MMTTeX: Connecting Content and Narration-Oriented Document Formats

Florian Rabe

Universities Paris-Sud and Erlangen-Nuremberg

Abstract. Narrative, presentation-oriented assistant systems for mathematics such as LaTeX on the one hand and formal, content-oriented ones such as proof assistants and computer algebra systems on the other hand have so far been developed and used largely independently. The former excel at communicating mathematical knowledge and the latter at certifying its correctness.

MMTTeX aims at combining the advantages of the two paradigms. Concretely, we use LaTeX for the narrative and MMT for the content-oriented representation. Formal objects may be written in MMT and imported into LaTeX documents or written in the LaTeX document directly. In the latter case, MMT parses and checks the formal content during LaTeX compilation and substitutes it with LaTeX presentation macros.

Besides checking the formal objects, this allows generating higher-quality LaTeX than could easily be produced by hand, e.g., by inserting hyperlinks and tooltips into formulas. Moreover, it allows reusing formalizations across narrative documents as well as between formal and narrative ones. As a case study, the present document was already written with MMTTeX.

1 Introduction

A major open problem in mathematical document authoring is to elegantly combine formal and informal mathematical knowledge. Multiple proof assistants and controlled natural language systems have developed custom formats for that purpose, e.g., [Wen11,TB85,WAB07,CFK+09]. Other languages L allow for integrating LaTeX into L-source files in a literate programming style (e.g., lhs2tex[1] for Haskell) or for integrating L-source chunks into LaTeX files (e.g., SageTeX[2] for SageMath). The design of MMTTeX is close to the latter, i.e., to combine MMT and LaTeX sources.

The goal of MMTTeX is very similar to sTeX [Koh08], where authors add content markup that allows converting a LaTeX to an OMDoc document. MMTTeX differs in several ways: authors write *fully formal* content in MMT syntax [RK13] directly in the LaTeX source, either in addition to or instead of informal text; and MMT is used to type-check the formal parts during LaTeX compilation workflow. This enables several advanced features: Formal content in LaTeX

[1] https://www.andres-loeh.de/lhs2tex/

[2] https://github.com/sagemath/sagetex

sources can use or be used by MMT content written elsewhere; in particular, background knowledge formalized elsewhere can be used inside the LaTeX document. And formulas written in MMT syntax are not only type-checked but result in high-quality pdf by, e.g., displaying inferred types as tooltips or adding hyperlinks to operator occurrences.

Online Resources All resources are available as a part of the MMT repository, specifically at `https://github.com/UniFormal/MMT/tree/devel/src/latex-mmt` for the current version. These resources include the MMT and LaTeX side of the implementation, the system documentation, and the sources of this paper, which is itself written with MMTTeX.

Acknowledgments Both this paper and the MMTTeX implementation were redone from scratch by the author. But that work benefited substantially from two unpublished prototypes that were previously developed in collaboration with Deyan Ginev, Mihnea Iancu, and Michael Kohlhase. The author was supported by DFG grant RA-18723-1 OAF and EU grant Horizon 2020 ERI 676541 OpenDreamKit.

2 Design and Behavior

2.1 Overview

MMTTeX consists of two components:
- an MMT plugin `latex-mmt` that converts MMT theories to LaTeX packages,
- a small LaTeX package `mmttex.sty` (about 100 loc with only standard dependencies), which allows for embedding MMT content.

The two components are tied together in bibtex-style, i.e.,
1. While running LaTeX on `doc.tex`, `mmttex.sty` produces an additional output file `doc.tex.mmt`, which is a regular MMT file. `doc.tex.mmt` contains all formal content that was embedded in the `tex` document.
2. `latex-mmt` is run on `doc.tex.mmt` and produces `doc.tex.mmt.sty`. This type-checks the embedded formal content and generates macro definitions for rendering it in the following step.
3. When running LaTeX the next time, the package `doc.tex.mmt.sty` is included at the beginning. Now all embedded formal content is rendered using the macro definitions from the previous step. If the formal content changed, `doc.tex.mmt` also changes.

Note that `latex-mmt` only needs to be re-run if the formal content document changed. That is important for sharing documents with colleagues or publishers who want to or can only run plain LaTeX: by simply supplying `doc.tex.mmt.sty` along with `doc.tex`, running plain LaTeX is sufficient to build `doc.pdf`.

