

`omdoc.sty/cls`: Semantic Markup for Open Mathematical Documents in \LaTeX

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

November 29, 2015

Abstract

The `omdoc` package is part of the \LaTeX collection, a version of $\text{\TeX}/\text{\LaTeX}$ that allows to markup $\text{\TeX}/\text{\LaTeX}$ documents semantically without leaving the document format, essentially turning $\text{\TeX}/\text{\LaTeX}$ into a document format for mathematical knowledge management (MKM).

This package supplies an infrastructure for writing OMDoc documents in \LaTeX . This includes a simple structure sharing mechanism for \LaTeX that allows to move from a copy-and-paste document development model to a copy-and-reference model, which conserves space and simplifies document management. The augmented structure can be used by MKM systems for added-value services, either directly from the \LaTeX sources, or after translation.

Contents

1	Introduction	3
2	The User Interface	3
2.1	Package and Class Options	3
2.2	Document Structure	4
2.3	Ignoring Inputs	5
2.4	Structure Sharing	5
2.5	Colors	6
3	Limitations	6
4	Implementation: The OMDoc Class	7
4.1	Class Options	7
4.2	Setting up Namespaces and Schemata for LaTeXML	8
4.3	Beefing up the <code>document</code> environment	9
5	Implementation: OMDoc Package	10
5.1	Package Options	10
5.2	Document Structure	11
5.3	Front and Backmatter	14
5.4	Ignoring Inputs	15
5.5	Structure Sharing	16
5.6	Colors	17
5.7	L ^A T _E X Commands we interpret differently	18
5.8	Leftovers	18

1 Introduction

$\mathcal{S}\text{TEX}$ is a version of $\text{T}\text{E}\text{X}/\text{L}\text{A}\text{T}\text{E}\text{X}$ that allows to markup $\text{T}\text{E}\text{X}/\text{L}\text{A}\text{T}\text{E}\text{X}$ documents semantically without leaving the document format, essentially turning $\text{T}\text{E}\text{X}/\text{L}\text{A}\text{T}\text{E}\text{X}$ into a document format for mathematical knowledge management (MKM). The package supports direct translation to the OMDoc format [Koh06]

The `omdoc` package supplies macros and environment that allow to label document fragments and to reference them later in the same document or in other documents. In essence, this enhances the document-as-trees model to documents-as-directed-acyclic-graphs (DAG) model. This structure can be used by MKM systems for added-value services, either directly from the $\mathcal{S}\text{TEX}$ sources, or after translation. Currently, trans-document referencing provided by this package can only be used in the $\mathcal{S}\text{TEX}$ collection.

DAG models of documents allow to replace the “Copy and Paste” in the source document with a label-and-reference model where document are shared in the document source and the formatter does the copying during document formatting/presentation.¹

2 The User Interface

The `omdoc` package generates four files: `omdoc.cls`, `omdoc.sty` and their $\text{L}\text{A}\text{T}\text{E}\text{X}\text{ML}$ bindings `omdoc.cls.ltxml` and `omdoc.sty.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. The OMDoc class is a minimally changed variant of the standard `article` class that includes the functionality provided by `omdoc.sty`. Most importantly, `omdoc.cls` sets up the $\text{L}\text{A}\text{T}\text{E}\text{X}\text{ML}$ infrastructure and thus should be used if OMDoc is to be generated from the $\mathcal{S}\text{TEX}$ sources. The rest of the documentation pertains to the functionality introduced by `omdoc.sty`.

2.1 Package and Class Options

The `omdoc` package and class accept the following options:

<code>report</code>	load <code>report.cls</code> instead of <code>article.cls</code>
<code>book</code>	load <code>book.cls</code> instead of <code>article.cls</code>
<code>showignores</code>	show the the contents of the <code>ignore</code> environment after all
<code>showmeta</code>	show the metadata; see <code>metakeys.sty</code>
<code>showmods</code>	show modules; see <code>modules.sty</code>
<code>extrefs</code>	allow external references; see <code>sref.sty</code>
<code>defindex</code>	index definienda; see <code>statements.sty</code>

¹EDNOTE: integrate with `latexml`’s `XMRef` in the Math mode.

2.2 Document Structure

<code>document</code>	The top-level <code>document</code> environment can be given key/value information by the
<code>\documentkeys</code>	<code>\documentkeys</code> macro in the preamble ¹ . This can be used to give metadata about
<code>id</code>	the document. For the moment only the <code>id</code> key is used to give an identifier to the
	<code>omdoc</code> element resulting from the L ^A T _E X _{ML} transformation.
<code>omgroup</code>	The structure of the document is given by the <code>omgroup</code> environment just like
	in OMDoc. In the L ^A T _E X route, the <code>omgroup</code> environment is flexibly mapped
	to sectioning commands, inducing the proper sectioning level from the nesting
	of <code>omgroup</code> environments. Correspondingly, the <code>omgroup</code> environment takes an
	optional key/value argument for metadata followed by a regular argument for
	the (section) title of the <code>omgroup</code> . The optional metadata argument has the
<code>id</code>	keys <code>id</code> for an identifier, <code>creators</code> and <code>contributors</code> for the Dublin Core meta-
<code>creators</code>	data [DCM03]; see [Koh15a] for details of the format. The <code>short</code> allows to give
<code>contributors</code>	a short title for the generated section. If the title contains semantic macros, they
<code>short</code>	need to be protected by <code>\protect</code> , and we need to give the <code>loadmodules</code> key it
<code>loadmodules</code>	needs no value. For instance we would have

```

\begin{module}{foo}
\symdef{bar}{Ba_r}
...
\begin{omgroup}[id=barderviv,loadmodules]
{Introducing $\protect\bar$ Derivations}

