Semantic Markup for TEX/LATEX

Michael Kohlhase

School of Engineering & Science

International University Bremen, Germany

http://www.faculty.iu-bremen.de/mkohlhase

IUB

The MKM Authoring/Migration Problem

- Very interesting Systems for Mathematical Knowledge Management (MKM)
- They promise to navigate/index/search/adapt/... large corpora of MK
- Problem: where is the beef?
- Possible sources:
  - libraries from theorem proving- and program verification and computer algebra systems (most of us do that)
  - Write your own in MathML/OpenMath/OMDoc/... (very tedious)
  - convert from SGML/Office engineering documents (difficult to get)
  - adapt from MS PowerPoint documents (see later talk)
  - migrate from existing TeX/LaTeX documents (There's the beef)
- TeX/LaTeX is a power-user's interface to mathematics!

MKM Formats

- Definition: A MKM format is a content-oriented representation language for mathematics, that makes the structure of the mathematical knowledge in a document explicit enough that machines can operate on it.
- Examples: (so we get a feeling)
  - Document Markup: LaTeX, DocBook, TEI, OMDoc... (but not TeX)
  - Formula Markup: Mathematica, Maple, OpenMath, Content-MathML (but not Presentation-MathML)
  - Theory/Context Markup: Maya, Casl, OMDoc (but not TeX/LaTeX)
- Goal of this talk: Make TeX/LaTeX into a MKM format on all levels.
  - allow to add explicit structure markup without changing presentation in particular, provide infrastructure for formula and theory/context markup.
  - enable translation into traditional MKM formats. (solve (part of) the MKM authoring/migration problem)

TEX/LATEX as MKM Format: The Notation/Context Problem

- idiosyncratic notations that are introduced, extended, and discarded on the fly

$$\lambda X_{\alpha}.X =_{\alpha} \lambda Y_{\alpha}.Y \hat{=} \mathbf{I}^{\alpha}$$

  meaning of α depends on the context: object type vs. mnemonic vs. type label.

- even "standard notations" depend on the context, e.g. binomial coefficients: $\binom{n}{k}$, ${}_nC^k$, C^n_k, and C^k_n all mean the same thing: $\frac{n!}{k!(n-k)!}$ (cultural context)
- Notation scoping follows complex rules (notations must be introduced)
  - "*We will write* $\wp(S)$ *for the set of subsets of* S" (for the rest of the doc)
  - "*We will use the notation of [BrHa86], with the exception...*". (by reference)
  - "*Let* S *be a set and* $f\colon S \to S\ldots$" (scope local in definition)
  - "*where* w *is the...*" (scope local in preceding formula)
  - A book on group theory in Bourbaki series uses notation [Bourbaki: Algebra]

TeX/LaTeX as MKM Format: The Reconstruction Problem

- Mathematical communication relies on the inferential capability of the reader.
- semantically relevant arguments are left out (or ambiguous) to save notational overload (reader must disambiguate or fill in details.)

  $$\log_2(x) \text{ vs. } \log(x) \qquad [\![\mathbf{A}]\!]_\varphi^{\mathcal{M}} \text{ vs. } [\![\mathbf{A}]\!]$$

- condensed notation: $f(x+1)\pm 2\pi = g(x-1)\mp 2i$ (stands for 2 equations)
- ad hoc extensions: $\#(A\cup B) \leq \#A + \#B$ (exceptions for ∞)
- overt ambiguity: $\sin x/y$ vs. $\frac{\sin x}{y}$ vs. $\sin\frac{x}{y}$ vs. $-1 \leq \sin x/\pi \leq 1$
- size of the gaps varies with the intended readership and the space constraints.
- can be so substantial, that only a few specialists in the field can understand

The sTEX approach

- The reconstruction and the notation/context problem have to be solved to turn or translate TEX/LATEX into a MKM format
- Problem: This is impossible in the general case (AI-hard)
- Idea: Enable the author to make structure explicit and disambiguate meanings
  - use the TEX macro mechanism for this (well established)
  - the author knows the semantics best (at least he understands)
  - the burden is is alleviated by manageability savings (MKM on TEX/LATEX)
- sTEXApproach: Semantic preloading of TEX/LATEX documents.

