omtext: Semantic Markup for Mathematical Text Fragments in LaTeX[*]

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

April 4, 2016

Abstract

The omtext package is part of the sTeX collection, a version of TeX/LaTeX that allows to markup TeX/LaTeX documents semantically without leaving the document format, essentially turning TeX/LaTeX into a document format for mathematical knowledge management (MKM).

This package supplies an infrastructure for writing OMDoc text fragments in LaTeX.

[*]Version v1.1 (last revised 2015/11/22)

Contents

1 Introduction

The `omtext` package supplies macros and environment that allow to mark up mathematical texts in sTeX, a version of TeX/LaTeX that allows to markup TeX/LaTeX documents semantically without leaving the document format, essentially turning TeX/LaTeX into a document format for mathematical knowledge management (MKM). The package supports direct translation to the OMDoc format [Koh06]

2 The User Interface

2.1 Package Options

`showmeta` The `omtext` package takes a single option: `showmeta`. If this is set, then the metadata keys are shown (see [Koh15a] for details and customization options).

2.2 Mathematical Text

`omtext` The `omtext` environment is used for any text fragment that has a contribution to a text that needs to be marked up. It can have a title, which can be specified via the `title=` `title` key. Often it is also helpful to annotate the `type` key. The standard relations `type=` from rhetorical structure theory `abstract`, `introduction`, `conclusion`, `thesis`, `comment`, `antithesis`, `elaboration`, `motivation`, `evidence`, `transition`, `note`, `annote` are recommended as values. Note that some of them are unary relations `for=` like `introduction`, which calls for a target. In this case, a target using the `for` key should be specified. The `transition` relation is special in that it is binary (a "transition between two statements"), so additionally, a source should be specified `from=` using the `from` key.

Note that the values of the `title` and `type` keys are often displayed in the text. `display=` This can be turned off by setting the `display` key to the value `flow`. Sometimes we want to specify that a text is a continuation of another, this can be done by `continues=` giving the identifier of this in the `continues` key.

Finally, there is a set of keys that pertain to the mathematical formulae in `functions=` the text. The `functions` key allows to specify a list of identifiers that are to be `theory=` interpreted as functions in the generate content markup. The `theory` specifies a EdN:1 module (see [KGA15a]) that is to be pre-loaded in this one[1] Finally, `verbalizes` `verbalizes=` specifies a (more) formal statement (see [Koh15b]) that this text verbalizes or EdN:2 paraphrases.[2]

2.3 Phrase-Level Markup

`\phrase` The `phrase` macro allows to mark up phrases with semantic information. It takes `verbalizes=` an optional `KeyVal` argument with the keys `verbalizes` and `type` as above and `type=` `style, class, index` that are disregarded in the LaTeX, but copied into the gen-

`style` `class` `index`

[1] EdNote: this is not implemented yet.
[2] EdNote: MK:specify the form of the reference.

erated content markup.

\nlex \nlcex We use the `\nlex{⟨phrase⟩}` for marking up phrases that serve as natural language examples and `\nlcex{⟨phrase⟩}` for counter-examples (utterances that are not acceptable for some reason). In natural language examples, we sometimes use "co-reference markers" to specify the resolution of anaphora and the like. We

\coreft \corefs use the `\coreft{⟨phrase⟩}{⟨mark⟩}` to mark up the "target" of a co-reference and analogously `\corefs` for coreference source – e.g. for an anaphoric reference. The usage is the following:

```
\nlex{If \coreft{a farmer}1 owns \coreft{a donkey}2,
        \corefs{he}2 beats \corefs{it}2.}
```

is formatted to

If a farmer^{1} *owns a donkey*^{2}*, he*$_{2}$ *beats it*$_{2}$*.*

\sinlinequote The `sinlinequote` macro allows to mark up quotes inline and attribute them. The quote itself is given as the argument, possibly preceded by the a specification of the source in a an optional argument. For instance, we would quote Hamlet with

```
\sinlinequote[Hamlet, \cite{Shak:1603:Hamlet}]{To be or not to be}
```

which would appear as "*To be or not to be*" Hamlet, (Shakespeare 1603) in the text. The style in which inline quotations appear in the text can be adapted

\@sinlinequote \@@sinlinequote by specializing the macros `\@sinlinequote` — for quotations without source and `\@@sinlinequote` — for quotations with source.

