Intensional HTML

Bill Wadge, Gord Brown, m. c. schraefel, and Taner Yildirim

Department of Computer Science, University of Victoria
Victoria, B.C., V8W 3P6, CANADA
`wwadge@csr.uvic.ca, gdbrown@csr.uvic.ca,`
`mc@csr.uvic.ca, taner_yildirim@pml.com`

Abstract. Intensional HTML is a high-level Web authoring language that makes practical (using standard client and server software) the specification of pages and sites that exist in many different versions or variants.
Each page of IHTML defines an intension — an indexed family of actual (extensional) HTML pages which varies over a multi-dimensional author-specified version space. The version space is partially ordered by a refinement/specialization ordering. For example, `platform:mac` can be refined to `platform:mac+language:french` or to `platform:mac%k68` and the last two both refine to `platform:mac%k68+language:french`.
Authors can create multiple labeled versions of the IHTML source for a given page. Requests from clients specify both a page and a version, and the server software selects the appropriate source page and uses it to generate the requested actual HTML page.
Authors do not, however, have to provide separate source for each version. If the server-side software cannot find a source page with the exact version requested, it uses the page whose label most closely approximates the requested version. In other words, it treats the refinement ordering as a (reverse) inheritance ordering. Thus different versions can share source, and authors can write generic, multi-version code.

1 The Versioning Phenomenon

Many documents created for publication are produced in different variants or versions, corresponding, say, to different languages, different levels of expertise, different dates or different target audiences. In fact, most artifacts produced by humankind (documents or otherwise) appear in families of related versions, and the diversity in a family of documents (for example, user manuals) often simply reflects the corresponding diversity in a related family of more concrete entities.

The advent of the World Wide Web has, if anything, increased the pressure on authors to create multi-version documents, for a number of reasons. Firstly, the Web is international, and a truly international site must be available in many different languages. The bandwidth available to users varies greatly, so that some appreciate high quality graphics while others prefer purely text pages. Different browsers have different capabilities, for example, in terms of tables and frames. Some sites offer more material to paying subscribers while others may want certain information hidden from outsiders. Some sites would naturally offer different information (e.g. weather reports) to people in different parts of the world. Finally, site designers might want to offer sites that are customizable to take personal preferences (fonts, background colors) into account.

Furthermore, it seems at first sight that it should be easier to support on the web than in print mediums. Web documents are fetched on demand, in response to requests from individual users who could, if necessary, provide at the same time relevant personal information (such as language or level of expertise). In principle, server software could take these parameters as input and generate a made-to-measure version of the requested document.

In practice, however, there are very few sites which allow themselves to be reconfigured according to user preferences. In particular, there are very few truly multilingual sites. Even large international corporations, which typically provide multilingual versions of their front pages, soon revert to the "default" (i.e. English) version for their inner pages.

The problem lies with the very nature of HTML. HTML certainly allows pages to be generated on demand (through the CGI protocol) but provides no real support for authors who use it. CGI is an *escape* from HTML. Instead, authors usually stay with HTML and produce multi-version sites by cloning source files. Cloning works well in the short run, for a small number of versions, but breaks down when the version space is large and when the family of sites has to continue evolving.

In this paper we propose an alternate solution: we extend HTML to a (slightly) higher level language which allows users to specify families of sites without the user cloning or, in the current implementation, escaping through the CGI gate.

1.1 The Problem with Cloning

The easiest way to produce a variant of a web site is to make a copy of the HTML source and modify it. Unfortunately, the cloning (copying/modifying) approach to version creation can, in the long run, produce severe difficulties maintaining the resulting families of versions. The problems arise when changes are required in parts of the original that were copied unchanged into the new versions. The same changes have to be made many times, in the sources for all the versions which used the original code.

The inevitable result of the copy-and-modify approach is a large family of clones which, as a group, is almost impossible to change in any uniform way.

The obvious solution is ensure that the different members of the family share the *same* code (and not copies thereof) so that necessary changes are made only once and need not be propagated. To use Ted Nelson's terminology, we must arrange that different versions *transclude* the source they have in common.

