Why the LaTeX community should care about SGML

William F. Hammond

Abstract

Given that the universal format-to-format translator Pandoc is coming of age, LaTeX authors are tempted to think that whatever LaTeX they write can quickly be translated without worry to whatever other format may be required.

Of course, that is not exactly true, but the use of an XML profile of LaTeX can make it exactly true. However, an SGML profile of LaTeX can provide closer emulation of classical LaTeX than an XML profile.

Most actors in the world of markup have restricted their use of SGML to XML. For that reason software that handles SGML beyond the realm of XML seems to be falling out of maintenance. If the LaTeX community wishes to continue to be able to avail itself of the advantages of SGML for LaTeX source emulation, it may fall on the LaTeX community to maintain the extant SGML libraries.

1 Maximally useful source markup

Authors spend a great deal of time writing their articles and reports. Because of that investment of time it is important that tools used in writing are chosen carefully. One wants the fruit of one's writing to be presentable not only on the printed page but also on screens of various sizes from mobile telephones to full-wall monitors. It is better to write once and then use robust processing streams for the desired output formats rather than to edit manually for each output. Within the realm of TeX-like source, the best results flow from profiled LaTeX.

1.1 The concept of LaTeX Profiles

A LaTeX profile is a dialect of LaTeX with a fixed command vocabulary, where all macro expansions must be effective in that vocabulary, having dual existence under an SGML [4] document type with a canonical XML [2] shadow. I spoke about this concept in my talk [7] on the 100000_2-th (i.e., 32nd) anniversary of TUG in 2010.

While an instance of a LaTeX profile may be written directly as SGML or XML using the regular syntax, it is envisioned that one would want to use generalized LaTeX, i.e., write using the syntax of LaTeX.

The XML guise of a LaTeX profile can be styled, though with less than perfection, using CSS [1]. See my talk [8] at TUG 2014. This is not a new idea except insofar as it involves CSS-only rendering of most of LaTeX math.

My GELLMU Project, found at my university website, `albany.edu/~hammond/gellmu`, as well at CTAN (`ctan.org/pkg/gellmu`) provides a didactic model for the use of a LaTeX profile.

1.2 A side remark on accessibility

In reference to the first of Ross Moore's talks at this meeting, I want to suggest that HTML 5 [3], with Unicode text and MathML for math, obtained from a LaTeX profile can rather easily be made accessible for those with vision impairment.

2 Why SGML rather than XML?

The simple answer is that SGML provides better emulation of classical LaTeX than XML in that XML requires markup elements to have tags for both start and end, whereas with SGML it is commonly possible to omit an end tag and less commonly possible to omit a start tag and sometimes both. This is not the place to rehearse the conditions under which these tag omissions are allowed. The point is that classical LaTeX may be viewed as allowing many tag omissions, and for that reason one may construct approximate SGML models that are closer to classical LaTeX than any XML model can be.

One aspect of the overall idea of generalized LaTeX is that the processing should proceed through a pipeline with well-defined stages, and the first stage of that processing should involve only syntax. Thus, for a particular LaTeX markup structure, a human may see how to use code to reformulate it directly under an XML document type, but that formulation might require knowledge of both classical LaTeX vocabulary and the vocabulary of the XML document type.

3 Example: A simple table

There follows a mundane example of a centered table that would normally be created in LaTeX using a *tabular* environment inside a *center* environment.

long phrase	five	shorter
shorter	long phrase	five

The pattern of horizontal alignment in the cells of this table is “rcr”, which in LaTeX is normally furnished as an argument of the *tabular* environment. This table has two rows, each with three cells. The rows have horizontal borders and the cells have vertical borders.

In the GELLMU Didactic Production System this centered table can be marked up with

```
\begin{display}
\begin{tabular}{|r|c|r|}
\hline
long phrase & five & shorter \\
\hline
shorter & long phrase & five \\
\hline
\end{tabular}
\end{display}
```

One probably wants the cell separators ("&") to be viewed as tags for cells and the row separators ("\\") to be viewed as tags for rows.

The question here is not how to mark up a table in some corresponding XML but how to formulate XML markup that models this LaTeX construction. Where do the *hline*-s belong in that model? If only because the question about the *hline*-s requires some thought, the translation from this generalized LaTeX to an XML model cannot be just a matter of syntax.

But with a few syntactic conventions, including the recognition of argument syntax on the *tabular* environment to generate a generic "argument" "`<ag0>`", flagging the \\ as a generic "breaking" element "`<brk0>`", and recognizing the special syntactic role played in LaTeX by the character "&" leading to the element "`tabampcell`", one arrives in a straightforward fashion at this segment of SGML:

```
<display>
<tabular><ag0><vbr/>r<vbr
   />c<vbr/>r<vbr/></ag0>
<hline>
long phrase
  <tabampcell>five
  <tabampcell>shorter<brk0>
<hline>
shorter
  <tabampcell>long phrase
  <tabampcell>five<brk0>
<hline>
</tabular>
</display>
```

(In this example "`<vbr/>`" is an empty element representing the special character "|". In the GELLMU Didactic Production System all 33 of the printable non-alphanumeric ASCII characters have representation as empty elements with three-letter names. There are various context-dependent ways that any of these characters can be special after a format translation. Naming them makes it possible for last processing minute decisions to be made. On the other hand, use of the names is quite often optional in generalized LaTeX markup source, as here with "|", so long as emulation of TeX's *manmac* is not being engaged.)