2.4 Declarations (under development)

Declarations are special phrases that carry a lot of meaning in mathematics: they introduce and further specify the indentifiers available in formulae. They are

\vdec \vids \vrest marked up via the `\vdec` macro. Inside a declaration we can use the `\vids` macro to mark up the variable names and `\vrest` to mark up the restrictions. In the simplest case we have a single variable as in "…for all u", which we mark up as.

```
... for all \vdec{\vids[u]{$u$}}.
```

A more complex example has multiple identifiers embedded in a restriction, as in "Let $x, y, z \in \mathbb{R}$, such that $x + 2y = z$, then …", which we mark up as

```
Let \vdec{\vids[x,y,z]{$\minset{x,y,z}\Reals$},
   such that \vrest{$x+2y=z$}}, then \ldots''
```

EdN:3 EdN:4 EdN:5 EdN:6 3456

[3] EdNote: explain and make better examples

[4] EdNote: unify this with the vardef macros from (currently) the modules package.

[5] EdNote: talk with Frederic about this see what other examples there are.

[6] EdNote: how do we identify the variables in complex restriction patterns. maybe with LMXref, which we should reinstate for this.

2.5 Block-Level Markup

sblockquote The sblockquote environment is the big brother of the \sinlinequote macro. It also takes an optional argument to specify the source. Here the four internal macros \begin@sblockquote to \end@@sblockquote are used for styling and can be adapted by package integrators. Here a quote of Hamlet would marked up as

\begin@sblockquote
\end@@sblockquote

```
\begin{sblockquote}[Hamlet, \cite{Shak:1603:Hamlet}]\obeylines
  To be, or not to be: that is the question:
  Whether 'tis nobler in the mind to suffer
\end{sblockquote}
```

and would render as

To be, or not to be: that is the question:

Whether 'tis nobler in the mind to suffer

Hamlet, (Shakespeare 1603)

\lec The \lec macro takes one argument and sets it as a comment at the end of the line, making sure that if the content is too long it is pushed into a new line. We use it internally for placing the of source of the sblockquote environment above. The actual appearance of the line end comment is determined by the \@@lec macro, which can be customized in the document class.

\@@lec

2.6 Index Markup

The omtext package provides some extensions for the well-known indexing macros of LaTeX. The main reason for introducing these macros is that index markup in OMDoc wraps the indexed terms rather than just marking the spot for cross-referencing. Furthermore the index commands only indexes words unless thenoindex option is set in the \usepackage. The omtext package and class make the usual \index macro undefined[7].

noindex

EdN:7

\indextoo The \indextoo macro renders a word and marks it for the index. Sometimes, we want to index a slightly different form of the word, e.g. for non-standard plurals: while \indextoo{word}s works fine, we cannot use this for the word "datum", which has the plural "data". For this we have the macro \indexalt, which takes another argument for the displayed text, allowing us to use \indexalt{data}{datum}, which prints "data" but puts "datum" into the index.

\indexalt

The second set of macros adds an infrastructure for two-word compounds. Take for instance the compound "OMDoc document", which we usually want to add into the index under "OMDoc" and "document". \twintoo{OMDoc}{document} is a variant of \indextoo that will do just this. Again, we have a version that prints a variant: This is useful for situations like this the one in Figure 1:

\twintoo

The third set of macros does the same for two-word compounds with adjectives, e.g. "wonderful OMDoc document". \atwin{wonderful}{OMdoc}{document}

\atwintoo

[7]EdNote: implement this and issue the respective error message

```
We call group \twinalt{Abelian}{Abelian}{group}, iff \ldots
```

will result in the following

We call group Abelian, iff ...

and put “Abelian Group” into the index.

Example 1: Index markup

will make the necessary index entries under “wonderful” and “document”. Again, we have a variant `\atwinalt` whose first argument is the alternative text.

`\atwinalt`

All index macros take an optional first argument that is used for ordering the respective entries in the index.

3 Limitations

In this section we document known limitations. If you want to help alleviate them, please feel free to contact the package author. Some of them are currently discussed in the sTeX GitHub repository [sTeX].

