're not information architects.[24]

Many Web designers have migrated from print media to Web design, and this preexisting experience in creating static artifacts causes them to see a design as fixed and immutable once created. In this view, a designer creates and controls the development of what is in effect a piece of art, which should not be changed or violated once created. It can be difficult to convey to designers the idea that users often require Web pages to adapt to their needs, and this need sometimes goes beyond art.

We suggest that designers need a lightweight, no-frills approach to including semantic information within XHTML documents. In effect the addition of the semantic information should be seamless, indivisible, and have a low-cost design overhead in terms of both time and effort.

■ Designers need a lightweight, no-frills approach to including semantic information within XHTML documents ■

Transcoding

Transcoding is a technology used to adapt Web content so that it can be viewed on any of the increasingly diverse devices available at a given instant. It has been used for a number of years in the context of making incomplete or badly written hypertext accessible to visually impaired users and their accessibility technologies. Transcoding in this context normally involves the following elements:

1. Syntactic changes, for example, shrinking or removing images[25]
2. Semantic rearrangement and fragmentation of pages based on the meaning of a section[26,27]
3. Annotation of the page created by a reader[9]
4. Generated annotations created by the content management system[28]

There are a number of different ways that transcoding can take place. In one example, the original material (an HTML document, for example) is analyzed by a program that creates a separate version containing annotations. The annotations include information that instruct the reformatting process. Inaccessible elements are removed or altered. Available systems are typically based along these lines and usually address a specific problem set. Some are annotation-based;[25] others generate text-only versions;[26,27] some filter the content;[29] others are specifically used for small-scale device interaction.[28] Regardless of which system is used, it invariably does not transform all inaccessible

elements but rather a subset, leaving holes in the accessibility of the resulting transcode. Transcoding approaches are discussed in more detail in the following subsections.

Annotation

The goal of annotations for Web-content transcoding has been to provide better support either for audio rendering, and thus for visually impaired users, or for visual rendering in small-screen devices. The problem of rendering Web pages in audio has some similarities to the problem of displaying Web pages on small-screen devices. For example, in both cases only a small portion of the page is viewable at any point. However, there are major differences in requirements. Although the amount of information that can be accessed at one time on a small-screen device is certainly limited, the interaction is still visual. The provided visual rendering is persistent (i.e., the screen acts as an external memory device), as opposed to audio rendering, which is transient. Additionally, audio is less focused and more serial in nature than visual rendering, and the user cannot easily and quickly shift focus.

Various proxy-based systems to transcode Web pages have also been proposed, based on external annotations for visually impaired users.[30,31] Their main focus is on extracting visually fragmented groupings, as well as information about roles and importance. More specifically, eight different roles have been proposed for such annotation, including proper content, header, and footer roles. Such roles are mainly focused at an abstract level and are not rich enough to fully annotate the page for accessibility. They do not support deep understanding and analysis of pages, and consequently the supported transcoding is constrained by these proposed roles.

Other work has centered on small-screen devices and proposes a system to transcode an HTML document by fragmenting it into several documents.[32] The transcoding is based on an external annotation framework. In addition to the differences explained previously, because the focus is small-screen devices, the physical and performance constraints of these devices must be considered, including, for example, screen size, memory size, and connection bandwidth. These are not, however, the primary considerations for users accessing Web pages in audio.

Semantic transcoding

In semantic transcoding, semantics provides the machine understandability and knowledge reasoning, and transcoding provides the transformation technique. Presently, however, such systems are limited to page analysis,[33] wherein a page built according to a set template can be analyzed and transformed by semantic or semantic-like technologies. Annotation is another popular way to add semantics to legacy systems. These applications rely on the XHTML page being annotated by using a specified ontology or taxonomy. These annotations are normally stored in an annotation database and are then used when the page is processed to make the transformations.[34] However, even the few systems that try to remove themselves from XHTML presentation to a more service-oriented (XML) model still specifically require that the data source be modified, a serious limitation.[35]

Transcoding summary

Each of these types of transformations is fraught with problems with regard to the acceptability of the results generated. This is especially true when both sighted users and visually impaired users wish to use the same Web page. Automatic transcoding based on removing parts of a page results in too much information loss, and manual transcoding is nearly impossible when applied to dynamic Web sites. Most systems use their own custom proxy servers or client-side interfaces, and these systems require a significant set-up cost in terms of user time. Finally, some systems require custom automatic annotation by a content generator, and so are not available to every user and all systems.

