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

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

October 21, 2015

Abstract

The `omdoc` 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 documents in LaTeX. This includes a simple structure sharing mechanism for sTeX that allows to 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 sTeX sources, or after translation.

Contents

1 Introduction

SТEX is 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]

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 SТEX sources, or after translation. Currently, trans-document referencing provided by this package can only be used in the SТEX 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]

EdN:1

2 The User Interface

The `omdoc` package generates four files: `omdoc.cls`, `omdoc.sty` and their LATEXML 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 LATEXML infrastructure and thus should be used if OMDoc is to be generated from the SТEX 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:

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

---

[1]EDNOTE: integrate with latexml's XMRef in the Math mode.

2.2   Document Structure

document   The top-level `document` environment is augmented with an optional key/value argument that can be used to give metadata about the document. For the moment

id   only the `id` key is used to give an identifier to the `omdoc` element resulting from the L^AT_EXML transformation.

omgroup   The structure of the document is given by the `omgroup` environment just like in OMDoc. In the L^AT_EX route, the `omgroup` environment is flexibly mapped to sectioning commands, inducing the proper sectioning level from the nesting of `omgroup` environments. Correspondingly, the `omgroup` environment takes an optional key/value argument for metadata followed by a regular argument for the (section) title of the omgroup. The optional metadata argument has the

id   keys `id` for an identifier, `creators` and `contributors` for the Dublin Core meta-
creators   data [DCM03]; see [Koh15a] for details of the format. The `short` allows to give
contributors   a short title for the generated section. If the title contains semantic macros, they
short   need to be protected by `\protect`, and we need to give the `loadmodules` key it
loadmodules   needs no value. For instance we would have

```
\begin{module}{foo}
\symdef{bar}{B^a_r}
 ...
\begin{omgroup}[id=barderiv,loadmodules]
  {Introducing $\protect\bar$ Derivations}
```

ꜱT_EX 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 vari-

blindomgroup   ant `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".
- Th inner one groups the frontmatter[1] 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.

\currentsectionlevel   The `\currentsectionlevel` macro supplies the name of the current sectioning
\CurrentSectionLevel   level, e.g. "chapter", or "subsection". `\CurrentSectionLevel` is the capitalized variant. They are useful to write something like "In this `\currentsectionlevel`, we will..." in an `omgroup` environment, where we do not know which sectioning level we will end up.

---

[1]We 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

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   `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 sTeX 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
\STRcopy   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 LaTeXML 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 LaTeX. 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.

2.5 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{⟨something⟩}` writes ⟨*something*⟩ in blue. The macros `\red \green`, `\cyan`, `\magenta`, `\brown`, `\yellow`, `\orange`, `\gray`, and finally `\black` are analogous.

`\blue`
`\red`
`...`
`\black`

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 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 LaTeXML bindings (between ⟨*ltxml.cls⟩ and ⟨/ltxml.cls⟩ and ⟨*ltxml.sty⟩ and ⟨/ltxml.sty⟩ respetively). 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 LaTeXML (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.
\ifclass@book 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 \newif\ifclass@book\class@bookfalse
14 \DeclareOption{book}{\def\omdoc@class{book}\class@booktrue%
15 \PassOptionsToPackage{\CurrentOption}{omdoc}}
```

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

```
16 \DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}
17 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{\omdoc@class}}
18 \ProcessOptions
19 ⟨/cls⟩
20 ⟨∗ltxml.cls⟩
21 DeclareOption('report',sub {PassOptions('omdoc','sty',ToString(Digest(T_CS('\CurrentOption'))))
22 DeclareOption('book',sub {PassOptions('omdoc','sty',ToString(Digest(T_CS('\CurrentOption'))));
23 DeclareOption(undef,sub
24 {PassOptions('article','cls',ToString(Digest(T_CS('\CurrentOption'))));
25   PassOptions('omdoc','sty',ToString(Digest(T_CS('\CurrentOption'))));});
```

```
26 ProcessOptions();
27 ⟨/ltxml.cls⟩
```

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

```
28 ⟨∗cls⟩
29 \LoadClass{\omdoc@class}
30 \RequirePackage{etoolbox}
31 \RequirePackage{omdoc}
32 ⟨/cls⟩
33 ⟨∗ltxml.cls⟩
34 LoadClass('article');
35 RequirePackage('sref');
36 ⟨/ltxml.cls⟩
```

4.2 Setting up Namespaces and Schemata for LaTeXML

Now, we also need to register the namespace prefixes for LaTeXML to use.

