**omtext**: Semantic Markup for Mathematical Text Fragments in \LaTeX\*

Michael Kohlhase
Jacobs University, Bremen

[http://kwarc.info/kohlhase](http://kwarc.info/kohlhase)

January 28, 2012

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 v1.0 (last revised 2012/01/28)
1 Introduction

The omtext package supplies macros and environment that allow to mark up mathematical texts in T\(\TeX\)/\(\LaTeX\), 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 [Koh10a] 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 key. Often it is also helpful to annotate the type key. The standard relations from rhetorical structure theory abstract, introduction, conclusion, thesis, comment, antithesis, elaboration, motivation, evidence, transition, note, annotate are recommended as values. Note that some of them are unary relations 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 using the from key.

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 giving the identifier of this in the continues key.

continues=

functions The functions key allows to specify a list of identifiers that are to be interpreted as functions in the generate content markup. The theory specifies a module (see [KGA10]) that is to be pre-loaded in this one. Finally, verbalizes specifies a (more) formal statement (see [Koh10b]) that this text verbalizes.

2.3 Phrase-Level Markup

phrase The phrase environment allows to mark up phrases with semantic information. It takes an optional KeyVal argument with the keys verbalizes and type as above and style, class, index that are disregarded in the \(\LaTeX\), but copied into the generated content markup.

style

class

index

\sinlinequote

EdNote: 1

EdNote: this is not implemented yet.
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 by specializing the macros \@sinlinequote — for quotations without source and \@@sinlinequote — for quotations with source.

2.4 Block-Level Markup

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}[Hamlet, \cite{Shak:1603:Hamlet}] 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)

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.

2.5 Index Markup

The omtext package provides some extensions for the well-known indexing macros of \LaTeXX. 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 the noindex option is set in the \usepackage. The omtext package and class make the usual \index macro undefined\footnote{EdNote: implement this and issue the respective error message}.

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

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”. \texttt{\textbackslash twintoo\{OMDoc\}\{document\}} is a variant of \texttt{\textbackslash 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:

\begin{quote}
\texttt{We call group \textbackslash twinalt\{Abelian\}\{Abelian\}\{group\}, iff \ldots}
\end{quote}

will result in the following

We call group Abelian, iff . . .

and put “Abelian Group” into the index.

\textbf{Example 1: Index markup}

The third set of macros does the same for two-word compounds with adjectives, e.g. “wonderful OMDoc document”. \texttt{\textbackslash atwin\{wonderful\}\{OMDoc\}\{document\}} will make the necessary index entries under “wonderful” and “document”. Again, we have a variant \texttt{\textbackslash atwinalt} whose first argument is the alternative text.

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

\section{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 \LaTeX TRAC [Ste].

1. none reported yet
4 Implementation

The omtext package generates two files: the \LaTeXX package (all the code between \begin{package}{package} and \end{package}) and the \LaTeXXML bindings (between \begin{ltxml} and \end{ltxml}). We keep the corresponding code fragments together, since the documentation applies to both of them and to prevent them from getting out of sync.

4.1 Package Options

The initial setup for \LaTeXXML:

\begin{verbatim}
package LaTeXML::Package::Pool;
use strict;
use LaTeXML::Package;
use Cwd qw(cwd abs_path);
\end{verbatim}

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).

\begin{verbatim}
\DeclareOption{showmeta}{\PassOptionsToPackage{\CurrentOption}{metakeys}}
\newif\ifindex\indextrue
\DeclareOption{noindex}{\indexfalse}
\ProcessOptions
\ifindex\makeindex\fi
\end{verbatim}

Then we need to set up the packages by requiring the sref package to be loaded.

\begin{verbatim}
\requirepackage{sref}
\requirepackage{comment}
\end{verbatim}

4.2 Metadata

All the OMDoc elements allow to specify metadata in them, which is modeled by the \omdoc:metadatap element. Since the content of this element is precisely controlled by the Schema, we can afford to auto-open and auto-close it. Thus metadata elements from various sources will just be included into one \omdoc:metadatap element.

\footnote{\texttt{EdNote}: need an implementation for \LaTeXXML}
element, even if they are supplied by different \TeX\ bindings. Also we add numbering and location facilities.