Often transcoding systems lean toward solving the problems of one user group and potentially destroy the content, structure, and context for other non-target groups. This directly challenges the nature of the World Wide Web and of other open hypermedia systems in general.

THE LLIS APPROACH

Our LLIS Triage Tool is unique in that it uses an ontology created from a preexisting CSS to allow triage of XHTML components. An overview of the system can be seen in ***Figure 1***. The CSS (component 1 in Figure 1) is used to create concepts in the ontology (component 3). These concepts are marked with attributes such as `isRemovable`, and new container classes such as `removableCSSComponent`

are created. As described previously, by providing appropriate definitions for such concepts along with descriptions of specific concepts in the ontology, we can achieve the effect of classifying all removable concepts below `removableCSSComponent` in the hierarchy. Classification functionality is provided by an Ontology Service (component 5). When an XHTML document arrives in the Mozilla** browser with the LLIS sidebar (component 4), the document's object model is parsed to see if there exists a link element looking similar to (and explained later):

```
<link rel="ontology" rev="bo" type="text/owl"
href="http://www.blogger.com/ontology/blogger.owl"/>
```

The LLIS application then retrieves the ontology, much as Mozilla retrieves the CSS document. The ontology is passed to the Ontology Service, and LLIS can now ask questions based on the actions required. These questions are preconfigured and anchored to the buttons on the LLIS sidebar (see *Figure 2*). Once a button is selected, a message is sent to the Ontology Service asking, for example, that all of the items suitable for removal be returned to the LLIS. The Service complies, and the LLIS parses the Document Object Model (DOM), looking for removable components and discarding them. In this way all pages created in `blogger.com` (close to a million entries) can be modified by using this one simple ontology and tool. Although the ontology is specific to the site, the triage can also work for any number of other sites which have had a specific triage ontology created for them—as long as they

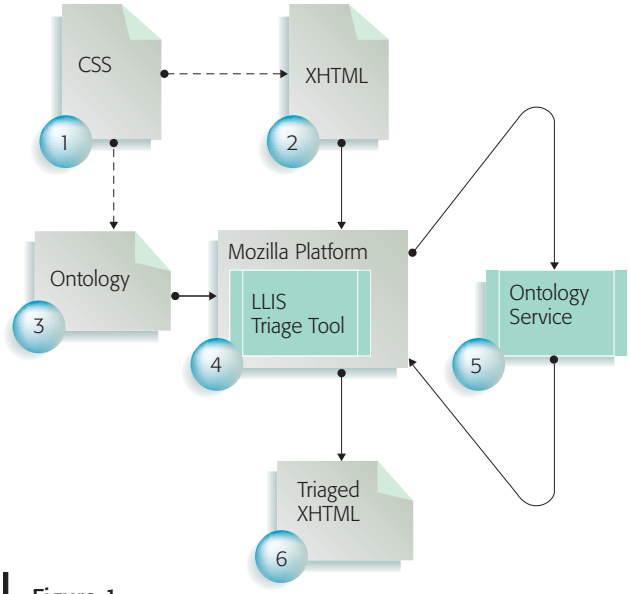

Figure 1
LLIS system overview

call the base ontology, which includes the removable container classes and the `isRemovable` concept. Although we have used `blogger.com` as an example for legacy sites, we are by no means limited to this kind of use.

Our system is in reality a process for associating ontology concepts with instances encoded within XHTML pages. We suggest that meaning should be encoded within the elements of XHTML and CSS by using ontologies that can be created in the normal

Figure 2
De-Fluffing the HiTokyo Web Log (see http://hitokyo.blogspot.com/ for the original site)

way. Ontological concepts and properties are encoded into both the elements and attributes of the XHTML document, and are used as identifiers within CSS that link presentation to XHTML elements. Our system revolves around a software process (described later) that converts an XHTML document into a series of instances and ontological concepts. Users view the document in a Web browser as they normally would. However, browsers that are aware of the semantics can use the ontological information to provide more intelligent access to the instances of information than was previously possible.

Because our system works with CSS, our method is unsuitable for legacy sites that do not use CSS to separate presentation from structure. However, it is suitable for all sites, old or new, which do separate style and presentation in this manner. Our system is therefore compatible with the earlier versions of the vast majority of modern (maintained) Web sites, and the number of sites that can be transformed in this way is increasing daily. Because we do not annotate or modify the actual XHTML document, our system does not force developers into costly and time-consuming reengineering to achieve compatibility with earlier versions.