At the next stage of processing — under an SGML transformation — using code with knowledge of markup vocabulary at both ends, the SGML segment above is transformed to the following XML segment:

```
<display>
 <tabular>
  <tabuhead>
  <tabharg><vbr/>r<vbr/>c<vbr
          />r<vbr/></tabharg>
    <hline/>
  </tabuhead>
  <tabubody>
   <taburow>
    <firstcell>long phrase</firstcell>
    <tabampcell>five</tabampcell>
    <tabampcell>shorter</tabampcell>
   </taburow>
   <taburow>
    <firstcell><hline/>shorter</firstcell>
    <tabampcell>long phrase</tabampcell>
    <tabampcell>five</tabampcell>
   </taburow>
   <taburow>
    <firstcell><hline/></firstcell>
   </taburow>
  </tabubody>
 </tabular>
</display>
```

4 SGML, LaTeX, decline, and authors

Like XML, SGML is a grammar for markup languages. XML has stricter rules than SGML. While formally SGML and XML have disjoint specifications, it is nonetheless the case that any XML document type admitting a "DTD" definition may be realized in a routine way as an SGML document type. In the other direction, most SGML document types admit an SGML normalization that can usually be transformed to an XML document type.[1]

4.1 Why XML now dominates SGML

One of the complications with SGML that led to the rise of XML is that an SGML document very rarely exists as a stand-alone file. An SGML document must always include or reference a formal document type definition and be associated, explicitly or implicitly, with an on-board SGML declaration. As a practical

[1] But, for example, there might be a challenge in this direction with an instance of an SGML document type that makes extensive use of SDATA.

matter an SGML user must have a collection of auxiliary documents just as a LaTeX user must have a collection of packages. While this can be practical for sharing among a group of authors, it makes sharing an SGML document across the web more difficult and less efficient than sharing an XML document.

SGML documents that are not XML have effectively vanished from the web.[2] The fact that any new SGML documents being generated are behind closed doors, together with a somewhat steeper learning curve for SGML than for XML, seems to have led to a loss of interest in SGML beyond XML.

4.2 Who the authors are

How do SGML and XML documents arise?

I believe most of the books and articles written by actual authors, whether for academic publication or for the popular press, are most likely written either with a word processor, such as provided by Microsoft, or in a TeX-family markup.

I believe that most extant SGML or XML documents are not actual source. For example, there are "markdown" languages from which basic XML documents can robustly be spawned. When SGML or XML documents are actual source, the creators are usually persons working, one way or another, in document technology. Those creators usually work with editing tools that have been adapted to minimize the distinctions between SGML and XML that I mentioned earlier in section 2.

I see the LaTeX community as still under challenge by the concern raised by Chris Rowley at the 2010 TUG meeting over "peak TeX" (analogous to "peak oil"), and here I've pointed to SGML (beyond XML) being in decline. One of the threats for the future of LaTeX is its difficulty in being converted to other formats for documents where that is sensible. The concept of a LaTeX profile provides a place where LaTeX and SGML can help each other to the benefit of both.

5 Libraries for parsing and processing

Because the rules for an XML document are somewhat more restrictive than the rules for an SGML document, libraries for processing XML are easier to construct and maintain than libraries for processing SGML. Indeed, the specification of SGML provides for variations of syntax, detailed character set specification, and myriad markup shortcuts to the point that I do not know whether any library was ever produced to provide functions for handling a full implementation of the SGML specification. However, the extent of SGML use that I think desirable for profiling LaTeX is well within the territory covered by SGML libraries.

[2] There was a time after the rise of XML that version 4 of HTML, an SGML "application" (document type), was the dominant markup for web pages. It worked because web browsers were required to have native knowledge of HTML. This continued for a while even after the early XML form of HTML was promoted and then appeared to gain widespread use through many web page instances that were not actually well-formed XML.

5.1 Simplicity with XML

With the rise of XML and the ease of writing software for parsing and transforming XML documents, many new options appeared. With XML a document need not be accompanied by a document type definition. It is sufficient that transforming software knows the markup vocabulary used with the document. Thus:

- An XML document may be rendered in a web browser solely by linking the document to a CSS stylesheet.
- An XML document may be transformed to another format by using an XSLT transformation that is defined by creating an XSLT stylesheet, which itself is an XML document.