24 ⟨∗ltxml⟩
25 Tag('omdoc:metadata',afterOpen=>\&numberIt,afterClose=>\&locateIt,autoClose=>1,autoOpen=>1);
26 ⟨/ltxml⟩

the itemize, description, and enumerate environments generate omdoc:li, omdoc:di with autoclose inside a CMP. This behavior will be overwritten later, so we remember that we are in a CMP by assigning _LastSeenCMP.

27 ⟨∗ltxml⟩
28 sub declareFunctions{
29   my ($stomach,$whatsit) = @_;
30   my $keyval = $whatsit->getArg(1);
31   my $funval = KeyVal($keyval,'functions') if KeyVal($keyval,'functions');
32   my @funsymbs = ParseKeyValList($funval);
33   #Unread the function declarations at the Gullet
34   foreach (@funsymbs) {
35     $stomach->getGullet->unread(Tokenize('\lxDeclare[role=FUNCTION]{'.$_.'$}')->unlist);
36   }
37   return;
38 }
39 Tag('omdoc: CMP', afterOpen => sub {AssignValue('_LastSeenCMP', $_[1], 'global');return;});
40 ⟨/ltxml⟩

the itemize, description, and enumerate environments originally introduced in the omtext package do double duty in OMDoc, outside a CMP they are transformed into a <omgroup layout='itemizedescriptionenumerate'>, where the text after the macros \item come to be the children. If that is only text, then it is enclosed in an <omtext>CMP>, otherwise it is left as it is. The optional argument of the \item is transformed into the <metadata>dc:title> of the generated item element.

41 ⟨∗ltxml⟩
42 DefParameterType('IfBeginFollows', sub {
43   my ($gullet) = @_;;
44   $gullet->skipSpaces;
45   my $next = $gullet->readToken;
46   $gullet->unread($next);
47 $next = ToString($next);
48   #Hm, falling back to regexp handling, the $gullet->ifNext approach didn't work
49   return 1 unless ($next=~/^\begin/);
50   return;
51   },
52   reversion=>'', optional=>1);
53 ⟨/ltxml⟩

4.3 Mathematical Text

We define the actions that are undertaken, when the keys are encountered. Here this is very simple, we just define an internal macro with the value, so that we can
use it later. Note that we allow math in the title field, so we do not declare it to be \texttt{Semiverbatim} (indeed not at all, which allows it by default).

\begin{verbatim}
\specialaddidkey{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}
\end{verbatim}

\begin{verbatim}
\specialaddidkey{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}
\end{verbatim}

\begin{verbatim}
\st@flow
\end{verbatim}

\texttt{omtext} We define this macro, so that we can test whether the display key has the value flow

\begin{verbatim}
\def\st@flow{flow}
\end{verbatim}

\texttt{omtext} The \texttt{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 \texttt{lec} macro so the trailing \texttt{par} does not get into the way.

\begin{verbatim}
\def\omtext@pre@skip\{\smallskip\}
\def\omtext@post@skip\{}
\providecommand{\stDMemph}[1]{\textbf{#1}}
\newenvironment{omtext}[1][\{}\bgroup\metasetkeys{omtext}{#1}\sref@label@id{this paragraph}%
\def\lec##1\{\@lec{##1}}
\ifx\omtext@display\st@flow\else\omtext@pre@skip\par
\fi\ignorespaces}
\end{verbatim}

\texttt{omtext}
We also make our life easier if defining an environment that is turned into something that contains `<CMP><body></CMP>`, use this method instead.

4.4 Phrase-level Markup

For the moment, we do disregard the most of the keys.

\begin{verbatim}
4.4 Phrase-level Markup

phrase For the moment, we do disregard the most of the keys
\end{verbatim}
For the moment, we do disregard the most of the keys

\def\nlex#1{\green{\sl{#1}}}
\def\nlcex#1{\ast\green{\sl{#1}}}

\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}

\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}

4.5 Block-Level Markup

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

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

\my*graphics \text{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 through.}

\textbf{4.6 Index Markup}

\% this is the main internal indexing command. It makes sure that the modules necessary for
\% interpreting the math in the index entries are loaded.
\% \%
this puts two-compound words into the index in various permutations

