LaTeX profiles as objects in the category of markup languages

William F. Hammond

Abstract

The mathematical notion of "category" in the context of markup languages raises the idea of widespread use of reliable automatic translations between markup languages.

LaTeX profiles, which are dialects of LaTeX with a fixed command vocabulary where all macro expansions must be effective in that vocabulary, are suitable domains for defining translations to other profiles and, where sensible, to other markup languages.

The construction of reliable translators from several journal-neutral LaTeX profiles to many journal-specific LaTeX profiles would eliminate the need for technical editing in the production flow for academic journals.

1 Profiled usage of LaTeX

We now have 15 years of experience with various efforts to translate LaTeX into HTML, the language of the World Wide Web. We know that the task is more difficult than it appeared to be to many of us in the mid-1990s. Part of what makes LaTeX difficult to translate is that LaTeX, contrary to the impression one might gain from an initial reading of Leslie Lamport's book [4], has never been entirely formalized as a language unto itself independent of any implementation for processing the language. Indeed, the only implementation of LaTeX is that originating with Lamport — now maintained by The LaTeX Project (`http://www.latex-project.org/`) — as a very large macro package under TeX.

As slide 1 says, success with such translation requires profiled usage of LaTeX.

Slide 1: Translation of LaTeX

Question: What works well with translation software?

Answer: Profiled usage of LaTeX.

- Carefully limited command vocabulary.
- Tuned translation software.

Slide 2 provides a succinct statement of what I wish to suggest.

Slide 2: Today's Suggestion

formalize profiled usage

1.1 LaTeX profiles

What might be involved in formalizing profiled usage? First, I am suggesting the notion of *LaTeX profile* as the framework for multi-purpose LaTeX documents. Slide 3 specifies what is meant by "LaTeX profile":

Slide 3: The Notion of LaTeX Profile

- A dialect of LaTeX with a fixed command vocabulary where all macro expansions must be effective in that vocabulary.
- A language essentially equivalent to an SGML document type with a canonical XML shadow.

My project on *Generalized Extensible LaTeX-Like MarkUp* (GELLMU), `http://www.albany.edu/~hammond/gellmu/`, begun in 1998, underlies what I am suggesting today. The GELLMU "didactic production system" provides, in particular, a fairly elaborate example of what might be regarded as a LaTeX profile although some names in its command vocabulary are not part of current standard LaTeX usage. Slide 4 shows the source markup for a minimal document instance under this profile.

Slide 4: A Simple Example

```
\documenttype{article}
\surtitle{LaTeX Profiles}
\title{\latex{} Profiles: An Example}
\begin{document}

It's easier to learn to write in a
\latex{} profile than to learn to
write \latex.

The numbers $pi$, $i = \sqrt{-1}$,
and $e = \func{exp}(1)$ are related
by the equation
\[ e^{i\pi} = -1 \ . \]
\end{document}
```

Slide 5 shows a typeset rendition of this minimal document instance.

Slide 5: LaTeX Profiles: An example

It's easier to learn to write in a LaTeX profile than to learn to write LaTeX.
The numbers π, $i = \sqrt{-1}$, and $e = \exp(1)$ are related by the equation

$$e^{i\pi} = -1 \,.$$

Slide 6: The GELLMU Project

- Demonstrates that the ideas in this presentation can be implemented
- Provides a didactic document type which may be viewed as close enough to being a LaTeX profile that it can serve as a base for constructing profiles

From the outset I should make clear that:

1. The totality of standard usage of classical LaTeX, as we have it, is not suitable for modeling as a LaTeX profile.
2. There should be many LaTeX profiles.

1.2 HTML translation history

A close examination of our 15 years of experience with the most successful projects for the automatic translation of LaTeX to HTML will suggest that such translations should take place in two stages: (a) first, capture the LaTeX document as an XML document under a document type that closely models LaTeX, and (b) second, translate from that document type to HTML. This is, in fact, the design in the GELLMU project (see [6], [7], and [8]) except that GELLMU source, though LaTeX-like, is not LaTeX nor a dialect of present LaTeX.