A Phenomenology of TeX/LaTeX macros

- Abbreviative Macros: define a new control sequence for a sequence of TeX tokens, which is expanded in document formatted.
- Semantic Macros: stand for semantic objects and expand to a presentation of the object. For instance a semantic macro for $\mathcal{C}^\infty(\mathbb{R})$ is

  ```
  \def\SmoothFunctionsOnReals{{\cal C}^\infty({\mathbb R})}
  ```

  an (even more semantic) variant would be

  ```
  \def\Reals{{\mathbb R}}
  \def\SmoothFunctionsOn#1{{\cal C}^\infty(#1)}
  \def\SmoothFunctionsOnReals{\SmoothFunctionsOn\Reals}
  ```

  first two are semantic, the last one abbreviative (only one char shorter)
- If we use `\binomcoeff{n}{k}` instead of `\left(n\atop k\right)` for $\left(n \atop k\right)$, we can change the notational standard by just changing the definition of the control sequence `\binomcoeff`.

IU^B)

A Phenomenology of TEX/LATEX macros (continued)

- Elliptive Macros: for leaving out "obvious" arguments

```
\def\interpret#1#2#3{{\left[\kern-0.18em\left[#1\right]\kern-0.18em\right]^{#2}_{#3}}}
\def\interm#1{\interpret{#1}{{\cal M}}{}}
\def\interp#1{\interpret{#1}{}{\phi}}
\def\interoo#1{\interpret{#1}{}{}}
```

- `\interpret{A}{\cal M}\phi` introduces $[\![A]\!]^{\mathcal{M}}_{\varphi}$.
- In the expressions $[\![A]\!]^{\mathcal{M}}$, $[\![A]\!]_{\varphi}$, and $[\![A]\!]$ we elide information: φ and $\mathcal{M}$ are relevant semantically, but not presented, since it can be inferred by the reader.

IUB

Converting TEX/LATEX Documents to XML

- HERMES [Anghelache] and TEX4HT [Gurari] use the TEX parser, seed the DVI file with semantic information, parse DVI for transformation.
- LATEXML [Miller] and SGLR/ELAN4 [van den Brand, Stuber] reimplement the TEX parser. (do not expand semantic macros)
- Case Study: Converting Intro Computer Science to OMDOC via semantic preloading and LATEXML
- LATEXML workflow: (used in our case study)
  - LATEXML $\hat{=}$ TEX parser + XML emitter + post-processing pipeline.
  - LATEXML bindings for the XML emitter, (for all LATEX packages as well)

```
DefConstructor("\Reals","<XMTok name='Reals'/>");
DefConstructor("\SmoothFunctionsOn{}",
               "<XMApp><XMTok name='SmoothFunctionsOn'/>#1</XMApp>");
DefMacro("\SmoothFunctionsOnReals","\SmoothFunctionsOn\Reals");
```

OMDoc in a Nutshell (three levels of modeling)

Formula level: OpenMath/C-MathML

- Objects as logical formulae
- semantics by ref. to theory level

```
<OMA>
  <OMS cd="arith1" name="plus"/>
  <OMS cd="nat" name="zero"/>
  <OMV name="N"/>
</OMA>
```

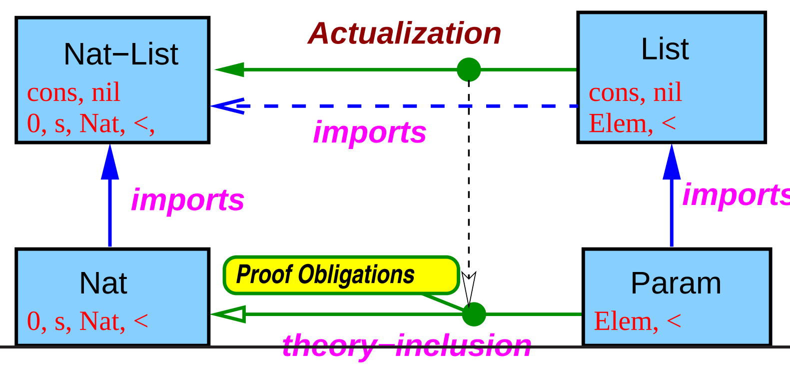
Statement level:

- Definition, Theorem, Proof, Example
- semantics explicit forms and refs.

```
<definition for="plus" type="rec.">
  <CMP>rec. eq. for plus</CMP>
  <FMP>X+0 = X</FMP>
  <FMP>X+s(Y) = s(X+Y)</FMP>
</definition>
```

