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

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

October 11, 2015

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.

Contents

2

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.

3

erated content markup.

\nlex   We use the `\nlex{`⟨*phrase*⟩`}` for marking up phrases that serve as natural
\nlcex   language examples and `\nlcex{`⟨*phrase*⟩`}` for counter-examples (utterances that
are not acceptable for some reason). In natural language examples, we sometimes
use "co-rereference markers" to specify the resolution of anaphora and the like.
\coreft   We use the `\coreft{`⟨*phrase*⟩`}{`⟨*mark*⟩`}` to mark up the "target" of a co-reference
\corefs   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   by specializing the macros `\@sinlinequote` — for quotations without source and
\@@sinlinequote   `\@@sinlinequote` — for quotations with source.

2.4   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
\begin@sblockquote   macros `\begin@sblockquote` to `\end@@sblockquote` are used for styling and can
\end@@sblockquote   be adapted by package integrators. Here a quote of Hamlet would marked up as

```
\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)

4

\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

\@@lec   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 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

noindex   the `noindex` option is set in the `\usepackage`. The `omtext` package and class make the usual `\index` macro undefined[3].

\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   `\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.

    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

\twintoo   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:

```
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

    The third set of macros does the same for two-word compounds with adjectives,

\atwintoo   e.g. "wonderful OMDoc document". `\atwin{wonderful}{OMdoc}{document}` will make the necessary index entries under "wonderful" and "document". Again,

\atwinalt   we have a variant `\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.

---

[3]EDNOTE: implement this and issue the respective error message

2.6   Support for **MathHub**

Much of the SₜₑX content is hosed on MathHub (`http://MathHub.info`), a portal and archive for flexiformal mathematics. MathHub offers GIT repositories (public and private escrow) for mathematical documentation projects, online and offline authoring and document development infrastructure, and a rich, interactive reading interface. The `modules` package supports repository-sensitive operations on MathHub.

Note that MathHub has two-level repository names of the form ⟨*group*⟩/⟨*repo*⟩, where ⟨*group*⟩ is a MathHub-unique repository group and ⟨*repo*⟩ a repository name that is ⟨*group*⟩-unique. The file and directory structure of a repository is arbitrary – except that it starts with the directory `source` because they are Math Archives in the sense of [Hor+11]. But this structure can be hidden from the SₜₑX author with MathHub-enabled versions of the `modules` macros.

`\mhcgraphics`   The `\mhcgraphics` macro is a variant of `\mycgraphics` with repository support. Instead of writing

```
\defpath{MathHub}{/user/foo/lmh/MathHub}
\mycgraphics{\MathHub{fooMH/bar/source/baz/foobar}}
```

we can simply write (assuming that `\MathHub` is defined as above)

```
\mhcgraphics[fooMH/bar]{baz/foobar}
```

Note that the `\mhcgraphics` form is more semantic, which allows more advanced document management features in MathHub.

If `baz/foobar` is the "current module", i.e. if we are on the MathHub path ...`MathHub/fooMH/bar`..., then stating the repository in the first optional argument is redundant, so we can just use

```
\mhcgraphics{baz/foobar}
```

Of course, neither LATₑX nor LATₑXMLknow about the repositories when they are called from a file system, so we can use the `\mhcurrentrepos` macro from the `modules` package to tell them. But this is only needed to initialize the infrastructure in the driver file. In particular, we do not need to set it in in each module, since the `\importmhmodule` macro sets the current repository automatically.

Caveat   if you want to use the MathHub support macros (let's call them mh-variants), then every time a module is imported or a document fragment is included from another repos, the mh-variant `\importmhmodule` must be used, so that the "current repository" is set accordingly. To be exact, we only need to use mh-variants, if the imported module or included document fragment use mh-variants.

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 TRAC [**sTeX:online**].