```

`blindomgroup`
`\blindomgroup`
`\currentsectionlevel`
`\CurrentSectionLevel`

\TeX automatically computes the sectioning level, from the nesting of `omgroup` environments. But sometimes, we want to skip levels (e.g. to use a subsection* as an introduction for a chapter). Therefore the `omdoc` package provides a variant `blindomgroup` that does not produce markup, but increments the sectioning level and logically groups document parts that belong together, but where traditional document markup relies on convention rather than explicit markup. The `blindomgroup` environment is useful e.g. for creating frontmatter at the correct level. Example 1 shows a typical setup for the outer document structure of a book with parts and chapters. We use two levels of `blindomgroup`:

- The outer one groups the introductory parts of the book (which we assume to have a sectioning hierarchy topping at the part level). This `blindomgroup` makes sure that the introductory remarks become a “chapter” instead of a “part”.
- The inner one groups the frontmatter² and makes the preface of the book a section-level construct. Note that here the `display=flow` on the `omgroup` environment prevents numbering as is traditional for prefaces.

The `\currentsectionlevel` macro supplies the name of the current sectioning level, e.g. “chapter”, or “subsection”. `\CurrentSectionLevel` is the capitalized variant. They are useful to write something like “In this `\currentsectionlevel`,

¹We cannot patch the document environment to accept an optional argument, since other packages we load already do; pity.

²We shied away from redefining the `frontmatter` to induce a `blindomgroup`, but this may be the “right” way to go in the future.

```

\begin{document}
\begin{blindomgroup}
\begin{blindomgroup}
\begin{frontmatter}
\maketitle\newpage
\begin{omgroup}[display=flow]{Preface}
... <<preface>> ...
\end{omgroup}
\clearpage\setcounter{tocdepth}{4}\tableofcontents\clearpage
\end{frontmatter}
\end{blindomgroup}
... <<introductory remarks>> ...
\end{blindomgroup}
\begin{omgroup}{Introduction}
... <<intro>> ...
\end{omgroup}
... <<more chapters>> ...
\bibliographystyle{alpha}\bibliography{kwarc}
\end{document}

```

Example 1: A typical Document Structure of a Book

we will...” in an `omgroup` environment, where we do not know which sectioning level we will end up.

2.3 Ignoring Inputs

ignore The `ignore` environment can be used for hiding text parts from the document structure. The body of the environment is not PDF or DVI output unless the `showignores` option is given to the `omdoc` class or `package`. But in the generated OMDoc result, the body is marked up with a `ignore` element. This is useful in two situations. For

editing One may want to hide unfinished or obsolete parts of a document

narrative/content markup In \LaTeX we mark up narrative-structured documents. In the generated OMDoc documents we want to be able to cache content objects that are not directly visible. For instance in the `statements` package [Koh15c] we use the `\inlinedef` macro to mark up phrase-level definitions, which verbalize more formal definitions. The latter can be hidden by an `ignore` and referenced by the `verbalizes` key in `\inlinedef`.

2.4 Structure Sharing

`\STRlabel` The `\STRlabel` macro takes two arguments: a label and the content and stores the the content for later use by `\STRcopy[⟨URL⟩]{⟨label⟩}`, which expands to the previously stored content. If the `\STRlabel` macro was in a different file, then we can give a URL `⟨URL⟩` that lets \LaTeX ML generate the correct reference.

`\STRsemantics` The `\STRlabel` macro has a variant `\STRsemantics`, where the label argument is optional, and which takes a third argument, which is ignored in L^AT_EX. This allows to specify the meaning of the content (whatever that may mean) in cases, where the source document is not formatted for presentation, but is transformed into some content markup format.²

2.5 Colors

For convenience, the `omdoc` package defines a couple of color macros for the `color` package: For instance `\blue` abbreviates `\textcolor{blue}`, so that `\blue{\textit{something}}` writes *something* in blue. The macros `\red` `\green`, `\cyan`, `\magenta`, `\brown`, `\yellow`, `\orange`, `\gray`, and finally `\black` are analogous.

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 sT_EX GitHub repository [sTeX].

1. when option `book` which uses `\pagestyle{headings}` is given and semantic macros are given in the `omgroup` titles, then they sometimes are not defined by the time the heading is formatted. Need to look into how the headings are made.

²EdNOTE: document LMID und LMXRef here if we decide to keep them.

