E-books and Graphics with LaTeXml

Deyan Ginev[1], Bruce R. Miller[2], and Silviu Oprea[3]

[1] Computer Science, Jacobs University Bremen, Germany
[2] National Institute of Standards and Technology, Gaithersburg, MD, USA
[3] University of Oxford, Oxford, UK.

Abstract. Marked by the highlights of native generation of EPUB E-books and `TikZ` support for creating SVG images, we present an annual report of LaTeXML development in 2013. LaTeXML provides a reimplementation of the TeX parser, geared towards preserving macro semantics; it supports an array of output formats, notably HTML5, EPUB, XHTML and its own LaTeX-near XML.
Other highlights include enhancing performance when used inside high-throughput build systems, via incorporating a native ZIP archive workflow, as well as a simplified installation procedure that now allows to deploy LaTeXML as a cloud service. To this end, we also introduce an official plugin-based scheme for publishing new features that go beyond the core scope of LaTeXML, such as web services or unconventional post-processors.
The software suite has now migrated to GitHub and we welcome forks and patches from the wider FLOSS community.

1 Introduction

Another busy year of LaTeXML [Mil] development has gone by; while we've not completely accomplished all the tasks we'd hoped for (c.f. [GM13]), we've finished others including some we hadn't originally planned. While it was originally developed for NIST's Digital Library of Mathematical Functions[4], where it continues to serve, we continue to find additional applications. One, carried out this year, was the natural extension of the system to generate EPUB documents.

A move to GitHub along with the adoption of coding standards, and a reorganization should enhance the ability of the community to both contribute to the core software and extend it through a plugin architecture being developed. **@Bruce: isn't it already developed? We just need CPAN.**

2 Reorganization

We have reorganized both our code development and our code base. In the first sense, we have moved our repository to GitHub[5] where you can more conveniently browse our code, or obtain the latest version. We have also ported our

[4] see `http://dlmf.nist.gov`
[5] see `https://github.com/brucemiller/LaTeXML`

Trac ticket database to GitHub's Issues, so you can also report bugs and request features at the same place.

Along with the move to github came more opportunities to share code and development which called for clearer code standards. We have made a commitment to code quality and formatting by adopting `perltidy` and `perlcritic` policies, which were adapted to the polyglot context of TEX, Perl, XML, XSLT, and more, and which are automatically enforced by `git` mechanisms.

In the second sense, we have reorganized the code itself to more clearly separate the modules related to the separate phases of processing. At the same time, we enable "conversion as an API", offering a connection and code sharing between those phases when more complex processing is called for, such as carrying a single TEX source file through the full processing to HTML, or even EPUB (see 3). In particular, it provides better support for daemonized processing, foundational to batch conversions and web service deployments.

This reorganization positions us to develop a plugin architecture that will allow modular extensions that cover both new LATEX styles and bindings, but also include enhanced postprocessing for more sophisticated applications such as sTEX. The authors have already refactored the three flavors of LATEXML web servers, an alternative grammar for math parsing, as well as an extension for converting TEX formulas into queries for the MathWebSearch search engine, as 5 separate repositories, also hosted on GitHub. The true power of the new contribution model is revealed when combined with Perl's CPAN distribution and dependency management system, which will allow for single command installation of any LaTeXML-based project and its full dependency tree.

3 E-books

The newest version of EPUB, version 3, is primarily a packaging of HTML pages representing chapters or sections into a structured ZIP archive. The big step forward for the scientific community is that it now calls for the use of MathML to represent mathematics. Since LATEXML is already generating HTML, with embedded MathML, and allows that output to be split into multiple pages as specified by the user, it seemed an obvious and natural extension to generate EPUB documents. Moreover, the web-service spin-off projects had already called for and drafted the compression of the resulting directory of generated content into a ZIP archive. Thus, with appropriate rearrangment of the pieces, and the addition of a Manifest of the correct structure, we have all the basic components needed to generate EPUB documents. We have generated a number of EPUB documents and successfully validated them against the official `idpf` validator[6].

We subsequently considered to also add support for Amazon's proprietary `mobi` E-book format. However, at the time of writing the `mobi` ecosystem is transitioning to the new Amazon Kindle Format 8 (`AKF8`), which aims to more fully align with EPUB 3.0. Finally, the lack of an open ecosystem around the

[6] see `http://validator.idpf.org/`

format prevented us from repeating the quick and painless design process for the EPUB output, so we did not venture further.

4 Graphics

Before we turn our attention to graphics, a brief digression may be in order. There are two main approaches currently used to generate HTML from TeX. The first approach, exemplified by `tex4ht`, uses the actual TeX engine to process the source by redefining certain commands to drop `\special` data into the normal `dvi` output file. Instead of `dvips`, a special `dvi` processor then deciphers the `dvi` and `\specials` to infer and construct the appropriate HTML. In the second approach, used by LaTeXML, a program is developed which emulates TeX for the most part but interprets some macros (called "Constructors" in LaTeXML) specially, so that it produces XML directly.