1. none reported yet

4 Implementation

The omtext package generates two files: the LATEX package (all the code between ⟨*package⟩ and ⟨/package⟩) and the LATEXML bindings (between ⟨*ltxml⟩ and ⟨/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 LATEXML:

```
1 ⟨*ltxml⟩
2 package LaTeXML::Package::Pool;
3 use strict;
4 use LaTeXML::Package;
5 use LaTeXML::Util::Pathname;
6 ⟨/ltxml⟩
```

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).[4]

```
7 ⟨*package⟩
8 \DeclareOption{showmeta}{\PassOptionsToPackage{\CurrentOption}{metakeys}}
9 \newif\ifindex\indextrue
10 \DeclareOption{noindex}{\indexfalse}
11 \ProcessOptions
12 \ifindex\makeindex\fi
13 ⟨/package⟩
14 ⟨*ltxml⟩
15 DeclareOption('showmeta',sub {PassOptions('metakeys','sty',ToString(Digest(T_CS('\CurrentOption
16 DeclareOption('noindex','');
17 ProcessOptions();
18 ⟨/ltxml⟩
```

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

```
19 ⟨*package⟩
20 \RequirePackage{sref}
21 \RequirePackage{xspace}
22 \RequirePackage{modules}
23 \RequirePackage{comment}
24 \RequirePackage{mdframed}
25 \RequirePackage{amssymb}
26 ⟨/package⟩
27 ⟨*ltxml⟩
28 RequirePackage('sref');
29 RequirePackage('xspace');
30 RequirePackage('modules');
```

---

[4]EDNOTE: need an implementation for LATEXML

8

```
31 RequirePackage('lxRDFa');
32 RequirePackage('amssymb');
```
33 ⟨/ltxml⟩

4.2 Metadata

All the OMDoc elements allow to specify metadata in them, which is modeled by the `omdoc:metadata` 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:metadata` element, even if they are supplied by different STEX bindings. Also we add numbering and location facilities.

34 ⟨∗ltxml⟩
```
35 Tag('omdoc:metadata',afterOpen=>\&numberIt,afterClose=>\&locateIt,autoClose=>1,autoOpen=>1);
```
36 ⟨/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`.

37 ⟨∗ltxml⟩
```
38 Tag('omdoc:CMP', afterOpen => sub {AssignValue('_LastSeenCMP', $_[1], 'global');return;});#$
```
39 ⟨/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.

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

4.3 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` infrastruc-

ture [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).

```
53 ⟨*package⟩
54 \srefaddidkey{omtext}
55 \addmetakey[]{omtext}{functions}
56 \addmetakey*{omtext}{display}
57 \addmetakey{omtext}{for}
58 \addmetakey{omtext}{from}
59 \addmetakey{omtext}{type}
60 \addmetakey*{omtext}{title}
61 \addmetakey*{omtext}{start}
62 \addmetakey{omtext}{theory}
63 \addmetakey{omtext}{continues}
64 \addmetakey{omtext}{verbalizes}
65 \addmetakey{omtext}{subject}
66 ⟨/package⟩
67 ⟨*ltxml⟩
68 DefKeyVal('omtext','functions','CommaList');
69 DefKeyVal('omtext','display','Semiverbatim');
70 DefKeyVal('omtext','for','Semiverbatim');
71 DefKeyVal('omtext','from','Semiverbatim');
72 DefKeyVal('omtext','type','Semiverbatim');
73 DefKeyVal('omtext','title','Plain'); #Math mode in titles.
74 DefKeyVal('omtext','start','Plain'); #Math mode in start phrases
75 DefKeyVal('omtext','theory','Semiverbatim');
76 DefKeyVal('omtext','continues','Semiverbatim');
77 DefKeyVal('omtext','verbalizes','Semiverbatim');
78 ⟨/ltxml⟩
```

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

```
79 % \ednote{MK: need to implement these in LaTeXML, I wonder whether there is a general
80 % mechanism like numberit.}\ednote{MK: this needs to be rethought in the light of
81 % |\usemodule|. It is probably obsolete. Is this used? Is this documented?}
82 ⟨*package⟩
83 \define@key{omtext}{require}{\requiremodules{#1}{sms}}
84 \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`

```
85 \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.

```
86 \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.