4 Implementation: The OMDoc Class

The functionality is spread over the `omdoc` class and package. The class provides the `document` environment and the `omdoc` element corresponds to it, whereas the package provides the concrete functionality.

`omdoc.dtx` generates four files: `omdoc.cls` (all the code between `<*cls>` and `</cls>`), `omdoc.sty` (between `<*package>` and `</package>`) and their \LaTeX ML bindings (between `<*ltxml.cls>` and `</ltxml.cls>` and `<*ltxml.sty>` and `</ltxml.sty>` respectively). We keep the corresponding code fragments together, since the documentation applies to both of them and to prevent them from getting out of sync.

The initial setup for \LaTeX ML (both package and class actually):

```
1 <ltxml.sty | ltxml.cls>
2 # -*- CPERL -*-
3 package LaTeXML::Package::Pool;
4 use strict;
5 use LaTeXML::Package;
6 use LaTeXML::Util::Pathname;
7 use Cwd qw(abs_path);
8 </ltxml.sty | ltxml.cls>
```

4.1 Class Options

To initialize the `omdoc` class, we declare and process the necessary options. For `omdoc.cls` this is quite simple. We have options `report` and `book`, which set the `\omdoc@class` macro and pass on the macro to `omdoc.sty` for further processing. The `book` option also sets the conditional to true for the frontmatter handling later.

`\omdoc@class`
`\ifclass@book`

```
9 <*cls>
10 \def\omdoc@class{article}
11 \DeclareOption{report}{\def\omdoc@class{report}%
12   \PassOptionsToPackage{\CurrentOption}{omdoc}
13   \PassOptionsToPackage{\CurrentOption}{stex}}
14 \newif\ifclass@book\class@bookfalse
15 \DeclareOption{book}{\def\omdoc@class{book}\class@booktrue%
16   \PassOptionsToPackage{\CurrentOption}{omdoc}
17   \PassOptionsToPackage{\CurrentOption}{stex}}
```

the rest of the options are only passed on to `omdoc.sty` and the class selected by the first options.

```
18 \DeclareOption*{\PassOptionsToClass{\CurrentOption}{\omdoc@class}
19   \PassOptionsToPackage{\CurrentOption}{omdoc}
20   \PassOptionsToPackage{\CurrentOption}{stex}}
21 \ProcessOptions
22 </cls>
23 <*ltxml.cls>
24 AssignValue('omdoc@class'=>'article');
25 DeclareOption('report', sub {AssignValue('omdoc@class' => 'report')};
```

```

26 PassOptions('stex','sty',ToString(Digest(T_CS('\CurrentOption'))));
27 PassOptions('omdoc','sty',ToString(Digest(T_CS('\CurrentOption'))));});
28 DeclareOption('book', sub {AssignValue('omdoc@class' => 'book');
29 PassOptions('stex','sty',ToString(Digest(T_CS('\CurrentOption'))));
30 PassOptions('omdoc','sty',ToString(Digest(T_CS('\CurrentOption'))));});
31 DeclareOption(undef, sub {PassOptions('article','cls',ToString(Digest(T_CS('\CurrentOption'))));
32 PassOptions('omdoc','sty',ToString(Digest(T_CS('\CurrentOption'))));
33 PassOptions('stex','sty',ToString(Digest(T_CS('\CurrentOption'))));});
34 ProcessOptions();
35 </ltxml.cls>

```

We load `article.cls`, and the desired packages. For the \LaTeX ML bindings, we make sure the right packages are loaded.

```

36 <*cls>
37 \LoadClass{\omdoc@class}
38 \RequirePackage{etoolbox}
39 \RequirePackage{omdoc}
40 \RequirePackage{stex}
41 </cls>
42 <*ltxml.cls>
43 my $class = LookupValue('omdoc@class');
44 LoadClass($class);
45 </ltxml.cls>

```

4.2 Setting up Namespaces and Schemata for \LaTeX ML

Now, we also need to register the namespace prefixes for \LaTeX ML to use.

```

46 <*ltxml.cls>
47 RegisterNamespace('omdoc'=>"http://omdoc.org/ns");
48 RegisterNamespace('om'=>"http://www.openmath.org/OpenMath");
49 RegisterNamespace('m'=>"http://www.w3.org/1998/Math/MathML");
50 RegisterNamespace('dc'=>"http://purl.org/dc/elements/1.1/");
51 RegisterNamespace('cc'=>"http://creativecommons.org/ns");
52 RegisterNamespace('stex'=>"http://kwarc.info/ns/sTeX");
53 RegisterNamespace('ltx'=>"http://dlmf.nist.gov/LaTeXML");
54 </ltxml.cls>

```

Since we are dealing with a class, we need to set up the document type in the \LaTeX ML bindings.

```

55 <*ltxml.cls>
56 RelaxNGSchema('omdoc+ltxml',
57     '#default'=>"http://omdoc.org/ns",
58     'om'=>"http://www.openmath.org/OpenMath",
59     'm'=>"http://www.w3.org/1998/Math/MathML",
60     'dc'=>"http://purl.org/dc/elements/1.1/",
61     'cc'=>"http://creativecommons.org/ns",
62     'stex'=>"http://kwarc.info/ns/sTeX",
63     'ltx'=>"http://dlmf.nist.gov/LaTeXML");
64 </ltxml.cls>

```


Then we load the `omdoc` package `omdoc.sty`, which contains the main body of functionality (e.g.sectioning/grouping). It can be loaded by classes other than `omdoc.cls` as well.

```
65 <*txml.cls>
66 \RequirePackage('omdoc');
67 \RequirePackage('stex');
68 </txml.cls>
```

4.3 Beefing up the document environment

Now, we will define the environments we need. The top-level one is the `document` environment, which we redefined so that we can provide keyval arguments.