1. none reported yet

4 Implementation

4.1 Package Options

We declare some switches which will modify the behavior according to the package options. Generally, an option xxx will just set the appropriate switches to true (otherwise they stay false).[8] EdN:8

```
⟨*package⟩
\newif\if@omtext@mh@\@omtext@mh@false
\DeclareOption{mh}{\@omtext@mh@true
\PassOptionsToPackage{\CurrentOption}{modules}}
\newif\ifindex\indextrue
\DeclareOption{noindex}{\indexfalse}
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{modules}}
\ProcessOptions
\ifindex\makeindex\fi
```

```
\if@omtext@mh@\RequirePackage{omtext-mh}\fi
\RequirePackage{xspace}
\RequirePackage{modules}
\RequirePackage{comment}
\RequirePackage{mdframed}
\RequirePackage{latexsym}
```

4.2 Mathematical Text

We define the actions that are undertaken, when the keys are encountered. The first set just records metadata; this is very simple via the \addmetakey infrastructure [Koh15a]. Note that we allow math in the title field, so we do not declare it to be Semiverbatim (indeed not at all, which allows it by default).

```
\srefaddidkey{omtext}
\addmetakey[]{omtext}{functions}
\addmetakey*{omtext}{display}
\addmetakey{omtext}{for}
\addmetakey{omtext}{from}
\addmetakey{omtext}{type}
\addmetakey*{omtext}{title}
\addmetakey*{omtext}{start}
\addmetakey{omtext}{theory}
\addmetakey{omtext}{continues}
\addmetakey{omtext}{verbalizes}
\addmetakey{omtext}{subject}
```

The next keys handle module loading (see [KGA15b]).

```
% \ednote{MK: need to implement these in LaTeXML, I wonder whether there is a general
% mechanism like numberit.}\ednote{MK: this needs to be rethought in the light of
% |\usemodule|. It is probably obsolete. Is this used? Is this documented?}
```

[8]EdNote: need an implementation for LaTeXML

```
\define@key{omtext}{require}{\requiremodules{#1}{sms}}
\define@key{omtext}{module}{\message{module: #1}\importmodule{#1}\def\omtext@theory{#1}}
```

`\st@flow` We define this macro, so that we can test whether the `display` key has the value `flow`

```
\def\st@flow{flow}
```

We define a switch that allows us to see whether we are inside an `omtext` environment or a statement. It will be used to give better error messages for inline statements.

```
\newif\if@in@omtext\@in@omtextfalse
```

`omtext` The `omtext` environment is different, it does not have a keyword that marks it. Instead, it can have a title, which is used in a similar way. We redefine the `\lec` macro so the trailing `\par` does not get into the way.

```
\def\omtext@pre@skip{\smallskip}
\def\omtext@post@skip{}
\providecommand{\stDMemph}[1]{\textbf{#1}}
\newenvironment{omtext}[1][]{\@in@omtexttrue%
\bgroup\metasetkeys{omtext}{#1}\sref@label@id{this paragraph}%
\def\lec##1{\@lec{##1}}%
\ifx\omtext@display\st@flow\else\omtext@pre@skip\par\noindent%
\ifx\omtext@title\@empty%
\ifx\omtext@start\@empty\else\stDMemph{\omtext@start}\xspace\fi%
\else\stDMemph{\omtext@title}:\xspace%
\ifx\omtext@start\@empty\else\omtext@start\xspace\fi%
\fi% omtext@title empty
\fi% omtext@display=flow
\ignorespaces}
{\egroup\omtext@post@skip\@in@omtextfalse}
```

4.3 Phrase-level Markup

`\phrase` For the moment, we do disregard the most of the keys

```
\srefaddidkey{phrase}
\addmetakey{phrase}{style}
\addmetakey{phrase}{class}
\addmetakey{phrase}{index}
\addmetakey{phrase}{verbalizes}
\addmetakey{phrase}{type}
\addmetakey{phrase}{only}
\newcommand\phrase[2][]{\metasetkeys{phrase}{#1}%
\ifx\prhase@only\@empty\only<\phrase@only>{#2}\else #2\fi}
```

`\coref*`

```
\providecommand\textsubscript[1]{\ensuremath{_{#1}}}
\newcommand\corefs[2]{#1\textsubscript{#2}}
\newcommand\coreft[2]{#1\textsuperscript{#2}}
```

\n*lex

```
\newcommand\nlex[1]{\green{\sl{#1}}}
\newcommand\nlcex[1]{*\green{\sl{#1}}}
```

sinlinequote

```
\def\@sinlinequote#1{``{\sl{#1}}''}
\def\@@sinlinequote#1#2{\@sinlinequote{#2}~#1}
\newcommand\sinlinequote[2][]
{\def\@opt{#1}\ifx\@opt\@empty\@sinlinequote{#2}\else\@@sinlinequote\@opt{#2}\fi}
```