```
37 ⟨∗ltxml.cls⟩
38 RegisterNamespace('omdoc'=>"http://omdoc.org/ns");
39 RegisterNamespace('om'=>"http://www.openmath.org/OpenMath");
40 RegisterNamespace('m'=>"http://www.w3.org/1998/Math/MathML");
41 RegisterNamespace('dc'=>"http://purl.org/dc/elements/1.1/");
42 RegisterNamespace('cc'=>"http://creativecommons.org/ns");
43 RegisterNamespace('stex'=>"http://kwarc.info/ns/sTeX");
44 RegisterNamespace('ltx'=>"http://dlmf.nist.gov/LaTeXML");
45 ⟨/ltxml.cls⟩
```

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

```
46 ⟨∗ltxml.cls⟩
47 RelaxNGSchema('omdoc+ltxml',
48        '#default'=>"http://omdoc.org/ns",
49        'om'=>"http://www.openmath.org/OpenMath",
50        'm'=>"http://www.w3.org/1998/Math/MathML",
51        'dc'=>"http://purl.org/dc/elements/1.1/",
52        'cc'=>"http://creativecommons.org/ns",
53       'stex'=>"http://kwarc.info/ns/sTeX",
54        'ltx'=>"http://dlmf.nist.gov/LaTeXML");
55 ⟨/ltxml.cls⟩
```

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

```
56 ⟨∗ltxml.cls⟩
57 RequirePackage('omdoc');
58 ⟨/ltxml.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.

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

```
59 ⟨∗cls⟩
60 \let\orig@document=\document
61 \srefaddidkey{document}
62 \renewcommand{\document}[1][]{\metasetkeys{document}{#1}\orig@document}
63 ⟨/cls⟩
64 ⟨∗ltxml.cls⟩
65 sub xmlBase {
66   my $baseuri = LookupValue('URLBASE');
67   $baseuri =~ s/\/$//g; # No trailing slashes
68   Tokenize($baseuri); }
69 DefEnvironment('{document} OptionalKeyVals:omdoc',
70       "<omdoc:omdoc "
71     .    "?&GetKeyVal(#1,'id')(xml:id='&GetKeyVal(#1,'id')')"
72     .     "(?&Tokenize(&LookupValue('SOURCEBASE'))"
73     .      "(xml:id='&Tokenize(&LookupValue('SOURCEBASE')).omdoc')()) "
74     .     "?&Tokenize(&LookupValue('URLBASE'))"
75     .      "(xml:base='&xmlBase()')()>"
76    .  "#body"
77    ."</omdoc:omdoc>",
78   beforeDigest=> sub { AssignValue(inPreamble=>0); },
79   afterDigest=> sub { $_[0]->getGullet->flush; return; },
80   afterDigestBegin => sub {
81    $_[1]->setProperty(id => Expand(T_CS('\thedocument@ID')));
82    if (my $ops = LookupValue('@at@begin@document')) {
83      Digest(Tokens(@$ops)); }
84    else {
85      return; } },
86   beforeDigestEnd => sub {
87    $_[0]->getGullet->flush;
88    if (my $ops = LookupValue('@at@end@document')) {
89      Digest(Tokens(@$ops)); }
90    else {
91      return; } },
92   mode => 'text');
93 Tag('omdoc:omdoc', 'afterOpen:late'=>\&insertFrontMatter,
94      afterOpen=>\&numberIt,afterClose=>\&locateIt);
95 ⟨/ltxml.cls⟩%$
```

