BLAHTEXML and multi-target document generation*

Gilles Van Assche

November 27, 2010

Abstract

BLAHTEX and BLAHTEXML are open-source tools for converting mathematical expressions written in the TEX syntax into MathML. This article focuses on a particular use case, where the source of a scientific document is written in XML and can be the input for a variety of output formats, ranging from LATEX articles to documents in OpenDocument format to web pages. We show that BLAHTEXML can play a central role in such a context, where the author wishes to enter equations in the TEX syntax and yet enable his document for publication not only with TEX but also in MathML-based formats.

Typing a mathematical expression using the syntax of TEX is much more convenient than in the MathML syntax. In fact, the latter was not designed to be typed by hand, but instead to be entered in a MathML editor or converted from another format. Yet, MathML is being adopted by an increasing number of programs and utilities, especially in browsers to display pages with formulas on the Web. To be able to use MathML while retaining the convenience of the TEX syntax, BLAHTEX(ML) provide a way to convert mathematical formulas from the syntax of TEX (or a large subset of it) to MathML [9].

BLAHTEXML differs from BLAHTEX in that it adds the ability to convert all TEX formulas in an XML file to MathML. The idea behind this new functionality stems from a specific use case of BLAHTEXML: the generation of documents in multiple formats from a single source [17]. This article focuses on a particular use case, where a document is written in XML and becomes the source for a variety of output formats, ranging from LATEX articles to documents in OpenDocument format to web pages. This approach is not new—actually, it is a fairly natural one—yet this article points out that BLAHTEXML fits nicely in the picture when it comes to scientific documents and papers.

The rest of the paper is organized as follows. First, we give an overview of XML technologies for scientific documents in Section 1. Then, Section 2 describes the functionality of BLAHTEX. The single-source approach for scientific documents is the content of Section 3, including information on BLAHTEXML in Section 3.1, XSLT in Section 3.2 and finally an example in Section 3.3.

1 XML technologies

The Extensible Markup Language, or XML, has become a popular way to express the content and structure of a document [3]. XML defines a generic syntax for enriching texts (or data) with humanly-readable tags. Alone, XML is hollow—it does not define the meaning of tags, nor how

*This article appears in **The Zpravodaj of the Czechoslovak TEX Users Group** [2]. The text of this article is licensed under the Attribution-NoDerivs Creative Commons license.

to process an XML document. Instead, it can be viewed as a common ground for applications that share a single syntax and a lot of standard tools to generate, query, transform and edit data or documents in a unified way. For instance, the Extensible Stylesheet Language Transformations (XSLT) language is an efficient way to generate XML documents or to transform one XML file into another [4].

An XML application is a restriction of the XML syntax to a well-defined set of tags and other conventions. Anyone is free to define his/her own XML application. As of interest for scientific documents, there are at least three XML applications that are important to mention: XHTML, the OpenDocument format and MathML. First, XHTML is an XML version of the famous HyperText Markup Language (HTML) that describes the content of a web page [5]. Retro-compatible with HTML, XHTML is a clean version of HTML that follows the XML syntax and consequently allows to use all the XML tools. Second, the OpenDocument format uses a container format (as a Zip file) that embeds XML files for the content and style information of the document [7]. Finally, MathML is an XML application that describes mathematical expressions [6]. It encodes the structure of such expressions in a standard way, so that software can display or process them.

MathML is used for embedded formulas in several applications, including XHTML and OpenDocument. For instance, MathML formulas can be included in XHTML web pages. Traditionally, mathematical expressions have been included as bitmap pictures—this is a solution with many drawbacks (e.g., poor, non-scalable display quality, increased load time), but of course one that works for all browsers. Formulas in MathML, on the contrary, provides a better alternative, which is supported by an increasing number of software, including many recent browsers (e.g., Firefox [11], Design Science's MathPlayer plug-in [13] for Microsoft Internet Explorer).

2 BLAHTEX

While MathML is becoming a universal way to express and exchange mathematical expressions, its syntax is extremely verbose, preventing the most courageous user from entering an equation of reasonable size by hand in a text editor. In fact, it is not the purpose of MathML for one to be able to actually type a formula in this syntax. Instead, there are interactive editors or converters to do so.

