Semantic Markup for TEX/LATEX

Michael Kohlhase
Electrical Engineering & Computer Science
International University Bremen
http://www.faculty.iu-bremen.de/mkohlhase

September 6, 2004

Abstract

We present a collection of TEX macro packages that allow to markup TEX/LATEX documents semantically without leaving the document format, essentially turning TEX/LATEX into a document format for mathematical knowledge management (MKM).

We analyze the current practice of semi-semantic markup in LATEX documents and supply a definition mechanism for semantic macros and a non-standard scoping construct for them, which is oriented at the semantic depency relation rather than the document structure.

We evaluate the STEX macro collection on a large case study: the course materials of a a two-semester course in Computer Science was annotated semantically and converted to the OMDoc MKM format Bruce Miller's LaTeXML system.

1 Introduction

One of the great problems of mathematical knowledge management (MKM) systems is to obtain access to a sufficiently large corpus of mathematical knowledge to allow the management/search/navigation techniques developed by the community to display their strength. Such systems usually expect the mathematical knowledge they operate on in the form of semantically enhanced documents.

We will use the term **MKM format** for a formal representation language for mathematics, that makes the structure of the mathematical knowledge in a document explicit enough that machines can operate on it. Examples of MKM formats include the various logic-based languages found in automated reasoning tools (see [RV01] for an overview), program specification languages (see e.g. [Ber89]), and the various XML-based, content-oriented markup languages for mathematics on the web, e.g. OpenMath [BCC+04], Content-MathML [ABC+03], or our own OMDoc (see Section 2.4).

Currently, a large part of mathematical knowledge is prepared in the form of TEX/LATEX documents. TEX [Knu84] is a document presentation format that combines complex page-description primitives with a powerful macro-expansion facility, which is utilized in LATEX (essentially a set of TEX macro packages, see [Lam94]) to achieve more content-oriented markup that can be adapted to particular tastes via specialized document styles. It is safe to say that LATEX largely restricts content markup to the document structure[1], and graphics,

[1] supplying macros e.g. for sections, paragraphs, theorems, definitions, etc.

leaving the user with the presentational TEX primitives for mathematical formulae. Therefore, even though LATEX goes a great step into the direction of an MKM format, it is not, as it lacks infrastructure for marking up the functional structure of formulae and mathematical statements, and their dependence on and contribution to the mathematical context.

In this paper, we will investigate how we can use the macro language of TEX to make it into an MKM format by supplying specialized macro packages, which will enable the author to add semantic information to the document in a way that does not change the visual appearance[2]. We speak of **semantic preloading** for this process and call our collection of macro packages STEX (Semantic TEX). Thus, STEX can serve as a conceptual interface between the document author and MKM systems: Technically, the semantically preloaded LATEX documents are transformed into the (usually XML-based) MKM representation formats, but conceptually, the ability to semantically annotate the source document is sufficient.

Concretely, we will present the STEX macro packages together with a case study, where we semantically preload the course materials for a two-semester course in Computer Science at International University Bremen and transform them to the OMDOC MKM format (see section 2.4) with the LATEXML system (see section 2.3), so that they can be used in the ACTIVEMATH system [MAF+01]. For this case study, we have added LATEXML bindings for the STEX macros, and a post-processor for the OMDOC language, but the STEX package should in principle be independent of these two choices, since it only supplies a general interface for semantic annotation in TEX/LATEX. Furthermore, we have semantically preloaded the LATEX sources for the course slides (380 slides, 8200 lies of LATEX code with 336kb). Almost all examples in this paper come from this case study.

2 Foundations and Related Work

Before we go into the details of the STEX package and the conversion interface, let us review the current situation. We will first try to classify TEX/LATEX macros used in mathematical documents with respect to their semantic contribution, and describe the semantic preloading process. Then we will survey TEX/LATEX conversion tools and look at the LATEXML system which we have used in our case study in more detail. Finally, we will review enough of the OMDOC format to make the presentation self-contained.

2.1 Semantic Preloading of TEX/LATEX Documents

Let us now re-consider why the conversion of TEX/LATEX documents to MKM formats is a non-trivial task. We can distinguish two different — albeit interrelated — problems:

The Notation/Context Problem Mathematicians have idiosyncratic notations that are introduced, extended, and discarded on the fly. Generally, this means that the parsing and semantics construction process has to be adapted on the fly as well. In particular, it depends on the context, what a piece of notation means. To go into only a few examples: The Greek letter α is used for numbering, as a variable name, as a type, and as a name for an operation, etc. In the formula

$$\lambda X_\alpha.X =_\alpha \lambda Y_\alpha.Y \hat{=} \mathbf{I}^\alpha \tag{1}$$

