The Idea of a Markup Interface for XML

William F. Hammond

October 22, 1999
Last re-processing March 20, 2006

1 The Value of a LaTeX-Like Markup User Interface for XML

A markup user interface (MUI) for an XML language (formally "XML application") is a markup language that admits a well-defined translation to the XML language.

Recent discussions in `news:comp.text.tex` show that at least two of us are thinking about MUIs for LaTeX-like XML languages. Jonathan Fine has been working on a *roff*-like MUI that he calls "Active TeX", and I have been working on a LaTeX-like MUI that I call GELLMU.

As I use the acronym MUI, I do intend it to be reminiscent of the acronym GUI for "graphical user interface". Both GUIs and MUIs have the intention of making life easier, or perhaps at least more familiar, for some authors. An MUI, unlike typical GUIs, has the possibility of giving the author full and rigorous control over content.

Furthermore, an MUI in the style of a pre-existing non-XML markup offers a convenient avenue for prototyping a new XML language to model the markup practice in the pre-existing markup.

Beyond that, it offers a route for conversion of legacy archives in the pre-existing markup to XML languages with minimal human intervention.

Please allow me to say a bit more about what I have in mind for GELLMU.

2 The Basic GELLMU Processing Design

The things that I have on hand, aside from LaTeX are:

1. GNU Emacs, version 20. (I believe that version 19 is OK.)
2. James Clark's `nsgmls`, a part of his `SP`.

My processing set-up is the following:

Syntactic translation from LaTeX-like markup to SGML My Elisp processor, which can be run interactively in GNU Emacs or in batch mode, performs syntactic translation to convert LaTeX-like markup to an SGML language (formally, application). The syntactic translator is largely ignorant of command names. Whatever command names are used become the names of SGML elements. There is a standard way to convert multiple argument/option sequences.

This processing stage traps syntax errors. (It will fail to detect an even number of missing `'$'` characters; but this error will show in the next stage.)

Validating parse of the SGML language At the validation stage the difference between SGML and XML is significant if one wants to have "math mode" be a global toggle since that may be modeled robustly only using SGML exclusions.

A validating parse is made using `nsgmls`. Of course, the language definition, i.e., SGML application definition, is crucial. It is contained in an SGML declaration and in an SGML document type definition (DTD).

The language definition that I am using is the heart of my personal production system. But I regard it only as didactic in the larger scheme of things. Others will certainly want things that I do not want.

As far as it goes, it models LaTeX rather closely in some ways and rather loosely in other ways. Where it departs from close modeling, the reason is usually related to having a document structure that is not print-centric.

The validating parse traps errors in language use.

Down-translation to XML This is done with the program called `sx` in the family of `SP` processors.

While it is possible to recover an equivalent SGML document from the down-translated XML, it is not possible under the XML umbrella to have as precise a language definition as under the wider SGML umbrella. That said, either form may be run through a processor that serves to enforce a tighter language definition.

SGML processing This can go anywhere that is sane for the language definition. One can use any programming language, but it helps to have a basic SGML library on hand. I am using David Megginson's `SGMLS.pm`, a Perl 5 library, and its interface `sgmlspl`.

In my personal production system I routinely format GELLMU "articles" for both LaTeX and HTML. Invalid HTML and error messages from LaTeX represent bugs in my processors, which I always repair as soon as possible. There may be box size complaints from TeX; they represent authoring content errors. These two formatters work either on the SGML or the XML version of an article.

At present my personal production language definition is not fully up-to-speed for journal articles nor for translation to XHTML-with-MathML, but it is serving me well for my classes. It gives me a sane way to have course handouts and web offerings in a bullet-proof way from a single source. At this time I simply cannot assume that more than about a third of my students have easy access to PDF readers. So I feel constrained to give them simple HTML with very limited pseudo-TeX for math.

Other formattings for GELLMU article that should be possible include translations to (1) DocBook, (2) TEI, and (3) Texinfo, though suboptimally so long as math is not available. These are not small jobs, and I may never undertake any of them. (Any project that undertakes to format an XML language in Texinfo should give serious consideration first to modeling Texinfo as an XML. It would also be desirable to minimize Info/TeX bifurcation in XTexinfo and to provide math for XTexinfo.)

If a time arrives when I can assume that three-fourths of my students have out-of-the-box browsers for XHTML-with-MathML, then I should be able to format the original GELLMU source directly for that — if by that time I am not able to squeeze it out of carefully-set LaTeX-4. (I exaggerate somewhat. In principle, I need to provide some math symbol declarations in the sources, and math symbol declaration handling is not yet present in my set-up.)

3 About this document

This document was prepared as a GELLMU "article". A copy of the HTML version of this document is available on the web at

`http://www.albany.edu/~hammond/gellmu/mui/`

along with a full list of anchored versions as follows:

GELLMU source: `mui.glm`.

SGML, “article” document type: `mui.sgml`.
XML, “article” document type: `mui.xml`.
formatting in LaTeX: `mui.ltx`.
DVI made from LaTeX: `mui.dvi`.
PDF made from LaTeX: `mui.pdf`.
formatting in HTML: `mui.html`.