EdN:3

`document` For the moment we do not use them on the \LaTeX level, but the document identifier is picked up by \LaTeX XML.³

```
69 <*cls>
70 \srefaddidkey{document}
71 \newcommand\documentkeys[1]{\metasetkeys{document}{#1}}
72 \let\orig@document=\document
73 \srefaddidkey{document}
74 \renewcommand{\document}[1][\metasetkeys{document}{#1}\orig@document}
75 </cls>
76 <*txml.cls>
77 sub xmlBase {
78   my $baseuri = LookupValue('URLBASE');
79   $baseuri =~ s/\$//g; # No trailing slashes
80   Tokenize($baseuri); }
81 DefConstructor('\documentkeys RequiredKeyVals:omdoc',
82               sub { activateKeys($_[1]); return; });
83 sub activateKeys {
84   my ($kv) = @_;
85   if ($kv) {
86     my @pairs = $kv->getPairs();
87     while(@pairs) {
88       my ($key, $val) = (shift(@pairs), shift(@pairs));
89       AssignValue('DOCUMENT'.$key => $val);
90     }
91   }
92 return; }
93 DefEnvironment('{document}',
94               "<omdoc:omdoc "
95               . "&LookupValue('DOCUMENTid')(xml:id='&LookupValue('DOCUMENTid')'"
96               . "(?&Tokenize(&LookupValue('SOURCEBASE')))"
97               . "(xml:id='&Tokenize(&LookupValue('SOURCEBASE')).omdoc')()) "
98               . "&Tokenize(&LookupValue('URLBASE'))"
99               . "(xml:base='&xmlBase()')()>"
100               . "#body"
```

³EDNOTE: faking documentkeys for now. @HANG, please implement

```

101     . "</omdoc:omdoc>",
102     beforeDigest=> sub { AssignValue(inPreamble=>0); },
103     afterDigest=> sub { $_[0]->getGullet->flush; return; },
104     afterDigestBegin => sub {
105         $_[1]->setProperty(id => LookupValue('DOCUMENTid'));
106         if (my $ops = LookupValue('@at@begin@document')) {
107             Digest(Tokens(@$ops)); }
108         else {
109             return; } },
110     beforeDigestEnd => sub {
111         $_[0]->getGullet->flush;
112         if (my $ops = LookupValue('@at@end@document')) {
113             Digest(Tokens(@$ops)); }
114         else {
115             return; } },
116     mode => 'text');
117 Tag('omdoc:omdoc', 'afterOpen:late'=>\&insertFrontMatter,
118     afterOpen=>\&numberIt,afterClose=>\&locateIt);
119 </ltxml.cls>%$

```

5 Implementation: OMDoc Package

5.1 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). The `report` and `book` options affect the sectioning behavior of the `omgroup` environment via the `\section@level` macro later.

`\section@level`

```

120 <*package>
121 \newif\ifshow@ignores\show@ignorefalse
122 \DeclareOption{showignores}{\show@ignorestrue}
123 \newcount\section@level\section@level=2
124 \DeclareOption{report}{\section@level=0}
125 \DeclareOption{book}{\section@level=0}
126 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{sref}}
127 \ProcessOptions
128 </package>
129 <*ltxml.sty>
130 \DeclareOption(undef, sub {PassOptions('sref','sty',ToString(Digest(T_CS('\CurrentOption')))); }
131 \ProcessOptions();
132 </ltxml.sty>

```

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

```

133 <*package>
134 \RequirePackage{sref}
135 \RequirePackage{xspace}

```

```

136 \RequirePackage{comment}
137 \RequirePackage{etoolbox}
138 \end{package}
139 \end{*ltxml.sty}
140 \RequirePackage('sref');
141 \RequirePackage('xspace');
142 \end{*ltxml.sty}

```

5.2 Document Structure

The structure of the document is given by the `omgroup` environment just like in OMDoc. The hierarchy is adjusted automatically according to the \LaTeX class in effect.

