System Description sTEX – A LATEX-based Ecosystem for Semantic/Active Mathematical Documents

Michael Kohlhase, Dennis Müller, Jan Frederik Schaefer

Computer Science, FAU Erlangen-Nürnberg

1 Introduction and History

In the sLATEX project [sLX], we explore how established communication and publication workflows – this mainly means LATEX in Mathematics and theoretical sciences – can be extended semantically for computer support. The central element of this endeavour is the sTEX package [Koh08; sTeX] which allows to **semantically preload** LATEX documents via special (semantic) macros. sTEX documents can be processed by `pdflatex` in the usual way, or via LATEXML [LTX], a LATEX-to-XML transformer, which has a sTEX plugin. The semantic annotations are exported into the generated PDF or OMDoc [Koh06] respectively where they can be used for added-value services. The sTEX packages (and classes) have been used to produce extensive course materials (3000+ pages of slides and integrated narrative), ca. 2500 exercise/exam problems, and the SMGloM, a multilingual mathematical glossary [Koh14], currently containing $\geq$ 2250 concepts in English (93%), German (71%) and Chinese (11%). This **sTEX corpus** together with the OMDoc/Mmt format have informed the development of the sTEX packages and document model.

While the original sTEX architecture and realization showed that semantic preloading of the mathematical documents and the deployment of active documents based on this is possible given enough motivation, scalability and the management of shared content – one of the potential side-benefits of semantic preloading – quickly became a problem. As a consequence we now host and manage all sTEX content as **mathematical archives** [Hor+11] on `https://MathHub.info` and extended sTEX with special path functionality for cross-references. As a side effect, MathHub can host the interactive HTML generated from the OMDoc in a central location.[1]

In spite of this, the use of sTEX never quite gained much traction outside the authors' research group and collaborative projects. In this system description we detail the effort over the last two years of making sTEX much more usable.

[1] e.g. SMGloM under `https://beta.mathhub.info/library/group/c21nbG9t`

2 The sTEX EcoSystem

2.1 Simplification of sTEX Workflows

Working with sTEX so far required using several external tools and modifying LATEX parameters, mostly related to sTEX's *module system*.

Local paths To find the actual source files containing modularly imported sTEX content, the previous workflow necessitated the creation of a `localpaths.tex` for every top-level `.tex` file, that stores the local file path to sTEX repositories. This workflow has been significantly simplified recently, by replacing the `localpaths.tex` files by a single `MATHHUB` system variable, which points ot the local MathHub clone. LATEX can now access without needing the `--shellescape` flag.

SMS mode sTEX allows the introduction of new semantic macros within `module` environments, using the `\symdef` command. Semantic macros defined in some external module are made locally available with the `\importmodule` command. Since modules can (and in practice often do) contain arbitrary additional content, for `\importmodule` to work, the *semantic* content of a module needs to be extracted from a `module` environment. Previously, this was achieved by a perl script that heuristically parsed a file `foo.tex` for `\symdef` and similar commands, and extracted those into a separate `foo.sms` file. `\importmodule` and related commands then used the `.sms` file to selectively load only the semantic content. This required both an external tool and posed a change management problem - every change to an sTEX module required rerunning the perl script to ensure the `.sms` file is consistent with the source `.tex` file.

The usage of `.sms` files has now been deprecated. Instead, `\importmodule` and related commands enter an “SMS mode” before inputting the required `.tex` file directly, in which everything other than selected sTEX commands (such as `\symdef`) is ignored, obviating the need for external tools or change management considerations. The overhead of multiply reading the narrative content of included files – redundant in SMS mode – turned out to be negligible.

File stack size The availability of a module system can quickly lead to deeply nested dependency trees on modules (and hence `.tex`-files), especially when using SMGloM modules. By default, LATEX has a (relatively low) upper limit for its *file stack*, determining the number of individual files that the TEX engine is allowed to have open at once. Consequently, sTEX users were advised to manually increase the file stack size of their local LATEX distribution, something most LATEX users are unfamiliar with or need admin rights.

