sTeX3 — A LaTeX-based ecosystem for semantic/active mathematical documents

Dennis Müller, Michael Kohlhase

This paper uses sTeX3. The semantically annotated XHTML version of this paper is available at: `tinyurl.com/tug22stex`

Abstract

We report on sTeX3 — a complete redesign and reimplementation (using LaTeX3) from the ground up of the sTeX ecosystem for semantic markup of mathematical documents. Specifically, we present:

1. The sTeX package that allows declaring semantic macros and provides a module system for organizing and importing semantic macros using logical identifiers. Semantic macros allow for annotating arbitrary LaTeX fragments, particularly symbolic notations and formulae, with their functional structure and formal semantics while keeping their presentation/layout intact. The module system induces a theory graph-structure on mathematical concepts, reflecting their dependencies and other semantic relations.
2. The RusTeX system, an implementation of the core TeX engine in Rust. It supports converting arbitrary LaTeX documents to XHTML. For sTeX3 documents, these are enriched with semantic annotations based on the OMDoc ontology.
3. An MMT integration: The RusTeX-generated XHTML can be imported and served by the MMT system (meta-meta-theory or meta-meta-tool, depending on desired emphasis) for semantically informed knowledge management services, e.g. linking symbols in formulae to their definitions, or "guided tour" mini-courses for any (semantically annotated) mathematical concept/object.

Generally, sTeX3 documents can be made not only *interactive* (by adding semantic services), but also "*active*" in that they actively adapt to reader preferences and pre-knowledge (if known).

1 Introduction

In mathematics (and adjacent disciplines), LaTeX is the de facto standard for typesetting *static* documents of all kinds. While LaTeX has thus established itself as the perfect tool for that job, since the advent of the internet a lot of functionalities have been developed and are commonly used (primarily via HTML) that allow for a more *active* interaction with documents than static formats allow for.

At the same time, computer scientists and mathematicians have developed techniques for representing the *formal semantics* of mathematical definitions, theorems, proofs and other statements in a computer-actionable manner. While the *strongest* of these techniques require significant expertise and effort to represent even relatively simple mathematical settings in their full formality, these are largely only required for the strongest forms of computer services (such as automated theorem proving); in contrast, relatively simple semantic annotations already allow for a plurality of useful services that can be integrated (primarily) in active documents.

To that end, we developed the sTeX [5, 11] package and related systems, and its recent redesign and reimplementation in the form of sTeX3.

sTeX is a standard LaTeX package that provides a mechanism for declaring semantic macros (representing distinct mathematical concepts), which can be used to annotate arbitrary document fragments with their semantics to an arbitrary degree of formality (we speak of *flexiformality* [4]). These semantic macros are collected in modules which can be imported anywhere (analogously to LaTeX packages), and are in turn collected in math archives [2] which can be developed communally. The main difference between modules and LaTeX packages is that the objects of modules are (mathematical) concepts, objects, and structures, not abbreviations and layout primitives. As a consequence, modules usually contain the corresponding definientia that specify the concepts, objects, structures, and possibly theorems that state their properties and relations to others, and proofs that justify these all in a neat self-contained package of reusable components. The overall effect of this is that documents and archives can be developed modularly in an "object-oriented" fashion.

Many such archives are available on `gl.mathhub.info`, in particular SMGloM, a multilingual mathematical glossary [10], currently containing ≥ 2250 concepts in English (93%), German (71%) and Chinese (11%).

In addition to being standard LaTeX documents, when converted to HTML the semantic information obtained from semantic macros (and other annotations) can be preserved in the form of HTML attributes. For those purposes, we implemented the RusTeX system, a plain TeX engine converting arbitrary LaTeX documents to XHTML.

The resulting XHTML documents can be imported and served by the MMT system [8, 9], which can interpret the semantic annotations and offer corresponding semantics-aware services, effectively transforming the (originally) statically typeset LaTeX

doi.org/10.47397/tb/43-2/tb134mueller-stex3

document into an active HTML document. Our collection of such active documents generated from sTeX can be browsed on `mmt.beta.vollki.kwarc.info/:sTeX`, including 3000+ pages of semantically annotated course notes and slides for various university lectures.

