Using LaTeX as a Semantic Markup Format

Michael Kohlhase
Computer Science
Jacobs University Bremen, Germany
http://kwarc.info/kohlhase

Abstract. 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, but mathematicians and publishers in Mathematics have heavily invested into the TEX/LaTeX format and workflow.

We analyze the current practice of semi-semantic markup in LaTeX documents and extend it by a markup infrastructure that allows to embed semantic annotations into LaTeX documents without changing their visual appearance. This collection of TEX macro packages is called sTEX (semantic TEX) as it allows to markup LaTeX documents semantically without leaving the time-tried TEX/LaTeX workflow, essentially turning LaTeX into an MKM format. At the heart of sTEX is a definition mechanism for semantic macros for mathematical objects and a non-standard scoping construct for them, which is oriented at the semantic dependency relation rather than the document structure.

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

1. Introduction

We will use the term **MKM format** for 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 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 XML-based, content-oriented markup languages for mathematics on the web, e.g. OpenMath [BCC+04], Content-MathML [ABC+03], or our own OMDoc [Koh06b]. The MKM community develops these languages with the aim to enable a *content commons* [HBK03], a large, community-developed body of semi-formalized mathematics that can serve as a vastly improved common resource for learning, teaching, and research in mathematics [Far05].

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. The latter allows to extend the functionality of the language, essentially turning TEX into a meta-format for developing task-specific vocabularies. This is utilized in LATEX (essentially a set of TEX macro packages, see [Lam94b]) 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 — supplying macros e.g. for sections, paragraphs, theorems, definitions, etc., and graphics, 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 already one. In particular, 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 article, 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 an author to add semantic information to the document in a way that does not change the visual appearance[1]. We speak of semantic preloading for this process and call our collection of macro packages sTEX (Semantic TEX)[2]. Thus, sTEX can serve as a conceptual interface between the document author and MKM systems: Technically, the semantically preloaded LATEX documents are transformed into (usually XML-based) MKM representation formats, but conceptually, the ability to semantically annotate the source document is sufficient for MKM.

Concretely, we will present the sTEX macro packages together with a case study, where we semantically preloaded the course materials for a two-semester course in Computer Science at Jacobs University and transformed them to the OMDoc MKM format. For this we used the LaTeXML system (see section 4.1). As a consequence, these materials can now be used in MKM systems like ActiveMath [MBA+01], SWiM [LK08], or *panta rhei* [MK07].

Before we go into the details of sTEX and conversion, 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 derive a methodology for semantic preloading from this, which is implemented in the sTEX packages we

[1] 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.

[2] sTEX is available from the CTAN network [CTA] and the development versions from [sTea].

present in Section 3. Then we will survey TeX/LaTeX conversion tools and look at the LaTeXML system which we have used in our case study.

2. Semantic Preloading of LaTeX Documents

Much of the semantic content in LaTeX documents is represented only implicitly, and has to be decoded by (human) readers for understanding and processing. For MKM purposes, i.e. for machine-processing this implicit content must be made explicit, which is a non-trivial task. To get a feeling for the intended target we will first take a look at the available MKM formats available for scientific documents before we describe the preloading process itself.

2.1. A Mathematical Knowledge Model for LaTeX Documents

First, the *Semantic Web* [BL98] is an approach to knowledge management that aims to be web-scalable. The underlying knowledge representation is provided in an ontology formalism like OWL-DL [MvH04]. This representation format is intentionally limited in its semantic expressiveness, so that inference stays decidable and web-scalable. Unfortunately, scientific knowledge can be only approximated very coarsely using this approach so far.

In contrast, the field of *Formal Methods* [Win90] use semantic formats with highly expressive knowledge representation components (usually first-order or higher-order logics). They are currently only used for security sensitive applications, such as formal program verification, since on the one hand they require the commitment to a particular logical system, and on the other hand the mathematical-logical formalization needed for formal verification is extremely time-consuming.

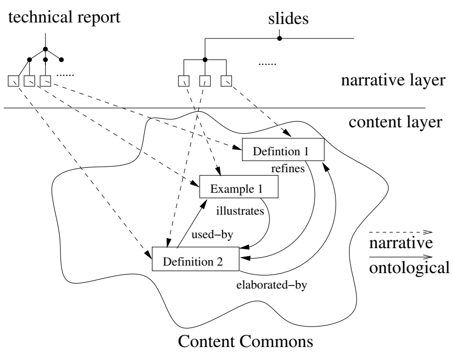
In contrast to those, the **structural/semantic approach** taken e.g. by the OMDoc [Koh06b] format does not require the full formalization of mathematical knowledge, but only the explicit markup of important structural properties. For instance, a statement will already be considered as "true" if there is a proof object that has certain structural properties, not only if there is a formally verifiable proof for it. Since the structural properties are logic-independent, a commitment to a particular logical system can be avoided without losing the automatic knowledge management, which is missing for semantically unannotated documents. Such document formats use a four-layered structure model of knowledge.

Object level. This represents objects such as complex numbers, derivatives, etc. for mathematics, molecules in chemistry, map specifiers for geo-sciences, or observables for physics. Semantic representation formats typically use functional characterizations that represent objects in terms of their logical structure, rather than specifying their presentation. This avoids ambiguities which would otherwise arise from domain specific representations.

Statement Level. The (natural/social/technological) sciences are concerned with modeling our environment, or more precisely, with statements about the objects in it. We can distinguish different types of statements, including model assumptions, their consequences, hypotheses, and measurement results. All of them have in common that they state relationships between objects and have to be verified or falsified in theories or experiments. Moreover, all these statements have a conventionalized structure, and a standardized set of relations among each other. For instance, a model is fully determined by its assumptions (also called *axioms*); all consequences are deductively derived from them (via *theorems* and *proofs*); hence, their experimental falsification uncovers false assumptions of the model. Proofs are only one example of *provenance information* that is encoded in the statement level, the trail from a measurement, via data processing, to presentation in a chart is another.

Theory/Context Level. Representations always depend on the ontological context; even the meaning of a single symbol is determined by its context — e.g. the glyph h can stand for the height of a triangle or Planck's quantum of action — and depending on the current assumptions, a statement can be true or false. Therefore, the sciences (with mathematics leading the way) have formed the habit of fixing and describing the context of a statement. Unfortunately, the structure of these context descriptions remain totally implicit, and thus cannot be used for computer-supported management. Semantic representation formats make this structure explicit. For instance in mathematical logic, a theory is the deductive closure of a set of axioms, that is, the (in general infinite) set of logical consequences of the model assumptions. Even though in principle this fully explains the phenomenon of context, important aspects like the re-use of theories, knowledge inheritance, and the management of theory changes are disregarded completely. Hence, formalisms that have a context level use elaborate inheritance structures for theories, e.g. in the form of ontologies for the Semantic Web or as "algebraic specifications" in program verification.