\newcommand{\@twin}[3]{\ifindex\def\@test{#1}%%%%
  \ifx\@test\@empty\def\@@idx{#2}\else\def\@@idx{#1}\fi%
  \@ifundefined{mod@id}{}
  {\index{\@@idx @#2!#3}%%%%
    \ifx\@test\@empty\def\@@idx{#3}\else\def\@@idx{#1}\fi%
    \index{\@@idx @#3!#2}}}%
  \ifx\@test\@empty\def\@@idx{#3}\else\def\@@idx{#1}\fi%
  \index{\@@idx @{\importmodule{\mod@id} #2}!{\importmodule{\mod@id} #3}}}%
}\newcommand{\twinalt}[4]{#2\@twin[#1]{#3}{#4}} % and use the word compound too

EdNote:5

this puts adjectivized two-compound words into the index in various permutations

\newcommand{\@atwin}[4]{\ifindex\def\@test{#1}%%%%
  \ifx\@test\@empty\def\@@idx{#2}\else\def\@@idx{#1}\fi%
  \@ifundefined{mod@id}{}
  {\index{\@@idx @#2!#3!#4}%%%%
    \ifx\@test\@empty\def\@@idx{#3}\else\def\@@idx{#1}\fi%
    \index{\@@idx @#3!#2 (#4)}}}%
  \ifx\@test\@empty\def\@@idx{#3}\else\def\@@idx{#1}\fi%
  \index{\@@idx @{\importmodule{\mod@id} #3}!{\importmodule{\mod@id} #2} ({\importmodule{\mod@id} #4})}}%
}\newcommand{\atwinalt}[5]{#2\@atwin[#1]{#3}{#4}{#4}} % and use it too

DefConstructor(‘\indextoo{}{}’),
  "<omdoc:idx>
     . "<omdoc:idt>#2</omdoc:idt>"
     . "<omdoc:id ?#1(sort-by='1')(>
     . "</omdoc:id>
     . "</omdoc:idx>";)
DefConstructor(‘\indexalt{}{}’),
  "<omdoc:idx>
     . "<omdoc:idt>#2</omdoc:idt>"
     . "<omdoc:id ?#1(sort-by='1'(>
     . "</omdoc:id>
     . "</omdoc:idx>";)}
The first think we have to take care of are the paragraphs, we want to generate OMDoc that uses the \texttt{ltx:p} element for paragraphs inside \texttt{CMP}s. For that we have modified the DTD only to allowed \texttt{ltx:p} elements in \texttt{omdoc: CMP} (in particular no text). Then we instruct the \texttt{\par} macro to close a \texttt{ltx:p} element if possible. The next \texttt{ltx:p} element is then opened automatically, since we make \texttt{ltx:p} and \texttt{omdoc: CMP} autoclose and autoopen.

\begin{verbatim}
# needed? DefConstructor('\par', sub { $_[0]->maybeCloseElement('ltx:p'); }, alias="\par\n");
Tag('omdoc: CMP', autoClose=>1, autoOpen=>1);
\end{verbatim}
the rest of the reinterpretations is quite simple, we either disregard presentation markup or we re-interpret it in terms of OMDoc. ⁶

```latex
\begin{verbatim}
\def\omspace#1{\hspace*{#1}}
\end{verbatim}
```

4.8 Providing IDs for OMDoc Elements

To provide default identifiers, we tag all OMDoc elements that allow \texttt{xml:id} attributes by executing the \texttt{numberIt} procedure below. Furthermore, we use the \texttt{locateIt} procedure to give source links.

```latex
\begin{verbatim}
Tag('omdoc:omtext',afterOpen=>\&numberIt,afterClose=>\&locateIt);
Tag('omdoc:omgroup',afterOpen=>\&numberIt,afterClose=>\&locateIt);
Tag('omdoc:CMP',afterOpen=>\&numberIt,afterClose=>\&locateIt);
Tag('omdoc:idx',afterOpen=>\&numberIt,afterClose=>\&locateIt);
Tag('omdoc:ide',afterOpen=>\&numberIt,afterClose=>\&locateIt);
Tag('omdoc:idt',afterOpen=>\&numberIt,afterClose=>\&locateIt);
Tag('omdoc:note',afterOpen=>\&numberIt,afterClose=>\&locateIt);
Tag('omdoc:metadata',afterOpen=>\&numberIt,afterClose=>\&locateIt);
Tag('omdoc:meta',afterOpen=>\&numberIt,afterClose=>\&locateIt);
Tag('omdoc:resource',afterOpen=>\&numberIt,afterClose=>\&locateIt);
Tag('ltx:p',afterOpen=>\&numberIt,afterClose=>\&locateIt);
\end{verbatim}
```