5 Implementation: OMDoc Package

5.1 Package Options

The package options come in two parts: the first we only pass on to the various other SₜₑX pacakges.

```
96 ⟨∗package⟩
97 \DeclareOption{showmeta}{\PassOptionsToPackage{\CurrentOption}{metakeys}}
98 \DeclareOption{showmods}{\PassOptionsToPackage{\CurrentOption}{modules}}
99 \DeclareOption{extrefs}{\PassOptionsToPackage{\CurrentOption}{sref}}
100 \DeclareOption{noauxreq}{\PassOptionsToPackage{\CurrentOption}{modules}}
101 \DeclareOption{defindex}{\PassOptionsToPackage{\CurrentOption}{statements}}
102 ⟨/package⟩
103 ⟨∗ltxml.sty⟩
104 DeclareOption('showmeta',sub {PassOptions('metakeys','sty',ToString(Digest(T_CS('\CurrentOption
105 DeclareOption('showmods',sub {PassOptions('modules','sty',ToString(Digest(T_CS('\CurrentOption'
106 DeclareOption('extrefs',sub {PassOptions('sref','sty',ToString(Digest(T_CS('\CurrentOption'))))
107 DeclareOption('noauxreq',sub {PassOptions('modules','sty',ToString(Digest(T_CS('\CurrentOption'
108 DeclareOption('defindex',sub {PassOptions('statements','sty',ToString(Digest(T_CS('\CurrentOpti
109 ⟨/ltxml.sty⟩
```

For the rest 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 af-
\section@level   fect the sectioning behavior of the omgroup environment via the \section@level macro later.

```
110 ⟨∗package⟩
111 \newif\ifshow@ignores\show@ignoresfalse
112 \DeclareOption{showignores}{\show@ignorestrue}
113 \newcount\section@level
114 \def\omdoc@class{article}\section@level=2
115 \DeclareOption{report}{\def\omdoc@class{report}\section@level=1}
116 \DeclareOption{book}{\def\omdoc@class{book}\section@level=0}
117 \DeclareOption*{}% accept all other options
118 \ProcessOptions
119 ⟨/package⟩
120 ⟨∗ltxml.sty⟩
121 DeclareOption('showignores','');
122 DeclareOption('report','');
123 DeclareOption('book','');
124 DeclareOption(undef, '');
125 ProcessOptions();
126 ⟨/ltxml.sty⟩
```

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

```
127 ⟨∗package⟩
128 \RequirePackage{sref}
129 \RequirePackage{xspace}
```

```
130 \RequirePackage{comment}
131 \RequirePackage{etoolbox}
132 ⟨/package⟩
133 ⟨∗ltxml.sty⟩
134 RequirePackage('sref');
135 RequirePackage('xspace');
136 RequirePackage('omtext');
137 ⟨/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.

\currentsectionlevel

```
138 ⟨∗package⟩
139 \def\currentsectionlevel{document\xspace}%
140 \def\Currentsectionlevel{Document\xspace}%
141 ⟨/package⟩
142 ⟨∗ltxml.sty⟩
143 DefMacro('\currentsectionlevel','\@currentsectionlevel\xspace');
144 DefMacro('\Currentsectionlevel','\@Currentsectionlevel\xspace');
145 DefConstructor('\@currentsectionlevel',
146               "<ltx:text class='omdoc-currentsectionlevel'>section</ltx:text>");
147 DefConstructor('\@CurrentSectionLevel',
148               "<ltx:text class='omdoc-Currentsectionlevel'>Section</ltx:text>");
149 ⟨/ltxml.sty⟩
```

blindomgroup

```
150 ⟨∗package⟩
151 \newcommand\at@begin@blindomgroup[1]{}
152 \newenvironment{blindomgroup}
153 {\advance\section@level by 1\at@begin@blindomgroup\setion@level}
154 {\advance\section@level by -1}
155 ⟨/package⟩
156 ⟨∗ltxml.sty⟩
157 DefEnvironment('{blindomgroup} OptionalKeyVals:omgroup',
158               "<omdoc:omgroup layout='invisible'"
159           .   "?&GetKeyVal(#1,'id')(xml:id='&GetKeyVal(#1,'id')')()"
160           .   "?&GetKeyVal(#1,'type')(type='&GetKeyVal(#1,'type')')()>\n"
161        . "#body\n"
162      . "</omdoc:omgroup>");
163 ⟨/ltxml.sty⟩
```

\omgroup@cl Convenience macro: defines the \currentsectionlevel macro from the keywords
in the arguments

```
164 ⟨∗package⟩
165 \newcommand\omgroup@cl[2]{%
```

```
166 \def\currentsectionlevel{#1\xspace}%
167 \def\Currentsectionlevel{#2\xspace}}
```

\omgroup@nonum   convenience macro: \omgroup@nonum{⟨level⟩}{⟨title⟩} makes an unnumbered sec-
                  tioning with title ⟨title⟩ at level ⟨level⟩.