Document Level. The structural/semantic formats support the separation of content and form on the discourse level. Concretely, documents are split into *narrative* and *content* layers, as suggested in the figure on the right: the lower level of the diagram represents the content of the knowledge (structured by the inherent semantic relations of the objects involved), and the upper part the form (structured, so that humans are motivated to concern themselves with the material, understand why some definitions are stated in just this way, and get the new information in easily digestible portions) see [Koh06b, KMM07] for details.



Of course, some of the features discussed here are not unique to OMDoc: for instance the format CNXML [HG07] used by the Connexions project [Tea06] covers the object layer, the documents layer, and part of the statement layer introduced above. Similarly, the LaTeX-based MMiSS format [KBLL+04] covers the statement- and (parts of) the context level. Finally, the OpenMath [BCC+04], MathML [ABC+03], and CML (Chemistry Markup Language) [MR+07] provide strong object levels representation infrastructures specialized to their respective disciplines, and have a flexible mechanism of meaning assignment via a simple context layer.

2.2. Semantic Macros in LaTeX

We will make use of the TeX macro mechanism for semantic 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.pdf` in a presentation format, which can be directly rendered for screen- or printer output. This basic and very simple mechanism can be put to various uses in documents: macros can be used e.g. to compute values for section numbers or footnotes making TeX sources more portable, they can be used as abbreviations to save the author typing effort, or they can be used for semantic annotation, which is what we will explore here. All of these uses do occur[3] in LaTeX documents. For our purposes here we distinguish them by their intent: **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. In contrast to this, **semantic macros** are used to represent the objects of discourse 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{\mathcal{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{\mathcal{C}^\infty(#1)}
\def\SmoothFunctionsOnReals{\SmoothFunctionsOn\Reals}
```

Semantic macros are commonly used to enhance the maintainability and reusability of source code. 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. We will now look at the problems involved in preloading documents at the different levels introduced in the last section.

[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.

2.3. Semantic Preloading at the Document and Statement Levels

We can consider many LaTeX macros and environments — e.g. the sectioning and front matter infrastructure as semantic macros at the document level. They specify the functional role of text fragments in the larger document context and leave the presentation of the respective visual cues to style files. The only difference to the semantic macros discussed above is that they make the document function explicit, whereas the ones above talk about mathematical objects. In fact, LaTeX offers a staggering lot of specialized classes for different types of documents. The narrative/content distinction is implicitly inscribed in them, but the content commons is difficult to realize, as TeX only supports the inclusion of whole files via the `\input` and `\include` commands. To build an information architecture based on referencing and linking content objects is possible but involves managing large collections of snippet files (see e.g. [BFGHS04] for a system based on this idea or [sTec] for the snippets of our case study).

On the statement level, the situation is similar, LaTeX supplies a variety of packages to mark up statements (including proofs); most notably the `amsthm` package. However, such packages largely neglect the relations between statements and leave them to be figured out from the presentation by a human reader; even Leslie Lamport's proposal in [Lam94a] failed to catch on in mathematics. Two notable exceptions are the MMiSSLaTeX [DLL+04] and Salt [GHMD07] approaches, which allow to annotate LaTeX documents with ontological relations to interrelate scientific documents and identify claims in them for search purposes. The MathLang format [KMW07] goes still a little farther, extending the semantic relations to the sub-sentence level, but is only implemented in the TeXmacs editor, not directly in LaTeX.

2.4. Semantic Preloading at the Object and Context Levels

While the situation of preloading LaTeX documents at the statement and document levels is reasonably well-understood and we have various implemented and tested approaches, the other two levels are largely untouched. The context level is traditionally left implicit in mathematical discourse, and at the object level, mathematicians rely on the extraordinary ability of mathematically literate readers to infer the functional structure from formulae. In this structure recovery process, we can distinguish three different — albeit interrelated — problems:

The Structural Disambiguation Problem. Over the last three millennia, mathematics has developed a complicated two-dimensional format for communicating formulae (see e. g. [Caj93, Wol00] for details). Structural properties of operators often result in special presentations, e.g. the scope of a radical expression is visualized by the length of its bar. Their mathematical properties give rise to placement (e.g. associative arithmetic operators are written infix), and their relative importance is expressed in terms of binding strength conventions which govern bracket elision. Changes in notation have been influential in shaping the way we calculate and think about mathematical concepts, and understanding mathematical notations is an essential part of any mathematics education. All of these make it difficult to determine the functional structure of an expression from its presentation.

The Elision Reconstruction Problem. Mathematical communication and notation makes use of 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.

The Notation/Context Problem. Mathematicians have idiosyncratic notations that are introduced, extended, and discarded on the fly. Generally, this means that the parsing and meaning 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

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

the first and third occurrence of the symbol α is the 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. Additionally, notation can depend on other forms of context. For instance, we have varied "standard notations" for binomial coefficients: $\binom{n}{k}$, ${}_nC^k$, C^n_k, and C^k_n all mean the same: $\frac{n!}{k!(n-k)!}$; the third notation is the French standard, whereas the last is Russian.

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. Similarly, 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`.

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].

All three problems have to be solved for a successful transformation of mathematical documents 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 to determine the semantics.

3. The sTeX Packages

We have seen above that for supporting the preloading of LaTeX documents, we have to provide semantic macros at all four levels of knowledge. We have developed these for two different document types: lecture slides for an entry-level course "General Computer Science" (GenCS), and course modules in the Connexions system. The first one focuses on semantic markup for an extensible vocabulary in the OMDoc format, whereas the second is geared at a faithful representation of the CnXML document model including its fixed content MathML representation of formulae. Even though these two document formats differ considerably, they share a lot of the underlying machinery, which suggests that this is independent of the particular MKM format, and therefore of general value. We will not go into the document-level infrastructure here, since it is rather straightforward. In the GenCS case this is a semantic extension of the `beamer` class with respect to OMDoc document elements, in the Connexions case, a custom-built, LaTeX document class, which also incorporates an infrastructure for the (relatively simple) CnXML statement level.

We will present the sTeX infrastructure for the OMDoc statement level, since it is rather complete and can be re-used in other document formats (see section 3.1). But the main contribution of the sTeX format is at the context- and object levels. In Section 2 we have identified three problems:

- The *structural disambiguation problem* can be solved by letting the author directly mark up the content form. To make this invisible in the presentation (i.e. the result of LaTeX formatting), we have to provide a potent formula presentation infrastructure that allows to recover the intended form from the sTeX markup and notation definitions. This is a central piece of the sTeX infrastructure, we will present it in section 3.2.
- 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*, which needs

to be represented. Therefore sTeX provides an infrastructure for context based on `modules` (see section 3.4)

- The *reconstruction problem*, we largely evade by enlisting the author to preload the document with semantic macros. To support this, sTeX provides a special infrastructure for flexible elision (see section 3.3).

All sTeX packages and classes are licensed under the LaTeX Project Public License (LPPL [Pro07]). The released version is available from the CTAN network [CTA] and the development versions from [sTea]. An sTeX package ⟪*name*⟫ is distributed as self-documenting LaTeX packages ⟪*name*⟫`.dtx` that contain

- documentation which can be extracted by running `latex` ⟪*name*⟫`.dtx`.
- a LaTeX macro package ⟪*name*⟫`.sty` that supplies the LaTeX macro definitions so that the LaTeX document can be formatted. It can be extracted by running the `latex` program over ⟪*name*⟫`.ins`, and
- a LaTeXML bindings package ⟪*name*⟫`.sty.ltxml` that defines the LaTeXML constructors and environments enabling the LaTeXML program to produce semantically enriched XML (see section 4.1 for an introduction). It can also be extracted by running the `latex` program over ⟪*name*⟫`.ins`.

3.1. Semantic Markup for Mathematical Statements

The LaTeX packages `statements.sty` and `sproof.sty` are the semantic basis for preloading mathematical statements (the text fragments for definitions, theorems, proofs,...) in the LaTeX documents. Let us look at the example in Figure 1 and at is preloaded LaTeX form in Figure 2 to get a feeling for the style of semantic markup of mathematical statements.

Theorem: *Let S_0, S_1, ... be a linear sequence of dominoes. If S_0 is pushed towards S_1 so that it falls, then all dominoes will fall.*

Proof: We prove that S_i falls in the direction of S_{i+1} by induction over i.

1. We have to consider two cases:
2. **Base case** ($i = 0$): We have assumed that "S_0 is pushed towards S_1, so that it falls"
3. **Step case** ($i > 0$):
   (a) We assume that S_{i-1} falls in the direction of S_i.
   (b) Thus it hits S_i and causes it to fall in the same direction, i.e. towards S_{i+1}.
4. Now, the assertion follows trivially, since if "S_i falls in the direction of S_{i+1}", then in particular "S_i falls". □

FIGURE 1. A Theorem with a Proof in Presentation-LaTeX

We see a presentation of a theorem with a proof, as we would find it in a beginners' textbook. Using the sTeX annotation infrastructure, the top-level structure of the discourse can be marked up using the specialized environments `assertion` and `sproof`. The proof structure that is presented as nested itemized lists in Figure 1 is classified as proof steps, a case analysis, justifications, etc. in Figure 2.

