Mixing Surface Languages for OMDoc

Mihnea Iancu, Michael Kohlhase, Florian Rabe, and Hang Yuan

Jacobs University, Bremen, Germany

1 Introduction

OMDoc [Koh06] is a representation format for mathematical knowledge and documents (both formal and informal ones). As the XML serialization of OMDoc is very verbose, OMDoc is almost exclusively edited in the form of concise, human-oriented surface languages, which may moreover focus on sublanguages of OMDoc. At the moment, two such surface languages are used primarily: first, sTeX [Koh08; sTeX] is based on LaTeX and thus heavily optimized for informal mathematical content; second, the MMT surface syntax [Rab14] is based on logic and most suitable for formal mathematics.

But in many situations — e.g. for informal comments in formal developments or for partially formalized informal documents — neither of these surface languages fully fits the bill. We present an approach that allows flexibly mixing surface languages and implement an infrastructure for mixing the sTeX and MMT formats. Most notably, this includes combining the two existing processing workflows despite the enormous differences between them.

In the MathHub system [Ian+14], we have over 5000 semi-formal sTeX source files that can benefit from further formalization and about 70 larger files in the newer MMT format than can benefit from better documentation.

2 Mixing Surface Languages

We now show the concrete syntax of the mixing regime. The main technical device here is to define suitable escaping mechanism into other languages. This technique is well-known from programming, where it has been pioneered as "literate programming" [Knu92] and is used e.g. for software documentation generators like JavaDoc or Doxygen. Although related, our system is different. We do not generate several output formats – program and documentation – from a single source. Instead, to profit from better locality, we generate *one* XML (OMDoc) document from a document with *multiple* surface languages.

We use three mechanisms for escaping into another language depending on the kind of OMDoc elements that the escaped region includes into the final OMDoc document.

1. **block-level inclusions** for OMDoc statement and module level elements
2. **inline inclusions** for OMDoc terms
3. **references** for referring to (parts of) other-language fragments.

Then, any surface-language for OMDoc can define specific syntax for all (or only some) of the mechanisms above. We extended the MMT and sTeX languages accordingly and present the concrete syntax below.

2.1 MMT inside sTeX

For sTeX we define MMT-specific escapes as follows. For *block-level inclusions* we use the `MMT` environment, for *inline inclusions* we use the `\mmt` macro or the `|` separator[1], and for *referencing* we use the `\mmtref` macro. Figure 1 shows all three of them:

1. the block-level inclusion in lines 3-6 has two declarations, the first for a neutral element `neut` (presented as `e`) and an axiom `eunit` that states the property of neutral elements for `neut` – we are assuming the declarations `base : type # G` for the base set and `op` : $\mathtt{G} \to \mathtt{G} \to \mathtt{G}$ `# 1 + 2` from the theory `semigroup` here.
2. We see the two forms of inline inclusion in line 8 and one more in line 10. Note that the `\mmt` macro and its short form can be used in math and text modes.
3. The `\mmtref` macro in line 10 references the subterm at path `type.1.2` of the declaration `eunit` in module `monoid`. The referenced subterm is the first child of the `ded` application, then second of the `forall` binding (i.e. `x + e = x + e = e`) in the `type` component. The module is an optional argument which defaults to current module so is omitted in the listing below; the reference to `base` in line 9 comes from an imported module, so we need the optional argument `semigroup` for `\mmtref`.

```
 1  \begin{module}[id=monoid,meta=http://kwarc.info/test?FOL]
 2   \importmodule{semigroup}
 3   \begin{MMT}[axioms]
 4    neut : G # e
 5    eunit : ded (forall [x] x + e = x + e = e)
 6   \end{MMT}
 7   \begin{definition}
 8    Let $\mvstruct{|G|,\mmt{+}}$ be a \trefi[semigroup]{semigroup},
 9    then we call an element \mmtref{neut} in \mmtref[semigroup]{base}
10    a \defi{unit}, iff \mmtref{eunit}[type.1.2] for all $x$ in |G|.
11   \end{definition}
12  \end{module}
```

Fig. 1. MMT in sTeX: formal declarations

[1] Actually, `|` is only the default, any character ⟨⟨char⟩⟩ can be activated as a short separator for the embedded format ⟨⟨format⟩⟩ via `\MakeShortMix`{⟨⟨format⟩⟩}{⟨⟨char⟩⟩} and deactivated with `\DeleteShortMix`{⟨⟨char⟩⟩}. Currently, ⟨⟨format⟩⟩ is always `MMT`, but this approach naturally scales to additional formats