```
87 \def\omtext@pre@skip{\smallskip}
88 \def\omtext@post@skip{}
89 \providecommand{\stDMemph}[1]{\textbf{#1}}
90 \newenvironment{omtext}[1][]{\@in@omtexttrue%
91 \bgroup\metasetkeys{omtext}{#1}\sref@label@id{this paragraph}%
92 \def\lec##1{\@lec{##1}}%
93 \ifx\omtext@display\st@flow\else\omtext@pre@skip\par\noindent%
94 \ifx\omtext@title\@empty%
95 \ifx\omtext@start\@empty\else\stDMemph{\omtext@start}\xspace\fi%
96 \else\stDMemph{\omtext@title}:\xspace%
97 \ifx\omtext@start\@empty\else\omtext@start\xspace\fi%
98 \fi% omtext@title empty
99 \fi% omtext@display=flow
100 \ignorespaces}
101 {\egroup\omtext@post@skip\@in@omtextfalse}
102 ⟨/package⟩
103 ⟨∗ltxml⟩
104 DefEnvironment('{omtext} OptionalKeyVals:omtext',
105   "<omdoc:omtext "
106     . "?&GetKeyVal(#1,'id')(xml:id='&GetKeyVal(#1,'id')')() "
107     . "?&GetKeyVal(#1,'type')(type='&GetKeyVal(#1,'type')')() "
108     . "?&GetKeyVal(#1,'for')(for='&GetKeyVal(#1,'for')')() "
109         . "?&GetKeyVal(#1,'from')(from='&GetKeyVal(#1,'from')')()>"
110   . "?&GetKeyVal(#1,'title')(<dc:title>&GetKeyVal(#1,'title')</dc:title>)()"
111   .     "?&GetKeyVal(#1,'start')(<ltx:text class='startemph'>&GetKeyVal(#1,'start')</ltx:text>)
112                 .     "#body"
113                 ."</omdoc:omtext>");
114 ⟨/ltxml⟩
```

4.4 Phrase-level Markup

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

```
115 ⟨∗package⟩
116 \srefaddidkey{phrase}
117 \addmetakey{phrase}{style}
118 \addmetakey{phrase}{class}
119 \addmetakey{phrase}{index}
120 \addmetakey{phrase}{verbalizes}
121 \addmetakey{phrase}{type}
122 \addmetakey{phrase}{only}
123 \newcommand\phrase[2][]{\metasetkeys{phrase}{#1}%
124 \ifx\prhase@only\@empty\only<\phrase@only>{#2}\else #2\fi}
125 ⟨/package⟩
126 ⟨∗ltxml⟩
127 DefKeyVal('phrase','id','Semiverbatim');
128 DefKeyVal('phrase','style','Semiverbatim');
129 DefKeyVal('phrase','class','Semiverbatim');
130 DefKeyVal('phrase','index','Semiverbatim');
131 DefKeyVal('phrase','verbalizes','Semiverbatim');
```

```
132 DefKeyVal('phrase','type','Semiverbatim');
133 DefKeyVal('phrase','only','Semiverbatim');
134 DefConstructor('\phrase OptionalKeyVals:phrase {}',
135        "<ltx:text %&GetKeyVals(#1) ?&GetKeyVal(#1,'only')(rel='beamer:only' content='&GetKeyVal
136 ⟨/ltxml⟩
```

\coref*

```
137 ⟨*package⟩
138 \providecommand\textsubscript[1]{\ensuremath{_{#1}}}
139 \newcommand\corefs[2]{#1\textsubscript{#2}}
140 \newcommand\coreft[2]{#1\textsuperscript{#2}}
141 ⟨/package⟩
142 ⟨*ltxml⟩
143 DefConstructor('\corefs{}',
144   "<ltx:text class='coref-source' stex:index='#2'>#1</ltx:text>");
145 DefConstructor('\coreft{}',
146   "<ltx:text class='coref-target' stex:index='#2'>#1</ltx:text>");
147 ⟨/ltxml⟩
```

\n*lex

```
148 ⟨*package⟩
149 \newcommand\nlex[1]{\green{\sl{#1}}}
150 \newcommand\nlcex[1]{*\green{\sl{#1}}}
151 ⟨/package⟩
152 ⟨*ltxml⟩
153 DefConstructor('\nlex{}',"<ltx:text class='nlex'>#1</ltx:text>");
154 DefConstructor('\nlcex{}',"<ltx:text class='nlcex'>#1</ltx:text>");
155 ⟨/ltxml⟩
```

sinlinequote