The two stage design is also integral to the remarkably successful translator *LaTeXML*,[1] initiated around 2001 at the U.S. National Institute of Standards and Technology (NIST) under the very capable leadership of Bruce Miller for the purpose of providing a translation route to HTML (actually XHTML+MathML) for the *NIST Handbook of Mathematical Functions*. As time passed *LaTeXML* became the translation engine for the ambitious project "arXMLiv",[2] led by Michael Kohlhase of Jacobs University in Bremen, for translation to XHTML +MathML of Paul Ginsparg's large e-print archive,[3] originally housed at Los Alamos and now located at Cornell.

[1] `http://http://dlmf.nist.gov/LaTeXML/`

[2] `http://kwarc.info/projects/arXMLiv/`

[3] `http://www.arxiv.org`

An older very successful project for translation from LaTeX to HTML (including, as desired, XHTML+MathML) is *tex4ht*,[4] developed by the late Eitan Gurari of Ohio State University, in the years after 1995. While its core technique is the use of special-loading in DVI files generated by LaTeX with *tex4ht* macros, since the time that the project's scope was extended to generate a number of XML formats other than HTML, it has seemed clear to me that the *tex4ht* design would be improved by generating XML under a document type modeling the supported parts of LaTeX and then using standard XML libraries for translation to HTML and the various other XML document types.

2 The advantage of using a LaTeX profile

The crux of the problem in translating LaTeX documents to HTML is that LaTeX, as a whole, is not well-defined as a language unto itself. In the LaTeX community there is a well-known "newbie" question: *How can I know if my LaTeX document is correct?* A commonly heard answer is that a LaTeX document is correct if it runs seamlessly through LaTeX, the program.[5]

The whole of LaTeX, with maximally sane mixing and matching of packages and arbitrary "legal" (see Lamport [4], Appendix E) excursions into the world of plain TeX, is suitable for translation to printer languages (either specific printer languages or, more commonly DVI and PDF) but not suitable for reliable automatic translation to HTML or to common author-level document formats.

2.1 Language specification

When a list of LaTeX commands, perhaps 500 to 1500 in number,[6] is fixed, and when rules for usage of those commands in relation to each other, i.e., what commands are allowed to appear in a given context, are given, then one has something that is essentially equivalent to an SGML[7] document type. It is straightforward to construct an XML[8] shadow.

[4] `http://www.cse.ohio-state.edu/~gurari/TeX4ht/`

[5] A tougher standard, probably unreasonably tough, might be that a LaTeX document is correct if it runs through LaTeX and also through the *tex4ht* driver script `"mzlatex"` for generating XHTML+MathML.

[6] Imagine culling these commands from the LaTeX core and from one's favorite packages; but note that one is just assembling a list of commands and is not in any way thereby adopting segments of packages.

[7] *Standard Generalized Markup Language*, an ISO standard. See Goldfarb's *Handbook* [2] for a copy of the standard, and see `http://www.sgmlsource.com/` for amendments to the standard.

[8] *Extensible Markup Language*, a World Wide Web Consortium (W3C) standard [1].

The reason for the dual track is that dialects of classical LaTeX can be more closely modeled with SGML than with XML, which is SGML dumbed down for use on the World Wide Web. The difference here between SGML and XML is a question of convenience for authors. For example, if we want blank lines to begin new paragraphs without the tediously redundant explicit closing of previous paragraphs, then we want the umbrella of SGML in-house for the formal structuring of a LaTeX profile rather than the XML umbrella.

When a document language is implemented under the SGML umbrella, then

1. It is possible to know with varying degrees of precision, as required, when a document instance is technically correct.
2. Correct document instances can be translated automatically to other suitable formats with a very high degree of reliability.
3. Software libraries are available for most computer languages to facilitate automated translation and other forms of automatic processing such as, for example, automatic extraction of metadata.

2.2 Categories: A metaphor

At this point it is relevant to mention the mathematical notion of *category*.

