System Description: A Semantics-Aware LaTeX-to-Office Converter

Lukas Kohlhase and Michael Kohlhase

Mathematics/Computer Science
Jacobs University Bremen

Abstract. We present a LaTeX-to-Office conversion plugin for LaTeXML that can bridge the divide between publication practices in the theoretical disciplines (LaTeX) and the applied ones (predominantly Office). The advantage of this plugin over other converters is that LaTeXML conserves enough of the document- and formula structure, that the transformed structures can be edited and processed further.

1 Problem & State of the Art

Technical documents from the STEM fields (Science, Technology, Engineering, and Mathematics) augment the text with structured objects – images, mathematical/chemical formulae, diagrams, and tables – that carry essential parts of the information. There are two camps with different techniques for authoring documents. The more theoretical disciplines (Mathematics, Physics, and Computer Science) prefer LaTeX, while the more applied ones (e.g. Life Sciences, Chemistry, Engineering) use Office Suites almost exclusively. Transforming between these two document formatting approaches is non-trivial: The TeX/LaTeX paradigm relies on in-document macros to "program" documents, empowering authors to automate document aspects and leading to community-supplied domain-specific extensions via LaTeX packages. Office suites rely on document styles that adapt visual parameters of the underlying document markup either document-wide or for individual elements.

This incompatibility of document preparation approaches causes friction in cross-paradigm collaboration as each camp deems their approach vastly superior and the other's insufferable. In this paper, we will discuss the transformation from TeX/LaTeX to Office documents. The converse direction would also be useful, but uses different methods.

copy from PDF	paste (libreoffice)
$h_{\mu_\varphi}(f) + \int_X \varphi d\mu_\varphi = \sup_{\mathcal{M}(f,X)} \{h_\mu(f) + \int_X \varphi d\mu\},$	h_{μ_ϕ}(f)+□ ϕdμ_ϕ = sup {h_μ(f)+□ ϕdμ},

Fig. 1. Copy & Paste into Word Processors

There are several methods to transform papers from LaTeX to an Office word processor. The first method is to just generate a PDF file and then open this file in Word/LibreOffice or copy/paste a fragment. This achieves the goal of looking like the desired PDF document, just in Office. There are two problems with this route: *i*) mathematical formulae are not preserved (see Figure **??**) *ii*) even if the result looks OK the results have lost their links (e.g. for citations/references or label/ref), or become difficult to edit, because they do not conform to the styling system of the word processor. The fundamental problem is that this process converts only the appearance of the document and loses all meaning that was encoded in the markup macros that were expanded during PDF generation. This is especially blatant when looking at the math in a document, which is either treated as text or images and cannot be edited/processed further. The same holds true for references, they are essentially treated as parts of text with a linked number in front of them, complicating adding new references substantially.

The other way of transforming LaTeX to Word, by transforming the LaTeX source file directly, avoids these problems. `latex2rtf` [**latex2rtf:on**] is a widely used system that uses a custom parser to convert a non-trivial fragment of LaTeX to the RTF format understood by most office systems. The system works well, but coverage is limited by the LaTeX parser and the aging RTF format. TeX4ht [**tex4ht:online**], which uses the TeX parser itself and seeds the output with custom directives that are parsed to create HTML has a post-processor that generates ODF. Its coverage of LaTeX is unlimited, but the intermediate format HTML somewhat limits the range of document fragments that can be generated.

Here we present a similar approach, only that we extend the backend of the LaTeXML system [**Miller:latexml:online**] to generate WML – the file format of MS Word – and ODT – that of Libre- and OpenOffice. Like `latex2rtf`, the LaTeXML system directly parses LaTeX source files. The main difference to TeX4ht is that LaTeXML generates an XML representation that is structurally near to the LaTeX sources and thus preserves the author-supplied semantics for further processing. Coverage for TeX primitives is complete, semantics-preserving LaTeXML bindings are available for most commonly used LaTeX packages.

2 The Office Formats

WML and ODT follow the same architectural paradigm: they are both zip-packaged directories of XML files that contain document content, metadata, and styling. We will use WML in our presentation here and point out differences in ODT as we go along.

The main content of a WML document – text, document structure, placement of images, tables etc. – is represented by special content markup elements in an XML file `document.xml`. All elements contain styling information, usually by referencing a `style` element in the file `style.xml`, which can be modified by adding local settings in children of the `properties` child. The other important kind of file are the `.rels` files, which are again XML. These files contain `relationship` elements, which detail the relations between elements in `document.xml` and external resources (e.g. for hyperlinks) or resources in the WML package (e.g. the image data files). The WML package additionally contains miscellaneous XML files; e.g. `settings.xml`, which is used to make the state of the word processor applications persistent and `fonttable.xml`, which contains extra information about fonts.