```
156 ⟨*package⟩
157 \def\@sinlinequote#1{``{\sl{#1}}''}
158 \def\@@sinlinequote#1#2{\@sinlinequote{#2}~#1}
159 \newcommand\sinlinequote[2][]
160 {\def\@opt{#1}\ifx\@opt\@empty\@sinlinequote{#2}\else\@@sinlinequote\@opt{#2}\fi}
161 ⟨/package⟩
162 ⟨*ltxml⟩
163 DefConstructor('\sinlinequote [] {}',
164               "<ltx:quote type='inlinequote'>"
165               . "?#1(<dc:source>#1</dc:source>\n)()"
166               . "#2"
167             . "</ltx:quote>");
168 ⟨/ltxml⟩
```

4.5   Block-Level Markup

sblockquote

```
169 ⟨*package⟩
```

12

```
170 \def\begin@sblockquote{\begin{quote}\sl}
171 \def\end@sblockquote{\end{quote}}
172 \def\begin@@sblockquote#1{\begin@sblockquote}
173 \def\end@@sblockquote#1{\def\@@lec##1{{\rm ##1}}\@lec{#1}\end@sblockquote}
174 \newenvironment{sblockquote}[1][]
175   {\def\@opt{#1}\ifx\@opt\@empty\begin@sblockquote\else\begin@@sblockquote\@opt\fi}
176   {\ifx\@opt\@empty\end@sblockquote\else\end@@sblockquote\@opt\fi}
177 ⟨/package⟩
178 ⟨∗ltxml⟩
179 DefEnvironment('{sblockquote} []',
180   "<ltx:quote>?#1(<ltx:note role='source'>#1</ltx:note>)()#body</ltx:quote>");
181 ⟨/ltxml⟩
```

sboxquote

```
182 ⟨∗package⟩
183 \newenvironment{sboxquote}[1][]
184 {\def\@@src{#1}\begin{mdframed}[leftmargin=.5cm,rightmargin=.5cm]}
185 {\@lec{\rm\@@src}\end{mdframed}}
186 ⟨/package⟩
187 ⟨∗ltxml⟩
188 DefEnvironment('{sboxquote} []',
189   "<ltx:quote class='boxed'>?#1(<ltx:note role='source'>#1</ltx:note>)()#body</ltx:quote>");
190 ⟨/ltxml⟩
```

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.

```
191 ⟨∗package⟩
192 \providecommand{\@@lec}[1]{(#1)}
193 \def\@lec#1{\strut\hfil\strut\null\nobreak\hfill\@@lec{#1}}
194 \def\lec#1{\@lec{#1}\par}
195 ⟨/package⟩
196 ⟨∗ltxml⟩
197 DefConstructor('\lec{}',
198     "\n<omdoc:note type='line-end-comment'>#1</omdoc:note>");
199 ⟨/ltxml⟩
```

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

```
200 ⟨ltxml⟩RawTeX('
201 ⟨∗ltxml | package⟩
202 \newcommand\mygraphics[2][]{\includegraphics[#1]{#2}}
203 \newcommand\mycgraphics[2][]{\begin{center}\mygraphics[#1]{#2}\end{center}}
204 \newcommand\mybgraphics[2][]{\fbox{\mygraphics[#1]{#2}}}
205 \newcommand\mycbgraphics[2][]{\begin{center}\fbox{\mygraphics[#1]{#2}}\end{center}}
```

13

206 ⟨/ltxml | package⟩
207 ⟨ltxml⟩');

4.6 Index Markup

`\omdoc@index` this is the main internal indexing command. It makes sure that the modules neces-
sary 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 currentlyimported
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.

```
208 ⟨∗package⟩
209 \addmetakey{omdoc@index}{at}
210 \addmetakey[false]{omdoc@index}{loadmodules}[true]
211 \newcommand\omdoc@index[2][]{\ifindex%
212 \metasetkeys{omdoc@index}{#1}%
213 \@bsphack\begingroup\@sanitize%
214 \ifx\omdoc@index@loadmodules\@true%
215 \protected@write\@indexfile{}{\string\indexentry%
216 {\ifx\omdoc@index@at\@empty\else\omdoc@index@at @\fi%
217 {\string\importmodules{\@ifundefined{mod@id}\imported@modules\mod@id}%
218 #2}}{\thepage}}%
219 \else%
220 \protected@write\@indexfile{}{\string\indexentry%
221 {\ifx\omdoc@index@at\@empty\else\omdoc@index@at @\fi#2}{\thepage}}%
222 \fi% loadmodules
223 \endgroup\@esphack\fi}%ifindex
```

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

`\indexalt`