[2]However, semantic annotation will make the author more aware of the functional structure of the document and thus may in fact entice the author to use presentation in a more consistent way than she would usually have.

the first and third occurrence of the symbol α is as a type of the bound variables X and Y, whereas the second one is an indicator that the equality operation is that of α-equality (the name is derived from the process of "alphabetic renaming"); the final and fourth occurrence of α — as an upper index on the combinator $\mathbf{I}$ — selects one of an infinite collection of identity combinators (identity function on type α, which incidentally as an operation has type $\alpha \to \alpha$). This example also shows that the notion of context can be extremely fine-granular in mathematics. Notation can also depend on other forms of context: We have varied "standard notations" for binomial coefficients: $\binom{n}{k}$, ${}_nC^k$, C^n_k, and C^k_n all mean the same thing: $\frac{n!}{k!(n-k)!}$; the third notation is the French standard, whereas the last one is the Russian one, so the context for determining the notation must in some cases include the cultural background of the author (whatever that means in practice).

The admissibility of symbols and notations in mathematical documents follows complex rules. In principle, a notation or symbol (more precisely a certain glyph that stands for a mathematical object or concept) must be introduced before it can be used. A notation can be introduced explicitly by a statement like "*We will write $\wp(S)$ for the set of subsets of S*", or by reference as in "*We will use the notation of [BrHa86] in the following, with the exception...*". The scope of a notation can be local, e.g. in a definition which begins with "*Let S be a set...*" or even only in the immediately preceding formula, if it is followed by "*where w is the...*". Finally, notation can be given by convention: If we open a book on group theory in the Bourbaki series on Algebra, we expect notation introduced in [Bou74].

The Reconstruction Problem Mathematical communication and notation relies on the inferential capability of the reader. Often, semantically relevant arguments are left out (or left ambiguous) to save notational overload relying on the reader to disambiguate or fill in the details. Of course the size of the gaps to be filled in varies greatly with the intended readership and the space constraints. It can be so substantial, that only a few specialists in the field can understand (e.g. enough to translate) a given piece of mathematical document.

Both of these problems have to be solved for a successful transformation into an MKM format, in which the meaning has to be made explicit, and all ambiguities have to be resolved. Of course, this is impossible in the general case except for the solution of the general "artificial intelligence problem" of achieving human-like intelligence in machines. Since we cannot rely on this problem to be solved anytime soon, we will burden the author with marking up the source documents with additional information that helps the transformation process determine the semantics.

We will make use the TEX macro mechanism for this markup: TEX allows to define so-called **macros**, which are expanded in the formatting process, that transforms a TEX source file `doc.tex` into a document `doc.dvi` in the (device-independent) presentation format, which can be directly rendered for the screen- or printer output. This basic and very simple mechanism can be put to various uses in documents: macros can be used to compute values for section numbers or footnotes and make TEX sources more portable, they can be used as abbreviations to save the author typing effort, and they can be used for semantic annotation,

which is what we will explore here. All of these uses are common[3] in LaTeX documents.

Abbreviative Macros define a new name for a sequence of TeX tokens, in essence, the macro just stands for the sum of tokens; this is the traditional function of LaTeX.

Semantic Macros stand for semantic objects and expand to a presentation (technically a sequence of TeX tokens of course) of the object. For instance $\mathcal{C}^\infty(\mathbb{R})$ stands for the set of arbitrarily differentiable ("smooth") functions on the real numbers. So a TeX definition

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

not only abbreviates the more complicated expression in the definiens, but also encapsulates the information that this expression represents a distinct mathematical object. A variant macro definition for $\mathcal{C}^\infty(\mathbb{R})$ would be

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

Here, we characterize the first two definitions as "semantic" and the third one as "abbreviative".

Semantic macros are commonly used to enhance the maintainability and reusability of source code. If we for instance use three different semantic macros for the glyph α in example (1), we can readily distinguish them (e.g. in searches, or for replacement) in the LaTeX source. If we use the semantic macro `\binomcoeff{n}{k}` instead of the presentation markup `\left(n\atop k\right)` for a binomial coefficient, then we can change the notational standard by just changing the definition of the control sequence `\binomcoeff`.

Elliptive Macros Finally, there are uses of macros that we will call **elliptive**, since they are used to elide (i.e. leave out) "obvious" arguments. Consider the family of macros in Figure 1: `\interpret` introduces the notation $[\![A]\!]^{\mathcal{M}}_{\varphi}$ for the denotation of a formula A in a model $\mathcal{M}$ under a variable assignment φ. The macro `\intermp` is abbreviative (it just saves typing). The case of `\interm`, `\interp`, and `\interoo` is interesting. In the expressions $[\![A]\!]^{\mathcal{M}}$, $[\![A]\!]_{\varphi}$, and $[\![A]\!]$ we do not abbreviate information, but really elide it, i.e. the information that is left out (the assignment φ for `\interm{A}`, the model $\mathcal{M}$ for `\interp{A}`, and both for `\interoo{A}`) is relevant semantically, we just do not present it, since it can be inferred by the reader.

