— sTeX Blue Note* —
Rethinking Modules and Semantic Macros in sTeX

Michael Kohlhase
Computer Science, Jacobs University.de

October 20, 2014

Abstract

In this note, we document the state of rethinking the sTeX infrastructure in terms of the SMGloM.

1 Introduction

We have been using sTeX as the encoding for the Semantic Multilingual Glossary of Mathematics (SMGloM; see [Gin+14]). The SMGloM data model has been taxing the representational capabilities of sTeX with respect to multilingual support and verbalization definitions; see [Koh14a], which we assume as background reading for this note. Multilinguality support has been started with in [KG14] and will (no longer) be covered in this note.

2 Mixed Presentation/Content Markup

Currently, sTeX produces content markup in the OpenMath encoding. But often sTeX formulae often contain bits of presentational LaTeX, which LaTeXML has to convert into OpenMath heuristically, which often leads to non-optimal results. Therefore we want to rethink the representation of formulae, instead of insisting on homogeneous content markup in OpenMath, we switch to MathML allow mixed presentation/content MathML, which conforms much more closely to user input (preserving presentational bits) and postpones full semantification to later stages of processing. Let us make an example: consider the formula $(a+b)^n$ encoded as `\exp{a+b}n`, where we have a semantic macro `\exp` defined by `\symdef{exp}[2]{#1^{#2}}` in module `arith` Then we should create

```
<math>
  <apply>
    <csymbol cd="arith">exp</csymbol>
    <mrow><ci>a</ci><mo>+</mo><ci>b</ci></mrow>
    <ci>n</ci>
  </apply>
</math>
```

Note that MathML does indeed allow to freely mix content and presentation MathML, here we have an application produced by the semantic macro `\exp` applied to the presentational $a+b$, where a and b are "content identifiers".

A side effect of the switch to MathML is that complex variable names are much nicer in MathML: x_5 is just

*Inspired by the "blue book" in Alan Bundy's group at the University of Edinburgh, sTeX blue notes, are documents used for fixing and discussing ϵ-baked ideas in projects by the sTeX group (see `http://github.com/KWARC/sTeX`). Unless specified otherwise, they are for project-internal discussions only. Please only distribute outside the sTeX group after consultation with the author.

```
<ci name="x5"><msub><mi>x</mi><mn>5</mn></msub></ci>
```

Finally, there is another effect of the switch to MathML: we finally have a good representation of formulae with text in them, e.g. the set

$$\{O \in \wp(X) \mid O \text{ is the union of open balls}\}$$

which we can encode as

```
\setst{O}{\inset{O}{\powerset{X}}}{\text{\ensuremath{O} is the union of open balls}}
```

given suitable semantic macros `\setst`, `\inset`, and `\powerset`. This should generate the mixed representation

```
<bind>
  <apply>
    <csymbol cd="sets">setst</csymbol>
    <apply><csymbol cd="sets">powerset</csymbol></apply>
  </apply>
  <bvar><ci>O</ci></bvar>
  <mtext><math><ci>O</ci></math> is the union of open balls</mtext>
</bind>
```

3 Verbalization Definitions

Currently, STEX only supports notation definitions for symbols, but we also need verbalization definitions for flexiformal mathematics; see [Koh14a] for a description of the concept and background on their use and [Koh14b, section 5] for first ideas towards an STEX encoding. We will extend the latter here.

3.1 OMDoc markup for Notation and Verbalization Definitions

In OMDoc, notation definitions are supplements to the `symbol` declaration. We have the following markup – in a simple case.

Listing 1: A classical Notation Definition

```
<symbol name="foo"/>
<notation for="foo">
  <prototype>
    <om:OMS cd="⟪CD⟫" name="foo"/>
  </prototype>
  <rendering>
    <msubsup><mi>f</mi><mi>o</mi><mi>o</mi></msubsup>
  </rendering>
<notation>
```

where ⟪CD⟫ is the current theory. For functional/binding symbols, the prototype is an OpenMath application/binding expression, where the argument positions are meta-variables `<expr name="\meta{name}/>"` which are being picked up in the `rendering` element as corresponding recursive calls `<render name="\meta{name}"/>`.

For verbalization definitions, we want to reuse notation definitions, so a mathematical concept "*big array raster*" may be given a symbol name `bar` and the notation definition.

Listing 2: A classical Verbalization Definition

```
<symbol name="bar"/>
<notation for="bar">
  <prototype>
    <om:OMS cd="⟪CD⟫" name="bar"/>
  </prototype>
  <rendering>big array raster</rendering>
<notation>
```

Note that verbalizations are part of text, so the contents of the `rendering` element are as well. In cases, where a symbol has both notations and verbalizations, e.g. addition, which has the notation + and the verbalization "*plus*", the notation element has multiple `rendering` children.