```
224 \newcommand\indexalt[3][]{{#2}\omdoc@index[#1]{#3}}            % word in text and index
225 ⟨/package⟩
226 ⟨∗ltxml⟩
227 DefConstructor('\indexalt[]{}{}',
228          "<omdoc:idx>"
229       .  "<omdoc:idt>#2</omdoc:idt>"
230       .  "<omdoc:ide ?#1(sort-by='#1')()>"
231         .   "<omdoc:idp>#3</omdoc:idp>"
232       .  "</omdoc:ide>"
233       ."</omdoc:idx>");
234 ⟨/ltxml⟩
```

`\indextoo`

```
235 ⟨∗package⟩
236 \newcommand\indextoo[2][]{{#2}\omdoc@index[#1]{#2}}            % word in text and index
237 ⟨/package⟩
238 ⟨∗ltxml⟩
239 DefConstructor('\indextoo[]{}',
```

14

```
240        "<omdoc:idx>"
241      . "<omdoc:idt>#2</omdoc:idt>"
242      . "<omdoc:ide ?#1(sort-by='#1')()>"
243        . "<omdoc:idp>#2</omdoc:idp>"
244      . "</omdoc:ide>"
245      ."</omdoc:idx>");
246 ⟨/ltxml⟩
```

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

```
247 ⟨∗package⟩
248 \newcommand\@twin[3][]{\omdoc@index[#1]{#2!#3}\omdoc@index[#1]{#3!#2}}
```

And again we have two interface macros building on this

\twinalt

```
249 \newcommand\twinalt[4][]{#2\@twin[#1]{#3}{#4}}
250 ⟨/package⟩
251 ⟨∗ltxml⟩
252 DefConstructor('\twinalt[]{}{}{}',
253        "<omdoc:idx>"
254      . "<omdoc:idt>#2</omdoc:idt>"
255      . "<omdoc:ide ?#1(sort-by='#1')()>"
256        . "<omdoc:idp>#2</omdoc:idp>"
257        . "<omdoc:idp>#3</omdoc:idp>"
258      . "</omdoc:ide>"
259      ."</omdoc:idx>");
260 ⟨/ltxml⟩
```

\twinalt

```
261 ⟨∗package⟩
262 \newcommand\twintoo[3][]{{#2 #3}\@twin[#1]{#2}{#3}} % and use the word compound too
263 ⟨/package⟩
264 ⟨∗ltxml⟩
265 DefConstructor('\twintoo[]{}{}',
266        "<omdoc:idx>"
267      . "<omdoc:idt>#2 #3</omdoc:idt>"
268      . "<omdoc:ide ?#1(sort-by='#1')()>"
269        . "<omdoc:idp>#2</omdoc:idp>"
270        . "<omdoc:idp>#3</omdoc:idp>"
271      . "</omdoc:ide>"
272      ."</omdoc:idx>");
273 ⟨/ltxml⟩
```

EdN:5   **\@atwin** this puts adjectivized two-compound words into the index in various permutations[5]

```
274 ⟨∗package⟩
275 \newcommand\@atwin[4][]{\omdoc@index[#1]{#2!#3!#4}\omdoc@index[#1]{#3!#2 (#4)}}
```

and the two interface macros for this case:

---

[5]EDNOTE: what to do with the optional argument here and below?

\@atwinalt

```
276 \newcommand\atwinalt[5][]{#2\@atwin[#1]{#3}{#4}{#4}}
277 ⟨/package⟩
278 ⟨∗ltxml⟩
279 DefConstructor('\atwinalt[]{}{}{}{}',
280        "<omdoc:idx>"
281      .  "<omdoc:idt>#2</omdoc:idt>"
282      .  "<omdoc:ide ?#1(sort-by='#1')()>"
283      .    "<omdoc:idp>#2</omdoc:idp>"
284      .    "<omdoc:idp>#3</omdoc:idp>"
285      .    "<omdoc:idp>#4</omdoc:idp>"
286      .  "</omdoc:ide>"
287      ."</omdoc:idx>");
288 ⟨/ltxml⟩
```

\atwintoo