4.4 Declarations (under development)

The declaration macros are still under development (i.e. the macros) are still under development and may change at any time. Currently they are completely empty.

```
\newcommand\vdec[1]{#1}
\newcommand\vids[2][]{#2}
\newcommand\vrest[2][]{#2}
```

4.5 Block-Level Markup

sblockquote

```
\def\begin@sblockquote{\begin{quote}\sl}
\def\end@sblockquote{\end{quote}}
\def\begin@@sblockquote#1{\begin@sblockquote}
\def\end@@sblockquote#1{\def\@@lec##1{{\rm ##1}}\@lec{#1}\end@sblockquote}
\newenvironment{sblockquote}[1][]
  {\def\@opt{#1}\ifx\@opt\@empty\begin@sblockquote\else\begin@@sblockquote\@opt\fi}
  {\ifx\@opt\@empty\end@sblockquote\else\end@@sblockquote\@opt\fi}
```

sboxquote

```
\newenvironment{sboxquote}[1][]
{\def\@@src{#1}\begin{mdframed}[leftmargin=.5cm,rightmargin=.5cm]}
{\@lec{\rm\@@src}\end{mdframed}}
```

The line end comment macro makes sure that it will not be forced on the next line unless necessary.

\lec The actual appearance of the line end comment is determined by the `\@@lec` macro, which can be customized in the document class. The basic one here is provided so that it is not missing.

```
\providecommand{\@@lec}[1]{(#1)}
\def\@lec#1{\strut\hfil\strut\null\nobreak\hfill\@@lec{#1}}
\def\lec#1{\@lec{#1}\par}
```

\my*graphics We set up a special treatment for including graphics to respect the intended OMDoc document structure. The main work is done in the transformation stylesheet though.

```
\newcommand\mygraphics[2][]{\includegraphics[#1]{#2}}
\newcommand\mycgraphics[2][]{\begin{center}\mygraphics[#1]{#2}\end{center}}
\newcommand\mybgraphics[2][]{\fbox{\mygraphics[#1]{#2}}}
\newcommand\mycbgraphics[2][]{\begin{center}\fbox{\mygraphics[#1]{#2}}\end{center}}
```

4.6 Index Markup

\omdoc@index* these are the main internal indexing commands – dividing them into three is awful, but I did not get list processing running. It makes sure that the modules necessary for interpreting the math in the index entries are loaded. If the `loadmodules` key is given, we import the module we are in otherwise all the currently imported modules. We do not have to require the module files, since the index is a the end of the document. If the `at` key is given, then we use that for sorting in the index.

```
\addmetakey{omdoc@index}{at}
\addmetakey[false]{omdoc@index}{loadmodules}[true]
\newcommand\omdoc@indexi[2][]{\ifindex%
\metasetkeys{omdoc@index}{#1}%
\@bsphack\begingroup\@sanitize%
\protected@write\@indexfile{}{\string\indexentry%
{\ifx\omdoc@index@at\@empty\else\omdoc@index@at @\fi%
\ifx\omdoc@index@loadmodules\@true%
\string\withusedmodules{\@ifundefined{mod@id}\used@modules\mod@id}{#2}%
\else #2\fi% loadmodules
}{\thepage}}%
\endgroup\@esphack\fi}%ifindex
\newcommand\omdoc@indexii[3][]{\ifindex%
\metasetkeys{omdoc@index}{#1}%
\@bsphack\begingroup\@sanitize%
\protected@write\@indexfile{}{\string\indexentry%
{\ifx\omdoc@index@at\@empty\else\omdoc@index@at @\fi%
\ifx\omdoc@index@loadmodules\@true%
\string\withusedmodules{\@ifundefined{mod@id}\used@modules\mod@id}{#2}!%
\string\withusedmodules{\@ifundefined{mod@id}\used@modules\mod@id}{#3}%
\else #2!#3\fi% loadmodules
}{\thepage}}%
\endgroup\@esphack\fi}%ifindex
\newcommand\omdoc@indexiii[4][]{\ifindex%
\metasetkeys{omdoc@index}{#1}%
\@bsphack\begingroup\@sanitize%
\protected@write\@indexfile{}{\string\indexentry%
{\ifx\omdoc@index@at\@empty\else\omdoc@index@at @\fi%
\ifx\omdoc@index@loadmodules\@true%
\string\withusedmodules{\@ifundefined{mod@id}\used@modules\mod@id}{#2}!%
\string\withusedmodules{\@ifundefined{mod@id}\used@modules\mod@id}{#3}!%
\string\withusedmodules{\@ifundefined{mod@id}\used@modules\mod@id}{#4}%
\else #2!#3\fi% loadmodules
}{\thepage}}%
\endgroup\@esphack\fi}%ifindex
```