Unlike MathML, the syntax of mathematical expressions in TEX is the de-facto standard in the scientific community and is simple enough to be entered by hand. This is where BLAHTEX comes into play: It allows one to enter formulas using the syntax of TEX and to convert them into MathML.

BLAHTEX was written by David Harvey, who targeted his program to support equations in MediaWiki, the engine behind Wikipedia [8]. In this context, writers enter text in a rather simple syntax called *wiki text* and MediaWiki generates the HTML code to be displayed in a browser. To keep the syntax simple, writers are allowed to enter equations in the TEX syntax. Currently, texvc converts the mathematical formulas of Wikipedia to either HTML or PNG bitmaps [16]. As an alternative, a MediaWiki extension using BLAHTEX is able to convert each of these into MathML [10]. Like texvc, BLAHTEX processes each equation individually.

The syntax supported by BLAHTEX is a subset of the TEX syntax, but the chosen subset is large enough for most purposes. For instance, it supports a long list of symbols, commands and environments compatible with TEX, LATEX and AMS-LATEX, as well as macros via `\newcommand`. The complete list can be found in the user manual [9].

Internally, BLAHTEX processes everything as Unicode, from the Greek letters to mathematical operators to text in languages other than English. As a convenient extension to the TEX syntax,

```
<blahtex>
<mathml>
<markup>
<msqrt>
  <msup>
    <mi>x</mi>
    <mn>2</mn>
  </msup>
  <mo lspace="0.222em" rspace="0.222em">+</mo>
  <mi>&#x3B1;</mi>
</msqrt>
</markup>
</mathml>
</blahtex>
```

Figure 1: Sample MathML output provided by BLAHTEX

```
<blahtex>
<png>
<md5>068bd5f892d1f87b0371fa570af10712</md5>
</png>
</blahtex>
```

Figure 2: Sample PNG file name output

BLAHTEX accepts a number of mathematical symbols to be directly entered in Unicode as an alias to the TEX command. E.g., BLAHTEX makes no difference between the multiplication sign "×" entered as is and the `\times` command.

A nice thing about BLAHTEX is that it makes a good attempt at providing the same spacing between operators as TEX does. It determines the proper spacing and provides it in the generated MathML code as `lspace` and `rspace` attributes. Although the rendering of MathML varies from browser to browser, this helps getting a consistent look, as close as possible to TEX's appearance.

We now illustrate the use of BLAHTEX through some examples.

The first way to use BLAHTEX, with the `--mathml` option, is to convert an equation given at standard input into MathML at standard output. For instance, typing: `echo '\sqrt{x^2+\alpha}' | blahtex --mathml` produces the output in Figure 1. In this example, the MathML fragment is enclosed in `blahtex/mathml/markup`. Note that the MathML fragment produced does not contain any namespace information; ideally, the MathML namespace should be added when enclosing this fragment in an actual XML file. In the case of a syntax error, explicit information is given in a `blahtex/error` element.

The second way to use BLAHTEX, with the `--png` option, is to convert an equation into a PNG file. BLAHTEX calls TEX to produce this bitmap picture. The name of the output file is automatically generated from the MD5 digest of the TEX code. Hence, if the same formula appears several times, only one PNG file is produced. To be able to determine the name of the PNG file, the digest is provided in the `blahtex/png/md5` element of the XML fragment at the standard output. For instance, typing `echo '\sqrt{x^2+\alpha}' | blahtex --png` produces the file `068bd5f892d1f87b0371fa570af10712.png` displaying $\sqrt{x^2+\alpha}$ and the XML fragment of Figure 2.

```
<?xml version="1.0"?>
<equations xmlns:b="http://gva.noekeon.org/blahtexml">
 <equation b:inline="x+y"/>
 <equation b:block="\exp(-\gamma x)"/>
</equations>
```

Figure 3: Sample input file for BLAHTEXML

3 Single-source approach for scientific documents

When writing a scientific document, the writer wishes to concentrate on the content and not worry too much about the technical details of the typesetting system. The purpose of LaTeX, as a layer on top of TeX, is indeed to provide separation between content and presentation. However, it does not forbid the writer to enter specific commands to control details of some presentation aspects, as the need naturally arises in practice. Also, one often has a predetermined target in mind for a document (e.g., an article for a specific journal, a report, a thesis) when writing it. Having specific presentation requirements (e.g., the journal's layout) is not a problem for a single document. However, if one wishes to re-use material between various documents, a simple copy & paste may not be enough: Some presentation-oriented commands need to be adapted as the layout conventions for different targets may not be identical. For instance, different LaTeX class files may have slightly different syntaxes. To enter the abstract of an article, one may require to enclose it in a `\abstract` command, while others require it in an environment delimited by `\begin{abstract}` and `\end{abstract}`. As another example, the highest heading level of an article is `\section`, while it is `\chapter` for a report. Moving a section to another document or to another level may require adapting all the heading commands.