We also have to number some \LaTeX{} tags, so that we do not get into trouble with the OMDoc tags inside them.

```latex
\begin{verbatim}
Tag('ltx:tabular',afterOpen=>\&numberIt,afterClose=>\&locateIt);
Tag('ltx:thead',afterOpen=>\&numberIt,afterClose=>\&locateIt);
Tag('ltx:td',afterOpen=>\&numberIt,afterClose=>\&locateIt);
Tag('ltx:tr',afterOpen=>\&numberIt,afterClose=>\&locateIt);
Tag('ltx:caption',afterOpen=>\&numberIt,afterClose=>\&locateIt);
\end{verbatim}
```

The \texttt{numberIt} procedure gets the prefix from first parent with an \texttt{xml:id} attribute and then extends it with a label that reflects the number of preceding siblings, provided that there is not already an identifier. Additionally, it estimates an XPointer position in the original document of the command sequence which produced the

⁶\texttt{EdNote}: MK: we should probably let \LaTeX{} deal with these and allow more text in the \texttt{omdoc+ltml.xsl}
tag. The `locateIt` subroutine is a sibling of `numberIt` as it is required as an `afterClose` handle for tags produced by LaTeX environments, as opposed to commands. `locateIt` estimates an XPointer end position of the LaTeX environment, allowing to meaningfully locate the entire environment at the source.

```perl
324 ⟨ltxml⟩
325 sub numberIt { 326 my($document,$node,$whatsit)=@_; 327 my($parents)=$document->findnodes('ancestor::*[@xml:id]',$node); 328 my $prefix= ($parents ? $parents[$#parents]->getAttribute('xml:id')."." : ''); 329 my($siblings)=$document->findnodes('preceding-sibling::*[@xml:id]',$node); 330 my $n = scalar($siblings)+1; 331 my $id = ($node =>$node->getAttribute('xml:id')); 332 my $localname = $node->localname; 333 $node->setAttribute('xml:id'=>$prefix."$localname$n") unless $id; 334 my $about = $node =>$node->getAttribute('about'); 335 $node->setAttribute('about'=>'#'.$node->getAttribute('xml:id')) unless $about; 336 # Also, provide locators: 337 my $locator = $whatsit->getProperty('locator'); 338 # Need to inherit locators if missing: 339 $locator = ($parents ? $parents[$#parents]->getAttribute('stex:srcref') : '') unless $locator; 340 if ($locator) { 341 # There is a BUG with namespace declarations (or am I using the API wrongly??) which 342 # does not recognize the stex namespace. Hence, I need to redefine it... 343 my $parent=$document->getNode; 344 if(! defined $parent->lookupNamespacePrefix("http://kwarc.info/ns/sTeX")) 345 { # namespace not already declared? 346 $document->getDocument->documentElement->setNamespace("http://kwarc.info/ns/sTeX","stex",0); 347 } 348 $node->setAttribute('stex:srcref'=>$locator); 349 }return;}
350
351 sub locateIt { 352 my($document,$node,$whatsit)=@_; 353 # Estimate trailer locator:
354 my $trailer = $whatsit->getProperty('trailer'); 355 return unless $trailer; # Nothing we can do if the trailer isn't defined 356 $trailer = $trailer->getLocator; 357 return unless ($trailer && $trailer!~/^\s*$/); # Useless if broken 358 my $locator = $node->getAttribute('stex:srcref'); 359 if ($locator) { 360 $locator =~ /^(.+from=\d+;\d+)/; 361 my $from = $1; 362 $trailer =~ /(,to=\d+;\d+.+)$/; 363 my $to = $1; 364 $locator = $from.$to; 365 } else { 366 $locator = $trailer; # This should never happen 367 } 368 my $parent = $document->getNode;
```
if(! defined $parent->lookupNamespacePrefix("http://kwarc.info/ns/sTeX"))
{
    # namespace not already declared?
    $document->getDocument->documentElement->setNamespace("http://kwarc.info/ns/sTeX","stex",0);
}

$node->setAttribute('stex:srcref' => $locator);
return;

⟨/ltxml⟩$
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.

Abelian group, 5

Abelian group, 5
References


[Ste] *Semantic Markup for \TeX*. Project Homepage. URL: http://trac.kwarc.info/sTeX/ (visited on 02/22/2011).