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

Figure 1: Elliptive Macros

[3]Of course, the actual frequency and distribution of macros among the categories below depends on the tastes of the individual author and the purpose of the document.

Elliptive macros are often used to specialize semantic macros as in our example. However, in contrast to semantic macros, they are not semantically complete, since they do not specify the full semantic information behind the mathematical object.

Obviously, to use TEX/LATEX as an MKM format, we need to maximize the use of semantic macros over the use of direct presentational notations. We call the process of converting presentation markup into semantic markup in the form of suitable semantic macros semantic preloading of a document. For an existing TEX/LATEX document, this is a relatively tedious process, as it involves heavy editing of the document[4], but if well-designed collections of semantic conventions are used during document creation, the notational overhead is easily outweighed by the inherent manageability and reusability benefits discussed above.

For the use of elliptive macros, the case is not so clear, since the presentation system in TEX/LATEX is rather inflexible, as it was originally designed for printed documents. In the case of other MKM formats, where the presentation engine may vary, we can make use of a complex user interaction during presentation, for instance, the user could request to see the elided arguments, or change the notation on the fly. So, if we look at TEX/LATEX as an MKM format, we can see an added value in supplying the elided arguments (via an upgraded presentation engine based on e.g. dvi specials, PDF, or dynamic XHTML+MATHML). In any case, we assume that semantic pre-loading makes all elliptive notations explicit in elliptive macros.

2.2 Tools for TEX/LATEX to XML Conversion

The conversion of TEX/LATEX documents has been supported by various tools, of which none has been totally satisfactory yet. For an overview and a test suite see [Mat].

The MMISSLATEX [DLL+04] to OMDOC converter is based on a set of LATEX styles [Koh01, Chapter 4] that use TEX's file output facility to generate OMDOC files directly.

The CONNEXIONS project has developed a LATEX package [WHB03] facilitating content markup in LATEX for the LATEX-to-CNXML conversion, but does not have a transformation tool that makes use of it.

Romeo Anghelache's HERMES [Ang] and Eitan Gurari's TEX4HT systems use special TEX macros to seed the `dvi` file generated by TEX with semantic information. The `dvi` file is then parsed by a custom parser to recover the text and semantic traces which are then combined to form the output XML document. While HERMES attempts to recover as much of the mathematical formulae as Content-MATHML, it has to revert to Presentation-MATHML where it does not have semantic information. TEX4HT directly aims for Presentation-MATHML.

The latter two systems rely on the TEX parser for dealing with the intricacies of the TEX macro language (e.g. TEX allows to change the tokenization (via "catcodes")and the grammar at run-time). In contrast to this, Bruce Miller's LATEXML [Mil] system and the SGLR/ELAN4 system [vdBS03] re-implement a parser for a large fragment of the TEX language. This has the distinct advantage that we can control the parsing process: We want to expand abbreviative macros and recursively work on the resulting token sequence, while we want to directly translate semantic macros, since they directly correspond to the content representations we want to obtain. The LATEXML and SGLR/ELAN4 systems allow us to do just this.

[4] although tools like regular expression replacement facilities e.g. in the `emacs` editor or one-shot conversion programs e.g. in `perl` can be a great help on uniformly marked up document corpora.

In the conversion experiment that drove the development of the sTEX package, we chose the LaTeXML system, whose LaTeX parser seems to have larger coverage. We assume that a solution based on the SGLR/Elan4 system would work similarly. Eventually, a combination of both systems might be the way to go: the Elan4 system could be integrated into post-processing phase of the LaTeXML work-flow. Systems like Hermes or TeX4HT could be used with sTEX, given suitable sTEX bindings provided we find a way to distinguish semantic- from abbreviative macros.

2.3 The LaTeXML LaTeX to Xml Converter

The LaTeXML system, consists of a TeX parser, an Xml emitter, and a post-processing pipeline. To cope with LaTeX documents, the system needs to supply LaTeXML **bindings** (i.e. special directives for the Xml emitter) for the LaTeX packages. So every LaTeX package `package.sty` comes with a LaTeXML binding file `package.ltxml`, a Perl file which contains LaTeXML constructor-, abbreviation-, and environment definitions, e.g.

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

`DefConstructor` is used for semantic macros, whereas `DefMacro` is used for abbreviative macros. The latter is used, since the `latexml` program does not read `package.sty` and needs to be told, which sequence of tokens to recurse on. The LaTeXML distribution contains LaTeXML bindings for the most common base LaTeX packages.