This is easier said than done. In fact, exactly the same issues arise in the production of software (the documents being programs), and software version management is one of the most difficult problems in software engineering. Indeed, the success of the object oriented approach is due in part to class inheritance, which allows code to be shared (reused) on a large scale.

HTML certainly allows two different pages to link to a third, but this is a very crude form of sharing. Links are essentially pointers, and the problem with sharing via pointers is that you also share everything the shared object itself has pointers to.

Consider the problem of supporting English and French versions of a simple slide show. The slide show consists of a sequence of pages of text and/or graphics, each linked to the next page in the sequence. Obviously, we have to create separate English and French versions of pages with text on them. But we also have to make clones of any pages that have only images, even though the French and English versions will appear identical on the screen. The problem is that the English version of the page in question must be linked to the rest of the English version of the show, while the French version of the slide must be linked to the rest of the French version. Two separate source files are required.

2 The IHTML Solution

In this paper we present IHTML (Intensional HTML), an HTML-based authoring language which incorporates an object-oriented (inheritance based) approach to hypertext versioning. IHTML allows authors to define, with a single source file, a whole indexed family of HTML variants based on the file in question. These variants are generated on demand, then discarded after use. In a sense, IHTML automates the cloning process, and eliminates the maintenance problem by ensuring that the clones are short lived.

IHTML is *intensional* because IHTML source has both intensional and extensional meanings. The intension is the whole indexed family of HTML pages; the extensions are the different individual HTML pages.

The main feature of IHTML is that authors can provide multiple sources for the same page, each source labeled with a different version. The IHTML server-side software accepts requests for particular versions of particular pages, and generates the actual HTML from the appropriate IHTML source files.

For example, in the case of the slide show, the author could name the pages `slide1`, `slide2`, `slide3`, etc., and provide for each of these pages two source files, one English and one French.

IHTML authors do not, however, have to provide separate source files for every possible version. The IHTML index space ("version space") is partially ordered by a refinement relation, and the source for a more refined version is by default inherited from the less refined (more generic) versions.

When the server-side software receives a request for a particular version of a particular page (or part thereof), it looks for a source file labeled with the requested version. If there is no such source, it looks for a source file whose label most closely approximates the requested version (if there is no such source file, or no best source file, it reports an error). More refined versions can therefore by default transclude source from more generic ones, and a relatively small number of source files can define a very large family of pages.

For example, suppose that the fifth slide is purely graphical. The author can provide a *single* source file, labeled as the standard (so-called "vanilla") version. When a request comes for, say, the French version of page `slide5`, the server software first looks for a source file for that page labeled as the French version. When it finds none, it uses the more general standard version. Requests for the English version are similarly referred to the single standard source.

2.1 IHTML Links and Includes

Normally, links in IHTML source look exactly like links in ordinary HTML. They are interpreted, however, as denoting a whole family of links, each connecting a given version of the page they appear in to the *corresponding* version of the page they link to.

For example, suppose that the generic IHTML source for `slide5` contains the link `<a href=page6>next page</a>`. This is interpreted as meaning that the English version of `page5` is linked to the English version of `page6`, and that the French version of `page5` is linked to the French version of `page6`. When the server software generates the French version of `page5` from the generic source, it makes the generic link into a link to the French version of `page6` (this might be the only change made).

IHTML also has a "server-side include" feature which causes the contents of an included file to be incorporated (by copying) into the HTML page under construction. For example, each page of the slide show might have `<!--#include virtual=header -->` at the top and `<!--#include virtual=footer -->` at the bottom, to include a standard header and footer in all slides. (The syntax was chosen to reflect the server-side include syntax of the Apache WWW server.)

The IHTML includes are also generic; for example, the English version of `page4` will include the English version of the header. When processing an include, the server software looks for the version of the named source file whose version label most closely approximates the "current" version, i.e. the requested version of the file in which the include appears. Note that it looks for the *requested* version of the included file: the including page may exist in only a single, generic version, but a more specific version of the included file will be used if one can be found.

The include facility is very important for IHTML because it allows authors to break the source components into pieces smaller than a whole page. This allows the author to isolate the parts of a page that actually vary, and write more generic source for the parts (such as headers and footers) that do not. Conversely, the author may write generic source for a page as a whole, and include content which varies over whatever dimensions are appropriate.