But, in our new, multilingual infrastructure, symbol and notation both go into the module signature, whereas the verbalizations go into the language bindings. Therefore we propose to ungroup the notation definitions, use the `prototype` and `rendering` elements directly, and cross-reference them. For plus this would give rise to the following OMDoc markup

Listing 3: Proposed Notation Definition

```
<symbol name="plus"/>
<protoype for="plus" name="plus.proto">
  <om:OMS cd="arithmetics" name="plus"/>
</prototype>
<rendering for="plus.proto">
  <msubsup><mi>f</mi><mi>o</mi><mi>o</mi></msubsup>
</rendering>
```

in the module signature and

Listing 4: Proposed Verbalization Definition

```
<rendering for="plus.proto">plus</rendering>
```

in the (English) language binding. Note that with the ungrouping, we can also be more flexible about where to put language-specific rendering (see Listing 5 for an example).

3.2 Direct STEX Encoding of Verbalization Definitions

In STEX use that the `\symdef` macro for notation definitions, e.g. `\symdef{foo}{f^o_o}` creates a semantic macro `\foo` that expands to f_o^o. Note that such semantic macros are only intended to be used in math mode (they usually lead to telltale errors in text mode). In the LATEXML workflow does two more things: it creates a `symbol` element and a `notation` element as in Listing **??**.

It seems natural to use the `\symdef`/`\symvariant` macros for verbalization definitions as well. If there is already semantic macro for `\bar`, we can simply use

```
\symvariant{bar}{en}{\text{big array raster}}
```

With this, `\bar[de]` expands to `big array raster`. Note that the user has to keep track on which variants are math mode and which are text mode, and make sure that he uses the right one in each situation. But we can hide this from the use by making `\bar` be mode-sensitive[1]: in math mode –this can be checked with `\ifmmode` in TEX, it selects the internal variant defined by `\symdef`/`\symvariant`, and in text mode it selects the internal macro defined by two new macros `\verbdef`/`\verbvariant`. Note that as we are in a language binding for verbalization definitions[2], we do no have to specify the language; moreover, we can have variants for the math/text modes separately. This is useful e.g. for greatest common divisors, which have language-sensitive notations and verbalizations (see Listing 5).

Listing 5: Notation and Verbalization Definitions for Greatest Common Divisor

```
\begin{modsig}{gcd}
  \symdef[name=gcd]{gcdOp}{\text{gcd}}
  \symdef{gcd}[1]{\prefix\gcdOp{#1}}
\end{modsig}

\begin{modnl}{gcd}{en}
  \verbdef{gcdOp}{greatest common divisor}
  \verbdef{gcd}[2]{\gcdOp of #1 and #2}
  \verbvariant{gcdOp}{plural}{greatest common divisors}
  \verbvariant{gcd}[2]{plural}{\gcdOp[plural] of #1 and #2}
```

[1] Thanks to Deyan Ginev for this suggestion

[2] We will not – for the moment – support verbalization definitions in the monolingual setting.

```
\end{modnl}

\begin{modnl}{gcd}{en}
  \verbdef{gcdOp}{\text{gcd}}
  \verbdef{gcd}[2]{\gcdOp von #1 und #2}
  \symvariant{gcdOp}{de}{\text{ggT}}
\end{modnl}
```

Note that the thus generated semantic macros are not only mode-sensitive, but also language-sensitive: for every language there is a "first" `\verbdef` followed by `\verbvariants`. These generate semantic macros, that react to the value of the switch `\stex@lang` – implicitly set by the `modnl` environment or explicitly by `sTeXselectlanguage`[1] and select the right variant in the right language. EdN:1

3.3 Implicit Verbalization Definitions from Definienda

But in most situations, an explicit `\verbdef` is unnecessary, since we have the definiendum markup. In the situation of Listing 6, we have a symbol `bar` generated by the `\symdef` and a definiendum for the symbol `bar` marked up by the `\defiii` macro – see [Koh14c] for details on `\def*`. Note that the optional argument of `\defiii` is used to specify the symbol name, here `bar` here. We could let LaTeXML let generate the equivalent of a `verbdef` as above implicitly, freeing the user from writing down specifications twice.