For the Xml conversion, the `latexml` program is run, say on a file `doc.tex`. `latexml` loads the LaTeXML bindings for the LaTeX packages used in `doc.tex` and generates an Xml file `doc.tex.xml`, which closely mimics the structure of the parse tree of the LaTeX source. The structure of this file is determined by the LaTeXML packages and must correspond to a format-specific document type definition (DTD), which is usually a relatively simple extension of Bruce Miller's LaTeX DTD.

In the semantic post-processing phase, the LaTeX-near representation in `doc.tex.xml` is transformed into the target format by the `latexmlpost` program. This program applies a pipeline of intelligent filters to `doc.tex.xml`. The LaTeXML program supplies various filters, e.g. for processing HtML tables, including graphics, or converting formulae to Presentation-MathML. Other filters like transformation to OpenMath and Content-MathML are currently under development. The filters can also consist of regular Xml-to-Xml transformation process, e.g. an XslT style sheet. Eventually, post-processing will include semantic disambiguation information like types, part-of-speech analysis, etc. to alleviate the semantic markup density for authors.

2.4 OMDoc, an Open Format for Mathematical Documents

OMDoc (see [Koh04]) is an Xml-based document format for representing technical documents and their underlying knowledge. To achieve content- and context markup for mathematical knowledge, OMDoc uses three levels of modeling

Mathematical Formulae At the lowest level of mathematical formulae, OMDoc uses the established standards OpenMath [BCC+04] and Content-MathML [ABC+03]. These

provide content markup for the structure of mathematical formulae and context markup in the form of URI references in the symbol representations.

Mathematical Statements OMDoc provides an original markup scheme for making the structure of mathematical statements explicit. Again, we have content and context markup aspects. For instance the definition in the second row of Figure 2 contains an informal description of the definition as a first child and a formal description in the two recursive equations in the second and third children supported by the `type`, which states that this is a recursive definition. The context markup in this example is simple: it states that this piece of markup pertains to a symbol declaration for the symbol `plus` in the current theory (presumably the theory `arith1`).

Mathematical Theories At this level, OMDoc supplies original markup for clustering sets of statements into theories, and for specifying relations between theories by morphisms. By using this scheme, mathematical knowledge can be structured into reusable chunks. Theories also serve as the primary notion of context in OMDoc, they are the natural target for the context aspect of formula and statement markup.

Level	Example

Formula level: OpenMath/C-MathML
• Objects as logical formulae
• semantics by pointing to theory level
	`<OMA>`
	`  <OMS cd="arith1" name="plus"/>`
	`  <OMV name="X"/>`
	`  <OMS cd="nat" name="zero"/>`
	`</OMA>`

Statement level:
• Definition, Theorem, Proof, Example
• structure explicit in forms and references
	`<definition for="#plus" type="rec.">`
	`  <CMP>rec. eq. for plus</CMP>`
	`  <FMP>$X+0=0$</FMP>`
	`  <FMP>$X+s(Y)=s(X+Y)$</FMP>`
	`</definition>`

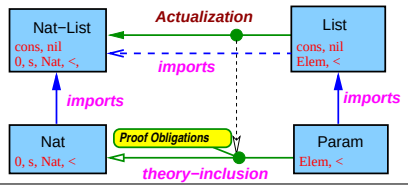
Theory level: Development Graph • inheritance via symbol-mapping • theory-inclusion by proof-obligations • local (one-step) vs. global links	

Figure 2: OMDoc in a Nutshell (the three levels of modeling)

All levels are augmented by markup for various auxiliary information that is present in mathematical documents, e.g. notation declarations, exercises, experimental data, program code, etc.

3 The sTeX Packages

In this section we will describe the sTeX distribution, which is available from the author upon request. We expect to release future versions of sTeX under the Gnu Lesser General

Public License (LGPL [FSF99]) at `http://www.faculty.iu-bremen.de/mkohlhase/kwarc` when they have stabilized further.

The STEX distribution is consists of several groups of files which share a common root, which we will collectively call an STEX package. An STEX package has (at least) four parts:

LATEX Macro Package (extension `.sty`) that supplies the LATEX macro definitions so that the LATEX document can be formatted.

LATEXML **Constructor Package** (extension `.ltxml`) that defines the LATEXML constructors and environments enabling the LATEXML program to produce semantically enriched XML.

Document Type Definition (DTD) (extensions `.dtd` for top-level DTDs, `.mod` for DTD modules, and `.ent` for entities of DTD modules) that specifies the intended XML output format

XSLT **Module** for the semantic post-processing phase of LATEXML

Thus an STEX package `doc` consists of files `doc.sty`, `doc.ltxml`, `doc.dtd`, `doc.mod`, `doc.ent`, `doc.xsl`.