2.2 The IHTML Version Space

The families of pages specified by IHTML are indexed by (subspaces of) the algebraically defined version space described in [1]. In the terminology of intensional logic [2], the elements of this space are *possible worlds*; each individual possible world (version) determines a particular extension, i.e. an actual HTML page.

The elements of the space described in [1] are expressions built up from identifiers using the operations + and %.

The % operator is the subversion operator: $V\%s$ is (by definition) a refinement of V. For example, `Mac%k68` is a subversion of `Mac`.

The + operator is the version join operator: the least upper bound in the refinement ordering. Intuitively, version $V+W$ is the most general version which incorporates the modifications/refinements of *both* versions V and W. For example, the `Mac%k68+french` version might be the version which is designed for 68K Macs *and* uses French as its interface language.

Elements (versions) are partially ordered by a refinement operator: $\subseteq$. This operator can be read as "is refined by", or "is more general than". For example, $V \subseteq W$ says that W refines V, or that V is more general (closer to "vanilla") than W.

These ideas are formalized in the axioms presented in [1], for example:

$$V \subseteq V\%W \tag{1}$$

or

$$V\%(W_1 + W_2) = V\%W_1 + V\%W_2 \ . \tag{2}$$

The elements of this version space are equivalence classes of expressions together with the coarsest order which satisfies the axioms. This space is similar to Prolog's set of Herbrand terms: a convenient collection of abstract symbolic objects to which we can attach meanings.

The IHTML version space extends that of [1] in one important way: it allows explicit dimensions [3].

For example, we interpreted the term `french` in the above expression as referring to the French language. What if we were producing information about cooking and also needed to specify the cuisine? In the IHTML space, we can use arbitrary identifiers as dimensional "multipliers" and form sums that specify coordinates for each of the given dimensions. This enlarged space includes expressions such as

```
platform:Mac%K68 + lang:french + cuisine:chinese .
```

The extra rules are:

$$D\!:\!\varepsilon \equiv \varepsilon \ , \tag{3}$$

$$D\!:\!(V + V') \equiv D\!:\!V + D\!:\!V' \ , \tag{4}$$

$$D\!:\!V \subseteq E\!:\!V' \leftrightarrow (D \equiv E) \text{ and } (V \subseteq V') \ . \tag{5}$$

(Here ε is the most general version: the standard or "vanilla" version.)

It should be clear now how to compare two dimension sums. In general:

$$D_0\!:\!V_0 + D_1\!:\!V_1 + \ldots \subseteq E_0\!:\!W_0 + E_1\!:\!W_1 + \ldots \tag{6}$$

if and only if for each D_i, either $V_i = \varepsilon$ or D_i is equal to E_j for some j and in that case $V_i \subseteq W_j$. (We assume that $D_i \neq D_j$ for $i \neq j$, and likewise for all E_i and E_j. In other words, there are no duplicates among the dimensions in a sum.)

2.3 Transversion Links

The second distinguishing feature of IHTML is the ability to define what we call *transversion* links. These are links that lead from the current version of the source page to a *different* version of the target page — different in a way specified in the tag. Transversion links allow visitors to the site to move from one version to another, without necessarily filling in forms or composing complex URLs. At the same time, they give the author full control over the way in which different versions of the site are interconnected.

A transversion link has the same format as an ordinary link, except that the tag may contain assignments to dimension identifiers. The link is interpreted as leading from a given version of the source page to the modified version of the target page — the modifications resulting from altering the coordinates of the given dimensions as specified.

For example, the author of the slide show might include, in the English versions of the source of the title page, a link of the form

```
<a href=page1 vmod="language:french"> .
```

In the (say) `language:english + background:blue` version of the title page, this will be interpreted as a link to the `language:french + background:blue` version of the first page. The "vmod" attribute defines a "version modifier" which is applied to the version of the current page.

Notice that following this link will take the reader to the French version of the whole slide show. The reason is that the French version of page 1 is linked to the French version of page 2, and so on. The English and French versions coexist as sort of parallel universes, and the transversion links let the reader move from one of these universes to the other.

