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

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

December 5, 2016

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

sTeX 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 sTeX sources, or after translation. Currently, trans-document referencing provided by this package can only be used in the sTeX 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 two files: `omdoc.cls`, and `omdoc.sty`. The OMDoc class is a minimally changed variant of the standard `article` class that includes the functionality provided by `omdoc.sty`. 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`

2.2 Document Structure

`document`
`\documentkeys`
`id`

The top-level `document` environment can be given key/value information by the `\documentkeys` macro in the preamble[1]. This can be used to give metadata about the document. For the moment only the `id` key is used to give an identifier to the `omdoc` element resulting from the LaTeXML transformation.

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

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

omgroup

The structure of the document is given by the `omgroup` environment just like in OMDoc. In the LaTeX 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 keys `id` for an identifier, `creators` and `contributors` for the Dublin Core metadata [DCM03]; see [Koh16a] for details of the format. The `short` allows to give a short title for the generated section. If the title contains semantic macros, they need to be protected by `\protect`, and we need to give the `loadmodules` key it needs no value. For instance we would have

id
creators
contributors
short
loadmodules

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

STeX 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`:

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[2] 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
\CurrentSectionLevel

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`, 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

[2] 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

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 [Koh16c] 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
\STRcopy

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 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]

EdN:2

[2] EdNote: document LMID und LMXREf here if we decide to keep them.

2.5 Colors

\blue
\red
...
\black

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.

3 Limitations

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

1. 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.

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.

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`

```
⟨*cls⟩
\def\omdoc@class{article}
\DeclareOption{report}{\def\omdoc@class{report}%
  \PassOptionsToPackage{\CurrentOption}{omdoc}
  \PassOptionsToPackage{\CurrentOption}{stex}}
\newif\ifclass@book\class@bookfalse
\DeclareOption{book}{\def\omdoc@class{book}\class@booktrue%
  \PassOptionsToPackage{\CurrentOption}{omdoc}
  \PassOptionsToPackage{\CurrentOption}{stex}}
```

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

```
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{\omdoc@class}
                          \PassOptionsToPackage{\CurrentOption}{omdoc}
                          \PassOptionsToPackage{\CurrentOption}{stex}}
\ProcessOptions
```

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

```
\LoadClass{\omdoc@class}
\RequirePackage{omdoc}
\RequirePackage{stex}
```

4.2 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.[3]

EdN:3

```
\srefaddidkey{document}
\newcommand\documentkeys[1]{\metasetkeys{document}{#1}}
\let\orig@document=\document
\srefaddidkey{document}
```

[3] EdNote: faking documentkeys for now. @HANG, please implement

```
\renewcommand{\document}[1][]{\metasetkeys{document}{#1}\orig@document}
⟨/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`

```
⟨*package⟩
\newif\ifshow@ignores\show@ignoresfalse
\DeclareOption{showignores}{\show@ignorestrue}
\newcount\section@level\section@level=2
\newif\ifclass@book\class@bookfalse
\DeclareOption{report}{\section@level=0}
\DeclareOption{book}{\section@level=0\class@booktrue}
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{sref}}
\ProcessOptions
```

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

```
\RequirePackage{etoolbox}
\RequirePackage{sref}
\RequirePackage{xspace}
\RequirePackage{comment}
\RequirePackage{pathsuris}
```

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`

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`, wich will be instantiated by CSS later.[4]

EdN:4

```
\def\current@section@level{document}%
\newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
\newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

blindomgroup

```
\newcommand\at@begin@blindomgroup[1]{}
\newenvironment{blindomgroup}
{\advance\section@level by 1\at@begin@blindomgroup\setion@level}
{\advance\section@level by -1}
```

\omgroup@nonum convenience macro: `\omgroup@nonum{`⟨*level*⟩`}{`⟨*title*⟩`}` makes an unnumbered sectioning with title ⟨*title*⟩ at level ⟨*level*⟩.

```
\newcommand\omgroup@nonum[2]{%
\ifx\hyper@anchor\@undefined\else\phantomsection\fi%
\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. In the end we call `\sref@label@id` to enable crossreferencing.

```
\newcommand\omgroup@num[2]{%
\edef\@@ID{\sref@id}
\ifx\omgroup@short\@empty% no short title
\@nameuse{#1}{#2}%
\else% we have a short title
\@ifundefined{rdfmeta@sectioning}%
  {\@nameuse{#1}[\omgroup@short]{#2}}%
  {\@nameuse{rdfmeta@#1@old}[\omgroup@short]{#2}}%
\fi%
\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\@@ID}
```

omgroup

```
\def\@true{true}
\def\@false{false}
\srefaddidkey{omgroup}
\addmetakey{omgroup}{date}
\addmetakey{omgroup}{creators}
\addmetakey{omgroup}{contributors}
\addmetakey{omgroup}{srccite}
\addmetakey{omgroup}{type}
\addmetakey*{omgroup}{short}
\addmetakey*{omgroup}{display}
\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.

```
\newif\if@@num\@@numtrue
\newif\if@frontmatter\@frontmatterfalse
\newif\if@backmatter\@backmatterfalse
\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.

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

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.[5]

EdN:5

```
\newcommand\omgroup@redefine@addtocontents[1]{%
\edef\@@import{#1}%
\@for\@I:=\@@import\do{%
\edef\@path{\csname module@\@I  @path\endcsname}%
\@ifundefined{tf@toc}\relax%
     {\protected@write\tf@toc{}{\string\@requiremodules{\@path}{sms}}}}
\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
\def\addcontentsline##1##2##3{%
\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}}
\else% hyperref.sty not loaded
\def\addcontentsline##1##2##3{%
\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}{\@cu
\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 sectioning command from `article.cls`.