Theory level: Development Graph

- inheritance via symbol-mapping
- theory-inclusion by proof-obligations
- local (one-step) vs. global links



IU^B

The sTEX Packages: Statement Markup

- Running example from 320101 General Computer Science I

> Theorem: *//.///.// is not a unary natural number.*
>
> Proof: We make use of the induction axiom P5:
>
> – we show that every unary natural number is different from //.///.// by convincing ourselves of the prerequisites of P5.
>
> – we have two cases:
>
> 1. base case: '/' is not //.///.// (obvious)
> 2. step case: If a number is different from //.///.//, then its successor is also different from //.///.//. (by inspection)
>
> – Thus we have considered all the cases and proven the theorem. □

sTEX Markup for the Example

```
\begin{assertion}[type=Theorem,id=not-un]{}
  $//.///.//$ is not a unary natural number.
\end{assertion}
\begin{proof}[id=not-un-pf,for=not-un]{We make use of the induction axiom P5:}
  \begin{step} we show that every unary natural number is different from $//.///.//$
    by convincing ourselves of the prerequisites of P5:
    \begin{justification}[method=apply-axiom,premises={ax5}]
      \begin{pfcases}{we have two cases}
        \begin{pfcase}[id=foo]{base case}
          \begin{step}[display=flow]'/' is not $//.///.//$
            \begin{justification}[method="trivial"]obvious\end{justification}
          \end{step}
        \end{pfcase}
        \begin{pfcase}[id=bar]{step case}
          \begin{step}[display=flow] If a number is different from $//.///.//$, then
             its successor is also different from $//.///.//$.
            \begin{justification}[method="blast-eq"]by inspection\end{justification}
          \end{step}
       ...
\end{proof}
```

The Generate OMDoc for the Example

```
<assertion type="theorem" id="not-un"
  <CMP><legacy format="TeX">//.///.//</legacy> is not a unary natural number.</CMP>
<assertion>
<proof id="not-un-pf" for="not-un">
  <CMP>We make use of the induction axiom P5:</CMP>
  <derive id="d1"/>
    <CMP>we show that every unary natural number is different from $//.///.//$
    by convincing ourselves of the prerequisites of P5</CMP>
    <method xref="apply">
      <premise xref="ax5"/>
      <proof id="foo"><metadata><Title>base case</Title></metadata>
        <derive id="c1"><CMP>'/' is not $//.///.//$</CMP>
          <method xref="trivial"><omtext><CMP>obvious</CMP></omtext></method>
        </derive>
      </proof>
      <proof id="bar"><metadata><Title>step case</Title></metadata>
        <derive id="c2">
          <CMP>If a number is different from $//.///.//$, then its
              successor is also different from $//.///.//$.</CMP>
          <method xref="eq-blast"><omtext><CMP>by inspection</CMP></omtext></method>
     ...
</proof>
```

sTeX Modules help with the Notation/Context Problem

- Note: *the context of notations coincides with the context of the concepts they denote*
- Idea: Use the theory structure for notational contexts
  - The scoping rules of TeX/LaTeX follow a hierarchical model:
  - a TeX macro is either globally defined or defined exactly inside the group given by the group induced curly braces hierarchy.
- Solution: provide explicit grouping for scope with inheritance.
  - new sTeX environment `module`,
  - new macro definition `\symdef`, scoped in `module`
  - specify the inheritance of `\symdef`-macros in `module` explicitly
  - `\symdef`-macros are undefined unless in home module or inherited.

sTeX Modules: Example

```
\begin{module}[id=pairs]\symdef{\pair}[2]{\langle#1,#2\rangle} ...\end{module}

\begin{module}[id=sets]
  \symdef{\member}[2]{#1\in #2}          % set membership
  \symdef{\mmember}[2]{#1\in #2}  ...    % aggregated set membership
\end{module}

\begin{module}[id=setoid,uses={pairs,sets}]
  \symdef{\sset}{{\cal S}}                % the base set
  \symdef{\sopa}{\circ}                   % the operation symbol
  \symdef{\sop}[2]{(#1\sopa #2)}            % the operation applied
  \begin{definition}[id=setoid-def]
    A pair $\pair\sset\sopa$ is called a setoid, if $\sset$ is closed under
    $\sopa$, i.e. if $\member{\sop{a}{b}}\sset$ for all $\mmember{a,b}\sset$.
  \end{definition}
\end{module}

\begin{module}[id=semigroup,uses=setoid]
  \begin{definition}[id=setoid-def]
    A setoid $\pair\sset\sopa$ is called a monoid, if $\sopa$ is associative on
    $\sset$, i.e. if $\sop{a}{\sop{b}{c}}=\sop{\sop{a}{b}}{c}$ for all $\mmember{a,b,c}\sset$.
  \end{definition}
\end{module}
```