```
\begin{assertion}[type=Theorem,id=domino-thm]
  Let $S_0$, $S_1$, \ldots be a linear sequence of dominoes. If $S_0$ is pushed
  towards $S_1$ so that it falls, then all dominoes will fall.
\end{assertion}
\begin{sproof}[for=domino-thm,id=domino-pf]{We prove that $S_i$ falls in the
    direction of $S_{i+1}$ by induction over $i$.}
  \begin{spfcases}{We have to consider two cases}
    \begin{spfcase}{Base case ($i=0$)}
      \begin{step}
        We have assumed that ``$S_0$ is pushed towards $S_1$, so that it falls''
      \end{step}
    \end{spfcase}
    \begin{spfcase}{Step case ($i>0$)}
      \begin{step}
        We assume that $S_{i-1}$ falls in the direction of $S_i$.
      \end{step}
      \begin{step}
        Thus it hits $S_i$ and causes it to fall in the same direction,
        i.e. towards $S_{i+1}$.
      \end{step}
    \end{spfcase}
  \end{spfcases}
  \begin{step}
    Now, the assertion follows trivially, since if ``$S_i$ falls in the
    direction of $S_{i+1}$'', then in particular ``$S_i$ falls''.
  \end{step}
\end{sproof}
```

FIGURE 2. A Theorem with a Proof Marked Up as Statements in sTeX

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 1, so that the changed representation is not visible to the reader.

3.2. Symbol Presentations for Structural Disambiguation

The `presentation` package supplies an infrastructure that allows to specify the presentation of semantic macros, including preference-based bracket elision. This allows to markup the functional structure of mathematical formulae without having to lose high-quality human-oriented presentation in LaTeX. Moreover, the notation definitions can be used by MKM systems for added-value services, either directly from the sTeX sources, or after translation. The setup for semantic macros via the `\symdef` form described in the `modules` package (see section 3.4) works well for simple mathematical functions: we make use of the macro application syntax in TeX to express function application. For a simple function called "foo", we would just declare `\symdef{foo}[1]{\prefix{foo}{#1}}` and have the concise and intuitive syntax `\foo{x}` for $foo(x)$. sTeX also includes a package `cmathml` for writing content MathML [ABC+03] expression, a set of about 100 functions for K-14[4] mathematics as part of the CONNEXIONS case study [Koh06a]. We will not go into this here, since it is a finite set that can be fixed with conventional methods. But mathematical notation is much more varied and interesting than just this: Many commonly-used functions deviate from the form $f(a_1, \ldots, a_n)$, where f is the function and the a_i are the arguments. For instance binomial coefficients: $\binom{n}{k}$, pairs: $\langle a, b\rangle$, sets: $\{x \in S \,|\, x^2 \neq 0\}$, or even simple addition: $3 + 5 + 7$. Note that in all these cases, the presentation is determined by the (functional) head of the expression, so we will bind the presentational infrastructure to the operator.

Mixfix Notations. For the presentation of ordinary operators, we will follow the approach used by the Isabelle theorem prover [NPW02]. There, the presentation of an n-ary function (i.e. one that takes n arguments) is specified as $\langle\!\langle pre\rangle\!\rangle\ \langle\!\langle arg_0\rangle\!\rangle\ \langle\!\langle mid_1\rangle\!\rangle \cdots \langle\!\langle mid_n\rangle\!\rangle\ \langle\!\langle arg_n\rangle\!\rangle\ \langle\!\langle post\rangle\!\rangle$, where the $\langle\!\langle arg_i\rangle\!\rangle$ are the arguments and $\langle\!\langle pre\rangle\!\rangle$, $\langle\!\langle post\rangle\!\rangle$, and the $\langle\!\langle mid_i\rangle\!\rangle$ are presentational material. For instance, in infix operators like the binary subset operator $\langle\!\langle pre\rangle\!\rangle$ and $\langle\!\langle post\rangle\!\rangle$ are empty, and $\langle\!\langle mid_1\rangle\!\rangle$ is $\subseteq$. For the ternary conditional operator in a programming language, we might have the presentation pattern `if`$\langle\!\langle arg_1\rangle\!\rangle$`then`$\langle\!\langle arg_2\rangle\!\rangle$`else`$\langle\!\langle arg_3\rangle\!\rangle$`fi` that utilizes all presentation positions.

The `presentation` package provides mixfix declaration macros `\mixfixi`, `\mixfixii`, and `\mixfixiii` for unary, binary, and ternary functions. The call pattern of these macros is just the presentation pattern above. In general, the mixfix declaration of arity i has $2i + 1$ arguments, where the even-numbered ones are for the arguments of the functions and the odd-numbered ones are for presentation material. Consider for instance the mixfix declaration for a pair operator in the second line of Figure 3 on page 16. As a commonly occurring special case the `\prefix` macro allows to specify a prefix presentation for a function. Note that it is better to specify `\symdef{uminus}[1]{\prefix{-}{#1}}` than just `\symdef{uminus}[1]{-#1}`, since we can specify the bracketing behavior in the former. The `\postfix` macro is similar, only that the function is presented after the argument as for e.g. the factorial function: 5! stands for the result of applying the factorial function to the number 5. Note that the function is still the first argument to the `\postfix` macro: we would specify the presentation for the factorial function with `\symdef{factorial}[1]{\postfix{!}{#1}}`.

Specifying the presentation of *n-ary associative operators* in `\symdef` forms is not straightforward, so we provide some infrastructure for that. As we cannot predict the number of arguments for n-ary operators, we have to give them

[4] Kindergarten to early college

all at once, if we want to maintain our use of TEX macro application to specify function application. So a semantic macro for an n-ary operator will be applied as `\nunion{`$\langle\langle a_1\rangle\rangle$`,...,`$\langle\langle a_n\rangle\rangle$`}`, where the sequence of n logical arguments $\langle\langle a_i\rangle\rangle$ are supplied as one TEX argument which contains a comma-separated list. We provide variants of the mixfix declarations which deal with associative arguments. For instance, the variant `\mixfixa` allows to specify n-ary associative operators. `\mixfixa{`⟪*pre*⟫`}{`⟪*arg*⟫`}{`⟪*post*⟫`}{`⟪*op*⟫`}` specifies a presentation, where ⟪*arg*⟫ is the associative argument and ⟪*op*⟫ is the corresponding operator that is mapped over the argument list; as above ⟪*pre*⟫ and ⟪*post*⟫ are prefix and postfix presentational material. Consider for instance, presentation for aggregated set membership symbol `\mmember` in Figure 3. Here the aggregation is an associative argument list which is separated by commas in the presentation. Note that the simpler definition `\infix{\in}{#1}{#2}` would have given the same presentation at the price of being less semantic.

The `\assoc` macro is a convenient abbreviation of a `\mixfixa` that can be used in cases, where ⟪*pre*⟫ and ⟪*post*⟫ are empty (i.e. in the majority of cases). It takes two arguments: the presentation of a binary operator, and a comma-separated list of arguments. It replaces the commas in the second argument with the operator in the first one. For instance `\assoc\cup{S_1,S_2,S_3}` will be formatted to $S_1 \cup S_2 \cup S_3$. Thus we can use `\def\nunion#1{\assoc\cup{#1}}` to define the n-ary operator for set union in TEX.

Precedence-Based Bracket Elision. A good example consists in the bracket elision rules in arithmetical expressions: $ax + y$ is actually $(ax) + y$, since multiplication "binds stronger" than addition. Note that we would not consider the "invisible times" operation as another elision, but as an alternative presentation. With the infrastructure above we can define infix symbols for set union and set intersection and combine them in one formula writing

`\nunion{\ninters{a,b},\ninters{c,d}}` (2)

But this yields $((a \cap b) \cup (c \cap d))$, and not $a \cap b \cup c \cap d$ as we would like, since $\cap$ binds stronger than $\cup$. Dropping outer brackets in the operators will not help in general: it gives the desired form for (2) but also $a \cup b \cap c \cup d$ for (3), where we would have liked to see $(a \cup b) \cap (c \cup d)$.

`\ninters{\nunion{a,b},\nunion{c,d}}` (3)

In mathematics, brackets are elided, whenever the author anticipates that the reader can understand the formula without them, or would be overwhelmed with them. To achieve this, there are sets of common conventions that govern bracket elision. The most common is to assign precedences to all operators, and elide brackets, if the precedence of the operator is lower than that of the context it is presented in. In our example above, we would assign $\cap$ a lower precedence than $\cup$ (and both a lower precedence than the initial precedence). To compute the presentation of (3) we start out with the `\ninters`, elide its brackets (since the precedence n of $\cup$ is lower than the initial precedence i), and set the context precedence for the arguments to n. When we present the arguments, we present the brackets, since the precedence of `nunion` is lower than the context precedence n.

The `presentation` package supplies optional keyval arguments to the mixfix declarations and their abbreviations that allow to specify precedences: The key `p` is used to specify the **operator precedence**, and the keys `p`$\langle\langle i\rangle\rangle$ can be used to specify the **argument precedences**. The latter will set the precedence level while processing the arguments, whereas the operator precedence invokes brackets, if it is larger than the current precedence level — which is set by the appropriate argument precedences by the dominating operators or the outer precedence. Note that in our semantic macro definition for $\circ$ in Figure 3 we have specified the number 400 for the operator precedence and 401 for the argument precedence to conventionalize associativity in the bracket: the slightly higher argument precedence gives room to elide the inner brackets. Of course, when we want to explain associativity, we cannot use these settings, since they would render the associativity assertion pointless as $a \circ b \circ c = a \circ b \circ c$. Therefore we have locally redefined them in the `definition` (see Section 3.4 for details).

3.3. Flexible Elision for Reconstruction

There are several situations in which it is desirable to display only some parts of the presentation: mathematicians gloss over parts of the formulae, e. g. leaving out arguments, if they are non-essential, conventionalized or can be deduced from the context. Indeed this is part of what makes mathematics so hard to read for beginners, but also what makes mathematical language so efficient for the initiates. A common example is the use of $\log(x)$ or even $\log x$ for $\log_{10}(x)$ or similarly $[\![t]\!]$ for $[\![t]\!]^{\varphi}_{\mathcal{M}}$, if there is only one model $\mathcal{M}$ in the context and φ is the most salient variable assignment.

Typically, these elisions are confusing for readers who are getting acquainted with a topic, but become more and more helpful as the reader advances. For experienced readers more is elided to focus on relevant material, for beginners representations are more explicit. In the process of writing a mathematical document for traditional (print) media, an author has to decide on the intended audience and design the level of elision (which need not be constant over the document though). With electronic media we have new possibilities: we can make elisions flexible. The author still chooses the elision level for the initial presentation, but the reader can adapt it to her level of competence and comfort, making details more or less explicit.

To provide this functionality, the `presentation` package provides the `\elide` macro. It allows to associate a text with an integer **visibility level** and to group them into **elision groups**. High levels mean high elidability.

Elision can take various forms in print and digital media. In static media like traditional print on paper or the PostScript format, we have to fix the elision level, and can decide at presentation/formatting time which elidable tokens will be printed and which will not. In this case, the presentation algorithm will take visibility thresholds T_g for every elidability group g as a user parameter and then elide all tokens in visibility group g with level $l > T_g$. We specify this threshold via the `\setelevel` macro. For instance in the example below, we have two type annotations `par` for type parameters and `typ` for type annotations themselves.