Now, we make two interface macros that make use of this:

\indexalt

```
\newcommand\indexalt[3][]{{#2}\omdoc@indexi[#1]{#3}}               % word in text and index
```

\indextoo

```
\newcommand\indextoo[2][]{{#2}\omdoc@indexi[#1]{#2}}               % word in text and index
```

\@twin this puts two-compound words into the index in various permutations

```
\newcommand\@twin[3][]{\omdoc@indexii[#1]{#2}{#3}\omdoc@indexii[#1]{#3}{#2}}
```

And again we have two interface macros building on this

\twinalt

```
\newcommand\twinalt[4][]{#2\@twin[#1]{#3}{#4}}
```

\twinalt

```
\newcommand\twintoo[3][]{{#2 #3}\@twin[#1]{#2}{#3}} % and use the word compound too
```

EdN:9 \@atwin this puts adjectivized two-compound words into the index in various permutations[9]

```
\newcommand\@atwin[4][]{\omdoc@indexiii[#1]{#2}{#3}{#4}\omdoc@indexii[#1]{#3}{#2 (#4)}}
```

and the two interface macros for this case:

\@atwinalt

```
\newcommand\atwinalt[5][]{#2\@atwin[#1]{#3}{#4}{#4}}
```

\atwintoo

```
\newcommand\atwintoo[4][]{{#2 #3 #4}\@atwin[#1]{#2}{#3}{#4}}         % and use it too
```

4.7 Miscellaneous

Some shortcuts that use math symbols but are not mathematical at all; in particular, they should not be translated by LaTeXML.

```
\newcommand\hateq{\ensuremath{\hat=}\xspace}
\newcommand\hatequiv{\ensuremath{\hat\equiv}\xspace}
\@ifundefined{ergo}%
{\newcommand\ergo{\ensuremath{\leadsto}\xspace}}%
{\renewcommand\ergo{\ensuremath{\leadsto}\xspace}}%
\newcommand{\reflect@squig}[2]{\reflectbox{$\m@th#1\rightsquigarrow$}}%
\newcommand\ogre{\ensuremath{\mathrel{\mathpalette\reflect@squig\relax}}\xspace}%
⟨/package⟩
```

[9]EdNote: what to do with the optional argument here and below?

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

References

[KGA15a] Michael Kohlhase, Deyan Ginev, and Rares Ambrus. *modules.sty: Semantic Macros and Module Scoping in sTeX*. Tech. rep. 2015. URL: https://github.com/KWARC/sTeX/raw/master/sty/modules/modules.pdf.

[KGA15b] Michael Kohlhase, Deyan Ginev, and Rares Ambrus. *modules.sty: Semantic Macros and Module Scoping in sTeX*. Tech. rep. Comprehensive TEX Archive Network (CTAN), 2015. URL: http://www.ctan.org/get/macros/latex/contrib/stex/modules/modules.pdf.

[KGA15c] Michael Kohlhase, Deyan Ginev, and Rares Ambrus. *modules.sty: Semantic Macros and Module Scoping in sTeX*. Self-documenting LATEX package. 2015.

[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.

[Koh15a] Michael Kohlhase. *metakeys.sty: A generic framework for extensible Metadata in LATEX*. Tech. rep. Comprehensive TEX Archive Network (CTAN), 2015. URL: http://www.ctan.org/tex-archive/macros/latex/contrib/stex/metakeys/metakeys.pdf.

[Koh15b] Michael Kohlhase. *statements.sty: Structural Markup for Mathematical Statements*. Tech. rep. 2015. URL: https://github.com/KWARC/sTeX/raw/master/sty/statements/statements.pdf.

[sTeX] *KWARC/sTeX*. URL: https://svn.kwarc.info/repos/stex (visited on 05/15/2015).