```
168 \newcommand\omgroup@nonum[2]{%
169 \ifx\hyper@anchor\@undefined\else\phantomsection\fi%
170 \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}}
```

\omgroup@num   convenience macro: \omgroup@nonum{⟨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.

```
171 \newcommand\omgroup@num[2]{\sref@label@id{\omdoc@sect@Name~\@nameuse{the#1}}%
172 \ifx\omgroup@short\@empty\@nameuse{#1}{#2}%
173 \else\@ifundefined{rdfmeta@sectioning}{\@nameuse{#1}[\omgroup@short]{#2}}%
174 {\@nameuse{rdfmeta@#2@old}[\omgroup@short]{#2}}\fi}
175 ⟨/package⟩
```

omgroup

```
176 ⟨*package⟩
177 \def\@true{true}
178 \def\@false{false}
179 \srefaddidkey{omgroup}
180 \addmetakey{omgroup}{date}
181 \addmetakey{omgroup}{creators}
182 \addmetakey{omgroup}{contributors}
183 \addmetakey{omgroup}{srccite}
184 \addmetakey{omgroup}{type}
185 \addmetakey*{omgroup}{short}
186 \addmetakey*{omgroup}{display}
187 \addmetakey[false]{omgroup}{loadmodules}[true]
```

                  we define a switch for numbering lines and a hook for the beginning of groups:
\at@begin@omgroup The \at@begin@omgroup macro allows customization. It is run at the beginning
                  of the omgroup, i.e. after the section heading.

```
188 \newif\if@@num\@@numtrue
189 \newif\if@frontmatter\@frontmatterfalse
190 \newif\if@backmatter\@backmatterfalse
191 \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.

```
192 \addmetakey{omdoc@sect}{name}
193 \addmetakey{omdoc@sect}{Name}
194 \addmetakey[false]{omdoc@sect}{clear}[true]
195 \addmetakey{omdoc@sect}{ref}
196 \addmetakey[false]{omdoc@sect}{num}[true]
197 \newcommand\omdoc@sectioning[3][]{\metasetkeys{omdoc@sect}{#1}%
```

```
198 \ifx\omdoc@sect@clear\@true\cleardoublepage\fi%
199 \if@@num% numbering not overridden by frontmatter, etc.
200 \ifx\omdoc@sect@num\@true\omgroup@num{#2}{#3}\else\omgroup@nonum{#2}{#3}\fi
201 \omgroup@cl\omdoc@sect@name\omdoc@sect@Name
202 \else\omgroup@nonum{#2}{#3}\fi}
```

and another one, if redefines the \addtocontentsline macro of LATEX to import
the respective macros. It takes as an argument a list of module names.[2]

```
203 \newcommand\omgroup@redefine@addtocontents[1]{\edef\@@import{#1}%
204 \@for\@I:=\@@import\do{\edef\@path{\csname module@\@I  @path\endcsname}%
205 \@ifundefined{tf@toc}\relax{\protected@write\tf@toc{}{\string\@requiremodules{\@path}{sms}}}}}
206 \ifx\hyper@anchor\@undefined% hyperref.sty loaded?
207 \def\addcontentsline##1##2##3{%
208 \addtocontents{##1}{\protect\contentsline{##2}{\string\importmodules{#1}##3}{\thepage}}}
209 \else\def\addcontentsline##1##2##3{%
210 \addtocontents{##1}{\protect\contentsline{##2}{\string\importmodules{#1}##3}{\thepage}{\@curren
211 \fi}% hypreref.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 sectinoning command
from article.cls.