Slide 7: Notion of Category

- A category consists of:
  1. Objects
  2. Arrows between objects
- Rule: An arrow followed by a second is also an arrow
- Relevance: to suggest a way of thinking about markup
- (No plans for actually using category theory)

My reason for introducing the concept of *category* here is not for the purpose of applying category theory but for the purpose of suggesting different ways of thinking about how the community handles its documents. In particular, I'm trying to suggest a way of thinking about how the systematic use of translations between document formats can improve the way we operate with our documents. The largest relevant category here is the category of all markup languages:

Slide 8: The Category of Markup Languages

- Markup languages are the objects
- Translations are the arrows

Slide 9 shows the principal objects and arrows in the category of markup languages associated with the GELLMU didactic production system.

Slide 9: Objects and Arrows in GELLMU

$$\begin{tikzcd}
\boxed{\text{GELLMU source}} \arrow[r] & \boxed{\text{SGML}} \arrow[d] & \\
\boxed{\text{Outside SGML source}} \arrow[r] & \boxed{\text{Author-level XML}} \arrow[d] \arrow[dr] & \\
 & \boxed{\text{PDF}} & \boxed{\text{HTML}}
\end{tikzcd}$$

Classical LaTeX is, more or less, a markup language. While many useful arrows can point toward LaTeX, few useful arrows point from LaTeX toward markup languages other than printer languages.

Slide 10: Classical LaTeX: An object in the category

(to the extent that classical LaTeX is a well-defined language)

- LaTeX is a reasonable translation target (for author-level markup languages).
- LaTeX is a poor domain for translation to languages other than printer languages.

Classical LaTeX does not fall under the umbrella of SGML, but classical HTML does. While classical HTML is not under the umbrella of XML, there is a variant of HTML called XHTML that is.

2.3 SGML and XML

Slide 11: SGML & XML

- SGML is a subcategory of the category of all markup languages
- XML is a subcategory of SGML
- XML is SGML made suitable for the World Wide Web

SGML is designed so as to facilitate the construction of arrows emanating from an SGML document type.

Slide 12: Good domains for translation

- Author-level SGML and XML document types are, by design, good domains for translation, i.e., arrows can flow **from** these document types.
- Arrows can be "chained"; these pipelines work well.

Let me suggest that in a properly designed system the chaining of arrows, i.e., translations, should take place at the command line. This makes it possible for the user to switch components (translators) from time to time, and it provides the opportunity for the user to run correctness tests at intermediate stages.

While the use of translations between document formats has not been part of regular LaTeX practice, the use of translations has not been far away.

2.4 Example: *Texinfo*

Texinfo, the language of the GNU Documentation System, was originally a vehicle for the simultaneous generation from a single source of (1) print output (via DVI) and (2) "Info", an early form of hypertext predating HTML. Historically, *Texinfo* had been processed by the TeX engine with *Texinfo* macros for print and either by the GNU Emacs Lisp engine or by a free-standing C program, called *makeinfo*, for Info hypertext. When HTML came along, it became easily possible for the program *makeinfo* to generate HTML as well as Info. It has been obvious from the beginning to anyone asking the question that *Texinfo*, which has a careful language definition, is equivalent to an SGML document type. This was formalized in the year 2000 with Daniele Giacomini's *Sgmltexi*,[9] and a few years later an independent XML version of *Texinfo* was incorporated in the *Texinfo* distribution.

[9] `http://www.archive.org/details/sgmltexi`

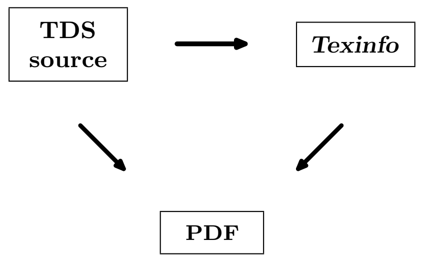
Because of its different markup syntax and command vocabulary relative to LaTeX, it is not sensible to think of *Texinfo* as a LaTeX profile. But aside from those differences, it is a long-standing example of something that is the same type of object as a LaTeX profile.