```
$\mathbf{I}\elide{tan}{500}{^\alpha}
          \elide{typ}{100}{_{\alpha\to\alpha}}
    :=\lambda{X\elide{ty}{500}{_\alpha}}.X$
```

The visibility levels in the example encode how redundant the author thinks the elided parts of the formula are: low values show high redundancy. In our example the intuition is that the type parameter on the $\mathbf{I}$ combinator and the type annotation on the bound variable X in the λ expression are of the same obviousness to the reader. So in a document that contains `\setegroup{typ}{1000}` and `\setegroup{tan}{1000}` will show $\mathbf{I} := \lambda X.X$ eliding all redundant information. If we have both values at 400, then we will see $\mathbf{I}^\alpha := \lambda X_\alpha.X$, and only if the threshold for `typ` dips below 100, then we see the full information: $\mathbf{I}^\alpha_{\alpha\to\alpha} := \lambda X_\alpha.X$.

In an output format that is capable of interactively changing its appearance, e.g. dynamic XHTML+MathML (i.e. XHTML with embedded Presentation MathML formulas, which can be manipulated via JavaScript in browsers), an application can export the information about elision groups and levels to the target format, and can then dynamically change the visibility thresholds by user interaction. Here the visibility threshold would also be used, but it only determines the default rendering; a user can subsequently fine-tune the document dynamically to reveal elided material to support understanding or to elide more to increase conciseness.

The price the author has to pay for this enhanced user experience is that she has to specify elided parts of a formula that would have been left out in conventional LaTeX. Some of this can be alleviated by good coding practices. Let us consider the log base case which is often elided in mathematics, since the reader is expected to pick it up from context. Using semantic macros, we can mimic this behavior: defining two semantic macros: `\logC` which picks up the log base from the context via the `\logbase` macro and `\logB` which takes it as a (first) argument.

