The Hyperlatex Story

Otfried Schwarzkopf

Dept. of Computer Science, Postech,
San 31, Hyoja-Dong, Pohang 790-784, South Korea

May 26, 1998

Abstract

Hyperlatex is a little package that allows you to use a LaTeX-like language to prepare documents in HTML, and, at the same time, to produce a neatly printed document from your input. It is possible to use arbitrary LaTeX commands for the typesetting of the printed output by escaping them in the input file.

About two years ago my drawing editor Ipe[1] was getting sufficiently complex to merit a real manual instead of a simple "README" file. Of course, Ipe should be able to show its manual on-line, on the other hand I also wanted to be able to print a well-formatted manual on paper. My first attempt at this used the `latexinfo` system to write the manual, so I was able to print it nicely and to have the on-line version as an info file. However, info files are simply text files, and it is impossible to include figures in the on-line manual. Quite a shortcoming when you are trying to write a manual for a figure editor! The second problem was that the first Ipe users were Emacs-illiterate, and they found it very hard to cope with the info reader (that's where the `alt.religion.emacs.haters` pun in the Hyperlatex manual comes from).

At that time HTML and HTML-readers like `mosaic` became widely used. They solved both problems—an HTML document can include figures, and HTML readers are basically designed to be fool proof (how else could one explain the success of the World Wide Web?). So, as the next step, I used the `latex2html` converter. I was now able to write the manual in plain LaTeX (unadorned with the special commands that `latexinfo` required), and `latex2html` would turn it into a set of HTML file.

But I soon found that I had a hard time making `latex2html` generate the kind of HTML that I wanted. This was no flaw with `latex2html`, but with the general approach of converting from LaTeX. In my eyes, conversion is not a solution to HTML authoring. A well written HTML document must differ from a printed copy in a number of rather subtle ways. I doubt that these differences can be recognized mechanically, and I believe that converted LaTeX can never be as readable as a document written in HTML.

This is most prominent in the formulation of cross references in a document. A LaTeX converter can turn the reference into a hyperlink, but it will have to keep the text the same. If we wrote "More details can be found in the classical analysis by Harakiri [8]", then the converter may turn "[8]" into a hyperlink to the bibliography in the HTML document. In handwritten HTML, however, we would probably leave out the "[8]" alltogether, and make the *name* "Harakiri" a hyperlink.

The same holds for references to sections and pages. The Ipe manual says "This parameter can be set in the configuration panel (Section 11.1)". A converted document would have the "11.1" as a hyperlink. Much nicer HTML is to write "This parameter can be set in the configuration panel", with "configuration panel" a hyperlink to the section that describes it. If the printed copy reads "We will study this more closely on page 42," then a converter must turn the "42" into a symbol that is a hyperlink to the text that appears on page 42. What we would really like to write is "We will later study this

[1] Ipe is an attempt to fully integrate LaTeX text with postscript drawing information. Ipe stores files in a format that is at the same time a legal postscript and a legal LaTeX-file, and the drawing editor runs LaTeX in the background to determine the size of text objects. Maybe I'll write a TUGboat article about it sometime.

more closely," with "later" a hyperlink—after all, it makes no sense to even allude to page numbers in an HTML document.

The Ipe manual also says "Such a file is at the same time a legal Encapsulated Postscript file and a legal LaTeX file—see Section 13." In the HTML copy the "Such a file" is a hyperlink to Section 13, and there's no need for the "—see Section 13" anymore.

There are also differences between LaTeX copy and HTML copy that have to do with the fact that HTML is still a somewhat enhanced text format. Many LaTeX concepts are hard to represent in HTML.

For instance, how do you present a mathematical expression like x_i or $a^2 + b^2 = c^2$ in HTML? `latex2html` converts these to little bitmaps. That is quite sophisticated, but is it the best representation? I don't think so. With current technology, bitmaps eat too much transmission time, and they only look good when the resolution of the browser is nearly the same as the resolution at which the bitmap has been created, which is not a realistic assumption.

Isn't there an easier way? If x_i is the ith element of an array, then I would write it as `x[i]` in HTML. If it's a variable in a program, I'd probably write `xi`. In another context, I might want to write *x_i*. To write Pythagoras's theorem, I might simply use `a^2 + b^2 = c^2`, or maybe `a*a + b*b = c*c`. To express "For any $\varepsilon > 0$ there is a $\delta > 0$ such that for $|x - x_0| < \delta$ we have $|f(x) - f(x_0)| < \varepsilon$" in HTML, I would write "For any *eps* > *0* there is a *delta* > *0* such that for |*x-x0*| < *delta* we have |*f(x)-f(x0)*| < *eps*."

Of course a converter could be told to translate ε to *eps*. But the best representation in HTML very often depends on the context, and is beyond the reach of any (non-human) converter.

So I ended up not using `latex2html`. `latex2html` is a good general converter and I had and have no ambition to improve on that. (And before anybody accuses me of being unfair—yes, all the differences described above can be achieved using the `texonly` and `htmlonly` environments of `latex2html`, and its macros for making cross references. But I soon found this too cumbersome.)