```
289 ⟨∗package⟩
290 \newcommand\atwintoo[4][]{{#2 #3 #4}\@atwin[#1]{#2}{#3}{#4}}          % and use it too
291 ⟨/package⟩
292 ⟨∗ltxml⟩
293 DefConstructor('\atwintoo[]{}{}{}',
294        "<omdoc:idx>"
295      .  "<omdoc:idt>#2 #3</omdoc:idt>"
296      .  "<omdoc:ide ?#1(sort-by='#1')()>"
297      .    "<omdoc:idp>#2</omdoc:idp>"
298      .    "<omdoc:idp>#3</omdoc:idp>"
299      .    "<omdoc:idp>#4</omdoc:idp>"
300      .  "</omdoc:ide>"
301      ."</omdoc:idx>");
302 ⟨/ltxml⟩
```

4.7   LaTeX Commands we interpret differently

The first think we have to take care of are the paragraphs, we want to generate
OMDoc that uses the `ltx:p` element for paragraphs inside CMPs. For that we have
modified the DTD only to allowed `ltx:p` elements in `omdoc:CMP` (in particular
no text). Then we instruct the `\par` macro to close a `ltx:p` element if possible.
The next `ltx:p` element is then opened automatically, since we make `ltx:p` and
`omdoc:CMP` autoclose and autoopen.

```
303 ⟨∗ltxml⟩
304 Tag('omdoc:CMP', autoClose=>1, autoOpen=>1);
305 Tag('omdoc:omtext', autoClose=>1, autoOpen=>1);
306 Tag('ltx:p', autoClose=>1, autoOpen=>1);
307 ⟨/ltxml⟩
```

the rest of the reinterpretations is quite simple, we either disregard presentational
markup or we re-interpret it in terms of OMDoc.[6]

EdN:6

---

[6]EDNOTE: MK: we should probably let LaTeXML deal with these and allow more text in the
omdoc+ltxml.xsl

```
308 ⟨∗package⟩
309 \def\omspace#1{\hspace*{#1}}
310 ⟨/package⟩
311 ⟨∗ltxml⟩
312 DefConstructor('\footnote[]{}',
313        "<omdoc:note type='foot' ?#1(mark='#1')>#2</omdoc:note>");
314 DefConstructor('\footnotemark[]',"");
315 DefConstructor('\footnotetext[]{}',
316        "<omdoc:note class='foot' ?#1(mark='#1')>#2</omdoc:note>");
317 ⟨/ltxml⟩
```

4.8   Providing IDs for OMDoc Elements

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

```
318 ⟨∗ltxml⟩
319 Tag('omdoc:omtext',afterOpen=>\&numberIt,afterClose=>\&locateIt);
320 Tag('omdoc:omgroup',afterOpen=>\&numberIt,afterClose=>\&locateIt);
321 Tag('omdoc:CMP',afterOpen=>\&numberIt,afterClose=>\&locateIt);
322 Tag('omdoc:idx',afterOpen=>\&numberIt,afterClose=>\&locateIt);
323 Tag('omdoc:ide',afterOpen=>\&numberIt,afterClose=>\&locateIt);
324 Tag('omdoc:idt',afterOpen=>\&numberIt,afterClose=>\&locateIt);
325 Tag('omdoc:note',afterOpen=>\&numberIt,afterClose=>\&locateIt);
326 Tag('omdoc:metadata',afterOpen=>\&numberIt,afterClose=>\&locateIt);
327 Tag('omdoc:meta',afterOpen=>\&numberIt,afterClose=>\&locateIt);
328 Tag('omdoc:resource',afterOpen=>\&numberIt,afterClose=>\&locateIt);
329 Tag('omdoc:recurse',afterOpen=>\&numberIt,afterClose=>\&locateIt);
330 Tag('omdoc:imports',afterOpen=>\&numberIt,afterClose=>\&locateIt);
331 Tag('omdoc:theory',afterOpen=>\&numberIt,afterClose=>\&locateIt);
332 Tag('omdoc:ignore',afterOpen=>\&numberIt,afterClose=>\&locateIt);
333 Tag('omdoc:ref',afterOpen=>\&numberIt,afterClose=>\&locateIt);
334 ⟨/ltxml⟩
```

We also have to number some LaTeXML tags, so that we do not get into trouble with the OMDoctags inside them.