IHTML also allows transversion includes, with a similar syntax. For example, `<!--#include virtual=footer.html vmod="language:english" -->` will include the footer in a version like the current version except that the `language` component is `english`.

IHTML also allows links to conventional (unversioned) HTML webware. We call these *extensional* (or *external*) links. One could think of an extensional link as a transversion link in which all the coordinates are set to ε. Accordingly, the syntax for an extensional link is `<a href=... version="">`. IHTML also has an extensional include tag.

2.4 IHTML with Existing Browsers

It might seem from what has been said that IHTML requires its own version of the server and client software. The requests from the client consist of a URL *and* a version. Satisfying such a request involves searching for the appropriate IHTML source file and then transforming the generic source into the particular HTML file corresponding to the version in question.

In fact, a prototype implementation of IHTML has been developed that uses existing browsers and existing server technology, by the author Yildirim (as part of his Master's thesis [4], completed June 1997). The basic idea is to ensure that all links to an IHTML-specified page go through a CGI script. The call to the script has two arguments, the generic URL and (a representation of) the particular version requested. The CGI script invokes the server-side software which locates the appropriate source file and produces the HTML.

The server software itself ensures that all links are CGI calls. When the software generates HTML from IHTML, it transforms the normal-looking generic IHTML links into CGI calls with the appropriate parameters. The first parameter, the URL, is taken directly from the IHTML source of the link. The second parameter, a version, is normally the same as the version included in the client request. In the case of a transversion link, however, the software modifies this version according to the information in the IHTML source of the link.

The CGI-based implementation proves that the basic concept works well, but it has some limitations. Performance is sometimes a problem, since there is non-trivial overhead in running a large CGI script for every HTTP request. In addition, URLs look strange to the user of the browser: everything starts with the same path (the path to the CGI script), and there is an unexpected numeric argument at the end of the URL. For these reasons (along with some implementation-level problems), authors Wadge and Brown have produced an entirely new implementation.

2.5 Current Implementation

The new version is an enhancement of the Apache WWW server [5]. Implemented as an Apache plug-in module, it traps HTTP requests for versioned entities (HTML pages, images, and so on), handling them separately, while letting requests for non-versioned things be handled by the server as usual. In this model, the user sees normal URLs, except that the name of a page has a version embedded in it. For example, a (partial) URL for the "`language:turkish`" version of page "`zork.html`" would be "`zork.M1lw9qG3L4Bzjuee.html`", given that "`M1lw9qG3L4Bzjuee`" is the encoding of "`language:turkish`". This version, although implemented as a modified server, still uses normal browser software (an essential feature, if the idea is to have any practical value whatsoever).

The implementation has three basic components: tools to aid the site designer in constructing a versioned site, software in the server to translate the URL in an HTTP request to a particular version of a particular filename, and software to translate IHTML source files to HTML.

Tools. The current tool set is Unix-based, consisting of modified "ls" and "cp" commands, as well as a front end to the "vi" text editor (or some other editor specified by the user). "icp" and "ivi" allow the user to specify the version of the file that is to be copied or edited; "ils" lists the versions in which an IHTML entity exists, in addition to the usual "ls" information. For example, to edit the `language:english+graphics:lowres` version of page `zork.html`, the site designer would use the command "`ivi -v language:english+graphics:lowres zork.html`". To list the available versions of the file, "`ils zork.html`". To copy an unversioned file `alpha.html` to the `colour:blue` version of `zork.html`, "`icp -v colour:blue alpha.html zork.html`".

URL to Filename Translation. Two steps in this process are different from the usual translation process. First, the software must decide whether the URL refers to a versioned object at all; if it doesn't, the IHTML component declines to handle the request, and processing continues normally. If it does, then the second step is to find the most relevant version of the requested file. If the exact version which was requested exists, it is chosen. If not, the software chooses the most relevant version (the maximum element of the set of less specific versions), assuming one exists. If none exists (the set has no unique maximum element, or there are no less specific versions), the usual HTTP "404" error is returned, indicating that the URL doesn't exist.