Instead, I turned back to the lisp macros from the `latexinfo`-package and changed them to generate HTML output instead of info files. Of course this was intended to be a hack, and never meant for wide use... How could I know that I would end up by having to write a short manual for Hyperlatex itself, and would even be invited to write a TUGboat article about it?

Although I still keep getting Email messages saying "Hey, your Hyperlatex converter is crap. It fell over immediately when I tried to convert this LaTeX file!", Hyperlatex was not intended to be general LaTeX-to-HTML converter—for the reasons explained above.

The idea of Hyperlatex is to make it possible to write a document that will look like a flawless LaTeX document when printed and like a handwritten HTML document when viewed with an HTML browser. In this it completely follows the philosophy of `latexinfo` (and `texinfo`). Like `latexinfo`, it defines it's own input format—the *Hyperlatex markup language*—and provides two converters to turn a document written in Hyperlatex markup into a DVI file or a set of HTML documents. If you have written a document *sample.tex* in Hyperlatex markup,[2] you simply run LaTeX on your file to generate a DVI file, which you can print as usual.

On the other hand, you can type

```
hyperlatex sample.tex
```

to generate a set of HTML files, probably called *sample.html*, *sample_1.html*, *sample_2.html* and so on. (The command `hyperlatex` is a simple shell script that calls GNU Emacs in batch mode and runs the Emacs lisp macros that implement the conversion to HTML. It is also possible to call these macros directly from inside Emacs.)

Obviously, this approach has the disadvantage that you have to learn a "new" language to generate HTML files. However, the mental effort for this is quite limited. The Hyperlatex markup language is simply a well-defined subset of LaTeX that has been extended with commands to create hyperlinks, to control the conversion to HTML, and to add concepts of HTML such as horizontal rules and embedded images. Furthermore, you can use Hyperlatex perfectly well without knowing anything about HTML markup.

[2] Yes, I do use the extension `.tex` for my Hyperlatex files, sharing it with TeX and LaTeX. The Hyperlatex format is much more similar to LaTeX than LaTeX is to TeX, so this seems justified.

The fact that Hyperlatex defines only a restricted subset of LaTeX does not mean that you have to restrict yourself in what you can do in the printed copy. Hyperlatex provides many commands that allow you to include arbitrary LaTeX commands (including commands from any package that you'd like to use) which will be processed to create your printed output, but which will be ignored in the HTML document. However, you do have to specify that *explicitely*. Whenever Hyperlatex encounters a LaTeX command outside its restricted subset, it will complain bitterly.

The rationale behind this is that when you are writing your document, you should keep both the printed document and the HTML output in mind. Whenever you want to use a LaTeX command with no defined HTML equivalent, you are thus forced to specify this equivalent. If, for instance, you have marked a logical separation between paragraphs with LaTeX's `\bigskip` command (a command not in Hyperlatex's restricted set, since there is no HTML equivalent), then Hyperlatex will complain, since very probably you would also want to mark this separation in the HTML output. So you would have to write

```
\texonly{\bigskip}
\htmlrule
```

to imply that the separation will be a `\bigskip` in the printed version and a horizontal rule in the HTML-version. Even better, you could define a command `\separate` in the preamble and give it a different meaning in DVI and HTML output. If you believe that `\bigskip` should always be ignored in the HTML version, then you can state so in the preamble as follows.

```
\H\newcommand{\bigskip}{}
```

This philosophy implies that in general an existing LaTeX-file will not make it through Hyperlatex. In many cases, however, it will suffice to go through the file once, adding the necessary markup that specifies how Hyperlatex should treat the unknown commands.

The `latex2html` converter will convert any environment for which it does not have a built-in translation to HTML to a bitmap. This option exists in Hyperlatex as well, but again you have to explicitely ask for it by enclosing the unknown environment in a `gif` environment.

(If you are interested in how this is done: unlike `latex2html`, Hyperlatex does not create a temporary LaTeX file with the `gif` environments. In fact, the *gif*-making is mostly implemented in TeX! The *hyperlatex.sty* package defines the `gif` environment as follows if the flag for *gif*-making is set.

```
\def\gif{\setbox\@gifbox=\vbox\bgroup}
\def\endgif{\egroup\shipout
  \copy\@gifbox\unvbox\@gifbox}
```

This means that the contents of the `gif` environment is put in a box which is shipped out on a separate page of the DVI file without disturbing LaTeX too much. Later, a shell script extracts the extra pages from the DVI file using `dvips` and turns them into bitmaps using `ghostscript`. The shell script itself is created by the LaTeX run.)

Hyperlatex implements most LaTeX commands that have a clear HTML analog, such as sectioning, font styles and sizes, displays and quotations, lists, accents (well, the ones defined in HTML), verbatim text, and there's even a weak implementation of `tabular`. I tried to be faithful to the spirit of LaTeX when adding new commands. (`latexinfo` had some commands with non-LaTeX syntax, those all had to go. I also changed the parser such that it much more closely mimics TeX's parsing.) One of the basic rules is that the meaning of no LaTeX-command has been changed (with the exception of two shortcuts that I found very useful, `\=` and `\+`—tabbing is not supported in HTML).