EdN:4 `\currentsectionlevel` For the `\currentsectionlevel` and `\Currentsectionlevel` macros we use an internal macro `\current@section@level` that only contains the keyword (no markup). We initialize it with “document” as a default. In the generated OMDoc, we only generate a text element of class `omdoc_currentsectionlevel`, which will be instantiated by CSS later.⁴

```

143 \end{*package}
144 \def\current@section@level{document}%
145 \newcommand\currentsectionlevel{\lowercase\expandafter\current@section@level\xspace}%
146 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
147 \end{package}
148 \end{*ltxml.sty}
149 \DefMacro('currentsectionlevel', '@currentsectionlevel\xspace');
150 \DefMacro('Currentsectionlevel', '@Currentsectionlevel\xspace');
151 \DefConstructor('@currentsectionlevel',
152               "<ltx:text class='omdoc-currentsectionlevel'>section</ltx:text>");
153 \DefConstructor('@Currentsectionlevel',
154               "<ltx:text class='omdoc-Currentsectionlevel'>Section</ltx:text>");
155 \end{*ltxml.sty}

```

`blindomgroup`

```

156 \end{*package}
157 \newcommand\at@begin@blindomgroup[1]{
158 \newenvironment{blindomgroup}
159 {\advance\section@level by 1\at@begin@blindomgroup\section@level}
160 {\advance\section@level by -1}
161 \end{package}
162 \end{*ltxml.sty}
163 \DefEnvironment('{blindomgroup} OptionalKeyVals:omgroup',
164               "<omdoc:omgroup layout='invisible'"
165               . "&GetKeyVal(#1,'id')(xml:id='&GetKeyVal(#1,'id')')()"
166               . "&GetKeyVal(#1,'type')(type='&GetKeyVal(#1,'type')')()>\n"
167               . "#body\n"

```

⁴EdNOTE: MK: we may have to experiment with the more powerful uppercasing macro from `mfirstuc.sty` once we internationalize.

```

168 . "</omdoc:omgroup>");
169 </ltxml.sty>

\omgroup@nonum convenience macro: \omgroup@nonum{<level>}{<title>} makes an unnumbered sectioning with title <title> at level <level>.
170 <*package>
171 \newcommand\omgroup@nonum[2]{%
172 \ifx\hyper@anchor\@undefined\else\phantomsection\fi%
173 \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}}

\omgroup@num convenience macro: \omgroup@num{<level>}{<title>} makes numbered sectioning with title <title> at level <level>. We have to check the short key was given in the omgroup environment and – if it is use it. But how to do that depends on whether the rdfmeta package has been loaded.
174 \newcommand\omgroup@num[2]{%
175 \sref@label@id{\omdoc@sect@name~\@nameuse{the#1}}%
176 \ifx\omgroup@short\@empty\@nameuse{#1}{#2}%
177 \else\@ifundefined{rdfmeta@sectioning}{\@nameuse{#1}[\omgroup@short]{#2}}%
178 {\@nameuse{rdfmeta@#1old}[\omgroup@short]{#2}}%
179 \fi}
180 </package>

omgroup
181 <*package>
182 \def\@true{true}
183 \def\@false{false}
184 \srefaddidkey{omgroup}
185 \addmetakey{omgroup}{date}
186 \addmetakey{omgroup}{creators}
187 \addmetakey{omgroup}{contributors}
188 \addmetakey{omgroup}{srccite}
189 \addmetakey{omgroup}{type}
190 \addmetakey*{omgroup}{short}
191 \addmetakey*{omgroup}{display}
192 \addmetakey[false]{omgroup}{loadmodules}[true]

\at@begin@omgroup we define a switch for numbering lines and a hook for the beginning of groups:
The \at@begin@omgroup macro allows customization. It is run at the beginning of the omgroup, i.e. after the section heading.
193 \newif\if@@num\@@numtrue
194 \newif\if@frontmatter\@frontmatterfalse
195 \newif\if@backmatter\@backmatterfalse
196 \newcommand\at@begin@omgroup[3][]{\{}

Then we define a helper macro that takes care of the sectioning magic. It comes with its own key/value interface for customization.
197 \addmetakey{omdoc@sect}{name}
198 \addmetakey[false]{omdoc@sect}{clear}[true]
199 \addmetakey{omdoc@sect}{ref}

```

```

200 \addmetakey[false]{omdoc@sect}{num}[true]
201 \newcommand\omdoc@sectioning[3][\metasetkeys{omdoc@sect}{#1}%
202 \ifx\omdoc@sect@clear\@true\cleardoublepage\fi%
203 \if@num% numbering not overridden by frontmatter, etc.
204 \ifx\omdoc@sect@num\@true\omgroup@num{#2}{#3}\else\omgroup@nonum{#2}{#3}\fi%
205 \def\current@section@level{\omdoc@sect@name}%
206 \else\omgroup@nonum{#2}{#3}\fi}

```

and another one, if redefines the `\addtocontentsline` macro of L^AT_EX to import the respective macros. It takes as an argument a list of module names.⁵

```

207 \newcommand\omgroup@redefine@addtocontents[1]{\edef\@import{#1}%
208 \@for\@I:=\@import\do{\edef\@path{\csname module@\@I @path\endcsname}%
209 \@ifundefined{tf@toc}\relax{\protected@write\tf@toc}{\string\@requiremodules{\@path}{sms}}}}
210 \ifx\hyper@anchor\@undefined% hyperref.sty loaded?
211 \def\addcontentsline##1##2##3{%
212 \addtocontents{##1}{\protect\contentsline{##2}{\string\importmodules{#1}##3}{\thepage}}}
213 \else\def\addcontentsline##1##2##3{%
214 \addtocontents{##1}{\protect\contentsline{##2}{\string\importmodules{#1}##3}{\thepage}{\@current}}
215 \fi}% hyperref.sty loaded?