```
335 ⟨∗ltxml⟩
336 Tag('ltx:p',afterOpen=>\&numberIt,afterClose=>\&locateIt);
337 Tag('ltx:tabular',afterOpen=>\&numberIt,afterClose=>\&locateIt);
338 Tag('ltx:thead',afterOpen=>\&numberIt,afterClose=>\&locateIt);
339 Tag('ltx:td',afterOpen=>\&numberIt,afterClose=>\&locateIt);
340 Tag('ltx:tr',afterOpen=>\&numberIt,afterClose=>\&locateIt);
341 Tag('ltx:caption',afterOpen=>\&numberIt,afterClose=>\&locateIt);
342 Tag('ltx:Math',afterOpen=>\&numberIt,afterClose=>\&locateIt);
343 ⟨/ltxml⟩
```

The `numberIt` procedure gets the prefix from first parent with an `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

17

position in the original document of the command sequence which produced the 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.

```
344 ⟨∗ltxml⟩
345 sub numberIt {
346   my($document,$node,$whatsit)=@_;
347   my(@parents)=$document->findnodes('ancestor::*[@xml:id]',$node);
348   my $prefix= (@parents ? $parents[$#parents]->getAttribute('xml:id')."." : '');
349   my(@siblings)=$document->findnodes('preceding-sibling::*[@xml:id]',$node);
350   my $n = scalar(@siblings)+1;
351   my $id = ($node -> getAttribute('xml:id'));
352   my $localname = $node->localname;
353   $node->setAttribute('xml:id'=>$prefix."$localname$n") unless $id;
354   my $about = $node -> getAttribute('about');
355   $node->setAttribute('about'=>'#'.$node->getAttribute('xml:id')) unless $about;
356   #Also, provide locators:
357   my $locator = $whatsit && $whatsit->getProperty('locator');
358   #Need to inherit locators if missing:
359   $locator = (@parents ? $parents[$#parents]->getAttribute('stex:srcref') : '') unless $locator
360   if ($locator) {
361     # There is a BUG with namespace declarations (or am I using the API wrongly??) which
362     # does not recognize the stex namespace. Hence, I need to redeclare it...
363     my $parent=$document->getNode;
364     if(! defined $parent->lookupNamespacePrefix("http://kwarc.info/ns/sTeX"))
365       { # namespace not already declared?
366         $document->getDocument->documentElement->setNamespace("http://kwarc.info/ns/sTeX","stex
367       }
368     $node->setAttribute('stex:srcref'=>$locator);
369   }return;}
370
371 sub locateIt {
372   my($document,$node,$whatsit)=@_;
373   #Estimate trailer and locator:
374   my $locator = $node->getAttribute('stex:srcref');
375   return unless $locator; # Nothing to do here...
376   my $trailer = $whatsit && $whatsit->getProperty('trailer');
377   $trailer = $trailer->getLocator if $trailer;
378   $trailer = $locator unless $trailer; # bootstrap
379   # TODO: Both should be local, or both remote, any mixture or undefinedness will produce garba
380   my $file_path = LookupValue('SOURCEFILE');
381   my $baselocal = LookupValue('BASELOCAL');
382   # Hmm, we only care about relative paths, so let's just do a URL->pathname map
383   $file_path=~s/^\w+\:\/// if $file_path;
384   $baselocal=~s/^\w+\:\/// if $baselocal;
385   if ($file_path && $baselocal && ($locator =~ s/^([^\#]+)\#/\#/)) {
386     my $relative_path = pathname_relative($file_path,$baselocal);
387     $locator = $relative_path.$locator;
```

```
388    }
389    if ($locator =~ /^(.+from=\d+;\d+)/) {
390      my $from = $1;
391      if ($trailer =~ /(,to=\d+;\d+.+)$/) {
392        my $to = $1;
393        $locator = $from.$to;
394      } else { Error("stex","locator",undef, "Trailer is garbled, expect nonsense in stex:srcref
395    } else { Error("stex","locator",undef, "Locator \"$locator\" is garbled, expect nonsense in s
396    my $parent = $document->getNode;
397    if(! defined $parent->lookupNamespacePrefix("http://kwarc.info/ns/sTeX"))
398      { # namespace not already declared?
399        $document->getDocument->documentElement->setNamespace("http://kwarc.info/ns/sTeX","stex",
400      }
401    $node->setAttribute('stex:srcref' => $locator);
402    return;
403 }
404 ⟨/ltxml⟩#$
```

4.9   Miscellaneous

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