As an object in the category of markup languages *Texinfo* can serve usefully as both origin and target for arrows. This is often the case with author-level SGML and XML document types.

2.5 Example: The *tdsguide* document class

Another instance of the use of format translations lies in the production during the late 1990s of the specification document for the TeX Directory System (TDS), `http://mirror.ctan.org/tds.zip`. Ulrik Vieth devised a set-up under which the TDS specification could be written in LaTeX and processed either as a LaTeX document or as a *Texinfo* document (for Info and HTML outputs). Of course, there was a finite list of LaTeX commands (and environments) associated with the TDS LaTeX source. In effect, this system can be said to amount to the ad hoc construction — not formalized — of (1) a LaTeX profile and (2) a program for translating that LaTeX profile to *Texinfo*.

Slide 13: *tdsguide*: Two Routes to PDF



Note that although the LaTeX source `tds.tex` is very nicely structured, there are two reasons why a direct-to-html translation program might be expected to have trouble with it. First, it is written under the document class *tdsguide*; I am not aware of any direct-to-html translator with knowledge of that document class. Second, the document uses *manmac* syntax, which is part of plain TeX rather than LaTeX.

Moreover, inasmuch as it is reasonable to think about constructing a translation to regular LaTeX from any author-level XML document type, it is, in particular, reasonable to imagine the possibility of someone constructing a translation from (the XML guise of) *Texinfo* to LaTeX. That would offer a third (and different) route to PDF for *tdsguide* documents. Beyond that it is reasonable to think about constructing a translation from *Texinfo* to GELLMU article, and, following that with the translation to LaTeX provided for GELLMU article would provide a fourth route to PDF for *tdsguide* documents.

3 How LaTeX profiles might be deployed

There are many advantages.

3.1 Traditional LaTeX authors

We know that a very substantial portion of the community of LaTeX authors is interested in being able to "pull" HTML from LaTeX documents. There are frequent questions in the Usenet newsgroup `comp.text.tex` about using translation programs. There is a recurring theme in discussions there where the participants are seeking a LaTeX authoring interface — a better form of LaTeX markup — that is robust for translation to other formats, particularly HTML. Another recurring theme that is sometimes intertwined with the former theme is that of having a version of LaTeX that enables an author to focus on content. Of course, there is some irony in this latter theme in that Lamport in his book [4] explicitly states that this is the purpose of LaTeX — as is indeed the case, when LaTeX is used well. Thus, the latter theme may be interpreted to be about somehow "enforcing" the use of LaTeX as Lamport intended.

One or more LaTeX profiles sponsored by the LaTeX project could meet these requests.

3.2 LaTeX for beginners

With a suitably literate system for constructing LaTeX profiles it should be a relatively simple matter to produce a command glossary for a given LaTeX profile.

Combine the availability of a definitive command glossary with the enforcement of sane markup provided by routine structural validation in the process of running LaTeX on a document instance under a LaTeX profile, and it is not difficult to see why it should be easier for a beginner to learn a LaTeX profile than to learn classical LaTeX.

3.3 LaTeX's interface with established SGML document types

Although provision for typesetting SGML and XML has been a stated goal of the LaTeX3 Project, almost all of the Project's work so far has been focused on developing the infrastructure for writing document classes and packages. The community is just beginning to harvest the results of that work.

Another aspect of the situation with SGML and XML in relation to LaTeX is that while LaTeX is used by a very large community of authors, it does not seem to be the case that large crowds of original authors have migrated to well-known document types such as that of the Text Encoding Initiative (TEI), `http://www.tei-c.org`, which provides a vehicle for capturing electronic versions of classical printed texts, and the *DocBook*[10] document type, of the Organization for the Advancement of Structured Information Standards (OASIS), oriented (like *Texinfo*) toward documentation of technical work.

Well supported SGML and XML document types such as these typically are accompanied by formatting code for reaching both HTML for online presentation and PDF for print presentation.