2.2 Formal Content in LaTeX Documents

`mmttex.sty` provides presentation-*irrelevant* and presentation-*relevant* macros for embedding formal content in addition to resp. instead of informal text.

Presentation-irrelevant macros only affect `doc.tex.mmt` and do not produce anything that is visible in the pdf document. These macros can be used to embed a (partial) formalization of the informal text. The formalization can occur as a single big chunk, be interspersed with the LaTeX source akin to parallel markup, or be anything in between. Importantly, if split into multiple chunks, one formal chunk may introduce names that are referred to in other formal chunks, and LaTeX environments are used to build nested scopes for these names.

At the lowest level, this is implemented by a single macro that takes a string and appends it to `doc.tex.mmt`. On top, we provide a suite of syntactic sugar that mimics the structure of the MMT language.

As a simple example, we will now define the theory of groups informally and embed its parallel MMT formalization into this paper. Of course, the embedded formalization is invisible in the pdf. Therefore, we added listings in gray that show the presentation-irrelevant macros that occur in the LaTeX sources of this paper and that embed the formalization. If this is confusing, readers may want to inspect the source code of this paper at the URL given above.

Our example will refer to the theory SFOLEQ, which formalizes sorted first-order logic with equality and is defined in the `examples` archive of MMT.[3] To refer to it conveniently, we will import its namespace under the abbreviation ex.

```
\mmtimport{ex}{http://cds.omdoc.org/examples}
\begin{mmttheory}{Group}{ex:?SFOLEQ}
```

A group consists of

- a set U,

  ```
  \mmtconstant{U}{sort}{}{}
  ```

- an operation $U \to U \to U$, written as infix $*$,

  ```
  \mmtconstant{operation}{tm U --> tm U --> tm U}{}{1 * 2 prec 50}
  ```

- an element e of U called the unit

  ```
  \mmtconstant{unit}{tm U}{}{e}
  ```

- an inverse element function $U \to U$, written as postfix $'$ and with higher precedence than $*$.

  ```
  \mmtconstant{inv}{tm U --> tm U}{}{1 ' prec 60}
  ```

We omit the axioms.

```
\end{mmttheory}
```

[3] See `https://gl.mathhub.info/MMT/examples/blob/master/source/logic/sfol.mmt`.

Here the environment mmttheory wraps around the theory. It takes two arguments: the name and the meta-theory, i.e., the logic in which the theory is written.

The macro mmtconstant introduces a constant declaration inside a theory. It takes 4 arguments: the name, type, definiens, and notation. All but the name may be empty.

We can also use the MMT module system, e.g., the following creates a theory that extends Group with a definition of division (where square brackets are the notation for λ-abstraction employed by SFOLEQ):

```
\begin{mmttheory}{Division}{ex:?SFOLEQ}
\mmtinclude{?Group}
\mmtconstant{division}
   {tm U --> tm U --> tm U}{[x,y]  x*y'}{1  /  2  prec  50}
```

Note that we have not closed the theory yet, i.e., future formal objects will be processed in the scope of Division.

Presentation-relevant macros result in changes to the pdf document. The most important such macro provided by `mmttex.sty` is one that takes a math formula in MMT syntax and parses, type-checks, and renders it. For this macro, we provide special syntax that allows using quotes instead of dollars to have formulas processed by MMT: if we write "F" (including the quotes) instead of \$$F$\$, then F is considered to be a math formula in MMT syntax and processed by MMT. For example, the formula " forall [x] x / x = e" is parsed by MMT relative to the current theory, i.e., Division; then MMT type-checks it and substitutes it with LaTeX commands. In the previous sentence, the LaTeX source of the quoted formula is additionally wrapped into a verbatim macro to avoid processing by MMT; if we remove that wrapper, the quoted formula is rendered into pdf as $\forall[x]\frac{x}{x}\doteq e$.

Type checking the above formula infers the type tm U of the bound variable x. This is shown as a tooltip when hovering over the binding location of x. (Tooltip display is supported by many but not all pdf viewers. Unfortunately, pdf tooltips are limited to strings so that we cannot show tooltips containing LaTeX or MathML renderings even though we could generate them easily.) Similarly, the sort argument of equality is inferred. Moreover, every operator carries a hyperlink to the point of its declaration. Currently, these links point to the MMT server, which is assumed to run locally.