IU^B)

The Result of the Example

Definition: A pair $\langle \mathcal{S}, \circ \rangle$ is called a setoid, if $\mathcal{S}$ is closed under $\circ$, i.e. if $(a \circ b) \in \mathcal{S}$ for all $a, b \in \mathcal{S}$.

Definition: A setoid $\langle \mathcal{S}, \circ \rangle$ is called a monoid, if $\circ$ is associative on $\mathcal{S}$, i.e. if $(a \circ (b \circ c)) = ((a \circ b) \circ c)$ for all $a, b, c \in \mathcal{S}$.

- Empirically: Explicit module structure
  - is a little overhead (can be automated)
  - Feels safer (but I might be brainwashed)
- In our case study: 320 slides, 160 modules, depth ~ 20

IU$^{\mathbf{B}}$)

sTEX Modules and LaTeXML Bindings

- Idea: Supply the LaTeXML bindings together with the semantic macros

```
\symdef{\pair}[2]{\langle{#1},{#2}\rangle}
\latexmldef{\pair}[2]{<XMApp>
                        <XMTok cd='pairs' name='pair'/>
                        <XMArg>#1</XMArg><XMArg>#2</XMArg>
                      </XMApp>}
```

  or shorter: `\latexmlconstructor{\pair}[args=2,cd=pairs,name=pair]`

- sTEX moves macro definitions back into documents (like in OOP)
- differentiate macros for "late binding effects"
  - late binding enables styling (good for presentational macros)
  - late bindings potentially changes meanings (bad for semantic macros)
- Empirically: sTEX modules are great candidates for semantic reuse

Elliptive Macros

- elliptive macros differ from semantic macros only in their LaTeXML binding
- preload the elided arguments, but do not show them

```
\elldef{\interm}[2]{\interpret{#1}{\cM}{}}       % supply, but do not show third arg
\elldef{\interp}[2]{\interpret{#1}{}{\assign}}   % supply, but do not show second arg
\elldef{\interoo}[3]{\interpret{#1}{}{}}         % supply, but do not show both
```

- LaTeXML bindings instruct to elide arguments in the transformation

```
\latexmlelide{\interm}{3}{\interpret}            % elide third arg
\latexmlelide{\interp}{2}{\interpret}            % elide second arg
\latexmlelide{\interp}{2,3}{\interpret}          % elide both
```

- equivalent to (make use hinting for presentation engine)

```
\latexmldef{\interm}[2]{%
 <XMApp>
   <XMTok cd='booeaneval' name='interpret'/>
   <XMArg>#1</XMArg>
   <XMArg elide='yes'>#2</XMArg>
   <XMArg>
     <XMTok cd='booleaneval' name='assign'/>
   </XMArg>
 </XMApp>}
```

```
<OMA>
  <OMS cd='booeaneval' name='interpret'/>
  <OMV name="A"/>
   <OMS style="display:none"
        cd="booeaneval" name="themodel"/>
  <OMS cd="booeaneval" name="assign"/>

</OMA>
```

Conclusion and Further Work

- turn TEX/LATEX into a MKM format by enabling semantic preloading (finally)
- sTEX+LATEXML $\hat{=}$ invasive editor for TEX/LATEX
- together with CPOINT and NB2OMDOC covers paradigmatic document formats.
- Future work (this is just the beginning)
  - semantically preload the OMDOC book
  - *.aux files for external modules (import modulo renaming/re-presentation?)
  - improve LATEXML postprocessing (type-analysis,part-of-speech,...)
  - HERMES/TEX4HT/generalized bindings?
  - more output formats XHTML+MATHML, CONNEXIONS
- Acknowledgments: David Carlisle (TEX/LATEX consulting)
  Bruce Miller (extended LATEXML)
  Ioan Sucan (preloaded General CS I/II slides)