My guess is that XSLT is probably the most widely used transformation language for XML today. Personally, I find it cumbersome to write for XSLT. I would much rather code in a traditional programming language. In particular, it is not pleasant trying to code for XSLT when the translation target is LaTeX.

5.2 The libraries OpenSP and SGMLSPM

I believe that the most widely deployed library for handling SGML is OpenSP. It is a C++ library for parsing and transforming SGML documents that was spawned from James Clark's SP. Before the rise of XML, I believe Clark's SP was dominant. For example, at some point during the time of Sun Microsystems' Solaris operating system, the system manual pages were re-coded from Roff source to a variant of Docbook SGML, and SP was deployed for generating various output formats. This arrangement for system manual pages is found today in Ubuntu systems though with OpenSP rather than SP. It's not always understood that since every XML document may be construed as an SGML document, OpenSP can be used with XML documents.

I might also mention OpenJade, which was spawned from Jade, also by James Clark. OpenJade provides an engine for *Document Style Semantics and Specification Language (DSSSL)*, which is an early transformation language for SGML that is written in SGML — thus, a forerunner of XSLT — under an SGML declaration that makes one of its stylesheets look like Lisp code. Usually package management systems that house OpenSP also house OpenJade; I mention this because online searches for "`openjade`" can be easier than for "`opensp`".

Finally, I want to mention the Perl software SGMLSPM/sgmlspl for SGML transformations written by David Megginson of Ottawa and released as GPL software in 1995 that, to my knowledge — I use it daily[3] — has never since needed repair. SGMLSPM/sgmlspl enables one to write a handler for each element in an SGML (or XML) document type to treat the rendering of that element in a translation. It is designed to accept a parsed stream from OpenSP and generate an output stream in the target format. This works well for writing HTML, LaTeX, and just about any output format. If the handlers are not written mindlessly, a single run of OpenSP piped to SGMLSPM/sgmlspl can handle a very large document. Also there is the advantage that inside one of those handlers, the transformation writer has the full power of Perl.

5.3 OpenSP needs maintenance

Unfortunately, as it is today, OpenSP only supports the basic multilingual plane of Unicode (U+0000–U+FFFF). Tackling the task of adding support for all of Unicode might not at first glance seem hard, but it is daunting because of the complexity of OpenSP that arises from the extent of its coverage of SGML beyond XML and its character handling. To my mind this task might be a good master's thesis project for someone in Computer Science. Prior to modification the code will require much study. The task needs someone with stamina and good eyes.

References

[1] Bert Bos, Tantek Çelik, Ian Hickson, & Håkon Wium Lie, *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification*, World Wide Web Consortium Recommendation, 7 June 2011. `w3.org/TR/2011/REC-CSS2-20110607`

[2] Tim Bray, Jean Paoli, C.M. Sperberg-McQueen, Eve Maler, & François Yergeau, *Extensible Markup Language (XML) 1.0 (Fifth Edition)*, World Wide Web Consortium Recommendation, 26 November 2008. `w3.org/TR/2008/REC-xml-20081126`

[3] S. Faulkner, A. Eicholz, et al., *HTML 5.2*, World Wide Web Consortium Recommendation, 14 December 2017. `w3.org/TR/2017/REC-html52-20171214`

[4] Charles F. Goldfarb, *The SGML Handbook*, Clarendon Press, Oxford, 1990.

[5] William F. Hammond, "GELLMU: A Bridge for Authors from LaTeX to XML", *TUGboat* 22:3 (2001), pp. 204–207. `tug.org/TUGboat/tb22-3/tb72hammond.pdf`

[6] William F. Hammond, "Dual presentation with math from one source using GELLMU", *TUGboat* 28:3 (2007), pp. 306–311. `tug.org/TUGboat/tb28-3/tb90hammond.pdf` A video of the presentation at TUG 2007, July 2007, in San Diego is available at `http://zeeba.tv/conferences/tug-2007`.

[7] William F. Hammond, "LaTeX profiles as objects in the category of markup languages", *TUGboat* 31:2 (2010), pp. 240–247. `tug.org/TUGboat/tb31-2/tb98hammond.pdf`. A video of the presentation at TUG 2010, June 2010, in San Francisco is available at `http://zeeba.tv/conferences/tug-2010`.

[8] William F. Hammond, "Can LaTeX profiles be rendered adequately with static CSS?", *TUGboat* 35:2 (2014), pp. 212–218. `tug.org/TUGboat/tb35-2/tb110hammond.pdf` A video recording of the presentation at TUG 2014, June 2014, in Portland, Oregon is available at `http://zeeba.tv/conferences/text/tex/tug-2014`.

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

[10] MacFarlane, John, *Pandoc: A Universal Document Converter.* `pandoc.org`

◇ William F. Hammond
University at Albany,
Albany, New York, and
San Diego, California
`whammond (at) albany dot edu`
`https://www.albany.edu/~hammond/`

[3] After incorporating one additional feature written by Dave Walden.