```
\provideEdefault{logbase}{10}
\symdef{logB}[2]{\prefix{\mathrm{log}\elide{base}{100}{_{#1}}}{#2}}
\abbrdef{logC}[1]{\logB{\fromEcontext{logbase}}{#1}}
```

Here we use the `provideEdefault` macro to initialize a LaTeX token register for the `logbase` default, which we can pick up from the elision context using `\fromEcontext` in the definition of `\logC`. Thus `\logC{x}` would render as $\log_{10}(x)$ with a threshold of 50 for `base` and as $\log_2$, if the local TeX group e.g. given by the `assertion` environment contains a `\setEdefault{logbase}{2}`.

Note that sTeX implements an elision approach already proposed for OMDoc and MathML3 [ABC+08] in [KMR08]. This uses numerical precedence values to control bracket elision for mathematical operators and generalizes it to a general mechanism for flexible elision. This approach differs from the bracket elision approach proposed in [AFNW07] for use in the Plato system, which uses a pre-order for precedences. We have come to the conclusion that while generalizing the pre-order system to a partial order on operators is very appealing from a mathematical perspective, it seems dangerous in a distributed and multi-author situation which we intend to support with our sTeX and OMDoc systems: one author can trivialize (part of) the precedence relation by introducing a pair that completes a cycle. Even though concrete numbers are less elegant, and allow less incremental development of the precedence order, they enforce locality in the specification, which we value higher for our purposes.

3.4. sTeX Modules for the Notation/Context Problem

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 frameworks, scoping of concepts is governed by 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 grouping induced by the curly braces hierarchy of TeX.

In sTeX, the package `modules` provides the LaTeX environment `module` for specifying the theory structure and uses the macros `\symdef` and `\abbrdef` for defining macros; these definition mechanisms correspond to the classes of macros discussed in Section 2.2. 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 3.

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 visibility is governed by the `import` relation specified by the `\importmodule` macro in the `module` environment. In particular, the macros `\pair` and `\sset` are defined in modules `setoid` and `semigroup` (since the `import` relation is transitive). With these symbol definitions, we get the text in Figure 4.

The `\symdef` form defines a two-stage presentation infrastructure for semantic macros. On the surface `\symdef{sop}[2]{...}` defines a macro `\sop` which can be inherited along the `importmodule` relation. Internally `\symdef` defines an internal macro `\modules@sop@pres`, which carries out the presentation which can locally be redefined via the `\redefine` macro. We have made use of this to locally change the bracket elision behavior of `\sop` in the second definition of Figure 3.