Notably, this paper itself uses sTeX. The semantically enriched version of it is linked above. Additionally, the source files are available on Overleaf at `www.overleaf.com/read/rvjbsnfshvhg` for demonstration purposes.

2 The sTeX package

For a detailed description of sTeX we refer to the documentation [6].

2.1 Modules and symbols

A module is opened via
`\begin{smodule}{⟨name⟩}`.

Within a module, we can declare a new *symbol* with a corresponding semantic macro using `\symdecl`; for example, a symbol named `natural-number` with semantic macro `\Nat` would be declared with:[1]

```
\symdecl{Nat}[name=natural-number]
```

We can now reference our new symbol using e.g. `\symname`, where `\symname{Nat}` now yields the (annotated!) text “natural number”. Additionally, we can provide a new notation for the symbol using `\notation`, as in `\notation{Nat}{\mathbb N}`, allowing us to now use the semantic macro in math mode to print $\mathbb{N}$, or in text mode to annotate arbitrary text via `\Nat{⟨text⟩}`.

Semantic macros can also take arguments and be provided with additional semantic information, e.g. “types”. While the latter are ignored by LaTeX, the MMT system can use these for additional services, e.g. type checking (see below). Furthermore, the `\symdef` macro combines the (usually used in conjunction) functionalities of `\symdecl` and `\notation`. For example,

```
\symdef{plus}[
  name=addition,
  args=2, op=+,
  type=\funspace{\Nat,\Nat}{\Nat}
]{#1 + #2}
```

declares `\plus` to be a binary function of type $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$, and immediately provides it with an appropriate notation, after which `$\plus ab$` yields “$a + b$”. The `op=+` in the above declaration allows us to refer to addition *itself* (rather than its application to arguments) via `\plus!`, yielding just $+$.

[1] See the source files of this paper for direct demonstrations of the examples here.

Analogously, we can introduce variables using `\vardef` (unlike symbols that have object-oriented scope, variables are local to the current TeX group).

2.2 Statements

Complex statements can be semantically marked-up using appropriate environments. For example, the following slightly simplified syntax allows us to declare commutativity as a predicate on binary operations and semantically annotate its definiens directly:

```
\symdecl{commutative}[args=1]
\begin{sdefinition}[for=commutative]
  \vardef{setA}{\comp{A}}
  \vardef{varop}[op=\circ,args=2]
    {#1 \circ #2}

  A binary operation
  $\fun{\varop!}{\setA,\setA}\setA$ is
  called \definame{commutative}, iff
  \definiens{
    \foral{
      \arg[2]{
        $\eq{\varop{a}{b},\varop{b}{a}}$}
      \comp{for all}
      \arg[1]{
        $\inset{a,b}\setA$
      }.
    }
  }
\end{sdefinition}
```

yielding:

Definition 2.1. A binary operation $\circ : A \times A \to A$ is called **commutative**, iff $a \circ b = b \circ a$ for all $a, b \in A$.

(See the source files and/or documentation for details on the syntax.)

Similarly, we can mark up e.g. *theorems*, like

```
\begin{sassertion}[type=theorem,
    name=commutativity-of-addition]
  \conclusion{
    \commutative{
      \arg{\plus{\comp{Addition}}} is
      \comp{commutative}
    }
  }.
\end{sassertion}
```

yielding

Theorem 2.1. *Addition is commutative.*

...and allowing us to now refer to commutativity of addition like any other symbol (e.g. via `\symname`).

The naming convention of prefixing environment names with `s-` (as in e.g. `sdefinition`) is to allow for functionality with respect to semantic optional