Presentation-oriented commands may become even more problematic when the output format is not LaTeX's original target (i.e., a Postscript or PDF file) but, say, a web page. It would be excessive for a converter from LaTeX to HTML to support all the presentation-oriented aspects of the document. At least some of them do not make sense at all, such as the page format, whilst others might just be very difficult to convert.

While there is no miracle solution to these problems, we think that the best solution is to generate different output formats from a source file in a common syntax. The common syntax may or may not be related to one of the output formats. The point is, however, that the common syntax should focus on the content and that, if necessary, some common presentation aspects can be added to it, provided that it does not privilege or exclude one of the output formats specifically.

3.1 Using BLAHTEXML

The idea of a common syntax naturally extends to the mathematical expressions, which can then be converted into an appropriate set of formats, depending on the target output format. This is where BLAHTEXML comes into play. Assuming a document written in a syntax based on XML, BLAHTEXML converts each equation found in the document into either MathML, nominal TeX syntax, PNG bitmap image files, or all three formats. The syntax of BLAHTEX(ML) is indeed TeX-oriented. Yet, the subset supported by BLAHTEX(ML) excludes TeX-specific presentation aspects that could not be converted into MathML.

BLAHTEXML provides the `--xmlin` option, which does not exist in BLAHTEX. With this option, BLAHTEXML processes an input file given at standard input. Such an input file may look like the example of Figure 3. The equations are given as attributes (`inline` or `block`) in the BLAHTEXML

```
<?xml version="1.0" encoding="UTF-8"?>
<equations xmlns:b="http://gva.noekeon.org/blahtexml">
 <equation>
  <math xmlns="http://www.w3.org/1998/Math/MathML">
   <mi>x</mi>
   <mo lspace="0.222em" rspace="0.222em">+</mo>
   <mi>y</mi>
  </math>
 </equation>
 <equation>
  <math xmlns="http://www.w3.org/1998/Math/MathML" display="block">
   <mi>exp</mi>
   <mo lspace="0" rspace="0" stretchy="false">(</mo>
   <mo lspace="0" rspace="0">-</mo>
   <mi>&#x3B3;</mi>
   <mspace width="0"></mspace>
   <mi>x</mi>
   <mo lspace="0" rspace="0" stretchy="false">)</mo>
  </math>
 </equation>
</equations>
```

Figure 4: The output file given by BLAHTEXML for the input file in Figure 3

namespace. Whenever BLAHTEXML meets such an equation, it expands it into the equivalent MathML code. The corresponding output is given in Figure 4. Note that by using a namespace, attributes containing equations can be added to any XML file independently of the syntax of other applications.

In addition to the MathML representation of the equations, the `--annotate-TeX` and `--annotate-PNG` options cause BLAHTEXML to produces an `annotation` element with the equation in nominal TEX syntax and another `annotation` element with the name of the PNG file containing a bitmap rendering. The generated MathML code and both new elements are enclosed in a `semantics` element, to conform to the MathML syntax. From the same example as above, this would generate the output of Figure 5.

3.2 Using XSLT

In document generation from a source in a common syntax, the source file of a document must be parsed before contents in the target format can be generated. Restricting the common syntax to an XML application, parsing XML can be done with various tools or can be programmed in different languages with appropriate libraries. Among the available tools, the XSLT language is particularly well suited for transforming an XML source file into either another XML file or a text file. Let us briefly introduce this tool and explain why it is well suited to our particular use case.

In XSLT, a *stylesheet* defines a transformation from XML into either XML or text. In its simplest form, it is a declarative language that specifies the piece of text or XML data to generate when it encounters a given XML tag in the source file. To apply a given stylesheet to a source file, one uses an *XSLT processor*.