```
\begin{module}[id=pairs]
  \symdef{pair}[2]{\mixfixii[p=0]\langle{#1},{#2}\rangle}
  ...
\end{module}

\begin{module}[id=sets]
  \symdef{member}[2]{\infix[p=600]\in{#1}{#2}}            % set membership
  \symdef{mmember}[2]{\mixfixai[p=600]{}{#1}\in{#2}{}{,}} % aggregated membership
  ...
\end{module}

\begin{module}[id=setoid]
  \importmodule{pairs}                            % import from pairs
  \importmodule{sets}                             % import from set
  \symdef{sset}{\mathcal{S}}                      % the base set
  \symdef{sopa}{\circ}                            % the operation symbol
  \symdef{sop}[2]{\infix[p=400,pi=401]\circ{#1}{#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]
  \importmodule{setoid}
  \begin{definition}[id=setoid-def]
    \redefine{sop}[2]{\infix[p=400]\circ{#1}{#2}} % to explain associativity
    A setoid $\pair\sset\sopa$ is called a semigroup, 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}
  \begin{notation}
    Note that we will elide brackets for associative operators, so that both sides
    of the equation above would be written as $\sop{\sop{a}{b}}{c}$.
  \end{notation}
\end{module}
```

FIGURE 3. Semantic Scoping of Semantic Macros via Modules

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

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 is what is often called "*late binding*" in programming. Depending on the viewpoint, late binding can be a problem or a feature: in content-oriented document management, late binding of style information

...
...
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 semigroup, if $\circ$ is associative on $\mathcal{S}$, i.e. if $a \circ (b \circ c) = (a \circ b) \circ c$ for all $a, b, c \in \mathcal{S}$.
Notation: Note that we will elide brackets for associative operators, so that both sides of the equation above would be written as $a \circ b \circ c$.

FIGURE 4. The Result of Modules in Figure 3

is used to adapt presentation, in programming, late binding of (changing) program 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. Therefore 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.

4. Tools for TeX/LaTeX to XML Conversion

The need for translating LaTeX documents into other formats has been long realized and there are various tools that attempt this at different levels of sophistication. We will disregard simple approaches like the venerable `latex2html` translator that have only limited support for user macro definitions, since these are essential for semantic preloading as we have seen above. The remaining ones fall into two categories that differ in the approach towards parsing the TeX/LaTeX documents.

Romeo Anghelache's HERMES [Ang07] 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 "category codes", and the grammar at run-time). In contrast to this, Bruce Miller's LATEXML [Mil07] 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.

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. Systems like Hermes or TeX4HT could be used with sTeX, given suitable sTeX bindings provided we find a way to distinguish semantic macros from abbreviative macros.

4.1. The LaTeXML 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 semantic macros in LaTeX packages. Concretely, every LaTeX package and class must be accompanied by a LaTeXML binding file, a Perl file which contains LaTeXML constructor, abbreviation, and environment definitions, e.g.

Listing 1

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

`DefConstructor` is used for semantic macros, whereas `DefMacro` is used for abbreviative ones. The latter is used, since the `latexml` program does not read the package or class file 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 a temporary LTXML document, which closely mimics the structure of the parse tree of the LaTeX source. The LTXML format provides XML counterparts of all core TeX/LaTeX concepts, serves as a target format for LaTeXML, and thus legitimizes the XML fragments in the LaTeXML bindings.

In the semantic post-processing phase, the LaTeX-near representation is transformed into the target format by the `latexmlpost` program. This program applies a pipeline of intelligent filters to its input. 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.

We have gained extensive experience with the LaTeXML converter in the arXMLiv project [SK08, ArX07b], which aims at translating the Cornell e-Print

Archive (arχiv [ArX07a]) to XHTML+MathML form. The main technical task of the arXMLiv project is to supply LaTeXML bindings for the (thousands of) LaTeX classes and packages used in the arχiv collection. For this we have developed a distributed build system that continuously runs LaTeXML over the arχiv collection and collects statistics about e.g. the most sorely missing LaTeXML bindings. We have processed more than half of the arχiv collection (one run is a processor-year-size undertaking) and already have a success rate of over 58% (i.e. over 58% of the documents ran through without LaTeXML noticing an error).

4.2. Using LaTeXML with sTeX

We are using the LaTeXML program to convert semantically preloaded sTeX documents to our OMDoc format [Koh06b]. As sTeX documents are valid LaTeX, LaTeXML program can be used for this without change, we only have to supply the necessary LaTeXML bindings for the packages described in the last section. The main interesting point here is that even though the `latexml` program internally needs to have access to `DefConstructor` directives (see Section 4.1) for semantic macros, we do not have to supply them manually: As we define semantic macros using `symdef`, the binding for this meta-macro can be used to construct `DefConstructor`s in memory only. Thus we only have to supply bindings for the macros used in the sTeX format itself. The semantic macros for the domain of discourse are generated virtually on the fly.

Note that the LaTeXML bindings for the sTeX package differ from those for LaTeXML, in that they do not generate XML in the LTXML format, but in OMDoc[5]. The LaTeXML program supports the generation of non-LTXML namespaces gracefully, since it supplies a general interface for semantic annotation in TeX/LaTeX.

We have tested the conversion on a two-semester course in Computer Science at Jacobs University. We have semantically preloaded the LaTeX sources for the course slides (444 slides, 9600 lines of LaTeX code with 530kb; the module dependency depth is 18.). Almost all examples in this paper come from this case study. For instance, transforming the sTeX code from Figure 2 yields the OMDoc document below:

```
<assertion type="Theorem" xml:id="domino−thm">
  Let S_0, S_1,\ldots be a linear sequence of dominoes. If S_0 is pushed
  towards S_1 so that it falls, then all dominoes will fall.
</assertion>
<proof for="domino−thm" xml:id="domino−pf">
  <CMP>We prove that S_i falls in the direction of S_{i+1} by induction over i.</CMP>
  <derive>
    <CMP>We have to consider two cases</CMP>
    <method xref="by−cases">
      <proof>
        <metadata><dc:title>base case: i = 0</dc:title></metadata>
```

[5] Actually, the immediate target format is OMDoc with special LTXML parts mixed-in, e.g. for tables. For these it is simpler to make use of the LaTeXML bindings directly and transform them to OMDoc by XSLT postprocessing rather than trying to adapt the (very complex) LaTeXML bindings.