3.1 Semantic Markup for Mathematical Statements

The LATEX style file `statements.sty` is the semantic basis for annotating mathematical statements (the text fragments for definitions, theorems, proofs,...) in the LATEX sources with semantic information so that it can be transformed to XML and hence to OMDOC via LATEXML. Let us look at the example in Figure 3 to get a feeling for the style of markup.

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. □

Figure 3: A Theorem with a Proof in Presentation-LATEX

We see a presentation of a theorem with a proof, as we would find it on a slide in a beginners course on natural numbers. Using the annotation infrastructure provided by the `statements` package, the structure of the discourse can be marked up using the specialized environments `assertion`, `proof` on the top-level, and the proof structure that is presented

as nested itemized lists in Figure 3 is classified as proof steps, a case analysis, justifications, etc in Figure 4.

```
\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{pfcase}
      \end{pfcases}
    \end{justification}
  \end{step}
  \begin{pfcomment}
    Thus we have considered all the cases and proven the theorem.
  \end{pfcomment}
\end{proof}
```

Figure 4: A Theorem with a Proof marked up as Statements in

All of these environments take keyword arguments (using David Carlisle's `keyval` [Car99] package). Currently the keys are `id`, `for`, `prefix`, `type`, `display`, `continues` for statements, and `method`, `premises`, and `args` for justifications to augment the segmentation and classification of text fragments by the environments with semantic and context information. Of course, the LaTeX macros and environments are defined to re-create the presentation in Figure 3, so that the changed representation is not visible to the reader. Upon transformation, `latexml` transforms this into the OMDoc document in Figure 5.

The advantage for the LaTeX user is obvious: she does not have to cope with the XML syntax, does not have to learn the particular (unfamiliar) syntax of OMDoc documents, and does not have to supply information that can be inferred or defaulted. In essence a LaTeX author can use the tools she is accustomed to. Moreover, the LaTeX document can be the original in the document creation work-flow, and can therefore be edited and maintained by the original author. In fact, the content markup in the source document can also be used for other purposes, for instance, more aspects than before can be treated by LaTeX styles giving a flexible but uniform look and feel to documents, and structural constraints can be checked (e.g. have all the premises in a proof been introduced in the document?).

```
<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>
        </derive>
      </proof>
    </method>
  </derive>
  <omtext id="te1">
    <CMP>Thus we have considered all the cases and proven the theorem.</CMP>
  </omtext>
</proof>
```

Figure 5: The Content of Figure 3 in OMDoc

3.2 Explicit vs. Implicit Statement Markup

As we have seen in Figure 3, some structure of mathematical statements is given explicitly by graphic or linguistic cues, e.g. the boldface word "**Theorem:**" or the little box at the end of the proof and must be reproduced in the package. This kind of statements usually occupies a full paragraph. Other statements may not even occupy a full sentence, such as "*... is of the form $R(x,x)$, which we call a diagonal relation...*". To account for this, we distinguish two forms of statements: **block statements** have explicit discourse markers that delimit their content in the surrounding text, and **flow statements** that do not have explicit markers, they are interspersed with the surrounding text. Since they have the same semantic status, they must both be marked up, but styled differently. This is specified by the keyword `display=` in the statement environments, which can have the values `block` and `flow`.

Furthermore, mathematical documents often collect coherent statements into groups that share a common text prefix (a text snippet that sets up an environment for the statements) this is modeled by the `statement-group` environment, e.g. the one in Figure 6, which gives the result in Figure 7.

```
\begin{statement-group}{A function $f\colon X \rightarrow Y$ is called}
  \begin{itemize}
  \item \begin{definition}[id=injective,display=flow]{\defemph{injective}} iff
      $\allcdot{x,y\in X}{f(x) = f(y)\Rightarrow x=y}$.
    \end{definition}
  \item \begin{definition}[id=surjective,display=flow]{\defemph{surjective}} iff
      $\allcdot{y\in Y}{\excdot{x\in X}{f(x)=y}}$.
    \end{definition}
  \item \begin{definition}[id=bijective,display=flow]{\defemph{bijective}} iff $f$
      is injective and surjective.
    \end{definition}
  \end{itemize}
\end{statement-group}
```

Figure 6: A Statement Group of Definitions

A function $f\colon X \to Y$ is called

- **injective** iff $\forall x, y \in X. f(x) = f(y) \Rightarrow x = y$.
- **surjective** iff $\forall y \in Y. \exists x \in X. f(x) = y$.
- **bijective** iff f is injective and surjective.