Implementing semantic concepts

We are suggesting a simple and flexible system without significant semantic overhead. To achieve this, we use the following group of techniques to encode semantics directly into a page:

- *Class and ID attributes*—XHTML `class` or `ID` attributes are used to encode a piece of semantic information, in the form of a concept class or property, into a defined piece of XHTML delimited by the closing element identifier. This is normally achieved by using the `div` and `span` elements to unite the presentation style (CSS) and the semantic meaning (ontology) for the user.

- *Nonpresentational XHTML attributes*—We can leverage the implicit information contained in the names of XHTML elements if we have a corresponding ontology. Elements that are nonpresentational (for example, <address>) can be used to encapsulate meaning within the page.

- *Individuals*—Unique individuals are defined by use of the anchor element, wherein the `href` attribute is used to signify a unique item.

- *Namespaces*—We include namespaces in XHTML documents so that multiple ontologies can be used to describe one document. To implement this we use the `link` element of the XHTML header section. We use the `rel` attribute to signify the ontology, the `type` attribute to signify the type, the `rev` attribute to signify the namespace, and the `href` attribute to specify the URI (Uniform Resource Identifier) of the ontology. However, to circumvent the restrictions of annotation and modification this requirement may place on the developer, we also search for ontologies present in the root directory of the Web site (that is, we look for files with the ".owl" extension). Secondly, we allow the ontology to be directly encoded within the page. In fact, we follow the cues laid down by the originators of the CSS concept and link XHTML to ontologies along these lines.

The suggested approach provides a mechanism for encoding "lightweight" information. Of course this approach has its limitations. For example, we can capture simple instantiation of atomic classes along with property assertions, but not richer assertions such as instantiation of arbitrary class expressions. We stress that our approach is not intended as a replacement for other representations, but is rather a complementary mechanism. For example, we can still expect the class and property definitions in the ontology to be encoded using existing approaches, such as RDF/XML.

Using instances to increase accessibility

To test our system we extended our Triage Ontology with Web logging terms, and from these created a specific ontology for `blogger.com` (see ***Figure 3***). The ontology was created in OWL using the Protégé tool and comprises a small set of concepts and subconcepts derived from the `blogger.com` CSS template. Some of these concepts were further annotated with a Boolean property called `isRemovable` (as previously mentioned) and an integer property called `howImportant`. The ontology is better visualized in Figure 3, which shows the `blogger.com` hierarchy.

The hierarchical (subclass) relationships for the class `removableCSSComponents` are inferred (using the "racer" reasoner[36]) and show that `deleted-comment`, `description`, `footer`, `profilecontainer`, and `sidebar` can all be removed. Figure 3 also shows that although `profilecontainer` subconcepts

(like `profile-data`) were not marked as removable, this relationship has been inferred from the ontology.

Our LLIS Triage Tool, a Mozilla-based application (see Figure 2), has three types of functionality: *De-Fluff*, *ReOrder*, and *Toggle Menu*. De-Fluffing removes all information that is classified as removable based on its location in the ontology (not in CSS or XHTML). ReOrder rearranges the page so that the most important pieces of information are moved to the top of the document based on the ontology's `howImportant` restriction. Finally, Toggle Menu moves menu items from their current location to the top of the DOM (as a child of the DOM body). Interestingly, our ontology contains two concepts, `recently` and `archive-list`, which have no CSS entries but are used as CSS class identifiers in `blogger.com`. These two concepts encompass the list of recent posts and the monthly archive lists and can be used as menus for previous postings. Axioms asserting that the concepts `recently` and `archive-list` are subclasses of `menu` are thus added to the ontology. As discussed later, our application can then treat `recently` and `archive-list` as kinds of menus and perform appropriate operations upon them.

The LLIS application for triage

The LLIS application is a Mozilla sidebar which parses the XHTML DOM and finds instance, concept, and property terms referred to in the associated ontology that are also present within the DOM and CSS (see "ID/CLASS Results" in Figure 2). It does this by connecting[37] to an Ontology Service by means of the DIG (Description Logic Implementation Group)[38] interface. The ontology URL (Uniform Resource Locator) is supplied to the Ontology Service, and the LLIS application formulates and sends questions to this Service. These questions are based upon the specific task that the LLIS has been asked to perform. For instance, if we wanted to remove all the concepts (and therefore CSS blocks) with the restriction `isRemovable=true` in the ontology, we query the Ontology Service asking for all removable components. These components are variable and are based on the ontology for the specific document. Both the returned list and the DOM are parsed, and similarities are derived by matching `class` or `ID` attributes containing an ontological term in the XHTML and the Ontology Service list. If a match is found, that CSS block can be removed. A particularly useful aspect of this process is that the concept within the ontology does not have to be marked as removable because this can be inferred by the reasoner from the ontological structure of the OWL DL (Web Ontology Language Description Logic).