IHTML to HTML Translation. IHTML files may contain tags and/or tag attributes that must be transformed to standard HTML before files are sent to the browser. There are two basic transformations. First, a normal HTML tag, such as `img`, might have an attribute modified to include a version code: "`src=pic.gif`" becomes "`src="pic.zj94kaz9zll-_a.gif"`", supposing that "`zj94kaz9zll-_a`" is the representation of the version of interest. The modification will consider the current version of the page, as well as any version-modifying attributes present in the tag. Any tag attribute which refers to a file or URL is a candidate for modification. Second, any IHTML-specific tag will be replaced by whatever it indicates. For example, an `include` tag will be replaced by the appropriate version of the file named in its `virtual` attribute. With one exception, to be described shortly, text outside of tags is echoed verbatim to the browser.

Additional IHTML Features

Executable Includes. As with standard Apache server-parsed documents, IHTML files may include the output from the execution of arbitrary Unix programs (scripts or otherwise). The difference, naturally, is that IHTML executables may exist in multiple versions, with the appropriate version being chosen in the usual IHTML manner.

Explicit Versions. As well as links, includes, and so on which modify the current version of the page, it is also possible to specify the exact version of a link, include, or whatever. This feature is particularly useful when including links to other versioned sites, with version spaces different from that of the current site.

Structured Documents. HTML documents are already structured by their HTML tags, of course. However, IHTML allows a higher level of structure which describes which parts of a document to include in which versions. It behaves rather like a C-language "switch" statement: "If the current version is a refinement of version 'a', include the following IHTML fragment. If it is a refinement of version 'b', include this other fragment", and so on. This feature can save the site designer from creating many small files for minor variations of a file. Rather than, say, including a file which varies in the `background` dimension, to set the background colour for a document which doesn't otherwise vary in this dimension, the designer can simply specify a series of alternatives (based on the value of the `background` dimension) at the top of one file, which set the background colour to the appropriate value.

3 A Sample Intensional Site

Author Yildirim has produced a fairly elaborate multi-version home site (viewable using standard browsers) at URL

```
http://csr.uvic.ca/~taner/cgi-bin/scan.cgi .
```

(This site uses Yildirim's cgi-based implementation, but will soon be converted to run using the Apache-based implementation.)

At first sight, it looks like a fairly normal home page. One can follow links to related pages with Mr. Yildirim's biographical details, résumé, course work, and favorite bands and beers.

However, at the bottom of each of these pages is a link anchored to the phrase "*Turkish version of this site*". When we click it, the text on the page changes from English to Turkish.

The words are well chosen: if we proceed to explore the site again, we find the résumé, the course work, the bands and so on, but *all* these pages are in Turkish. We are now in the Turkish version of the whole *site*, not just of the home page. At the bottom there is an anchored phrase containing the words "*ingilizce versiyonu*" and, not surprisingly, it leads us back to the English version of the site.

Each page also offers us a similar transversion link to the text-only version (actually, versions) of the site.

Finally, on the home page there is a link anchored (in the English versions) to the phrase "*Background Options*". It takes us to a page with several small anchored images — background colors or patterns. Following, say, the link anchored to the orange square takes us to the home page of a version of the site in which *all* the pages have an orange background.

Note that we can view the different versions of the site with a standard browser, just by following links, without filling in forms or otherwise composing complex URLs. In fact, we do not need to know *anything* about the version algebra, including the fact that it exists at all.

The site described above uses an eight-dimensional version space. The dimensions are `language`, `display`, `background`, `date`, `category`, `order`, `text`, and `link`.

The `language` and `display` dimensions have two coordinate values each: `english/turkish` and `text/graphics`, respectively. In other words, the coordinate in the `language` dimension has either the value `english` or the value `turkish`, and the coordinate in the `graphics` dimension has either the value `text` or the value `graphics`.

The `background` dimension has thirty possible coordinate values and the `text` and `link` dimensions have twenty seven possible values.

Four distinct dates are used as the update dates.

The `category` dimension has six coordinate values: `humor`, `sports`, `music`, `movie`, `software`, `weather`.

The `order` dimension has two coordinate values: `ascending` and `descending`.