```
        <derive>
          <CMP>We have assumed that "S_0 is pushed towards S_1, so that it falls"</CMP>
        </derive>
      </proof>
       <proof>
        <metadata><dc:title>step case: i > 0</dc:title></metadata>
        <derive>
          <CMP>We assume that S_{i-1} falls in the direction of S_i.</CMP>
        </derive>
        <derive>
          <CMP>Thus it hits S_i and causes it to fall in the same direction,
            i.e. towards S_{i+1}.</CMP>
        </derive>
      </proof>
    </method>
  </derive>
  <derive>
    <CMP>Now, the assertion follows trivially, since if " S_i falls in the
    direction of S_{i+1}", then in particular "S_i falls".</CMMP>
  </derive>
</proof>
```

The advantage for the LaTeX user is obvious: she does not have to cope with the XML, 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. The price authors have to pay is that of preloading the LaTeX documents. For an existing TeX/LaTeX document, this is a relatively tedious process, as it involves heavy editing of the document[6], 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. In our case study we have measured the time overhead for semantic preloading to lie between 10% and 15% of the overall effort of creating the documents, if we use specialized tools like Darko Pesikan's sTeX mode [sTeb] for `emacs`, which helps build structured markup and maintain module dependencies. The overhead is (partially) recovered by the fact that, the content markup in the source document can 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?). Finally, we can base added-value services either on the generated OMDoc or on the sTeX source itself. For instance a simple Perl script was used to generate the module dependency graph in Figure 5 which covers the first semester of the course. Note that this generation of semantic information from the sTeX source is quite limited and restricted to the case, where the underlying TeX is parseable, as in the case of module headings for the graph. Once the author makes use of the inherent programmability of the TeX/LaTeX format, we need a fully capable TeX

[6] 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.

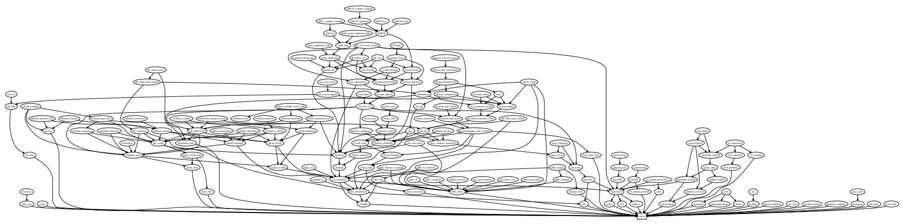


FIGURE 5. Module Graph of the "General Computer Science 1" Course

parser like the LaTeXML system. Then we can extract the more involved semantic properties — i.e. the ones that MKM is interested in — from the generated OMDoc.

5. Conclusion and Future Work

We have presented the sTeX format, a collection of macro-packages for semantic annotation of LaTeX documents and an extension of the LaTeXML system with bindings for the sTeX package. This allows us to semantically preload LaTeX documents and transform them into XML and ultimately into MKM formats like OMDoc or CnXML.

The system is being tested on a first-year computer science course at Jacobs University, on a set of Connexions modules, and recently the OMDoc specification [Koh06b]. An anonymous reviewer of this article raised the concern that our current "sTeX time overhead ratio" of 10-15% would not transfer to research mathematics articles. It seems that the low "semantics tax" comes from the fact that our case studies so far are relatively self-contained, so that we do not need to take into account semantic markup of pre-requisites. From our limited experiments with using sTeX for MKM research papers and conference presentations, it seems that if we neglect the initial investment of establishing the basic sTeX infrastructure for a field, the overhead for semantic annotation of research articles is below 10% and is even outweighed by the management benefits, e.g., where a Ph.D. thesis shares sTeX source fragments with pre-published research papers. Nevertheless it is important to further decrease the overhead for semantic preloading by fine-tuning the sTeX format and tools, but not — as that reviewer suggests — by allowing ambiguous parts in the sTeX source and then processing them with an expert system, but rather to integrate "expert system" functionality into sTeX-aware editors that will in a mixed-initiative process help the author produce semantic text. An area where we are experimenting with this is the `\term` macro, which allows to preload a technical term with information about its corresponding symbol. As the author cannot be bothered to supply this information for every single occurrence of the word in a text we have the `emacs` mode [sTeb] silently add the necessary information where it is obvious and hide it from the user in the editing buffer. Of course, the editor has to be aware of the current semantic context or document collection the author is working on. We consider the development of context-aware semantic editors a challenge for the MKM community in the next years. The sTEX format is no different from other MKM formats in this respect.

Work on the sTEX packages and their LaTeXML bindings is ongoing, driven by these case studies and user requests. A current weak point of the system is error handling. To understand the problems here, note that there are three possible classes of errors: *a) sTEX errors*: if the sTEX macros have been misused — here, more native error handling that reports errors in terms of sTEX concepts rather than falling through to the underlying LaTeX errors would be helpful for the user; *b) target format errors*: not all error-less sTEX documents can be transformed into valid documents of the respective target formats, as these usually impose more structural constraints — e.g. the OMDoc format does not allow `definition` elements inside an `omtext`. Such constraints could be checked by the sTEX packages already on the TEX level. *c) leftover LaTeX*: the LaTeX-based workflow allows to mix ordinary presentational macros into sTEX documents, for which either no LaTeXML bindings are provided by the sTEX package — leading to transformation errors — or which the target format does not admit. As the top-level `omdoc` class is based on LaTeX's `article.cls`, some leftover macros have not been disabled yet.

In essence, the sTEX package together with its LaTeXML bindings form an invasive editor for OMDoc in the sense discussed in [KK04, Koh05]: 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 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 (**C**Point [KK04] and NB2OMDOC [Sut06] 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 **C**Point).

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 was a student at Jacobs University who has faithfully carried the brunt of the editing load involved with semantic preloading the LaTeX slides for the GenCS1 course. Ioan and Deyan Ginev (another Jacobs undergrad) helped me with developing the LaTeXML bindings for sTEX. Finally, I am indebted to David Carlisle and Rares

Ambrus who helped me with the non-trivial TeX hacking involved in getting the sTeX modules to work.

References

[ABC+03] Ron Ausbrooks, Stephen Buswell, David Carlisle, Stéphane 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.