```
212 \newenvironment{omgroup}[2][]% keys, title
213 {\metasetkeys{omgroup}{#1}\sref@target%
214 \ifx\omgroup@display\st@flow\@@numfalse\fi
215 \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.

```
216 \ifx\omgroup@loadmodules\@true%
217 \omgroup@redefine@addtocontents{\@ifundefined{mod@id}\imported@modules%
218 {\@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.

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

---

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

```
232 ⟨*ltxml.sty⟩
233 DefEnvironment('{omgroup} OptionalKeyVals:omgroup {}',
234                "<omdoc:omgroup layout='sectioning'"
235              .   "?&GetKeyVal(#1,'id')(xml:id='&GetKeyVal(#1,'id')')()"
236              .   "?&GetKeyVal(#1,'type')(type='&GetKeyVal(#1,'type')')()>\n"
237          . "<dc:title>#2</dc:title>\n"
238        . "#body\n"
239      . "</omdoc:omgroup>");
240 ⟨/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

```
241 ⟨*package⟩
242 \providecommand\printindex{\IfFileExists{\jobname.ind}{\input{\jobname.ind}}{}}
243 ⟨/package⟩
244 ⟨*ltxml.sty⟩
245 DefConstructor('\printindex','<omdoc:index/>');
246 Tag('omdoc:index',afterOpen=>\&numberIt,afterClose=>\&locateIt);
247 ⟨/ltxml.sty⟩
```

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

```
248 ⟨*ltxml.sty⟩
249 DefConstructor('\tableofcontents',
250                "<omdoc:tableofcontents level='&ToString(&CounterValue('tocdepth'))'/>");
251 Tag('omdoc:tableofcontents',afterOpen=>\&numberIt,afterClose=>\&locateIt);
252 ⟨/ltxml.sty⟩
```

The case of the \bibliography command is similar

\bibliography

```
253 ⟨*ltxml.sty⟩
254 DefConstructor('\bibliography{}',"<omdoc:bibliography files='#1'/>");
255 Tag('omdoc:bibliography',afterOpen=>\&numberIt,afterClose=>\&locateIt);
256 ⟨/ltxml.sty⟩
```

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

```
257 ⟨*cls⟩
258 \ifclass@book
259 \renewenvironment{frontmatter}
260 {\@frontmattertrue\cleardoublepage\@mainmatterfalse\pagenumbering{roman}}
261 {\@frontmatterfalse\setcounter{page}{1}\pagenumbering{arabic}}
```

14

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

5.4   Ignoring Inputs

ignore

```
290 ⟨∗package⟩
291 \ifshow@ignores
292 \addmetakey{ignore}{type}
293 \addmetakey{ignore}{comment}
294 \newenvironment{ignore}[1][]
295 {\metasetkeys{ignore}{#1}\textless\ignore@type\textgreater\bgroup\itshape}
296 {\egroup\textless/\ignore@type\textgreater}
297 \renewenvironment{ignore}{}{}\else\excludecomment{ignore}\fi
298 ⟨/package⟩
299 ⟨∗ltxml.sty⟩
300 DefKeyVal('ignore','type','Semiverbatim');
301 DefKeyVal('ignore','comment','Semiverbatim');
302 DefEnvironment('{ignore} OptionalKeyVals:ignore',
303                 "<omdoc:ignore  %&GetKeyVals(#1)>#body</omdoc:ignore>");
304 Tag('omdoc:ignore',afterOpen=>\&numberIt,afterClose=>\&locateIt);
305 ⟨/ltxml.sty⟩
```

5.5 Structure Sharing

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

```
306 ⟨∗package⟩
307 \long\def\STRlabel#1#2{\STRlabeldef{#1}{#2}{#2}}
308 ⟨/package⟩
309 ⟨∗ltxml.sty⟩
310 DefConstructor('\STRlabel{}{}', sub {
311   my($document,$label,$object)=@_;
312   $document->absorb($object);
313   $document->addAttribute('xml:id'=>ToString($label)) if $label; });
314 ⟨/ltxml.sty⟩
```

\STRcopy The **\STRcopy** macro is used to call the expansion of a given label. In case the label is not defined it will issue a warning.[3]

```
315 ⟨∗package⟩
316 \newcommand\STRcopy[2][]{\expandafter\ifx\csname STR@#2\endcsname\relax
317 \message{STR warning: reference #2 undefined!}
318 \else\csname STR@#2\endcsname\fi}
319 ⟨/package⟩
320 ⟨∗ltxml.sty⟩
321 DefConstructor('\STRcopy[]{}',"<omdoc:ref xref='#1##2'/>");
322 Tag('omdoc:ref',afterOpen=>\&numberIt,afterClose=>\&locateIt);
323 ⟨/ltxml.sty⟩
```