Figure 7: A Statement Group of Definitions

3.3 STEX Modules solve the Notation/Context Problem

In Section 2.1, we have identified two problems, the *reconstruction problem*, we have evaded by enlisting the author to preload the document with semantic macros. The *Notation/Context problem*, can partially be solved by semantic macros as well, since they can be used to disambiguate between different semantic usages of notations. For the context problem we note that *the context of notations coincides with the context of the concepts they denote*. Thus the main idea for solving the context problem is to adapt the mechanism for concept scoping known from MKM languages to TEX/LATEX. In STEX, we inherit our intuition from the OMDoc format, which in turn builds on work in computational logic [FGT92, Far00], and algebraic specification (see e.g. [CoF98, AHMS00]). In these framework, scoping of concepts is governed by a grouping in collections of mathematical statements called "*theories*" or "*modules*", where the inheritance of concepts in theories is explicitly expressed in an inheritance relation.

Note that the scoping facilities offered by the TEX/LATEX format do not allow us to model these scoping rules. The visibility of semantic macros, like any TEX macros, is governed by the (hierarchical) grouping facility in TEX. In a nutshell, a TEX macro is either globally defined or defined exactly inside the group given by the group induced curly braces hierarchy.

In STEX, the package `statements` provides the LATEX environment `module` for specifying the theory structure and uses the macros `\symdef`, `\abbrdef elldef` for defining macros; the three definition mechanisms correspond to the three classes of macros discussed in Section 2.1. Like theories in OMDoc, the `module` environment governs the visibility of semantic macros in LATEX. A semantic macro is visible in its "home module" and in all modules that import macros from it. To get an intuition for the situation, let us consider the example in Figure 8.

Here we have four modules: `pairs`, `sets`, `setoid`, and `semigroup` where `setoid` imports semantic macros from the first two, and the last imports from it. We can see that macro

```
\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}
```

Figure 8: The Module Structure

visibility is governed by the `uses` relation specified in the keyword arguments to the `module` environment. In particular, the macros `\pair` and `\sset` are defined in modules `setoid` and `semigroup` (since the `uses` relation is transitive). With these symbol definitions, we get the text in Figure 9.

Note that the inheritance hierarchy does allow multiple inheritance. Generally, the `uses` relation on modules should be a directed acyclic graph (no inheritance cycles). In a case of a `\symdef` conflict, the first (leftmost in the inheritance tree induced by the `uses` relation) is taken.

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 $a, b, c \in \mathcal{S}$.

Figure 9: The Result of Modules in Figure 8

Note that the use of sTEX modules moves macro definitions that have traditionally been moved into separate files in the TEX/LATEX community back into the documents themselves. This is akin to the organization of functionality in object-oriented programming. The main reason for this is what is often called the "*late binding*" in programming. Depending on the

viewpoint, late binding can be a problem or feature: in content-oriented document management, late binding of style information is used to adapt presentation, in programming, late binding of program (changing) modules may cause problems with program semantics. We view late binding for semantic macros as a problem (we do not want to change the semantics), so we advise to use the modules approach presented here for semantic preloading. In particular in our experience, modules are the ideal candidates for re-use in semantically marked-up mathematical documents, as they are semantically and ontologically self-contained and the remaining dependency on context is made explicit by the inheritance relation.

3.4 LaTeXML Bindings for Semantic and Elliptive Macros

For the transformation into Xml, the `latexml` program needs to have access to the respective `DefConstructor` directives (see Section 2.3) for semantic macros; for abbreviative macros they are not needed, as `latexml` just reads the macro definition in the `\abbrdef` definition.

In our approach, sTeX supports modular specification of LaTeXML bindings for semantic macros using the `\latexmldef` companion to `\symdef`. This macro takes two arguments: The macro invocation pattern and the Xml pattern. For the symbol definition in the second line of Figure 8, we would add something like

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

or more concisely (using special abbreviative syntax):

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

which has the same effect of supplying the LaTeXML program with the necessary `DefConstructor` declarations. Of course, with a suitable extension, this approach would also be suitable for other TeX/LaTeX to Xml transformations.