Of special interest is the representation of mathematical formulae. WML uses a proprietary XML format for presentation markup together with a variant of TeX markup that is used for user input. The expression of the left is the –slightly abridged – representation of 1.5×10^7. The ODT format treats formulae as external objects; every single one has a subdirectory in the package which contains a presentation MathML file (for external communication), a user input file in the venerable StarOffice format, and an image of the formula (for display in the word processor).

```
<omml:oMath>
  <r><t>1.5</t></r>
  <sSup>
    <e><r><t>10</t></r></e>
    <sup><r><t>7</t></r></sup>
  </sSup>
  1.5\times 10^{7}
</omml:oMath>
```

3 Transformation

To create the WML/ODT files we first transform the `.tex` file to an intermediate XML-based LTXML format using LaTeXML. Then we use an XSLT stylesheet to generate `document.xml`. For LTXML elements that do not have a direct counterpart in WML we adapt existing WML elements. For instance, a LaTeX `quote` environment is represented by a WML `p` ("paragraph") element with a special style "`quote`" we added to `styles.xml`. This allows the user to later semantically work with the document, e.g. by changing all quotes to red. For WML formulae, we use a stylesheet supplied by Microsoft to transform the MathML generated by LaTeXML to the WML math format, for ODT formulae we make use of MathML and image generation in LaTeXML. The file `document.xml.rels` is generated by XSLT from `.tex.xml` and is placed into the directory structure the by the LaTeXML post-processor together with other supporting files such as images and some static files that are independent of the input document. The main file of interest here is `styles.xml`, which contains the style information of the document. This had to be adapted manually recreate the visual appearance of the PDF files generated by LaTeX. Finally the LaTeXML post-processor zips documents into the final WML/ODT file.

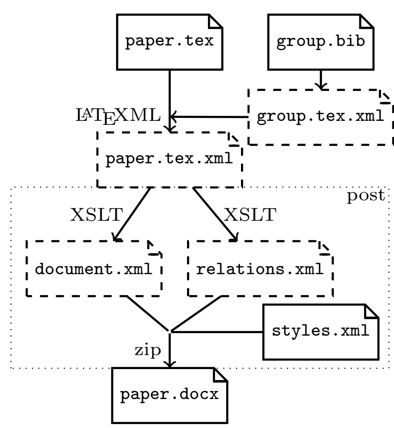

Fig. 2. The Transformation Process

The user does not see all these transformation, generation, and packaging steps: given a LaTeX paper, all she has to do is type

```
latexmlc paper.tex --destination=paper.docx
```

A transformation to ODT can be specified by choosing the destination `paper.odt`.

4 Conclusion

$$h_{\mu_\varphi}(f) + \int_X \varphi d\mu_\varphi = \sup_{M(f,X)} \{h_\mu(f) + \int_X \varphi d\mu\},$$

Fig. 3. Converted Formula in MS Word

We have presented a LaTeXML plugin that transforms LaTeX papers into Office documents in a one-line system call. The result of converting the formula from Figure **??** to MS Word is on the right. With the recent web front-end of LaTeXML, it will be simple to extend this to a web service. The LaTeXML Word Processing plugin is public domain and is available from GitHub at [**LaTeX2Office:github:on**]. The conversion makes crucial use of the fact that LaTeXML preserves more of the document and formula semantics than other systems that process LaTeX documents, this ensures that the core process in the transformation – the translation of LaTeXML XML to Office XML (WML or ODF) has enough information to generate the respective target document structures. The biggest limitations of the current transformation are that *i*) we cannot currently generate the text-based input format (StarMath or the WML TeX variant) and *ii*) citations and references are only partially converted into the "semantic" formats. This makes it difficult to edit formulae/references in the respective word processors after transformation. For ODF formulae, we want to make use of the TeXMaths plugin for Libreoffice, which uses LaTeX instead of StarMath for user input of formulae – but hides it in the comment area of the images which makes handling more difficult.

In the future we want to develop an "office package" for LaTeXand a corresponding LaTeXML binding, which allows the direct markup of higher-level structures – e.g. document metadata in LaTeX documents, so that it can be transferred to the office documents. Similarly, we want to extend the transformation to carry over even more semantics from the sTeX format into semantically extended office formats like CPoint or CWord; this would finally give us a way to cleanly interface the currently LaTeX-based document methods in the KWARC group to applied STEM disciplines.