Listing 6: Definiendum Markup in Language Bindings

```
\begin{modnl}[creators=miko,primary]{foo}{en}
\begin{definition}
  A \defiii[bar]{big}{array}{raster} ($\bar$) is a\ldots, it is much bigger
  than a \defiii[sar]{small}{array}{raster}.
\end{definition}
\end{modnl}

\begin{modnl}[creators=miko]{foo}{de}
\begin{definition}
  Ein \defiii[bar]{gro"ses}{Feld}{Raster} ($\bar$) ist ein\ldots, es
  ist viel gr"o"ser als ein \defiii[sar]{kleines}{Feld}{Raster}.
\end{definition}
\end{modnl}
```

But let us also look at a more interesting symbol: the "special linear group" already discussed in [Koh14b]. Here the sTeX verbalization definition would be

```
\verbdef[name=slgroup]{SLGroup}[2]{special linear group of order #1 over #2}
```

Here we have a problem with retrieving this from the definition without additional markup. A normal definition would have the form

```
\begin{definition}
  The \defiii[slgroup]{special}{linear}{group} \notatiendum{$\SLgroup{n}{F}$}
  of degree $n$ over a \trefi[field]{field} $F$ is ...
\end{definition}
```

In particular, the definiendum is discontiguous and usually only the "head" is explicitly emphasized by boldface font. In this situation, a "continuation markup might help – just exploring the syntax here:

```
\begin{definition}
  The \defiii[slgroup]{special}{linear}{group} \notatiendum{$\SLgroup{n}{F}$}
  \defc[slgroup]{of degree \defarg[1]{$n$}}
  \defc[slgroup]{over \defarg[2]{a \trefi[field]{field} $F$}} is ...
\end{definition}
```

[1] EdNote: MK@MK: this still needs to be implemented; also implement variants of the other babel selection mechanisms like foreignlanguage, etc. best in smultiling.dtx.

Here the `\defc` macro continues the definiendum started with the `\defiii` – we specify which one with the symbol name in the optional argument and the embedded `\defarg` macro escapes out of that and marks its argument as an argument specifier. I am not sure that this is better than just adding the explicit verbalization definition above. But maybe the inline markup gives us more structure.

An alternative would be to have a long definiendum markup and use `\notatiendum` to escape out of it. Something like

```
\begin{definition}
  The \definiendum[slgroup]{special linear group \notatiendum{$\SLgroup{n}{F}$}
  of degree \defarg[1]{$n$} over \defarg[2]{a \trefi[field]{field} $F$}} is ...
\end{definition}
```

This implies less markup work. But do we lose structure here? If we have optional arguments (and here both are), we would like to associate "*of order*" with the first argument and "*over*" with the second. So maybe something like

```
\begin{definition}
  The \definiendum[slgroup]{\defhead{special linear group}
    \notatiendum{$\SLgroup{n}{F}$}
    \defarg[1,opt]{of degree \arg{$n$}}
    \defarg[2,opt]{over \arg{a \trefi[field]{field} $F$}}} is ...
\end{definition}
```

is more useful. That would allow us to account for all the elision forms.

But that could also be done with the explicit verbalization definition

```
\verbdef[name=slgroup]{SLGroup}[2]{[special linear group] [of order #1] [over #2]}
```

where `[` and `]` group the elision groups. But maybe we also want to use curly braces instead of them. We have to see what works best.

4 Conclusion

We have described a set of new functionalities for sTeX and specified some aspects of them. Now, they need to be implemented and tested.

References

[Gin+14] Deyan Ginev et al. "The SMGLoM Project and System". 2014. URL: `http://gl.kwarc.info/smglom/internal/raw/master/cicm14-system/paper.pdf`.

[KG14] Michael Kohlhase and Deyan Ginev. *`smultiling.sty`: Multilinguality Support for sTeX*. Tech. rep. 2014. URL: `https://github.com/KWARC/sTeX/raw/master/sty/smultiling/smultiling.pdf`.

[Koh14a] Michael Kohlhase. "A Data Model and Encoding for a Semantic, Multilingual Terminology of Mathematics". In: *Intelligent Computer Mathematics*. Ed. by Stephan Watt et al. Lecture Notes in Computer Science. Springer, 2014, pp. 169–183. URL: `http://kwarc.info/kohlhase/papers/cicm14-smglom.pdf`.

[Koh14b] Michael Kohlhase. "A Data Model and Encoding for SMGloM". SMGloM Blue Note. 2014. URL: `http://gl.kwarc.info/smglom/blue/raw/master/datamdl/note.pdf`.

[Koh14c] Michael Kohlhase. *`statements.sty`: Structural Markup for Mathematical Statements*. Tech. rep. 2014. URL: `https://github.com/KWARC/sTeX/raw/master/sty/statements/statements.pdf`.