Some tentative and rather incomplete experiments[11] I have undertaken, with the GELLMU didactic document type playing the role of a LaTeX profile, suggest to me that the process of formatting SGML and XML documents for print and HTML can be improved by first translating to a suitable LaTeX profile. This may seem contrary to common sense. The first point, however, is that numbering and cross-references can be worked out at an early stage, so that one can have consistency in this regard between the print and HTML outputs. The second point is metaphorical. Think of the trip from a source document to an end format as a downhill journey. One can jump the entire distance, or one can move more slowly, checking in at way stations along a path; at each of these stages one strives to retain as much as possible for that stage of the author's expressed intention as found in the source.

3.4 Suggestions for LaTeX evolution

Slide 14 presents a capsule description of what I think the LaTeX project should undertake in order to support the generation of HTML and other non-traditional formats from LaTeX source.

Because the LaTeX project needs to continue to provide support for legacy documents, a document instance written under a LaTeX profile must be clearly identifiable at the outset. My suggestion, used from the outset in the GELLMU project, is that one of these new document instances, prepared under

[10] `http://www.oasis-open.org/docbook/`

[11] These involved using *sgmlspl*, about which there is a section in *The LaTeX Web Companion* [3], and which is the staged translation workhorse in the GELLMU project, to begin building a translator from *DocBook* to GELLMU *article*, sufficient for handling four particular document instances.

a LaTeX profile, should begin with `\documenttype` rather than `\documentclass`.

Slide 14: Suggestion for the LaTeX Project

- Sponsor one or more reference profiles.
- Sponsor translations from reference profiles to PDF and HTML.

The argument of *documenttype* should be the name of the root element in the XML document type corresponding to the LaTeX profile being used. The name of the root element in an XML document type falls far short of specifying completely what document type definition (command list) is involved. With GELLMU `\documenttype` has an option whose content is typically a string that serves as a key to (or name for) a data structure that has been made known to the syntactic translator. (See §3.1 of the *GELLMU Manual* [5].) If this option is missing, the syntactic translator looks for a default value for that key corresponding to the name of the root element. While presently in GELLMU the documenttype option points to information characterizing the document type and style choices are determined by the manner of running processors on a document instance, it would be possible with LaTeX profiles, if desired, to incorporate style choice information in the data structures behind these keys.

As with the ad hoc system created for the TDS specification (section 2.5) there will be various choices for processing a document instance prepared under a LaTeX profile. In particular, there are a number of choices for print formatting. Slide 15 indicates three possibilities:

Slide 15: Paper Typesetting of a LaTeX Profile

There are several possibilities:

- Translate to classical LaTeX
- Translate to ConTeXt
- Teach LaTeX itself to digest the profile

In case it is not completely clear, I should point out that the third of these routes might be substantially different from the first two in that the latter probably should involve first explicitly generating the XML shadow of the document instance and then translating from there. If one is also going to have HTML output, it would, of course, be expected that the HTML output would be translated from the same XML shadow. With the third route, however, there will be the question of bypassing generation of a formal XML shadow and then losing the advantage of structural validation — or maybe constructing the XML shadow for structural validation and for staging non-print outputs but generating the print output directly from the source.

At this point I hope that the suggestions in Slide 16 will be obvious.

Slide 16: Publishing

- Encourage maintainers of XML document types to reach HTML and PDF by translating first to reference LaTeX profiles.
- Encourage authors to submit articles to journals as LaTeX instances under reference profiles.

Technical editing by a journal should not be necessary when an article is submitted under a reference LaTeX profile. The journal will have its own profile that is a modification of a reference profile enabling the journal to incorporate metadata. For each article a journal will prepare metadata, and the journal's processing stream will merge that metadata with the author's source to generate a document instance under the journal's profile that, in turn, will be processed to end formats using the journal's formatting software.

4 Issues with implementation

Within the GELLMU project, particularly its didactic production system, there is a demonstration that everything discussed here is possible. The question is how these ideas might be introduced so as to provide as smooth as possible a transition in authoring techniques for those LaTeX users who wish to avail themselves of the advantages of source markup that is amenable to generalized processing rather than only processing for print by TeX engines.