This has been recently resolved by fully utilizing the difference between *reading* a module (in SMS mode) and *activating* a module. During SMS mode (when the containing `.tex` file is *open*), all semantic macros are merely stored in a separate helper macro. Only after the file has been fully read (and closed by LATEX), its content is *activated* by executing all semantic macros therein, avoiding recursive `\importmodule`-calls and ensuring that LATEX's file stack only ever increases by 1 for semantic imports, obviating the need to manually increasing the file stack size.

Standalone SMGloM files Previously, sTEX distinguished **documents** (with LATEX preamble and `document` environment) with **module** (without) and used external build tools to provide modules with preambles on the fly. With the need for external tools – see above – otherwise alleviated, we realized that the LATEX `standalone` package to make modules independently compilable by `pdflatex`: `standalone.sty` allows for using `\input` on `.tex` files that themselves have a header, without LATEX throwing an error. This had been possible earlier, but now it is documented best practice.

2.2 sTEXLS: An sTEX Language Server and IDE

The highly fragmented structure[2] of sTEX corpora can be a challenge when creating and editing sTEX content. Some of the difficulties can be alleviated with an IDE for sTEX. To avoid being tied to a specific editor, the sTEX IDE is based on **sTEXLS**, a language server that could be used for any editor supporting the language server protocol. sTEXLS has its roots in a bachelor's thesis [Pli18], which explored machine-learning-based approaches to find missing annotations of term references in sTEX documents. The student behind [Pli18] has continued working on sTEXLS, which now supports various features that help authoring sTEX content. Aside from enabling simple interactions like cross-file definition look-up, a key feature of sTEXLS is its ability to point out semantic problems in the source files (**semantic linting**). This ranges from minor issues like redundant imports to actual errors like references to non-existent concepts. sTEXLS addresses such errors by e.g. listing modules from which the concepts could be imported or by suggesting similar sounding concepts in case of a spelling mistake. These features are so useful that sTEXLS is now commonly used for the creation of sTEX content.

2.3 Generating Supplemental Material from sTEX Sources

The semantic annotations allow deriving a number of supplemental resources for an sTEX document. Concretely, we have tools to automatically generate dictionaries, glossaries and dependency graphs e.g. to supplement the lecture notes. These tools act directly on the sTEX sources, utilizing the the uniform structure of semantic annotations.

Dictionary generation exploits that definienda in a (natural language) definition are explicitly annotated by a semantic concept. E.g. the synonyms "*linear ordering*" and "*simple ordering*", as well as the German translation, "*lineare Ordnung*", are identified as same concept. An English-to-German dictionary

[2] As TEX cannot load document fragments natively, it is natural to prefer very small source files that only contain small semantically self-contained fragments

would then have two entries, one for "*linear ordering*" and one for "*simple ordering*". To generate the dictionary for a lecture, we only include concepts that have been referenced in the lecture notes/slides.

If we map words to their definition rather than their translations, we have a generated glossary. To make it "definitionally closed", we also include entries for all concepts referenced elsewhere in the glossary.

We already routinely generate dictionaries and glossaries for some of our lectures, which was appreciated by the students. We have also experimented with the generation of concept dependency graphs but our visualization efforts had limited success so far due to the sheer size of resulting graphs.

2.4 sTEXML2: Partial Preloading and XHTML Harvesting

So far, to obtain formal content from sTEX documents, these documents needed to be converted to OMDoc. To subsequently enable KM services, the resulting OMDoc needs to be additionally converted to the specific OMDoc dialect used by the Mmt system by splitting it into (formal) *content* OMDoc and (informal) *narrative* omdoc. This workflow requires:

1. Dedicated sTEX document classes for OMDoc,
2. an sTEX-Plugin for LaTeXML that allows generating OMDoc, partially overriding core methods of LaTeXML,
3. a suite of LaTeXML bindings for most (if not all) sTEX primitive macros, and
4. an Mmt component for importing the OMDoc generated by LaTeXML.

All of these components needed to be consistently kept in-sync with respect to any updates regarding the sTEX-package, LaTeXML, and Mmt, and as a result regularly suffered from bitrot and increasingly bloated and difficult to maintain implementations. Additionally, OMDoc generation was incompatible with the document classes commonly used by LaTeX users.