XSLT can become a bit complex when the task to perform diverges from its core abilities. However, in the context of multi-target document generation, XSLT is simple to program and to read.

```
<equations xmlns:b="http://gva.noekeon.org/blahtexml">
 <equation>
  <math xmlns="http://www.w3.org/1998/Math/MathML">
   <semantics>
    <mrow>[...]</mrow>
    <annotation encoding="TeX">x + y</annotation>
    <annotation encoding="image-file-PNG">
      ./f05c46190061a618fd432bf5471cc2ab.png</annotation>
   </semantics>
  </math>
 </equation>
 <equation>
  <math xmlns="http://www.w3.org/1998/Math/MathML" display="block">
   <semantics>
    <mrow>[...]</mrow>
    <annotation encoding="TeX">\exp ( - \gamma x )</annotation>
    <annotation encoding="image-file-PNG">
      ./df6bfcabef19b8a0ccdbd2077ae96e75.png</annotation>
   </semantics>
  </math>
 </equation>
</equations>
```

Figure 5: The output file given by BLAHTEXML for the input file in Figure 3 when additional annotations are requested

```
<xsl:template match="b">
  <xsl:text>\textbf{</xsl:text>
  <xsl:apply-templates/>
  </xsl:text>}</xsl:text>
</xsl:template>
```

Figure 6: Example of XSLT code to convert the bold `b` tag of XHTML to the `textbf` command in LaTeX

For instance, no explicit loops need to be written to go through the entire source file, as such loops are managed by the XSLT processor automatically. This reduces the work to writing the text or XML fragment to be generated corresponding to a given input XML element.

As a brief example, let us consider the conversion from XHTML to LaTeX using XSLT. The XHTML tag `b` indicates bold text. The equivalent LaTeX command would be `\textbf`. The piece of code in Figure 6 makes this conversion: It declares a template, which matches `b` tags. For all such tags, it then tells to output `\textbf{`, then to apply recursively other templates, e.g., to convert other tags or simply to write the text inside the `b` tag, and finally it concludes by outputting the closing brace `}`.

3.3 A simple example based on XML

On the BLAHTEXML web page, we provide an example of document generation system based on an XML syntax [1]. This is a working example, although with a reasonably simple functionality. The goal is not to rival with well-known systems, such as DocBook [15], with its definition of

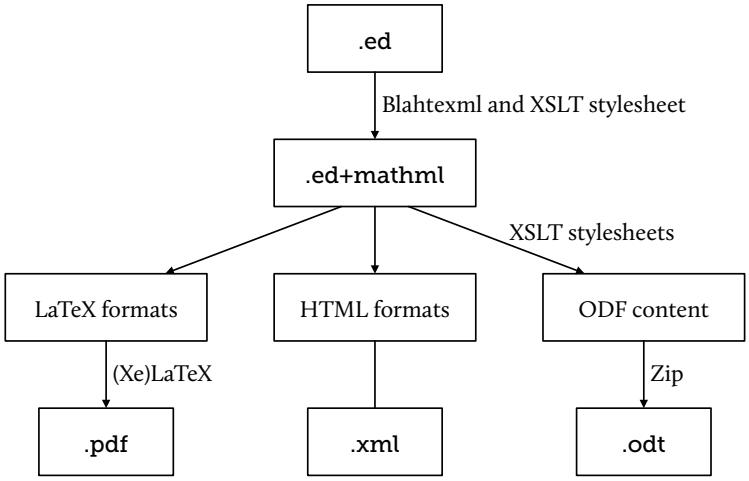

Figure 7: The general flow

almost 400 different tags. Instead, this working example proposes a clean and simple syntax, whose only ambition is to illustrate the use of BLAHTEXML for multi-target document generation in the scope of scientific documents and articles.

The proposed example is based only on open-source technologies: The general process is managed by `make` and the XSLT processing is performed by any XSLT processor. In the example, the processor used is `xsltproc` [14], although any XSLT processor could be used.

Let us briefly describe the syntax of the source file and then the process from the source file to a target output. The source file is a document in XML, which contains the text, the structure of the document and some meta-information. The input syntax is illustrated in the file `Sample.ed`, which contains some sample text and mathematical expressions. The root element of the XML file is `document`. In it, two child elements appear: `head` and `body`. In the former, information about the author(s), their affiliation and the title can be provided. The latter provides the contents and structure of the document.