Let us finally come to elliptive macros, these differ from truly semantic macros only in their LaTeXML binding, which must reflect the elision, as we have argued in Section 2.1. Let us consider the concrete examples of the elliptive macros in Figure 1.

```
\symdef{\cM}{{\cal M}}\symdef{\assign}{\phi} % constant, but unspecified model, assignment
\symdef{\interpret}[3]{{\left[\kern-0.18em\left[#1\right]\kern-0.18em\right]^{#2}_{#3}}}
\latexmlconstructor{\interpret}[args=3,cd=booleval,name=interpret]
\abbrdef{\intermp}[1]{\interpret{#1}{\cM}{\phi}}
\elldef{\interm}[2]{\interpret{#1}{\cM}{}}\latexmlelide{\interm}{3}{\interpret}
\elldef{\interp}[2]{\interpret{#1}{}{\assign}}\latexmlelide{\interp}{2}{\interpret}
\def{\interoo}[3]{\interpret{#1}{}{}}\latexmlelide{\interp}{2,3}{\interpret}
```

Here the `\latexmlelide` macro takes three arguments, the first is the control sequence for the elliptive macro, the second is a comma-separated list of arguments to be elided, and the third is the control sequence of the semantic macro, that this macro is an elliptive variant of. In our example, the macro `\latexmlelide{\interp}{2}{\interpret}` is equivalent to the declaration

```
\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>}
```

Here, the additional attribute elide on the `XMArg` element in the TEX-near representation (see Section 2.3) gives the hint to the post-processing that this argument should be elided in the target format. For instance, in a situation, where the model $\mathcal{M}$ is clear from the context, we would semantically mark up the formula $[\![A]\!]_\varphi$ (which actually stands for $[\![A]\!]_\varphi^{\mathcal{M}}$) as `\interm{A}{\cM}` in the TEX/LATEX source. Note that the macro `\interm` is mixed elliptive and abbreviative; the variable assignment φ is abbreviated by the macro, and the model is elided. Note furthermore, that this approach to elliptive macros assumes that the target format has a hinting system for the presentation engine. In the case of OMDoc, we can just make use of the Css system; LaTeXML post-processing would result in the OMDoc/OpenMath representation

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

for the elliptive notation `\interm{A}{\cM}`. In the simplest form of the OMDoc presentation, the Css `style` attribute is copied to the resulting XHtML+MathML output, where it instructs the user agent not to display the element.

3.5 Miscellaneous Packages

In our case study, we used an extension `mikoslides.cls` of the LATEX `seminar` class for representing slides. This class supplies the top-level structure for slides, it can easily replaced by any other top-level class in LATEX.

Instead of using Copy and Paste in the source document it is better to share the document content and let the formatter do the copying. We have also used the author's `structuresharing` package, which provides the macros `\STRlabel` and `\STRcopy`, and extended it with `\STRsemantics`. The `\STRlabel` macro takes two arguments: a label and the content and stores the the content for later use by `\STRcopy{label}`, which expands to the previously stored content. The `\STRlabel` macro has a variant `\STRsemantics`, where the label argument is optional, and which takes a third argument, which is ignored in LaTeX. This allows to specify the meaning of the content (whatever that may mean) in cases, where the source document is not formatted for presentation, but is transformed into some content markup format.

4 Conclusion and Future Work

We have presented a system STEX of macro-packages for semantic annotation of TEX/LATEX documents and an extension of the LaTeXML system with bindings for the STEX package. This allows us to semantically pre-load TEX/LATEX documents and transform them into Xml and ultimately into the OMDoc format.

The system is being tested on a first-year computer science course at International University Bremen. The next case study will be the OMDoc 1.2 report [Koh04].

In essence, the STEX package together with its LaTeXML bindings forms an invasive editor for OMDoc in the sense discussed in [KK04]: The author can stay in her accustomed

work-flow; in the case of TEX/LATEX, she can use the preferred text editor to "program" documents consisting of text, formulae, and control sequences for macros. The documents can even be presented in by the TEX formatter in the usual way. Only with the semantic preloading, they can be interpreted as MKM formats that contain the necessary semantic information, and can even be transformed into explicit MKM formats like OMDoc.

Thus the STEX/LaTeXML combination extends the available invasive editors for OMDoc to three (CPoint [KK04] and nb2omdoc [Sut04] being those for PPT and Mathematica). This covers the paradigmatic examples of scientific document creation formats (with the exception of MS Word a possible porting target of the VBA-based application CPoint).

In the future, we plan to extend the system with a CnXML back-end, which produces the input format for Connexions and e-learning document management system at Rice University. In particular, we will include the LATEX package [WHB03] developed there.

Acknowledgments

This work has profited significantly from discussions with Bruce Miller and Ioan Sucan. The former is the author of the LaTeXML system and has extended his system readily to meet the new demands from our project. The latter is a student a IUB who has faithfully carried the brunt of the editing load involved with semantic pre-loading the LATEX slides. Finally I am indebted to David Carlisle who helped me with the non-trivial hacking involved in getting the modules to work.

References

[ABC+03] Ron Ausbrooks, Stephen Buswell, David Carlisle, Stphane Dalmas, Stan Devitt, Angel Diaz, Max Froumentin, Roger Hunter, Patrick Ion, Michael Kohlhase, Robert Miner, Nico Poppelier, Bruce Smith, Neil Soiffer, Robert Sutor, and Stephen Watt. Mathematical Markup Language (MathML) version 2.0 (second edition). W3c recommendation, World Wide Web Consortium, 2003. Available at http://www.w3.org/TR/MathML2.