The first approach has the advantage of (usually) allowing the processing of arbitrary TeX and LaTeX packages, although the resulting HTML may not reflect the intended structure nor semantics. The challenges are in the TeX programming necessary to insert the `\specials`, generating valid HTML, and in whether sufficient semantic structure can be recovered from the `dvi`.

The second approach has the advantage of having more direct control of the generated output. It is easier, though not trivial, to extend to new XML structures. Furthermore, LaTeXML uses an intermediate XML format which preserves the semantic structure. It is fundamentally XML aware, so it produces valid XML. A feature of LaTeXML bindings is that macro control sequences can be defined to be "Constructors" which directly construct the XML representation of their content. The challenge, of course, is to emulate TeX sufficiently well to process complex packages, or alternatively, to develop LaTeXML-specific bindings for them.

In either approach, LaTeX packages that define macros with semantic intent must be dealt with individually or else the semantics will be lost.

Within that context, we were skeptical when Michael Kohlhase initially posed the proposition: Was LaTeXML's engine good enough to implement the `TikZ` package and generate SVG? Presumably any semantics implied by `TikZ` markup isn't so critical. The package is so large and complex, not to mention its development so fast-moving, that creating LaTeXML-specific bindings for all its commands impractical. However, it is designed to pass all processed graphics through a relatively small driver layer, and even has a `tex4ht` driver for producing SVG!

The main tasks, then, were to implement LaTeXML bindings for that driver and improve LaTeXML's engine to cope with the sophisticated TeX macro usage in the higher layers of `pgf` and `TikZ`.

Ultimately, we succeeded beyond our expectations. Although the results are not perfect, LaTeXML now successfully processes 3/4 of the first page[7] of `TikZ` examples on the TeXample.net website, generating valid HTML5, with text and

[7] see `http://www.texample.net/tikz/examples/all/`

MathML combined. In contrast, `tex4ht` succeeds on slightly more than half the examples, often producing invalid markup, and doesn't support MathML embedded in the SVG. It must be admitted, however, that LaTeXML is *very* slow at processing `TikZ` markup!

In the process, we have further improved the fidelity of the TeX emulation, introduced a (currently very rudimentary) mechanism for estimating the size of displayed objects and exercised the integration of both MathML and SVG into HTML. Additionally, LaTeXML now has its own TeX profiler, which offers binding developers per-macro feedback on inclusive and exclusive runtimes, helping to identify core conversion bottlenecks.(**@BRUCE: Is this accurate?**) These improvements are beneficial even outside the graphics milestone and contribute to an overall better LaTeXML ecosystem.

Areas needing further work are `TikZ`' matrix structure which currently clashes with LaTeXML's handling of alignments; inaccuracies of LaTeXML's sizing of objects; and, of course, examples involving other exotic packages not yet known to LaTeXML. We plan to test against the entire suite of examples at TeXample.net to discover other weaknesses and further improve the module.

Beyond `TikZ`, we are hoping to leverage this experience and apply it to supporting the `xy` package, another popular and powerful system. It seems to have a less well-defined driver layer and we are in the early stages of discovering the smallest set of macros that could serve that function. Nevertheless, we have had some preliminary, proof-of-concept, success. We already have minimal support, for the `pstricks` package, but with its Postscript oriented design, it is more time consuming to develop further bindings.

5 Outlook

The initial success with `TikZ` processing is quite gratifying, but it needs refinement, and look forward to testing on a larger scale. We also intend to extend our reach to the `xy` packages. Other E-book formats such as `AKF8` should be possible with specializations of manifest generation and other fine tuning. Surprisingly, generating Word and OpenOffice formats shares many features with E-books; of course finding the documentation and writing the XSLT transformations from LaTeXML's native XML to Word's will be challenging.

Our move to GitHub, the code reorganization and the plugin contribution model should make it easier for users to use and adapt the system, not to mention contributing back patches and improvements that will help our developement.

References

[GM13] Deyan Ginev and Bruce Miller. "LaTeXML 2012 - A Year of LaTeXML". In: *Intelligent Computer Mathematics.* Conferences on Intelligent Computer Mathematics. (Bath, UK, July 8–12, 2013). Ed. by Jacques Carette et al. Lecture Notes in Computer Science 7961. Springer, 2013, pp. 335–338. ISBN: 978-3-642-39319-8. DOI: `10.1007/978-3-642-39320-4`.

[Mil] Bruce Miller. *LaTeXML: A LaTeX to XML Converter*. URL: `http://dlmf.nist.gov/LaTeXML/` (visited on Mar. 12, 2013).