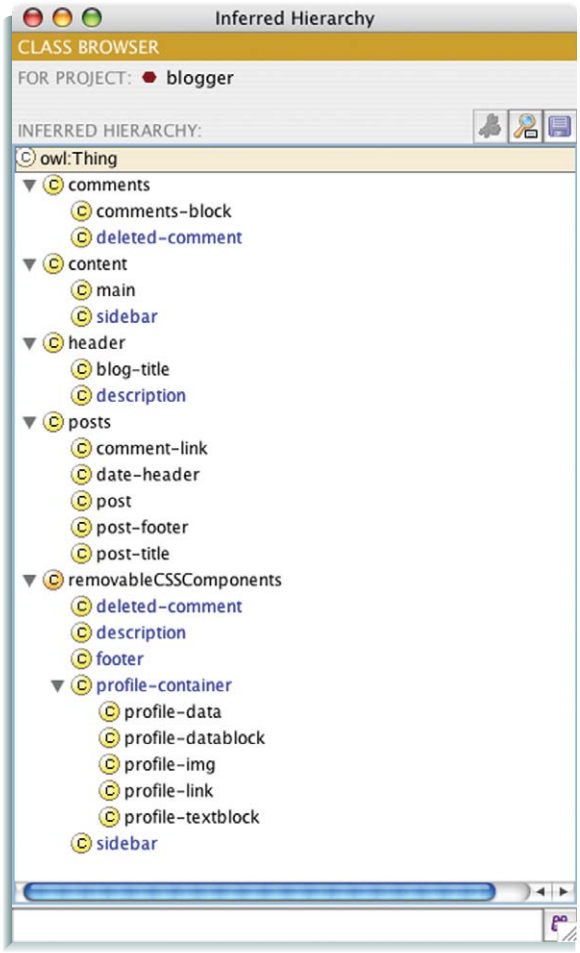

Figure 3
Inferred blogger.com ontology

This is also the case when the `is-a-kind-of` relationship is used with regard to the `recently` and `archive-list` concepts. LLIS knows how to process `menu` concepts (from the Triage Ontology); therefore, when it passes `archive-list` to the Ontology Service and asks whether it is a `menu`, the reasoner will respond that it is. We can therefore choose to move it either to the top of the document or back to its original position. The reordering of the DOM does not, in general, change the visual appearance of the presentation because CSS takes care of the page layout. The reordering does, however, move the information in the XHTML document, and changes

Figure 4
Example sidebar code from blogger.com

are noticeable if style information is removed. This is exactly the outcome for which we had hoped because accessibility technologies access the XHTML DOM as presented and often exclude the style and placement information.

Building a Web site suitable for triage

We now examine in more detail what is required to build a Web site suitable for triage. XHTML and CSS are built as part of the standard site creation. The only part that needs to be added for a new site is the `link` element in the XHTML head element pointing to the ontology that will be created to assist in the triage task.

The method for creating the site-specific ontology is the same, regardless of whether this is a legacy site or a newly created site. ***Figure 4*** shows an XHTML code fragment from a randomly chosen `blogger.com` entry. Notice the block structure of enclosing XHTML `div` elements and their associated `class` and `ID` attributes. These attributes can also be seen in the ontology (Figure 3), where they follow a similar block structure. It seems evident, then, that the ontology can be created from an analysis of the `class` and `ID` attributes found in the XHTML file (see Figure 4) that are also found in the style information of the CSS. Indeed the ontology for `blogger.com` was created by a simple analysis of the CSS style elements (see ***Figure 5***). Again, neither the original XHTML nor CSS documents are modified or destroyed in this process.

Once the user asks the triage tool to, for example, De-Fluff the XHTML document, a query is sent to the Ontology Service asking it to return all `removableCSSComponents`. These components in turn identify `class` and `ID` specifiers within the XHTML. We then simply remove the CSS blocks to which these specifiers refer.