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

```
324 ⟨∗package⟩
325 \newcommand\STRsemantics[3][]{#2\def\@test{#1}\ifx\@test\@empty\STRlabeldef{#1}{#2}\fi}
326 ⟨/package⟩
327 ⟨∗ltxml.sty⟩
328 DefConstructor('\STRsemantics[]{}{}', sub {
329   my($document,$label,$ignore,$object)=@_;
330   $document->absorb($object);
331   $document->addAttribute('xml:id'=>ToString($label)) if $label; });
332 ⟨/ltxml.sty⟩#$
```

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

```
333 ⟨∗package⟩
334 \def\STRlabeldef#1{\expandafter\gdef\csname STR@#1\endcsname}
335 ⟨/package⟩
336 ⟨∗ltxml.sty⟩
337 DefMacro('\STRlabeldef{}{}', "");
338 ⟨/ltxml.sty⟩
```

---

[3]EDNOTE: MK: we need to do something about the ref!

5.6 Colors

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

```
339 ⟨∗package⟩
340 \def\black#1{\textcolor{black}{#1}}
341 \def\gray#1{\textcolor{gray}{#1}}
342 \def\blue#1{\textcolor{blue}{#1}}
343 \def\red#1{\textcolor{red}{#1}}
344 \def\green#1{\textcolor{green}{#1}}
345 \def\cyan#1{\textcolor{cyan}{#1}}
346 \def\magenta#1{\textcolor{magenta}{#1}}
347 \def\brown#1{\textcolor{brown}{#1}}
348 \def\yellow#1{\textcolor{yellow}{#1}}
349 \def\orange#1{\textcolor{orange}{#1}}
350 ⟨/package⟩
```

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

```
351 ⟨∗ltxml.sty⟩
352 sub omdocColorMacro {
353   my ($color, @args) = @_;
354   my $tok_color = TokenizeInternal($color);
355   (T_BEGIN, T_CS('\@omdoc@color'), T_BEGIN, $tok_color->unlist,
356    T_END, T_CS('\@omdoc@color@content'), T_OTHER('['), $tok_color->unlist, T_OTHER(']'),
357    T_BEGIN, $args[1]->unlist, T_END, T_END); }
358 DefMacro('\@omdoc@color{}', sub { MergeFont(color=>$_[1]->toString); return; });#$
359 ⟨/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 ?

```
360 ⟨∗ltxml.sty⟩
361 DefConstructor('\@omdoc@color@content[]{}',
362   "?#isMath(#2)(<ltx:text ?#1(style='color:#1')()>#2</ltx:text>)");
363 foreach my $color(qw(black gray blue red green cyan magenta brown yellow orange)) {
364   DefMacro("\\".$color.'{}', sub { omdocColorMacro($color, @_); }); }#$
365 ⟨/ltxml.sty⟩
```

5.7 LaTeX Commands we interpret differently

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

```
366 ⟨∗ltxml.sty⟩
367 DefConstructor('\newpage','');
368 ⟨/ltxml.sty⟩
```

5.8 Leftovers

```
369 ⟨∗package⟩
```

```
370 \newcommand\baseURI[2][]{}
371 ⟨/package⟩
372 ⟨∗ltxml.sty⟩
373 DefMacro('\baseURI []Semiverbatim', sub {
374    my $baselocal = ToString(Digest($_[1]));
375    $baselocal = abs_path($baselocal) unless $baselocal=~/^(\w+):\/\//;
376    AssignValue('BASELOCAL'=>$baselocal,'global');
377    AssignValue('URLBASE'=>ToString(Digest($_[2])),'global');
378 });
379 ⟨/ltxml.sty⟩%$
```

EdN:4

^{4} and finally, we need to terminate the file with a success mark for perl.

```
380 ⟨ltxml.sty | ltxml.cls⟩1;
```

---

^{4}EDNOTE: 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 LaTeX documents*. Tech. rep. Comprehensive TeX 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 LaTeX*. Tech. rep. Comprehensive TeX 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 TeX Archive Network (CTAN), 2015. URL: `http://www.ctan.org/tex-archive/macros/latex/contrib/stex/statements/statements.pdf`.