```

now the `omgroup` environment itself. This takes care of the table of contents via the helper macro above and then selects the appropriate sectioning command from `article.cls`.

```

216 \newenvironment{omgroup}[2][\% keys, title
217 {\metasetkeys{omgroup}{#1}\sref@target%
218 \ifx\omgroup@display\st@flow\@numfalse\fi
219 \if@frontmatter\@numfalse\fi

```

If the `loadmodules` key is set on `\begin{omgroup}`, we redefine the `\addcontetsline` macro that determines how the sectioning commands below construct the entries for the table of contents.

```

220 \ifx\omgroup@loadmodules\@true%
221 \omgroup@redefine@addtocontents{\@ifundefined{mod@id}\imported@modules%
222 {\@ifundefined{module@\mod@id @path}\imported@modules{\mod@id}}\fi%

```

now we only need to construct the right sectioning depending on the value of `\section@level`.

```

223 \advance\section@level by 1
224 \ifcase\section@level%
225 \or\omdoc@sectioning[name=Part,clear,num]{part}{#2}%
226 \or\omdoc@sectioning[name=Chapter,clear,num]{chapter}{#2}%
227 \or\omdoc@sectioning[name=Section,num]{section}{#2}%
228 \or\omdoc@sectioning[name=Subsection,num]{subsection}{#2}%
229 \or\omdoc@sectioning[name=Subsubsection,num]{subsubsection}{#2}%
230 \or\omdoc@sectioning[name=Paragraph,ref=this paragraph]{paragraph}{#2}%
231 \or\omdoc@sectioning[name=Subparagraph,ref=this subparagraph]{paragraph}{#2}%
232 \fi% \ifcase
233 \at@begin@omgroup[#1]\section@level{#2}}}% for customization

```

⁵EDNOTE: MK: the extension `sms` is hard-coded here, but should not be. This will not work in multilingual settings.

```

234 {\advance\section@level by -1}
235 \end{package}
236 \let\section\section
237 DefEnvironment('{omgroup} OptionalKeyVals:omgroup {}',
238               "<omdoc:omgroup layout='sectioning'"
239               .   "?&GetKeyVal(#1,'id')(xml:id='&GetKeyVal(#1,'id')')()"
240               .   "?&GetKeyVal(#1,'type')(type='&GetKeyVal(#1,'type')')()>\n"
241               .   "<dc:title>#2</dc:title>\n"
242               .   "#body\n"
243               .   "</omdoc:omgroup>");
244 \end{ltxml.sty}

```

5.3 Front and Backmatter

Index markup is provided by the `omtext` package [Koh15b], so in the `omdoc` package we only need to supply the corresponding `\printindex` command, if it is not already defined

```

\printindex
245 \package
246 \providecommand\printindex{\IfFileExists{\jobname.ind}{\input{\jobname.ind}}{}}
247 \end{package}
248 \let\printindex\printindex
249 DefConstructor('\printindex','<omdoc:index/>');
250 Tag('omdoc:index',afterOpen=>\&numberIt,afterClose=>\&locateIt);
251 \end{ltxml.sty}

```

`\tableofcontents` The table of contents already exists in \LaTeX , so we only need to provide a \LaTeX ML binding for it.

```

252 \let\tableofcontents\tableofcontents
253 DefConstructor('\tableofcontents',
254               "<omdoc:tableofcontents level='&ToString(&CounterValue('tocdepth'))'/>");
255 Tag('omdoc:tableofcontents',afterOpen=>\&numberIt,afterClose=>\&locateIt);
256 \end{ltxml.sty}

```

The case of the `\bibliography` command is similar

`\bibliography`

```

257 \let\bibliography\bibliography
258 DefConstructor('\bibliography','<omdoc:bibliography files='#1'/>');
259 Tag('omdoc:bibliography',afterOpen=>\&numberIt,afterClose=>\&locateIt);
260 \end{ltxml.sty}

```

`frontmatter` `book.cls` already has a `\frontmatter` macro, so we have to redefine the front matter environment in this case.

```

261 \let\frontmatter\frontmatter
262 \ifclass@book
263 \renewenvironment{frontmatter}