The site consists of thirteen intensional pages, each available in all versions — a total of more than 4.5 million virtual HTML pages (not every page varies in every version). The original one-version HTML program for the site consisted of about 38K bytes divided into 13 different files. The IHTML source consists of 192 files, but most of them, 94, are tiny files — consisting typically of a single line (such as `background=image/sky.gif` or `text="#FF00FF" link="#0000FF"`).

There is a total of 107K bytes of IHTML source — and most of the extra is Turkish versions of the original English text. These 107K bytes of IHTML source supports a virtual site which would correspond to over 13G bytes of cloned HTML.

4 Future Directions

The intensional approach to variation described here can be applied to any indexable family of pages, whether or not one might currently consider them to be "versions" of a single page.

For example, the slides in the slide show are obviously indexed by the natural numbers. We can therefore consider them to be variants of "the" slide, and add a `page_number` dimension to our version space. This would allow us to have a single generic IHTML source for all the slides, which would have headers, footers, logos, color choices etc. The generic slide page would include a `body` file, which itself would vary over the `page_number` dimension.

If we have a number of different slide presentations, we could in turn consider them to be versions of "the" presentation, and write an even more generic page for all our presentations.

At a university site, the pages for different departments could be produced as versions of a single generic department page that varies over the `department` dimension. Similarly, the different faculty pages could be treated as a family varying over a `professor` dimension.

A page which changes every day, e.g. that of a newspaper, can clearly be indexed by the set of dates. If we add a `date` dimension, we can write generic source which specifies parts of the layout (such as mastheads) that are invariant. We can extend our scheme by allowing source pages to be labeled by *intervals*, with the understanding that the source is valid for requests whose date coordinate lies in the interval. This idea is described in more detail in [6], where it is pointed out that it amounts to treating the Web as a kind of reactive system.

5 Aggregating Lists

In [7] author schraefel describes how the IHTML approach could be applied to a non-technical document — specifically, an essay on Wuthering Heights. The plan is to allow the reader to specify a set of parameters which identify the aspects of the essay in which they are particularly interested. They could specify a particular character, or that character's relation to a second character, an issue, degree of depth (from an abstract to a full essay) and a degree of documentation (from none, to complete footnotes and a full bibliography).

It soon became clear that early versions of IHTML would have difficulty with some aspects of this design, in particular the parameters that define intensities, and the specification of lists.

The problem with the original IHTML is that it is based on the standard object-oriented inheritance convention: given a request for a particular version of the page, we search for the most relevant (least generic) applicable source document. In forming a list, however, we want to pull in all (not just the most) relevant items.

As a result, both implementations of IHTML have a feature which allows a list to be formed by taking *all* relevant versions, not just the most relevant. In the latest implementation, this takes the particularly simple form of a variant of the C-like switch statement which concatenates *all* the bodies whose version conditions are relevant.

This simple extension could have many applications — consider how much of the information on the Web is composed of lists of some sort. With the aggregation extension, users can fine-tune the size and criteria for forming a list, and choose various formats for displaying it.

References

1. J. Plaice and W. W. Wadge, "A New Approach to Version Control", *IEEE Transactions on Software Engineering*, March 1993, pp268–276.
2. R. Thomason, editor, *Formal Philosophy, Selected Papers of R. Montague*, Yale University Press, 1974.
3. J. Plaice and S. Ben Lamine, "Eduction: A General Model for Computing", in E. A. Ashcroft, editor, *Intensional Programming II*, Singapore: World Scientific, 1997. In Press.
4. T. Yildirim, *Intensional HTML*, MSc Thesis, Computer Science Department, University of Victoria, 1997.
5. B. Behlendorf et.al, *The Apach HTTP Server Project*, `httpd://www.apache.org/`.

6. W. Wadge and A. Yoder, "The Possible-World Wide Web", in Mehmet A. Orgun, Edward A. Ashcroft, editors, *Intensional Programming I*, pages 207–213. Singapore: World Scientific, 1996. (also available at `http://lucy.uvic.ca/oo.html`).
7. m. c. schraefel, *Talking to Antigone*, PhD Dissertation (interdisciplinary), University of Victoria, 1997.