Hyperlatex provides three different ways of treating parts of your document differently in LaTeX and HTML mode. There are simple commands `\texonly` and `\htmlonly`, which ignore their argument if in the wrong mode. There is also a command `\texorhtml` which takes two arguments of which only one is evaluated. Then there are two environments `iftex` and `ifhtml` which are convenient to ignore larger chunks of input in one mode. Finally, you can prefix a single line with `\T` or `\H`, so that you could write

```
We are now in
\T \LaTeX-mode.
\H Html-mode.
```

Hyperlinks are created using the `\link` and `\label` commands. `\link{`*anchor*`}{`*label*`}` typesets the text *anchor* and makes it an active hyperlink to the label *label* in the HTML document. To also create a reference in the printed document, you will need to use `\ref` or `\pageref`. This is facilitated by the optional argument of `\link`.

```
\link{anchor}[printed reference]{label}
```

The LaTeX output of this command will contain the anchor and the printed reference, while the HTML output will only show the anchor as a hyperlink to the position marked with the label. So you can write

```
This parameter can be set in the
\link{configuration panel}%
  [~(Section~\ref{con-panel})]{con-panel}.
```

The starred version `\link*` suppressed the anchor in the printed version, so that we can write

```
We will see
\link*{later}[in Section~\ref{sl}]{sl}
how this is done.
```

It is very common to use `\ref{`*label*`}` or `\pageref{`*label*`}` inside the optional argument, where *label* is the label set by the link command. In that case the reference can be abbreviated as `\Ref` or `\Pageref` (with capitals). These definitions are already active when the optional arguments are expanded, so we can write the example above as

```
This parameter can be set in the
\link{configuration panel}%
[~(Section~\Ref)]{con-panel}.
```

This even works when we need the `\Ref` command outside, but after the `\link` command, as for instance in

```
\link{Such a file}{ipe-file} is at
the same time ... a legal \LaTeX{}
file\texonly{---see Section~\Ref}.
```

References to external resources are made in exactly the same way, using the `\xlink` command, and references to the bibliography work using the same principle.

To facilitate typing short pieces of mathematics, Hyperlatex has a `\math` command whose argument is read in math mode in the printed version. In the HTML version, it is simply left untreated, so you can write `\math{x_i}` to get x_i in the HTML document. You could also use the optional argument, and writing `\math[\code{x[i]}]{x_i}` will give you `x[i]` in the HTML version. The Pythagorean example can by written as either `\math{a^2 + b^2 = c^2}`, or as `\math[a*a + b*b = c*c]{a^2 + b^2 = c^2}`.

The most important shortcomings in Hyperlatex 1.3 are that there are no footnotes, that `\newcommand` and `\newenvironment` can only be used without arguments, and that there is no support for the new `<table>`, `<fig>`, and `<math>` tag of HTML 3 (which are already supported by some browsers). Also, there are a few ideosyncrasies that still stem from Hyperlatex's origin in `latexinfo`. The most important of these is the treatment of special characters. While LaTeX has ten of these, `latexinfo` has only three, namely `\`, `{`, and `}`. (`latexinfo` is mainly used for software documentation, where one often has to use these characters without there special meaning, and since there is no math mode in info files, most of them are useless anyway.) In the first version of Hyperlatex, I had only added the unbreakable space `~`, so there were four special characters. Since my main use was to write the Ipe manual, I found it convenient that I didn't even have to escape the % sign, for instance. However, it soon turned out that most other Hyperlatex users found this more confusing than convenient, and in Hyperlatex 1.1 the percent sign became a comment. And now that HTML 3 declares a `<math>` tag, I find that it is time to return all 10 special characters to their special status, and this is going to be realized in Hyperlatex 1.4. That means that the $ sign can now be used to enclose math mode material, that & can be used as a separator in formulas, and # for parameters for new commands. I hope that this will make it easier for new Hyperlatex users.

Another problem is the growing number of HTML-dialects. By user's request, I had added the fontsize changing command when `netscape` become available. Some other `netscape`isms can be used using optional arguments (which simply generate raw HTML attributes). I also changed the `center` environment to use `netscape`'s `<center>` tag (it used to be the same as `quotation`). It's getting more messy all the time, and Hyperlatex 1.4, which I'm testing right now, will have a `\htmllevel` command to set the type of HTML that will be generated (`html2`, `netscape`, or `html3`).

When HTML level 3 is selected, then Hyperlatex 1.4 will generate `<math>` tags for math mode material. It will also translate LaTeX's `figure` and `table` environment to `<fig>` tags, and will have a fuller implementation of `tabular` using `<table>`.

Hyperlatex 1.4 will also have footnotes, and commands and environments with arguments and I hope that that will make Hyperlatex users even happier.

More information about Hyperlatex is available on the world wide web at *http://hobak.postech.ac.kr/~otfried/html/hyperlatex.html* or *http://www.cs.ruu.nl/~otfried/html/hyperlatex.html*.