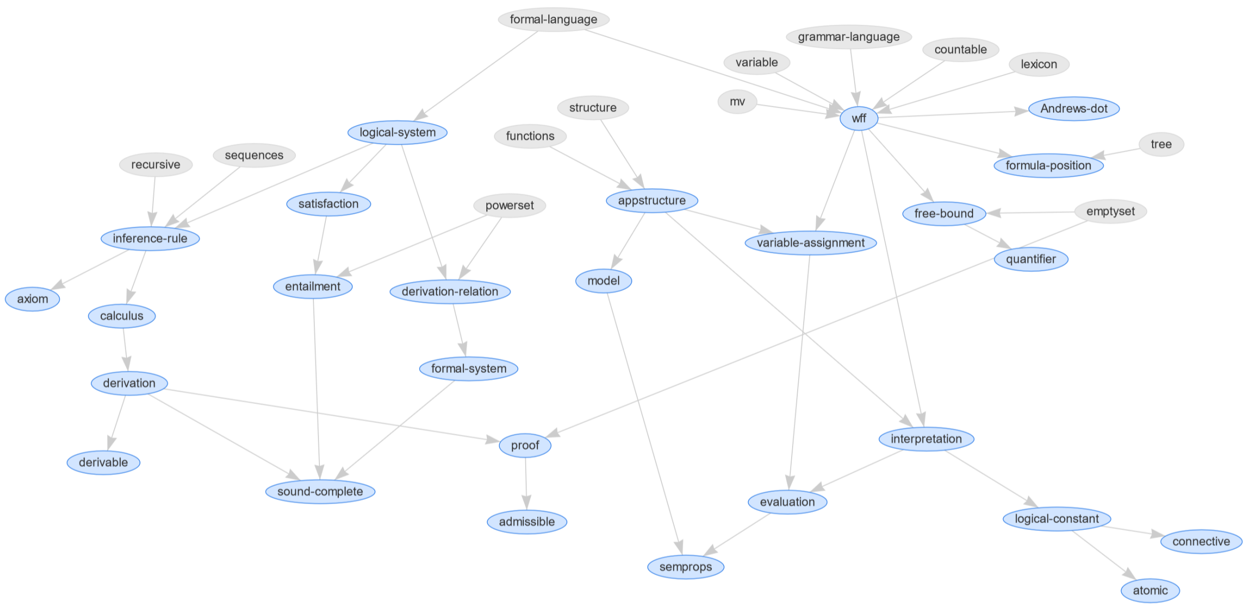

Figure 1: An example of a theory graph in the area of formal logic

arguments (e.g. `type=`, `for=`), while staying compatible with already existing environments. In fact, all the typesetting and semantic highlighting done by sTeX can be fully customized — in the case of the `sdefinition` environment, for example by deferring typesetting to a standard `definition` environment defined via the `amsthm` package (as in this paper).

2.3 Importing modules

The semantic macros `\eq` and `\foral` used in our definition above represent *equality* and *universal quantification* (i.e. "for all"). These are imported from existing sTeX modules, namely `mod?Equal` in the math archive `sTeX/MathBase/Relations` and `mod/syntax?UniversalQuantifier` in the archive `sTeX/Logic/General`. If we only want to *use* the semantic macros in these modules, we can use the syntax `\usemodule[`⟨*archive*⟩`]{`⟨*module*⟩`}`. If, however, we are currently *in* a module, the contents of which *depend* on the symbols we want to import, we can use `\importmodule[`⟨*archive*⟩`]{`⟨*module*⟩`}` instead, which additionally *exports* the contents of the thus-imported module whenever we import the current one. For example, this paper might never explicitly import the `Equal` module, but could still use its contents if it imports others that in turn (transitively) import `Equal`.

This import mechanism naturally induces a theory graph, with modules as nodes and the import-relation as edges (see Figure 1). sTeX and MMT support more complicated edges as well, that represent less trivial and thus more interesting *theory morphisms* between modules that knowledge can be translated along (see e.g. [9] for details).[2]

To let sTeX know where the required archives can be found, users can (among other ways) set a corresponding macro `\mathhub`, or set an environment variable `MATHHUB` once and for all. As a result, references to archives (and thus modules) are independent of the local filesystem. Notably, the number of modules imported in a given document can grow large very quickly — to allow for submission procedures (e.g. with *TUGboat* or `arxiv.org`) without needing to submit possibly hundreds of files, package options allow for storing and retrieving all semantic macros imported from external modules in/from a dedicated `\jobname.sms` file during compilation, which can be distributed alongside the document.

3 The RusTeX system

There are multiple existing applications to convert LaTeX documents to HTML, including but not limited to TeX4ht [1] and LaTeXML [7]. Unfortunately, all of these have turned out to be deficient for our purposes, primarily due to their lacking support for either commonly used packages and macros or for introducing the required XML attributes for semantic annotations. We therefore decided to add to the existing set of such conversion tools.