```

```

264 {\@frontmattertrue\cleardoublepage\@mainmatterfalse\pagenumbering{roman}}
265 {\@frontmatterfalse\setcounter{page}{1}\pagenumbering{arabic}}
266 \else
267 \newenvironment{frontmatter}
268 {\@frontmattertrue\pagenumbering{roman}}
269 {\@frontmatterfalse\setcounter{page}{1}\pagenumbering{arabic}}
270 \fi
271 \</cls>
272 \<*txml.cls>
273 DefEnvironment('{frontmatter}', '#body');
274 \</txml.cls>
275 % \End{macrocode}
276 % \end{environment}
277 %
278 % \begin{environment}{backmatter}
279 % |book.cls| already has a |\backmatter| macro, so we have to redefine the back
280 % matter environment in this case.
281 % \begin{macrocode}
282 \<*cls>
283 \ifclass@book
284 \renewenvironment{backmatter}
285 {\cleardoublepage\@mainmatterfalse\@backmattertrue}
286 {\@backmatterfalse}
287 \else
288 \newenvironment{backmatter}{\@backmattertrue}{\@backmatterfalse}
289 \fi
290 \</cls>
291 \<*txml.cls>
292 DefEnvironment('{backmatter}', '#body');
293 \</txml.cls>

```

5.4 Ignoring Inputs

ignore

```

294 \<*package>
295 \ifshow@ignores
296 \addmetakey{ignore}{type}
297 \addmetakey{ignore}{comment}
298 \newenvironment{ignore}[1]{}
299 {\metasetkeys{ignore}{#1}\textless\ignore@type\textgreater\bgroupt\itshape}
300 {\egroupt\textless\ignore@type\textgreater}
301 \renewenvironment{ignore}{}{} \else\excludecomment{ignore}\fi
302 \</package>
303 \<*txml.sty>
304 DefKeyVal('ignore', 'type', 'Semiverbatim');
305 DefKeyVal('ignore', 'comment', 'Semiverbatim');
306 DefEnvironment('{ignore} OptionalKeyVals:ignore',
307 " <omdoc:ignore %&GetKeyVals(#1)>#body</omdoc:ignore>");
308 Tag('omdoc:ignore', afterOpen=>\&numberIt, afterClose=>\&locateIt);

```

309 \langle /ltxml.sty \rangle

5.5 Structure Sharing

6

```

310  $\langle$ *package $\rangle$ 
311 \providecommand{\lxDocumentID}[1]{}%
312 \def\LXMID#1#2{\expandafter\gdef\csname xmarg#1\endcsname{#2}\csname xmarg#1\endcsname}
313 \def\XMRef#1{\csname xmarg#1\endcsname}
314  $\langle$ /package $\rangle$ 
315  $\langle$ *ltxml $\rangle$ 
316 # Set the id to used for the top-level document
317 DefMacro('lxDocumentID{ }', '\def\thedocument@ID{#1}');
318
319 # \LXMID{id}{math} Associate the identifier id with the given math expression.
320 DefMacro('LXMID{ }{ }', '@XMArg{#1}{#2}');
321
322 # \XMRef{id} Refer to the math expression associated with id.
323 # In presentation, this is similar to using a shorthand macro.
324 # In content situations, an XMRef is generated.
325 DefMacro('XMRef{ }', '@XMRef{#1}');
326  $\langle$ /ltxml $\rangle$ 

```

\backslash STRlabel The main macro, it it used to attach a label to some text expansion. Later on, using the \backslash STRcopy macro, the author can use this label to get the expansion originally assigned.

```

327  $\langle$ *package $\rangle$ 
328 \long\def\STRlabel#1#2{\STRlabeldef{#1}{#2}{#2}}
329  $\langle$ /package $\rangle$ 
330  $\langle$ *ltxml.sty $\rangle$ 
331 DefConstructor('STRlabel{ }{ }', sub {
332   my($document,$label,$object)=@_;
333   $document->absorb($object);
334   $document->addAttribute('xml:id'=>ToString($label)) if $label; });
335  $\langle$ /ltxml.sty $\rangle$ 

```

\backslash STRcopy The \backslash STRcopy macro is used to call the expansion of a given label. In case the label is not defined it will issue a warning.⁷

```

336  $\langle$ *package $\rangle$ 
337 \newcommand\STRcopy[2][ ]{\expandafter\ifx\csname STR@#2\endcsname\relax
338 \message{STR warning: reference #2 undefined!}
339 \else\csname STR@#2\endcsname\fi}
340  $\langle$ /package $\rangle$ 
341  $\langle$ *ltxml.sty $\rangle$ 
342 DefConstructor('STRcopy[ ]{ }', "<omdoc:ref xref='#1##2' />");

```

⁶EdNOTE: The following is simply copied over from the latexml package, which we eliminated, we should integrate better.