CONCLUSIONS

Our system suggests a method of encoding light-weight markup into Web pages to afford a low-cost, but very valuable semantic benefit. With the essence of the information design being abstracted from what the graphic or Web designer created, the current system gives a taste of how semantics can be represented simply and effectively within Web pages. Additionally, we have shown how this can be

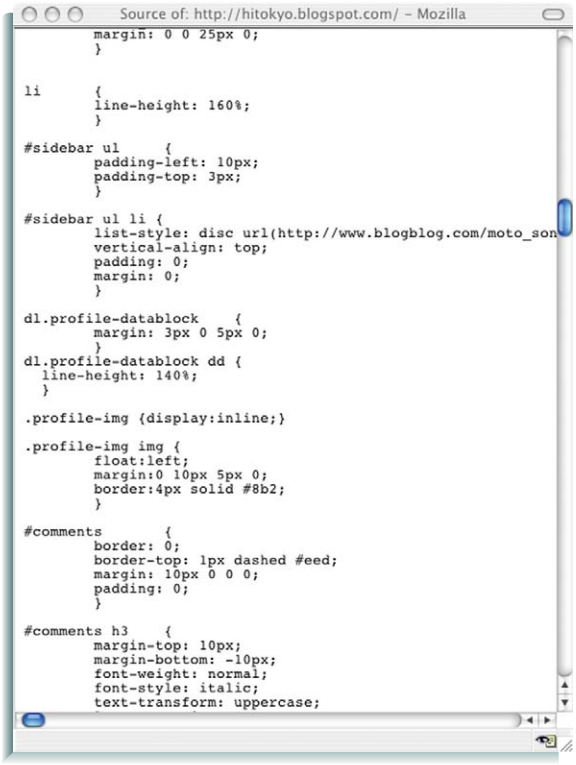

Figure 5
Example style sheet code from blogger.com

achieved without incurring a significant overhead with regard to marking up the semantic information and validating it to strict XHTML 1.0. We propose that the inclusion of semantic information directly in XHTML is the only effective way to assist users who are visually impaired to access Web pages and, at the same time, avoid decreasing or compromising the creative activity of authors and designers. Indeed, this work represents the first stage in a more elaborate system to enable semantic information to be freely accessible to all users. In particular, knowing the meaning of the information that is being encountered will allow visually impaired users to perform their own triage on that information, making the Web significantly more accessible to them.

**Trademark, service mark, or registered trademark of the Massachusetts Institute of Technology or the Mozilla Foundation.