[2] The full theory graph for (exemplary) the SMGloM can be navigated actively on `mmt.beta.vollki.kwarc.info/graphs/tgview.html?type=stexgraph&graphdata=smglom`.

RusTeX[3] is an implementation of a plain TeX engine using the programming language Rust, outputting XHTML. It implements merely the (vast majority of) primitives of TeX, eTeX and pdfTeX, and uses a locally installed LaTeX distribution (by processing the available `latex.ltx` file) to handle LaTeX documents. While this means that RusTeX behaves virtually identically to `pdflatex` (except for the output format), this comes at the cost of a priori no special treatment of standard LaTeX-macros (although RusTeX allows for adding special treatment of arbitrary macros on top). Instead, everything is expanded to primitive TeX *whatsits*, which are exported to (primarily) `<div>`-nodes, styled via CSS-classes depending on the whatsit.

Notably, however, with sTeX3 we opted for a mechanism analogous to the `pgf`-package: The relevant functionality is reduced to a mere handful of primitive macros for (HTML) annotations that a configuration file for a *backend* of choice (e.g. `pdflatex` or RusTeX) can provide. This means that sTeX can be easily made compatible with alternative conversion tools, provided they support the basic functionality required.

4 MMT integration and applications

MMT [8, 9] is a software system and API for a wide range of generic knowledge management services, providing algorithms for e.g. library management, parsing, (parametric) bi-directional type checking and reconstruction, term simplification, and various other computations on formal knowledge. The system uses a variant of the OMDoc [3] ontology, a representation format for semantically enriched mathematical documents.

The XHTML generated by RusTeX can be imported by the MMT system directly, extracting the semantic annotations and converting them to the corresponding OMDoc elements. As a result, the full suite of MMT services are available for sTeX documents.

Services thus enabled in active documents currently include:

1. *Disambiguation*: Every symbol is assigned a globally unique MMT URI (i.e. identifier) that unambiguously determines the semantics of the symbol, regardless of e.g. notations used. In this paper (in the PDF), this MMT URI is shown when hovering over a symbol reference.

   In the HTML version, hovering over a symbol reference shows (if available) the corresponding definition or theorem statement of the symbol, allowing for quick reminders of the meaning of terms and notations.
2. *Type checking*: For fully formally annotated document fragments, we can make use of MMT's type checking and inference mechanism: For example, in the definition and theorem in subsection 2.2, the MMT system can infer from our usage of `\definiens` that `commutative` is a unary predicate on binary operations, and uses that information to *type check* the theorem; that is, the system checks that the content of the `\conclusion`-macro is an actual proposition (which it determines by inferring the type of `commutative`), which recursively entails checking that `\plus` is indeed a binary operation (in this case on `\Nat`), and would warn us of a likely mistake otherwise.
3. *Guided tours*: Theory graphs provide us with the full semantic dependencies of a module. This allows us to generate small mini-courses that contain all prerequisite knowledge leading up to some intended concept in inverse dependency order: starting with the basics and ending with the concept to be explained. Clicking on a symbol reference in the XHTML opens a window linking to these guided tours.

Other features we are actively working on include:

1. *Notation selection*: Since sTeX allows for providing arbitrarily many notations for symbols, besides authors choosing the notation they prefer, in the HTML the document can in principle replace the notations used based on *readers'* preferences, making the resulting document more accessible for readers from different backgrounds with differing conventions.
2. *User-adapted guided tours*: In the context of classes at our university, we are working on modelling students' knowledge as probabilistic "*user models*" that allow us to generate guided tours specifically adapted to a user's prior knowledge. For example, by omitting already known concepts, selecting the most adequate examples, choosing their most familiar programming language for code snippets, etc.
3. *Flexible knowledge exploration/recommendation*: If we have a theory graph and a user model as above (possibly of a whole cohort of readers), we can use this information to recommend "useful knowledge items nearby" that might be interesting to the reader. These could be additional examples that help deepen understanding, theorems that give additional properties or relations, or even self-test problems. MMT can use the

[3] `github.com/slatex/RusTeX`

theory graph topology and user model information to determine what items are "nearby" the part of the theory graph that is (estimated to be) known to the reader.