[AHMS00] Serge Autexier, Dieter Hutter, Heiko Mantel, and Axel Schairer. Towards an evolutionary formal software-development using CASL. In C. Choppy and D. Bert, editors, *Proceedings Workshop on Algebraic Development Techniques, WADT-99*, number 1827 in LNCS. Springer, 2000.

[Ang] Romeo Anghelache. *Hermes: A content oriented LaTeX to XML+MathML conversion/authoring tool*. Web Manual at `http://www.psyx.org/hermes/doc.html`.

[BCC+04] Stephen Buswell, Olga Caprotti, David P. Carlisle, Michael C. Dewar, Marc Gaetano, and Michael Kohlhase. The Open Math standard, version 2.0. Technical report, The Open Math Society, 2004. `http://www.openmath.org/standard/om20`.

[Ber89] J. A. Bergstra. *Algebraic specification*. ACM Press, 1989.

[Bou74] Nicolas Bourbaki. *Algebra I*. Elements of Mahtematics. Springer Verlag, 1974.

[Car99] David Carlisle. *The* keyval *package*. The Comprehensive TEX Archive Network, 1999. Part of the TEX distribution.

[CoF98] Language Design Task Group CoFI. Casl — the CoFI algebraic specification language — summary, version 1.0. Technical report, `http://www.brics.dk/Projects/CoFI`, 1998.

[DLL+04] Christoph Dwertmann, Arne Lindow, Christoph Lüth, Markus Roggenbach, and Jan-Georg Smaus. *How to use and configure MMiSSLATEX*, 2004. avaialble at `http://www.informatik.uni-bremen.de/mmiss/tools_e.htm`.

[Far00] William Farmer. An infrastructure for intertheory reasoning. In David McAllester, editor, *Automated Deduction – CADE-17*, number 1831 in LNAI, pages 115–131. Springer Verlag, 2000.

[FGT92] William Farmer, Josuah Guttman, and Xavier Thayer. Little theories. In D. Kapur, editor, *Proceedings of the 11th Conference on Automated Deduction*, volume 607 of *LNCS*, pages 467–581, Saratoga Spings, NY, USA, 1992. Springer Verlag.

[FSF99] Free Software Foundation FSF. GNU lesser general public license. Software License available at `http://www.gnu.org/copyleft/lesser.html`, 1999.

[KK04] Andrea Kohlhase and Michael Kohlhase. CPoint: Dissolving the author's dilemma. In Andrea Asperti, Grzegorz Bancerek, and Andrej Trybulec, editors, *Mathematical Knowledge Management, MKM'04*, number 3119 in LNAI. Springer Verlag, 2004. forthcoming.

[Knu84] Donald E. Knuth. *The TEXbook*. Addison Wesley, 1984.

[Koh01] Michael Kohlhase. OMDoc: An open markup format for mathematical documents (version 1.1), 2001. `http://www.mathweb.org/omdoc/omdoc.ps`.

[Koh04] Michael Kohlhase. OMDoc an open markup format for mathematical documents (version 1.2), 2004. Manuscript, `http://www.mathweb.org/omdoc/omdoc1.2.ps`.

[Lam94] Leslie Lamport. *LATEX: A Document Preparation System, 2/e*. Addison Wesley, 1994.

[MAF+01] E. Melis, J. Buedenbender E. Andres, Adrian Frischauf, G. Goguadze, P. Libbrecht, M. Pollet, and C. Ullrich. The ACTIVEMATH learning environment. *Artificial Intelligence and Education*, 12(4), winter 2001 2001.

[Mat] Online Repository at `http://www.mathml.ca`.

[Mil] Bruce Miller. `LaTeXML`: A LATEX to xml converter. Web Manual at `http://dlmf.nist.gov/LaTeXML/`.

[RV01] Alan Robinson and Andrei Voronkov, editors. *Handbook of Automated Reasoning*, volume I and II. Elsevier Science and MIT Press, 2001.

[Sut04] Klaus Sutner. Converting MATHEMATICA notebooks to OMDoc. to appear in [Koh04], 2004.

[vdBS03] Mark van den Brand and Jürgen Stuber. Extracting mathematical semantics from latex documents. In *Proc. Intl. Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR 2003)*, number 2901 in LNCS, pages 160–173, Mumbai, India, 2003. Springer.

[WHB03] Rebecca Willett, Brent Hendricks, and Richard Baraniuk. CnxTeX - a LATEX style file to facilitate LATEX-to-xml conversion. available at `http://dsp.rice.edu/software`, 2003.