```
405 ⟨*package⟩
406 \newcommand\hateq{\ensuremath{\hat=}\xspace}
407 \newcommand\hatequiv{\ensuremath{\hat\equiv}\xspace}
408 \ifdef{\textleadsto}%
409 {\renewcommand\textleadsto{\ensuremath{\leadsto}\xspace}}%
410 {\newcommand\textleadsto{\ensuremath{\leadsto}\xspace}}%
411 \newcommand{\reflect@squig}[2]{\reflectbox{$\m@th#1\rightsquigarrow$}}%
412 \newcommand\texttoleads{\ensuremath{\mathrel{\mathpalette\reflect@squig\relax}}\xspace}%
413 ⟨/package⟩
414 ⟨*ltxml⟩
415 DefMacro('\hateq','\@hateq\xspace');
416 DefConstructor('\@hateq',"\x{2259}");
417 DefMacro('\hatequiv','\@hatequiv\xspace');
418 DefConstructor('\@hatequiv',"\x{2A6F}");
419 DefMacro('\textleadsto','\@textleadsto\xspace');
420 DefConstructor('\@textleadsto',"\x{219D}");
421 DefMacro('\texttoleads','\@texttoleads\xspace');
422 DefConstructor('\@texttoleads',"\x{2B3F}");
423 ⟨/ltxml⟩
```

4.10   Support for **MathHub**

\mh*graphics   Use the current value of \mh@currentrepos or the value of the mhrepos key if it
is given in \my*graphics.

```
424 ⟨*package⟩
425 \addmetakey{Gin}{mhrepos}
```

```
426 \newcommand\mhgraphics[2][]{\metasetkeys{Gin}{#1}%
427 \edef\mh@@repos{\mh@currentrepos}%
428 \ifx\Gin@mhrepos\@empty\mygraphics[#1]{\MathHub{\mh@currentrepos/source/#2}}%
429 \else\mygraphics[#1]{\MathHub{\Gin@mhrepos/source/#2}}\fi
430 \def\Gin@mhrepos{}\mhcurrentrepos\mh@@repos}
431 \newcommand\mhcgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
432 \newcommand\mhbgraphics[2][]{\fbox{\mhgraphics[#1]{#2}}}
433 \newcommand\mhcbgraphics[2][]{\begin{center}\fbox{\mhgraphics[#1]{#2}}\end{center}}
434 ⟨/package⟩
435 ⟨∗ltxml⟩
436 sub mhgraphics {
437   my ($gullet,$keyval,$arg2) = @_;
438   my $repo_path;
439   if ($keyval) {
440     $repo_path = ToString(GetKeyVal($keyval,'mhrepos')); }
441   if (! $repo_path) {
442     $repo_path = ToString(Digest(T_CS('\mh@currentrepos'))); }
443   else {
444     $keyval->setValue('mhrepos',undef); }
445   my $mathhub_base = ToString(Digest('\MathHub{}'));
446   my $finalpath = $mathhub_base.$repo_path.'/source/'.ToString($arg2);
447   return Invocation(T_CS('\@includegraphicx'), $keyval, T_OTHER($finalpath)); }#$
448 DefKeyVal('Gin','mhrepos','Semiverbatim');
449 DefMacro('\mhgraphics OptionalKeyVals:Gin {}', \&mhgraphics);
450 DefMacro('\mhcgraphics []{}','\begin{center}\mhgraphics[#1]{#2}\end{center}');
451 DefMacro('\mhbgraphics []{}','\fbox{\mhgraphics[#1]{#2}}');
452 ⟨/ltxml⟩
```

4.11   Finale

We need to terminate the file with a success mark for perl.

```
453 ⟨ltxml⟩1;
```

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

[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: http://kwarc.info/frabe/Research/HIJKR_dimensions_11.pdf.

[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.