CITED REFERENCES

1. C. Chen, "Structuring and Visualizing the WWW by Generalized Similarity Analysis," *Proceedings of the 8th ACM Conference on Hypertext (Hypertext'97)*, Southampton, UK, April 6–11, 1997, ACM, New York (1997), pp. 177–186.
2. R. Furuta, F. M. Shipman III, C. C. Marshall, D. Brenner, and H.-W. Hsieh, "Hypertext Paths and the World Wide Web: Experiences with Walden's Paths," *Proceedings of the 8th ACM Conference on Hypertext (Hypertext'97)*, Southampton, UK, April 6–11, 1997, ACM, New York (1997), pp. 167–176.
3. M. Brambring, "Mobility and Orientation Processes of the Blind," in *Electronic Spatial Sensing for the Blind: Contributions from Perception, Rehabilitation and Computer Vision, Proceedings of the NATO Advanced Research Workshop on Visual Spatial Prosthesis for the Blind*, Lake Arrowhead, CA, September 10–13, 1984, D. H. Warren and E. R. Strelow, Editors, Dordrecht; Lancaster: Nijhoff Publishers in cooperation with NATO Scientific Affairs Division (1985) pp. 493–508.
4. C. Asakawa and C. Laws, "Home Page Reader: IBM's Talking Web Browser," *Proceedings of the 1998 Closing the Gap Conference*, Minneapolis, MN, October 22–24, 1998. See also IBM Home Page Reader, IBM Corporation, http://www-3.ibm.com/able/solution_offerings/hpr.html.
5. *A Short Guide to Blindness*, Royal National Institute of the Blind, London, UK (February 1996).
6. D. Shea, CSS Zen Garden, http://www.csszengarden.com/.
7. A. G. Dodds, C. I. Howarth, and D. C. Carter, "The Mental Maps of the Blind," *Journal of Visual Impairment and Blindness* **76**, 5–12 (1982).
8. S. Harper, Y. Yesilada, and C. Goble, Editors, *Proceedings of the International Cross-Disciplinary Workshop on Web Accessibility (W4A 2004)*, New York, May 18, 2004, ACM, New York (2004).
9. T. Berners-Lee, *Weaving the Web*, Orion Business Books, London, UK (1999).
10. T. R. Gruber, "Towards Principles for the Design of Ontologies Used for Knowledge Sharing," *Proceedings of the International Workshop on Formal Ontology*, Padova, Italy, March, 1993. A substantially revised version is available at http://www.ececs.uc.edu/~mazlack/CS690.f2004/Semantic.Web.Ontology.Papers/Gruber.93b.pdf.
11. T. Berners-Lee, Semantic Web Architecture, World Wide Web Consortium (December 6, 2000), http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html.
12. *Resource Description Framework (RDF)*, World Wide Web Consortium, http://www.w3.org/RDF/.
13. D. Brickley and R. V. Guha, *RDF Vocabulary Description Language 1.0: RDF Schema, W3C Recommendation*, World Wide Web Consortium (February 10, 2004), http://www.w3.org/TR/rdf-schema/.
14. M. Dean and G. Schreiber, *OWL Web Ontology Language Reference, W3C Recommendation*, World Wide Web Consortium (February 10, 2004), http://www.w3.org/TR/owl-ref/.
15. I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen, "From SHIQ and RDF to OWL: The Making of a Web Ontology Language," *Journal of Web Semantics* **1**, No. 1, 7–26 (2003).
16. S. Bechhofer, I. Horrocks, and D. Turi, *Implementing the Instance Store*, Preprint CSPP-29, Department of Computer Science, University of Manchester (August 2004).
17. I. Horrocks, L. Li, D. Turi, and S. Bechhofer, "The Instance Store: DL Reasoning with Large Numbers of Individuals," *Proceedings of the 2004 Description Logics Workshop (DL 2004)*, Whistler, BC, Canada, June 6–8, 2004, pp. 31–40,

http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS//Vol-104/04Horrocks-final.pdf.