2.2 sTeX inside MMT

The most natural use case for using sTeX inside MMT is natural language comments. Consider for instance Figure 2, where we annotate a MMT theory of monoids with a definition of "neutral element" in sTeX. Here we only use block level inclusion with the start keyword `/sTeX` (the corresponding end is given by the MMT end block charater `^^`). From our current experience, the only use of embedding sTeX in MMT is for such comments and notation definitions, which are more expressive than the MMT notation definitions. But the example theory in Figure 2 shows that it is natural to nest MMT in sTeX in MMT– we only know of contrived examples of deeper nesting.

```
theory Monoid : http://cds.omdoc.org/testcases?Logic =
  unit  : tm ^_ # e ^^
  conn  : tm → tm → tm ^_ # 1 ∘ 2 prec 10 ^^
  conn_assoc : {X,Y,Z}  (X ∘ Y) ∘ Z = X ∘ (Y ∘ Z) ^^
  conn_neut : {X}  X ∘ e = X ^^
  /sTeX
  \begin{definition}
    We call an element |e| a \defii{neutral}{element} for a law
    of composition |conn|, if \mmtref{conn_neut}[type.1]
  \end{definition}
  ^^
^]
```

Fig. 2. sTeX in MMT: informal comments

3 Mixing OMDoc Generators

The *compilation* process for mixed-format source files works in incremental steps. First, it parses the top format producing a partial OMDoc document with cross-references – via OMDoc `<oref>` elements – for the snippets in other formats from the source file. Then, it *recurses* into processing those elements (if any) producing new partial documents and, finally, *merges* everything into one joint OMDoc document by resolving the cross-references.

Concretely, *parsing* a mixed-format file `x.f1` first produces a *partial* OMDoc file `x.f1.omdoc` with cross-references. Then, for each internal format `f2` we generate a source file `x.f1.f2` by *presenting* the OMDoc into the `f2` format. The process recurses for each new file to produce a complete OMDoc file which is then merged into the partial one above by replacing the cross-references with their corresponding generated OMDoc. See Figure 3 for an example compilation, where we use colors for different formats in source files and rounded corners for OMDoc documents.

To make this process work we give the escaped snippets machine-generated names and use them as identifiers for the cross-references. Then the corresponding elements in the auxiliary OMDoc documents are annotated with those identifiers which are later used for merging. For block-level inclusions we annotate by wrapping `omgroup` elements with the identifier as their name around the declarations that need to be transcluded. For inline inclusions we generate one symbol with the identifier as its name and the term as its definition. Finally, references use preexisting declarations so we just need to resolve them.

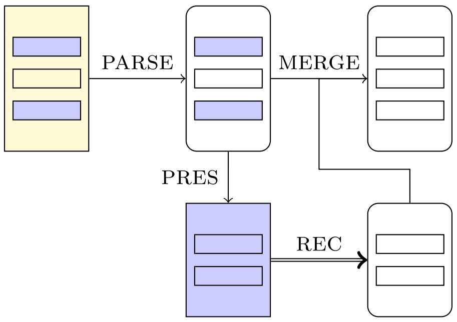

Fig. 3. Compilation Workflow

For the implementation we need, for each supported format, two things. First, extending the parser to support the dedicated syntax and produce the cross-references. Second, building a presenter into that format so that it can be processed when used inside other formats.

3.1 Compilation for sTeX and MMT

Figure 4 shows the process for generating OMDoc applied to the two mixed surface languages presented above in a mutually recursive diagram. On the left we have the "mso" processing path for MMT with sTeX inside and on the right its dual, the "smo" path, for sTeX with MMT inside. The dashed arrows constitute the base cases for the homogenous case and the double arrows the recursive calls.

MMT with sTeX   sTeX with MMT

$$\begin{tikzcd}
 & \mathtt{x.mmt} \arrow[d, "\text{MMT}"'] \arrow[dr, dashed, bend left, "\text{MMT}"] & & \mathtt{x.tex} \arrow[d, "\text{LaTeXML}"] \arrow[dl, dashed, bend right, "\text{LaTeXML}"'] & \\
\mathtt{x.mmt.tex} \arrow[d, Rightarrow, "\text{smo}"] & \mathtt{x.mmt.omdoc} \arrow[l] \arrow[dr] & & \mathtt{x.tex.omdoc} \arrow[r] \arrow[dl] & \mathtt{x.tex.mmt} \arrow[d, Rightarrow, "\text{mso}"'] \\
\mathtt{x.mmt.tex.omdoc} \arrow[rr, "\text{MMT}"'] & & \mathtt{x.omdoc} & & \mathtt{x.tex.mmt.omdoc} \arrow[ll, "\text{MMT}"]
\end{tikzcd}$$