4.1 Syntax

The discussion following slide 15 suggests that a possible route to print for a document instance under a LaTeX profile might bypass the profile's XML shadow. There are related questions:

1. Might a modern engine like `luatex` be a good vehicle for generating the SGML shadow? Similarly, might such an engine be a better vehicle than the GNU Emacs Lisp engine for generating the *Texinfo* version of the TDS specification (see section 2.5)?

2. Shall the engine used to generate the SGML shadow use knowledge (unlike the GELLMU syntactic translator) of the corresponding document type definition[12] in so doing? For example, shall this stage of processing be allowed to know that `\frac` takes two arguments, numerator and denominator, and to know what the names of those arguments are?

In regard to the second of these questions, if the processor for generation of the SGML shadow is allowed to use knowledge of command vocabulary, then the syntactical requirements for a document instance could be somewhat looser than in GELLMU. For example, as in LaTeX but not in GELLMU, spaces could be allowed between the successive arguments and options in a command invocation. This would be both possible and, in a sense, author-friendly. However, I do not recommend loose syntax. The syntactic tightness of GELLMU was motivated by that of *Texinfo*. The enforcement of syntactic tightness can help prevent author errors. Moreover, the overall design of processing is more "modular" when the first stage deals only with syntax.

Also, if the processor for generation of the SGML shadow incorporates knowledge of the command vocabulary, then it may be reasonable for that processor to write the XML shadow directly without first going through an SGML shadow and then relying on an SGML parser and a subsequent processor with knowledge of the command vocabulary to write the XML shadow.

4.2 Counters

While LaTeX has infrastructure for managing counters, there is nothing entirely parallel in the SGML world. Document types can provide hooks for counting, and the processors operating on those document types can take those into account. Section 5.5 in the *GELLMU Manual* [5], "Labels, References, and Anchors", explains the approach to this in the GELLMU didactic production system.

Where there are to be multiple end formats of a given document instance, it is best if counters, numbering, and cross-references are managed centrally, as in the GELLMU didactic production system, so that there is consistency among the various end formats.

[12] That the GELLMU syntactic translator operates only on syntax, largely without knowledge of command vocabulary, enables the syntactic translator to be employed for writing virtually any SGML or XML document type using directly the vocabulary of that document type with the advantage for the author of being able to use GELLMU's *newcommand* with arguments.

4.3 Names

For a faithful representation[13] of source markup under a LaTeX profile as an XML document every item of LaTeX markup needs to have a name.

For the document type of the GELLMU didactic production system the minimum number of characters in a command name is three. One and two character names are reserved for user macros. While this is not necessary, I recommend it.

A number of frequently used commands in classical LaTeX do not have names. The SGML and XML shadows need names for them. For example, `~` is "non-breaking space" in LaTeX. In the GELLMU didactic production system it becomes the empty element named *nbs*.[14]

While the argument in a LaTeX command such as `\emph` simply corresponds to the content of an element `<emph>` in the XML shadow, the arguments in a command taking more than one argument such as `\frac` need to be named. For example, `\frac{3}{7}` would be represented in the XML shadow as

```
<frac><numr>3</numr><denm>7</denm></frac> .
```

4.4 Special ASCII characters

One commonly sees the 128 characters in the ASCII character set in a table of 8 rows of 16 characters each. The first 2 rows are control characters that are, apart from newlines, non-printable and not used either in LaTeX or in SGML document types. Thus there are 6 rows of 16 characters that are the printable ASCII characters except for the very last of these, which is not printable and not relevant here. Within the ASCII realm, we need to deal with the 95 printable characters. Of these 62 are alphanumeric — upper- and lowercase letters and numerals. The 33 remaining printable ASCII characters are the "special" ASCII characters.

Within the realm of electronic formats each of the 33 special ASCII characters is a candidate for use as a control character of some type. For this reason it is wise that a name be provided for each of these characters in a LaTeX profile. This is the case in the GELLMU didactic document type. Thus, for example, `{` has markup meaning in LaTeX, and one may use `\{` to place an actual left brace in one's document; in the GELLMU didactic production system `\{` is represented by the empty element `<lbr/>` in the XML shadow. The character `@` is only a bit different. Normally it is perfectly safe to use this character for

[13] Faithful to the source apart from *newcommand*s that should appear expanded.