18. T. Berners-Lee, *RDF in HTML*, World Wide Web Consortium (2002), http://www.w3.org/2002/04/htmlrdf.
19. N. Kew, *Why Validate?* World Wide Web Consortium (September 24, 2001), http://lists.w3.org/Archives/Public/www-validator/2001Sep/0126.html.
20. S. B. Palmer, *RDF in HTML: Approaches* (June 2, 2002), http://infomesh.net/2002/rdfinhtml/.
21. D. Connolly, *HyperRDF: Using XHTML Authoring Tools with XSLT to Produce RDF Schemas*, World Wide Web Consortium (August 13, 2000), http://www.w3.org/2000/07/hs78/.
22. M. Altheim and S. B. Palmer, *Augmented Metadata in XHTML*, Sun Microsystems, Inc. (2002), http://infomesh.net/2002/augmeta/.
23. D. Hazaël-Massieux and D. Connolly, *Gleaning Resource Descriptions from Dialects of Languages (GRDDL)*, World Wide Web Consortium (December 7, 2004), http://www.w3.org/2004/01/rdxh/spec.
24. S. Harper, Y. Yesilada, and C. Goble, "Workshop Report: W4A—International Cross-Disciplinary Workshop on Web Accessibility 2004," *SIGCAPH Computers and the Physically Handicapped* **76**, 2–20 (2004).
25. M. Hori, G. Kondoh, K. Ono, S.-I. Hirose, and S. Singhal, "Annotation-Based Web Content Transcoding," *Proceedings of 9th International World Wide Web Conference (WWW9)*, Amsterdam, The Netherlands, May 15–19, 2000, http://www9.org/w9cdrom/169/169.html.
26. W. Myers, BETSIE (BBC Education Text to Speech Internet Enhancer) Home Page, British Broadcasting Corporation (BBC) Education, http://www.bbc.co.uk/education/betsie/.
27. {textualise;}, Codix.net, Ltd., http://codix.net/solutions/products/textualise/index.html.
28. O. Buyukkokten, H. G. Molina, A. Paepcke, and T. Winograd, "Power Browser: Efficient Web Browsing for PDAs," *Proceedings of the 2000 SIGCHI Conference on Human Factors in Computing Systems (CHI 2000)*, The Hague, The Netherlands, April 1–6, 2000, ACM, New York (2000), pp. 430–437.
29. WebCleaner—A Filtering HTTP Proxy, SourceForge®, http://webcleaner.sourceforge.net.
30. H. Takagi and C. Asakawa, "Transcoding Proxy for Nonvisual Web Access," *Proceedings of the Fourth International ACM SIGCAPH Conference on Assistive Technologies (ASSETS 2000)*, Arlington, VA, November 13–15, 2000, ACM, New York (2000), pp. 164–171.
31. C. Asakawa and H. Takagi, "Annotation-Based Transcoding for Nonvisual Web Access," *Proceedings of the Fourth International ACM SIGCAPH Conference on Assistive Technologies (ASSETS 2000)*, Arlington, VA, November 13–15, 2000, ACM, New York (2000), pp. 172–179.
32. M. Hori, G. Kondoh, K. Ono, S.-I. Hirose, and S. Singhal, "Annotation-Based Web Content Transcoding," *Proceedings of the 9th International World Wide Web Conference on Computer Networks: The International Journal of Computer and Telecommunications Networking*, Amsterdam, The Netherlands, May 15–19, 2000, North-Holland Publishing Co., Amsterdam, The Netherlands (2000), pp. 197–211.
33. L. Seeman, "The Semantic Web, Web Accessibility, and Device Independence," *Proceedings of the International Cross-Disciplinary Workshop on Web Accessibility (W4A 2004)*, New York, May 18, 2004, ACM, New York (2004), pp. 67–73.
34. Y. Yesilada, S. Harper, C. Goble, and R. Stevens, "Ontology Based Semantic Annotation for Enhancing Mobility Support for Visually Impaired Web Users," *Proceedings of the K-CAP 2003 Workshop on Knowledge Markup and Semantic Annotation (Semannot 2003)*, Sannibel, FL, October 26, 2003, http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-101/Yeliz_Yesilada-et-al.pdf.
35. A. W. Huang and N. Sundaresan, "A Semantic Transcoding System to Adapt Web Services for Users with Disabilities," *Proceedings of the Fourth International ACM SIGCAPH Conference on Assistive Technologies (ASSETS 2000)*, Arlington, VA, November 13–15, 2000, ACM, New York (2000), pp. 156–163.
36. I. Horrocks and P. F. Patel-Schneider, "Three Theses of Representation in the Semantic Web," *Proceedings of the Twelfth International World Wide Web Conference (WWW'03)*, Budapest, Hungary, May 20–24, 2003, ACM, New York (2003), pp. 39–47.
37. L. Carr, S. Kampa, W. Hall, S. Bechhofer, and C. Goble, "Ontologies and Hypertext," in *Handbook on Ontologies*, S. Staab and R. Studer, Editors, Springer, New York (2004), pp. 517–532.
38. S. Bechhofer, *The DIG Description Logic Interface: DIG/1.1*, University of Manchester (February 7, 2003), http://dl-Web.man.ac.uk/dig/2003/02/interface.pdf.

Accepted for publication January 14, 2005.
Published online August 5, 2005.

Simon Harper
School of Computer Science, University of Manchester, Manchester, M13 9PL, UK (sharper@cs.man.ac.uk). Dr. Harper is a Research Fellow in the Information Management Group of the Department of Computer Science at the University of Manchester where he is working on integrated hypermedia and database solutions in accessibility. He comes from an industrial background as a software consultant with a major energy provider. His research interests are a synthesis of hypermedia, cognition, and rehabilitation engineering. He also works on the *Journal of Web Semantics* and is a Web developer for the ACM SigWeb.

Sean Bechhofer
School of Computer Science, University of Manchester, Manchester, M13 9PL, UK (seanb@cs.man.ac.uk). Mr. Bechhofer is a Lecturer in the Information Management Group of the School of Computer Science at the University of Manchester. Over the last 10 years, he has worked on tools and infrastructure to support the use of Semantic Web technology, publishing numerous articles in journals, conferences, and books. He was a key participant in the COHSE (Conceptual Open Hypermedia Service) project, which combined open hypermedia with ontological services to produce an early example of a Semantic Web application. He was also mainly responsible for developing OilEd, one of the first ontology development tools to use reasoning. He was a member of the W3C WebOnt Ontology Language Working Group, providing key experience that led to the acceptance of OWL as a W3C recommendation and implementing one of the first validating parsers for OWL, along with supporting infrastructure for the use of OWL in applications. ■