This is implemented as follows:

1. An MMT formula "F" simply produces a macro call \mmt@X for a running counter X. If that macro is undefined, a placeholder is shown and the user is warned that a second compilation is needed. Additionally, a definition mmt@X = F in MMT syntax is placed into `doc.tex.mmt`.
2. When `latex-mmt` processes that definition, it additionally generates a macro definition \newcommand{\mmt@X}{$\overline{F}$} in `doc.tex.mmt.sty`, where $\overline{F}$ is the intended LaTeX representation of F.

3. During the next compilation, MMT @X produces the rendering of $\overline{F}$. If F did not type-check, additional a LaTeX error is produced with the error message.

Before we continue, we close the current theory:

\end{mmttheory}

2.3 Converting MMT Content To LaTeX

We run `latex-mmt` on every theory T that is part of the background knowledge, e.g., SFOLEQ, and on all theories that are part of `doc.tex.mmt`, resulting in one LaTeX package (sty file) each. This package contains one \RequirePackage macro for every dependency and one \newcommand macro for every constant declaration. `doc.tex.mmt.sty` is automatically included at the beginning of the document and thus brings all necessary generated LaTeX packages into scope.

The generated \newcommand macros use (only) the notation of the constant. For example, for the constant named operator from above, the generated command is essentially \newcommand{\operator}[2]{#1∗#2}. Technically, however, the macro definition is much more complex: Firstly, instead of #1∗#2, we produce a a macro definition that generates the right tooltips, hyperreferences, etc. Secondly, instead of \operator, we use the fully qualified MMT URI as the LaTeX macro name to avoid ambiguity when multiple theories define constants of the same local name.

The latter is an important technical difference between MMTTeX and sTeX [Koh08]: sTeX intentionally generates short human-friendly macro names because they are meant to be called by humans. That requires relatively complex scope management and dynamic loading of macro definitions to avoid ambiguity. But that is inessential in our case because our macros are called by generated LaTeX commands (namely those in the definiens of \mmt@X). Nonetheless, it would be easy to add macros to `mmttex.sty` for creating aliases with human-friendly names.

The conversion from MMT to LaTeX can be easily run in batch mode so that any content available in MMT can be easily used as background knowledge in LaTeX documents.

3 Conclusion

We have presented a system that allows embedding formal MMT content inside LaTeX documents. The formal content is type-checked in a way that does not affect any existing LaTeX work flows and results in higher quality LaTeX than could be easily produced manually. Moreover, the formal content may use and be used by any MMT content formalized elsewhere, which allows interlinking across document formats.

Of course, we are not able to verify the informal parts of a document this way — only those formulas that are written in MMT syntax are checked. But our design supports both gradual formalization and parallel formal-informal representations.

It is intriguing to apply the same technology to formal proofs. This is already possible for formal proof terms, but those often bear little resemblance to informal proofs. Once Mmt supports a language for structured proofs, that could be used to write formal proofs in MMTTeX. Morever, future work could apply Mmt as a middleware between LaTeX and other tools, e.g., Mmt could run a computation through a computer algebra system to verify a computation in LaTeX.

References

CFK+09. M. Cramer, B. Fisseni, P. Koepke, D. Kühlwein, B. Schröder, and J. Veldman. The Naproche Project Controlled Natural Language Proof Checking of Mathematical Texts. In N. Fuchs, editor, *Controlled Natural Language*, pages 170–186. Springer, 2009.

Koh08. M. Kohlhase. Using LaTeX as a Semantic Markup Format. *Mathematics in Computer Science*, 2(2):279–304, 2008.

RK13. F. Rabe and M. Kohlhase. A Scalable Module System. *Information and Computation*, 230(1):1–54, 2013.

TB85. A. Trybulec and H. Blair. Computer Assisted Reasoning with MIZAR. In A. Joshi, editor, *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, pages 26–28. Morgan Kaufmann, 1985.

WAB07. M. Wagner, S. Autexier, and C. Benzmüller. PLATO: A Mediator between Text-Editors and Proof Assistance Systems. In S. Autexier and C. Benzmüller, editors, *7th Workshop on User Interfaces for Theorem Provers (UITP'06)*, pages 87–107. Elsevier, 2007.

Wen11. M. Wenzel. Isabelle as a document-oriented proof assistant. In J. Davenport, W. Farmer, J. Urban, and F. Rabe, editors, *Intelligent Computer Mathematics*, pages 244–259. Springer, 2011.