The structure of the file was inspired from XHTML 2.0. Text paragraphs can be grouped in sections using the `section` element. Such sections can be nested, which mean they actually represent a chapter, a section or a subsection depending on the nesting depth. Section titles are provided in `h` elements. Text paragraphs are enclosed in `p` elements, and ordered and unordered lists in `ol` and `ul`, respectively, with each list item in `li`. Inside paragraphs or list items, plain text can be given. The text can be further formatted using emphasis (italic, `em`), a strong font (bold, `strong`), small capitals (`sc`), subscript (`sub`) and superscript (`sup`).

As of interest for BLAHTEXML specifically, inline mathematical expressions are written in `ieq` elements, and stand-alone formulas in `eq` elements. Inside such elements, the formula is given in BLAHTEX format.

The general processing flow is illustrate in Figure 7. To determine the sequence of steps from the source file to the output file, a `makefile` is provided. Depending on the target format, the following steps can be taken.

- As a the mathematical expressions are not written as attributes (but more simply inside `ieq` and `eq` elements), a first step consists in putting the equations in the appropriate attributes for BLAHTEXML. This preparation step is performed by the `PrepareForBlahtexml.xsl` XSLT

stylesheet.

- As a result of the previous step, the mathematical expressions are written as attributes in the BLAHTEXML namespace. This step now consists in converting these formulas using BLAHTEXML with the `--annotate-TeX` and `--annotate-PNG` options. As a result, all formulas are in three formats: MathML, TEX and as PNG files. Depending on the desired output format, the following steps will extract the format they need.
- Then, the XSLT stylesheets `Numbering.xsl` and `Referencing.xsl` process the resulting file to number sections and to resolve cross-references. This step is mainly done for XHTML output, as LATEX and OpenDocument formats have their own syntax to express numbered sections and references.
- The core of the output generation is performed by a format-specific XSLT stylesheet to produce XHTML, LATEX or OpenDocument code. More details on the various output formats are given below.
- Optionally, a last step finalizes the production and again depends on the desired output format. For instance, for a `.tex` file, LATEX (or XELATEX) is invoked to produce a PDF file. If the target format is OpenDocument, then the resulting XML file is packaged into a Zip file and renamed as `.odt`.

Let us give some more details about the generation of the possible output formats. To allow `make` to determine which sequence of operations to perform, the different output formats have specific extensions. For instance, to produce a PDF file from `Sample.ed` via LATEX using an IEEE class file, one has to type `make Sample.ieee.latex.pdf`. We will see the other extensions as we go.

For XHTML, the generation of the various tags is fairly straightforward, since to an element of our input syntax corresponds an element in XHTML. This part of the job is done by the `ToXHTML-common.xsl` stylesheet. Details about the display styles can be tuned via the `document.css` cascaded stylesheet. There are three flavors of XHTML output formats, depending on the way the mathematical expressions are handled.

- For equations in MathML, the extension is `.xhtmlmathml.xml` (e.g., `make Sample.xhtml-mathml.xml`).
- As a first alternate option for browsers that have no MathML support, the mathematical expressions can be displayed as bitmap pictures, using the PNG files produced earlier. For this, the extension is `.xhtmlpng.xml`.
- As a second alternate option, the mathematical expressions can be displayed with pure HTML tags, but in a rather approximate form. For instance, HTML can display text in superscript and subscripts, but if an expression (like $A_{\text{sub}}^{\text{sup}}$) requires both then the HTML code will not be able to put one above the other (e.g., the result might look like $A_{\text{sub}}{}^{\text{sup}}$). Other restrictions apply, for instance, for two-dimensional constructions such as matrices or fractions. Nevertheless, this option may be useful and sufficient if the formulas are simple. Here, the extension is `.xhtmlapprox.xml`.

For TEX and derivatives, there are also several flavors. In the provided example, the output is either LATEX-oriented or XELATEX-oriented. The latter has the advantage of an easy support of True Type and Open Type fonts. Here the XSLT stylesheet must output a text file that follows TEX's syntax conventions. The main part of the job is done by the `ToLaTeX-common.xsl` stylesheet. Then, a number of smaller XSLT stylesheets give specific generation rules, most notably a specific header, for each flavor. The flavors supported in this example are the following.