As a result, we have deprecated the direct OMDoc generation via LaTeXML and the sTEX-Plugin and are re-basing the sTEX packages on a very selective set of semantic primitives. Note that we only need to implement LaTeXML bindings for these and can reuse majority of sTEX functionality implemented in TEX – LaTeXML covers enough TEX/LaTeX primitives by now. This makes it much easier to maintain coherence between the LaTeX implementation and the LaTeXML bindings. We now use LaTeXML to generate XHTML (which LaTeXML supports natively) with OMDoc-annotations (provided by the package bindings alone). Crucially, this is compatible with all existing LaTeXML bindings, and yields documents that can be immediately inspected with a web browser without loss of document content or significantly impacting layout, maintaining the narrative structure of the original document while introducing *partial* OMDoc information where induced by semantic macros. Afterwards, the Mmt system can harvest the generated XHTML to extract the OMDoc fragments relevant for KM services. This simple change of approach realizes an old desideratum in the sLaTeX project: flexibly mixing (partial) sTEX functionality into arbitrary LaTeX document classes – this is called "sTEX light" in [KKL10].

3 Conclusion & Future Work

We have significantly improved the user-friendliness of the sLATEX ecosystem by minimizing the number of *required* external tools and simplifying the general workflow, while supplying additional *optional* tools for added-value services.

As future work, we intend to

1. extend the sTEX language to subsume all primitives of the MMT/OMDOC ontology, allowing sTEX to serve as a full surface language for MMT,
2. allow for semantic markup of arbitrary (in particular informal natural language) document fragments, and
3. improve and extend IDE support, e.g. by providing direct access and search functionality for the SMGloM.

References

[Hor+11] Fulya Horozal et al. "Combining Source, Content, Presentation, Narration, and Relational Representation". In: *Intelligent Computer Mathematics*. Ed. by James Davenport et al. LNAI 6824. Springer Verlag, 2011, pp. 212–227. ISBN: 978-3-642-22672-4. URL: `https://kwarc.info/frabe/Research/HIJKR_dimensions_11.pdf`.

[KKL10] Andrea Kohlhase, Michael Kohlhase, and Christoph Lange. "sTeX – A System for Flexible Formalization of Linked Data". In: *Proceedings of the 6th International Conference on Semantic Systems (I-Semantics) and the 5th International Conference on Pragmatic Web*. Ed. by Adrian Paschke et al. ACM, 2010. ISBN: 978-1-4503-0014-8. DOI: `10.1145/1839707.1839712`. arXiv: `1006.4474v1 [cs.SE]`.

[Koh06] Michael Kohlhase. *OMDoc – An open markup format for mathematical documents [Version 1.2]*. LNAI 4180. Springer Verlag, Aug. 2006. URL: `http://omdoc.org/pubs/omdoc1.2.pdf`.

[Koh08] Michael Kohlhase. "Using LATEX as a Semantic Markup Format". In: *Mathematics in Computer Science* 2.2 (2008), pp. 279–304. URL: `https://kwarc.info/kohlhase/papers/mcs08-stex.pdf`.

[Koh14] Michael Kohlhase. "A Data Model and Encoding for a Semantic, Multilingual Terminology of Mathematics". In: *Intelligent Computer Mathematics 2014*. Ed. by Stephan Watt et al. LNCS 8543. Springer, 2014, pp. 169–183. ISBN: 978-3-319-08433-6. URL: `https://kwarc.info/kohlhase/papers/cicm14-smglom.pdf`.

[LTX] Bruce Miller. *LaTeXML: A LATEX to XML Converter*. URL: `http://dlmf.nist.gov/LaTeXML/` (visited on 03/12/2021).

[Pli18] Marian Plivelic. "Using machine learning to support annotating of keywords in mathematical texts". B.Sc. Thesis. FAU Erlangen-Nürnberg, Feb. 2018. URL: `https://gl.kwarc.info/supervision/BSc-archive/blob/master/2018/Plivelic_Marian.pdf`.

[sLX] *sLaTeX: An Ecosystem for Semantically Enhanced LATEX*. URL: `https://github.com/sLaTeX` (visited on 03/11/2021).

[sTeX] *sTeX: A semantic Extension of TeX/LaTeX*. URL: `https://github.com/sLaTeX/sTeX` (visited on 05/11/2020).