[14] Of course, "non-breaking space" is the Unicode character U+00A0, but writing that in the XML shadow would make the shadow not a faithful representation of the source.

itself in a LaTeX document, and normally there is no harm in passing it through as itself to the XML shadow. If, however, the author, as in the case of the TDS specification (see 2.5), wants to translate the document to *Texinfo*, there is suddenly a problem since `@` has markup significance in *Texinfo*.

For each of these 33 special characters there can be found contexts where they have markup or "control" significance. Names should be available for all of them.

4.5 Non-ASCII characters

Because TeX and LaTeX originally handled only 7-bit characters, LaTeX has legacy names for characters that are no longer strictly necessary or, at least, almost no longer strictly necessary in the sense that we expect Unicode-capable modern TeX engines soon to be mainstreamed. For example, there is the character 'ł', U+0142, which may be entered as `\l`[15] in LaTeX and as `&lstrok;` in classical HTML.

For the long term it seems clear that names for Unicode characters beginning with the Latin 1 range of Unicode are not needed.

4.6 CSS

CSS stands for "Cascading Style Sheets", which is a web technology for controlling the appearance of HTML and of arbitrary author-level XML document types for display in mainstream web browsers.

At the point where some LaTeX documents have XML shadows, one may become interested in writing CSS sheets for governing the display of those XML shadows in web browsers. Beyond that the question arises whether there might some day be gain in using CSS sheets to govern, at least in part, the typesetting of such a document by LaTeX.

References

[1] Tim Bray, Jean Paoli, & C.M. Sperberg-McQueen, *Extensible Markup Language (XML) 1.0*, World Wide Web Consortium Recommendation, 10 February 1998, `http://www.w3.org/TR/1998/REC-xml-19980210`, currently superseded by `http://www.w3.org/TR/REC-xml/`.

[2] Charles F. Goldfarb, *The SGML Handbook*, Oxford University Press, 1990.

[3] Michel Goossens and Sebastian Rahtz et al., *The LaTeX Web Companion*, Addison-Wesley, 1999.

[4] Leslie Lamport, *LaTeX: A Document Preparation System*, 2nd edition, Addison-Wesley, 1994.

[5] William F. Hammond, *The GELLMU Manual*, 2007, `http://mirror.ctan.org/support/gellmu/doc/glman.pdf`, or `http://mirror.ctan.org/support/gellmu/doc/glman.xhtml` (XHTML+MathML).

[6] William F. Hammond, "GELLMU: A Bridge for Authors from LaTeX to XML", *TUGboat: The Communications of the TeX Users Group*, vol. 22 (2001), pp. 204–207; also available online at `http://www.tug.org/TUGboat/Contents/contents22-3.html`.

[7] William F. Hammond, "Dual presentation with math from one source using GELLMU", *TUGboat: The Communications of the TeX Users Group*, vol. 28 (2007), pp. 306–311; also available online at `http://www.tug.org/TUGboat/Contents/contents28-3.html`. A video recording of the presentation at TUG 2007, July 2007, in San Diego is available at `http://www.river-valley.tv/conferences/tex/tug2007/`.

[8] William F. Hammond, "Multipurpose LaTeX-like markup for math", talk given in the AMS-MAA Special Session *Putting Math on the Web the Correct Way* at the Joint Mathematics Meetings in San Diego in January 2008. This has not been published, but HTML slides that link to many examples are available on the web at `http://math.albany.edu/math/pers/hammond/Presen/JMM08/Putting/`.

◇ William F. Hammond
Dept. of Mathematics & Statistics
University at Albany
Albany, New York 12222
USA
`hammond (at) albany dot edu`
`http://www.albany.edu/~hammond/`

[15] In keeping with the thought that one-character names should be reserved for user macros, the name for this character in the GELLMU didactic production system is `\csll`.