- For a simple article in LaTeX, the specific stylesheet is ToLaTeX-article.xsl and the extension is .article.latex.tex (e.g., make Sample.article.latex.tex) for the .tex source file. To get the result directly as a PDF file, the .tex extension can be replaced by .pdf (e.g., make Sample.article.latex.pdf).
- For an article following the APS Physical Review conventions and using the revtex4-1 class file, the stylesheet is ToLaTeX-revtex.xsl and the extension is .revtex.latex.tex.
- For an article using the IEEEtran class file for the IEEE Transactions journals, the stylesheet is ToLaTeX-ieee.xsl and the extension is .ieee.latex.tex.
- For a simple article in XeLaTeX, the stylesheet is ToXeLaTeX-article.xsl and the extension is .article.xelatex.tex.

Adding a new flavor tailored to special needs is rather simple, as it suffices to add a new rule in the makefile and a new XSLT stylesheet based on one of the models above. Most of the specific stylesheets just define an alternate LaTeX file header.

Finally, for OpenDocument format, most of the job is done by the ToODT.xsl stylesheet. It produces a file called content.xml, which is then Zipped, together with the files provided in ODT-Template/, to make a .odt file. Here the target extension is simply .odt (e.g., make Sample.odt). Details about the display styles can be tuned in the ODT-Template/styles.xml file. The .odt file can be opened by any word processor supporting the standard OpenDocument format[1].

4 Conclusions

BLAHTEX(ML) can be useful for converting mathematical expressions written in the TeX syntax into MathML. In particular, we have shown that BLAHTEXML can perform this task in the scope of a multi-target document generation system for scientific documents. We have given an example to illustrate this purpose, where a document is written in a common XML-based syntax and various output formats can be generated from it, including various flavors of LaTeX.

Although fully working, the example given is rather simple from a functionality point of view. In this respect, future work may include the support for tables, figures, bibliographic entries and more output formats.

References

[1] G. Van Assche, *ExampleDoc*, http://gva.noekeon.org/blahtexml/exampledoc.

[2] ———, *Blahtexml and multi-target document generation*, The Zpravodaj of the Czechoslovak TeX Users Group (2012), no. 3, 137, http://bulletin.cstug.cz/.

[3] World Wide Web Consortium, *Extensible markup language (XML)*, http://www.w3.org/standards/xml/.

[4] ———, *Extensible stylesheet language transformation (XSLT)*, http://www.w3.org/standards/xml/transformation.

[1] At this time of writing, a bug in OpenOffice.org prevents the mathematical expressions from being displayed with a correct size after loading the document [12]. A possible workaround consists in double-clicking on the equations to open them in the integrated equation editor, which forces OpenOffice.org to resize the mathematical expressions appropriately. We hope this issue will be solved soon.

[5] ———, *HTML & CSS*, http://www.w3.org/standards/webdesign/htmlcss.

[6] ———, *Mathematical markup language (MathML)*, http://www.w3.org/standards/webdesign/math.

[7] Organization for the Advancement of Structured Information Standards, *Open document format for office applications (OpenDocument)*, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=office.

[8] Wikimedia Foundation, *MediaWiki*, http://www.mediawiki.org/.

[9] D. Harvey and G. Van Assche, *Blahtex and blahtexml version 0.8 manual*, http://gva.noekeon.org/blahtexml/.

[10] Mediawiki, *Extension:Blahtex*, http://www.mediawiki.org/wiki/Extension:Blahtex.

[11] Mozilla, *Firefox*, http://www.firefox.com.

[12] OpenOffice.org, *Issue 91779*, http://www.openoffice.org/issues/show_bug.cgi?id=91779.

[13] Design Science, *MathPlayer plug-in*, http://www.dessci.com/en/products/mathplayer/.

[14] D. Veillard, *The xsltproc tool*, http://xmlsoft.org/XSLT/xsltproc2.html.

[15] N. Walsh, *DocBook 5: The definitive guide*, O'Reilly, 2010.

[16] T. Wegrzanowski, *Texvc*, http://en.wikipedia.org/wiki/Texvc.

[17] Wikipedia, *Single source publishing*, http://en.wikipedia.org/wiki/Single_source_publishing.