To make these services as accessible to users as possible, we are actively developing a dedicated IDE in the form of a plugin for the VS Code editor using the *Language Server Protocol*.[4] The IDE integrates the MMT system (which in turn integrates RusTeX) and can preview the active XHTML document generated from LaTeX. Additionally, it allows for searching both local and remote (on `gl.mathhub.info`) sTeX content and downloading remotely available math archives directly.

5 Conclusion

The sTeX package now allows us to cover the complete spectrum from purely informal to fully formally annotated knowledge, directly in standard LaTeX documents. Via RusTeX and MMT, this makes formal knowledge management services available for LaTeX documents, allowing us to generate active documents that integrate semantically informed services for readers. The IDE bundles the whole toolchain required and makes it conveniently accessible to authors.

It is clear that semantic annotations constitute a considerable additional effort — in our experience up to 25–30% of the overall document development effort. Whether this investment can be amortized by the services that become available by it depends on the document or archive and on the context. We envision sTeX as an alternative to LaTeX primarily for documents with a high

- *impact*, i.e. which have many more readers than authors, or
- *inherent complexity*, which need semantic services to help readers understand them.

Some of the effort can surely be mitigated by advanced IDEs such as the one we are developing.

The main problem is that semantic annotations need semantic targets — i.e. annotated sTeX documents they can point to. This makes the first sTeX documents in a new domain very tedious to annotate, since we have to create archives for the "dependency cone". We aim to alleviate this by providing a community portal for flexiformal mathematics: `MathHub.info`, where math archives can be hosted, discussed, and maintained so that — over time — we can ensure that the "cost" of annotating a document is proportional to the size of the document and not to the size of the domain.

[4] `github.com/slatex/sTeX-IDE`

References

[1] E. Gurari, M. Hoftich, et al. *TEX4ht*. `tug.org/tex4ht/`.

[2] F. Horozal et al. Combining Source, Content, Presentation, Narration, and Relational Representation. *Intelligent Computer Mathematics*. Ed. by J. Davenport et al. LNAI 6824. Springer Verlag, 2011, pp. 212–227. `kwarc.info/frabe/Research/HIJKR_dimensions_11.pdf`.

[3] M. Kohlhase. *OMDoc — An Open Markup Format for Mathematical Documents [Version 1.2]*. LNAI 4180. Springer Verlag, Aug. 2006. `http://omdoc.org/pubs/omdoc1.2.pdf`.

[4] M. Kohlhase. The Flexiformalist Manifesto. *14th International Workshop on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2012)*. Ed. by A. Voronkov et al. Timisoara, Romania: IEEE Press, 2013, pp. 30–36. `kwarc.info/kohlhase/papers/synasc13.pdf`.

[5] M. Kohlhase. Using LaTeX as a Semantic Markup Format. *Mathematics in Computer Science* 2:2, 2008, pp. 279–304. `kwarc.info/kohlhase/papers/mcs08-stex.pdf`.

[6] M. Kohlhase and D. Müller. *The sTeX3 Package Collection*. `github.com/slatex/sTeX/blob/main/doc/stex-doc.pdf`.

[7] B. Miller. *LaTeXML: A LaTeX to XML Converter*. `mast.nist.gov/LaTeXML/`.

[8] *MMT — Language and System for the Uniform Representation of Knowledge*. `uniformal.github.io/`.

[9] F. Rabe and M. Kohlhase. A Scalable Module System. *Information & Computation* 0:230, 2013, pp. 1–54. `kwarc.info/frabe/Research/mmt.pdf`.

[10] *SMGloM: A Semantic, Multilingual Glossary for Mathematics*. `gl.mathhub.info/smglom/`.

[11] *sTeX: A Semantic Extension of TeX/LaTeX*. `github.com/sLaTeX/sTeX`.

◇ Dennis Müller
Friedrich-Alexander University, Erlangen-Nürnberg, DE
`dennis.mueller (at) fau.de`
ORCID 0000-0002-4482-4912

◇ Michael Kohlhase
Friedrich-Alexander University, Erlangen-Nürnberg, DE
`michael.kohlhase (at) fau.de`
ORCID 0000-0002-9859-6337