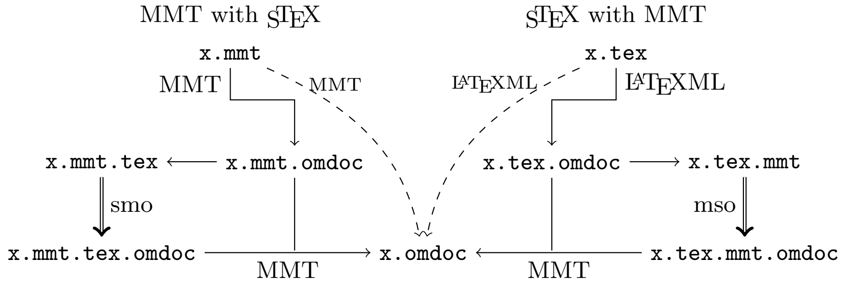

Fig. 4. Generating OMDoc from Mixed sTeX and MMT Sources

All the necessary sTeX extensions are implemented by providing LaTeXML bindings for the new macros and environments, and the corresponding XSLT stylesheets during the post-processing of LaTeXML transformation. The stylesheets have two primary functions, extracting the MMT elements into a MMT file and producing the cross-references. The enhancements for the MMT to OMDoc are analogous. With the explicit cross-reference information, the final MMT step just needs to dereference the names and paths involved and retract the cross-references.

The MMT code described above is available at `https://svn.kwarc.info/repos/MMT/src` in the `latex-mmt` and `stex-importer` projects. For LaTeXML,the code is at `https://github.com/KWARC/LaTeXML-Plugin-sTeX/tree/mmt`. Finally, the examples presented are available at `https://gl.kwarc.info/immt/papers/tree/master/mmt-stex/examples`.

4 Conclusion

Summary We have presented a simple but effective extension to the original two surface-to-base language compilation processes for OMDoc. Mixing surface languages allows to authors to utilize the respective strengths of the surface languages for flexiformal documents – documents of flexible forality; see [Koh13]. The next important step in this endeavor would be to similarly extend editors. There is (limited) editor support for sTeX [JK10] and MMT in JEdit [Rab14], but this language support would ideally change at the language borders as well to give the author full language support.

Generalizing To Other Surface Languages The mixing-by-interleaving-compilation approach is not restricted to OMDoc and the two surface languages in our experiment:

1. We have only used the admissibility of references in the surface and base languages and the ability of compilers to write auxiliary files.
2. Adding a surface language $\mathcal{L}$ increase the number and depth of paths in Figure 4 by one: each processing step with the $\mathcal{L}$-compiler eliminates the $\mathcal{L}$ snippets.

In fact, we have also experimented with a LaTeX-based workflow, where the end product is a PDF file (e.g. a scientific paper about MMT and sTeX). There we use MMT inclusions in regular LaTeX and let the MMT system present them. This is of less general use than the workflow presented in this paper.

Acknowledgements The authors gratefully acknowledge discussions with Deyan Ginev on an earlier PDF-oriented mixing process. We acknowledge financial support from the OpenDreamKit Horizon 2020 European Research Infrastructures project (#676541).

References

[Ian+14] Mihnea Iancu et al. "System Description: MathHub.info". In: *Intelligent Computer Mathematics 2014*. Conferences on Intelligent Computer Mathematics. (Coimbra, Portugal, July 7–11, 2014). Ed. by Stephan Watt et al. LNCS 8543. Springer, 2014, pp. 431–434. ISBN:

978-3-319-08433-6. URL: `http://kwarc.info/kohlhase/submit/cicm14-mathhub.pdf`.

[JK10] Constantin Jucovschi and Michael Kohlhase. "sTeXIDE: An Integrated Development Environment for sTeX Collections". In: *Intelligent Computer Mathematics*. Ed. by Serge Autexier et al. LNAI 6167. Springer Verlag, 2010. ISBN: 3642141277. URL: `http://kwarc.info/kohlhase/papers/mkm10-stexide.pdf`.

[Knu92] Donald E. Knuth. *Literate Programming*. The University of Chicago Press, 1992.

[Koh06] Michael Kohlhase. OMDoc – *An open markup format for mathematical documents [Version 1.2]*. LNAI 4180. Springer Verlag, Aug. 2006. URL: `http://omdoc.org/pubs/omdoc1.2.pdf`.

[Koh08] Michael Kohlhase. "Using LaTeX as a Semantic Markup Format". In: *Mathematics in Computer Science* 2.2 (2008), pp. 279–304. URL: `https://svn.kwarc.info/repos/stex/doc/mcs08/stex.pdf`.

[Koh13] Michael Kohlhase. "The Flexiformalist Manifesto". In: *14th International Workshop on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2012)*. Ed. by Andrei Voronkov et al. Timisoara, Romania: IEEE Press, 2013, pp. 30–36. ISBN: 978-1-4673-5026-6. URL: `http://kwarc.info/kohlhase/papers/synasc13.pdf`.

[Rab14] Florian Rabe. "A Logic-Independent IDE". In: *Workshop on User Interfaces for Theorem Provers*. Ed. by C. Benzmüller and B. Woltzenlogel Paleo. Elsevier, 2014, pp. 48–60. URL: `https://kwarc.info/people/frabe/Research/rabe_ui_14.pdf`.

[sTeX] *KWARC/sTeX*. URL: `https://github.com/KWARC/sTeX` (visited on 05/15/2015).