[ABC+08] Ron Ausbrooks, Bert Bos, Olga Caprotti, David Carlisle, Giorgi Chavchanidze, Ananth Coorg, Stphane Dalmas, Stan Devitt, Sam Dooley, Margaret Hinchcliffe, Patrick Ion, Michael Kohlhase, Azzeddine Lazrek, Dennis Leas, Paul Libbrecht, Manolis Mavrikis, Bruce Miller, Robert Miner, Murray Sargent, Kyle Siegrist, Neil Soiffer, Stephen Watt, and Mohamed Zergaoui. Mathematical Markup Language (MathML) version 3.0. W3C working draft of march april 9., World Wide Web Consortium, 2008. Available at http://www.w3.org/TR/MathML3.

[AFNW07] Serge Autexier, Armin Fiedler, Thomas Neumann, and Marc Wagner. Supporting user-defined notations when integrating scientific text-editors with proof assistance systems. In Kauers et al. [KKMW07], pages 176–190.

[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, pages 73–88. Springer Verlag, 2000.

[Ang07] Romeo Anghelache. Hermes - a semantic xml+mathml+unicode e-publishing/self-archiving tool for latex authored scientific articles. web page at http://hermes.roua.org/, 2007.

[ArX07a] `arXiv.org` e-Print archive, seen December2007. web page at http://www.arxiv.org.

[ArX07b] `arXMLiv: Translating the arXiv to XML+MathML`, seen December2007. web page at http://kwarc.info/projects/arXMLiv/.

[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.

[BFGHS04] Peter Baumgartner, Ulrich Furbach, Margret Gross-Hardt, and Alex Sinner. Living book – deduction, slicing, and interaction. *Journal of Automated Reasoning*, 32(3):259–286, 2004.

[BL98] Tim Berners-Lee. The semantic web, 1998. Available at http://www.w3.org/DesignIssues/Semantic.html, seen July 2006.

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

[Caj93] Florian Cajori. *A History of Mathematical Notations*. Courier Dover Publications, 1993. Originally published in 1929.

[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.

[CTA] CTAN the Comprehensive TEX Archive Network. web page at http://www.ctan.org.

[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.

[Far05] William M. Farmer. Mathematical knowledge management. In David G. Schwartz, editor, *Mathematical Knowledge Management*, pages 599–604. Idea Group Reference, 2005.

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

[GHMD07] Tudor Groza, Siegfried Handschuh, Knud Möller, and Stefan Decker. Salt - semantically annotated latex for scientific publications. In Enrico Franconi, Michael Kifer, and Wolfgang May, editors, *ESWC*, volume 4519 of *Lecture Notes in Computer Science*, pages 518–532. Springer, 2007.

[HBK03] Geneva Henry, Richard G. Baraniuk, and Christopher Kelty. The connexions project: Promoting open sharing of knowledge for education. In *Syllabus, Technology for Higher Education*, 2003.

[HG07] Brent Hendricks and Adan Galvan. The Connexions Markup Language (CNXML). http://cnx.org/aboutus/technology/cnxml/, 2007. Seen June 2007.

[KBLL+04] Bernd Krieg-Brückner, Arne Lindow, Christoph Lüth, Achim Mahnke, and George Russell. Semantic interrelation of documents via an ontology. In G. Engels and S. Seehusen, editors, *DeLFI 2004*, volume P-52 of *LNI*, pages 271–282. Springer-Verlag, 2004.

[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, pages 175–189. Springer Verlag, 2004.

[KKMW07] Manuel Kauers, Manfred Kerber, Robert Miner, and Wolfgang Windsteiger, editors. *MKM/Calculemus 2007*, number 4573 in LNAI. Springer Verlag, 2007.

[KMM07] Michael Kohlhase, Christine Müller, and Normen Müller. Documents with flexible notation contexts as interfaces to mathematical knowledge. In Paul Libbrecht, editor, *Mathematical User Interfaces Workshop 2007*, 2007.

[KMR08] Michael Kohlhase, Christine Müller, and Florian Rabe. Notations for living mathematical documents. In MKM08 [MKM08]. in press.

[KMW07] Fairouz Kamareddine, Robert Lamarand Manuel Maarek, and J. B. Wells. Restoring natural language as a computerised mathematics input method. In Kauers et al. [KKMW07], pages 280–295.

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

[Koh05] Andrea Kohlhase. Overcoming Proprietary Hurdles: CPoint as Invasive Editor. In Fred de Vries, Graham Attwell, Raymond Elferink, and Alexandra Tödt, editors, *Open Source for Education in Europe: Research and Practise*, pages 51–56, Heerlen, The Netherlands, November 2005. Open Universiteit Nederland, Open Universiteit Nederland. Proceedings at http://hdl.handle.net/1820/483.

[Koh06a] Michael Kohlhase. CnxLaTeX: A LaTeX-based syntax for connexions modules. Self-documenting LaTeX package, available at https://svn.kwarc.info/repos/stex/sty/cmathml.pdf, 2006.

[Koh06b] Michael Kohlhase. OMDoc – *An open markup format for mathematical documents [Version 1.2]*. Number 4180 in LNAI. Springer Verlag, 2006.

[Lam94a] Leslie Lamport. How to write a proof. *American Mathematical Monthly*, 102(7):600 – 608, 1994.

[Lam94b] Leslie Lamport. *LaTeX: A Document Preparation System, 2/e*. Addison Wesley, 1994.

[LK08] Christoph Lange and Michael Kohlhase. A Semantic Wiki for Mathematical Knowledge Management. In Jörg Rech, Björn Decker, and Eric Ras, editors, *Emerging Technologies for Semantic Work Environments: Techniques, Methods, and Applications*, pages 47–68. IGI Global, 2008. In press.

[MBA+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), 2001.

[Mil07] Bruce Miller. `LaTeXML`: A LaTeX to xml converter. Web Manual at http://dlmf.nist.gov/LaTeXML/, seen September2007.

[MK07] Christine Müller and Michael Kohlhase. panta rhei. In Alexander Hinneburg, editor, *Wissens- und Erfahrungsmanagement LWA (Lernen, Wissensentdeckung und Adaptivität) conference proceedings*, pages 318–323, 2007.

[MKM08] *Mathematical Knowledge Management, MKM'08*, LNAI. Springer Verlag, 2008.

[MR+07] Peter Murray-Rust et al. Chemical markup language (CML). http://cml.sourceforge.net/, seen January 2007.

[MvH04] Deborah L. McGuinness and Frank van Harmelen. OWL web ontology language overview. W3C recommendation, W3C, February 2004. Available at http://www.w3.org/TR/2004/REC-owl-features-20040210/.

[NPW02] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002.

[Pro07] LaTeX Project. The LaTeX project public license. Software License available at http://www.latex-project.org/lppl/, seen 2007.

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

[SK08] Heinrich Stamerjohanns and Michael Kohlhase. Transforming the arχiv to xml. In MKM08 [MKM08]. in press.

[sTea] Subversion repository at https://svn.kwarc.info/repos/stex.

[sTeb] Subversion repository at https://svn.kwarc.info/repos/stex/emacs.

[sTec] Subversion repository at https://svn.kwarc.info/repos/stex-content/slides.

[Sut06] Klaus Sutner. Converting MATHEMATICA notebooks to OMDoc. In OMDoc – *An open markup format for mathematical documents [Version 1.2]* [Koh06b], chapter 26.17.

[Tea06] Connexions Team. Connexions: Sharing knowledge and building communities. White paper at http://cnx.org/aboutus/publications/ConnexionsWhitePaper.pdf, 2006.

[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.

[Win90] Jeanette M. Wing. A specifier's introduction to formal methods. *IEEE Software*, 23(9):8–24, September 1990.

[Wol00] Stephen Wolfram. Mathematical notation: Past and future. In *MathML and Math on the Web: MathML International Conference*, Urbana Champaign, USA, October 2000. http://www.stephenwolfram.com/publications/talks/mathml/.

Michael Kohlhase
Computer Science
Jacobs University Bremen, Germany
http://kwarc.info/kohlhase