⁷EdNOTE: MK: we need to do something about the ref!


```

343 Tag('omdoc:ref',afterOpen=>\&numberIt,afterClose=>\&locateIt);
344 \ltxml.sty

```

`\STRsemantics` if we have a presentation form and a semantic form, then we can use

```

345 \*package>
346 \newcommand\STRsemantics[3][\def\@test{#1}\ifx\@test\@empty\STRlabeldef{#1}{#2}\fi}
347 \package>
348 \ltxml.sty>
349 DefConstructor('\STRsemantics[]{}', sub {
350   my($document,$label,$ignore,$object)=@_;
351   $document->absorb($object);
352   $document->addAttribute('xml:id'=>ToString($label)) if $label; });
353 \ltxml.sty>#

```

`\STRlabeldef` This is the macro that does the actual labeling. Is it called inside `\STRlabel`

```

354 \*package>
355 \def\STRlabeldef#1{\expandafter\gdef\csname STR@#1\endcsname}
356 \package>
357 \ltxml.sty>
358 DefMacro('\STRlabeldef{}{}', "");
359 \ltxml.sty>

```

5.6 Colors

`blue, red, green, magenta` We will use the following abbreviations for colors from `color.sty`

```

360 \*package>
361 \def\black#1{\textcolor{black}{#1}}
362 \def\gray#1{\textcolor{gray}{#1}}
363 \def\blue#1{\textcolor{blue}{#1}}
364 \def\red#1{\textcolor{red}{#1}}
365 \def\green#1{\textcolor{green}{#1}}
366 \def\cyan#1{\textcolor{cyan}{#1}}
367 \def\magenta#1{\textcolor{magenta}{#1}}
368 \def\brown#1{\textcolor{brown}{#1}}
369 \def\yellow#1{\textcolor{yellow}{#1}}
370 \def\orange#1{\textcolor{orange}{#1}}
371 \package>

```

For the L^AT_EXML bindings, we go a generic route, we replace `\blue{#1}` by `{\@omdoc@color{blue}\@omdoc@color@content{#1}}`.

```

372 \ltxml.sty>
373 sub omdocColorMacro {
374   my ($color, @args) = @_;
375   my $tok_color = TokenizeInternal($color);
376   (T_BEGIN, T_CS('\@omdoc@color'), T_BEGIN, $tok_color->unlist,
377    T_END, T_CS('\@omdoc@color@content'), T_OTHER(''), $tok_color->unlist, T_OTHER('')),
378   T_BEGIN, $args[1]->unlist, T_END, T_END); }
379 DefMacro('\@omdoc@color{}', sub { MergeFont(color=>$_[1]->toString); return; });#
380 \ltxml.sty>

```

Ideally, here we will remove the optional argument and have a conversion module add the attribute at the end (or maybe add it just for math?) or, we can take the attributes for style from the current font ?

```
381 <*ltxml.sty>
382 DefConstructor(' \@omdoc@color@content[]{}',
383   "?#isMath(#2)(<ltx:text ?#1(style='color:#1'())>#2</ltx:text>");
384 foreach my $color(qw(black gray blue red green cyan magenta brown yellow orange)) {
385   DefMacro("\\".$color.'{}', sub { omdocColorMacro($color, @_); }); }# $
386 </ltxml.sty>
```

5.7 L^AT_EX Commands we interpret differently

The reinterpretations are quite simple, we either disregard presentational markup or we re-interpret it in terms of OMDoc.

```
387 <*ltxml.sty>
388 DefConstructor('\newpage', '');
389 </ltxml.sty>
```

5.8 Leftovers

```
390 <*package>
391 \newcommand\baseURI[2] [] {}
392 </package>
393 <*ltxml.sty>
394 DefMacro('\baseURI []Semiverbatim', sub {
395   my $baselocal = ToString(Digest($_[1]));
396   $baselocal = abs_path($baselocal) unless $baselocal =~ /^(\\w+):\\\/\\\/;
397   AssignValue('BASELOCAL'=>$baselocal, 'global');
398   AssignValue('URLBASE'=>ToString(Digest($_[2])), 'global');
399 });
400 </ltxml.sty>%$
```

⁸ and finally, we need to terminate the file with a success mark for perl.

```
401 <ltxml.sty | ltxml.cls>1;
```

⁸EDNOTE: this should be handled differently, omdoc.sty should include url and give a new macro for it, which we then use in omdoc

References

- [DCM03] The DCMI Usage Board. *DCMI Metadata Terms*. DCMI Recommendation. Dublin Core Metadata Initiative, 2003. URL: <http://dublincore.org/documents/dcmi-terms/>.
- [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. *dcm.sty: An Infrastructure for marking up Dublin Core Metadata in L^AT_EX documents*. Tech. rep. Comprehensive T_EX Archive Network (CTAN), 2015. URL: <http://www.ctan.org/tex-archive/macros/latex/contrib/stex/dcm/dcm.pdf>.
- [Koh15b] Michael Kohlhase. *omtext: Semantic Markup for Mathematical Text Fragments in L^AT_EX*. Tech. rep. Comprehensive T_EX Archive Network (CTAN), 2015. URL: <http://www.ctan.org/tex-archive/macros/latex/contrib/stex/omtext/omtext.pdf>.
- [Koh15c] Michael Kohlhase. *statements.sty: Structural Markup for Mathematical Statements*. Tech. rep. Comprehensive T_EX Archive Network (CTAN), 2015. URL: <http://www.ctan.org/tex-archive/macros/latex/contrib/stex/statements/statements.pdf>.
- [sTeX] *KWARC/sTeX*. URL: <https://svn.kwarc.info/repos/stex> (visited on 05/15/2015).