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

```
\ifx\omgroup@loadmodules\@true%
\omgroup@redefine@addtocontents{\@ifundefined{mod@id}\used@modules%
{\@ifundefined{module@\mod@id @path}{\used@modules}\mod@id}}\fi%
```

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

```
\advance\section@level by 1\relax%
```

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

```
\ifcase\section@level%
\or\omdoc@sectioning[name=Part,clear,num]{part}{#2}%
\or\omdoc@sectioning[name=Chapter,clear,num]{chapter}{#2}%
\or\omdoc@sectioning[name=Section,num]{section}{#2}%
\or\omdoc@sectioning[name=Subsection,num]{subsection}{#2}%
\or\omdoc@sectioning[name=Subsubsection,num]{subsubsection}{#2}%
\or\omdoc@sectioning[name=Paragraph,ref=this paragraph]{paragraph}{#2}%
\or\omdoc@sectioning[name=Subparagraph,ref=this subparagraph]{paragraph}{#2}%
\fi% \ifcase
\at@begin@omgroup[#1]\section@level{#2}}% for customization
{\advance\section@level by -1}
```

5.3 Front and Backmatter

Index markup is provided by the `omtext` package [Koh16b], so in the `omdoc` package we only need to supply the corresponding `\printindex` command, if it is not already defined

`\printindex`

```
\providecommand\printindex{\IfFileExists{\jobname.ind}{\input{\jobname.ind}}{}}
```

⟨/package⟩

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

```
⟨*cls⟩
\ifclass@book
\renewenvironment{frontmatter}
{\@frontmattertrue\cleardoublepage\@mainmatterfalse\pagenumbering{roman}}
{\@frontmatterfalse\setcounter{page}{1}\pagenumbering{arabic}}
\else
\newenvironment{frontmatter}
{\@frontmattertrue\pagenumbering{roman}}
{\@frontmatterfalse\setcounter{page}{1}\pagenumbering{arabic}}
\fi
%    \End{macrocode}
% \end{environment}
%
% \begin{environment}{backmatter}
%   |book.cls| already has a |\backmatter| macro, so we have to redefine the back
%   matter environment in this case.
%    \begin{macrocode}
\ifclass@book
\renewenvironment{backmatter}
{\cleardoublepage\@mainmatterfalse\@backmattertrue}
{\@backmatterfalse}
\else
\newenvironment{backmatter}{\@backmattertrue}{\@backmatterfalse}
\fi
⟨/cls⟩
```

5.4 Ignoring Inputs

ignore

```
⟨*package⟩
\ifshow@ignores
\addmetakey{ignore}{type}
\addmetakey{ignore}{comment}
\newenvironment{ignore}[1][]
{\metasetkeys{ignore}{#1}\textless\ignore@type\textgreater\bgroup\itshape}
{\egroup\textless/\ignore@type\textgreater}
\renewenvironment{ignore}{}{}\else\excludecomment{ignore}\fi
```

5.5 Structure Sharing

EdN:6 [6]

```
\providecommand{\lxDocumentID}[1]{}%
\def\LXMID#1#2{\expandafter\gdef\csname xmarg#1\endcsname{#2}\csname xmarg#1\endcsname}
\def\LXMRef#1{\csname xmarg#1\endcsname}
```

\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.

```
\long\def\STRlabel#1#2{\STRlabeldef{#1}{#2}{#2}}
```

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

```
\newcommand\STRcopy[2][]{\expandafter\ifx\csname STR@#2\endcsname\relax
\message{STR warning: reference #2 undefined!}
\else\csname STR@#2\endcsname\fi}
```

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

```
\newcommand\STRsemantics[3][]{#2\def\@test{#1}\ifx\@test\@empty\STRlabeldef{#1}{#2}\fi}
```

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

```
\def\STRlabeldef#1{\expandafter\gdef\csname STR@#1\endcsname}
```

5.6 Colors

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

```
\def\black#1{\textcolor{black}{#1}}
\def\gray#1{\textcolor{gray}{#1}}
\def\blue#1{\textcolor{blue}{#1}}
\def\red#1{\textcolor{red}{#1}}
\def\green#1{\textcolor{green}{#1}}
\def\cyan#1{\textcolor{cyan}{#1}}
```

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

[7] EdNote: MK: we need to do something about the ref!

```
\def\magenta#1{\textcolor{magenta}{#1}}
\def\brown#1{\textcolor{brown}{#1}}
\def\yellow#1{\textcolor{yellow}{#1}}
\def\orange#1{\textcolor{orange}{#1}}
⟨/package⟩
```

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.

[Koh16a] Michael Kohlhase. *dcm.sty: An Infrastructure for marking up Dublin Core Metadata in LaTeX documents*. Tech. rep. Comprehensive TeX Archive Network (CTAN), 2016. URL: http://mirror.ctan.org/macros/latex/contrib/stex/sty/dcm/dcm.pdf.

[Koh16b] Michael Kohlhase. *omtext: Semantic Markup for Mathematical Text Fragments in LaTeX*. Tech. rep. Comprehensive TeX Archive Network (CTAN), 2016. URL: http://mirror.ctan.org/macros/latex/contrib/stex/sty/omtext/omtext.pdf.

[Koh16c] Michael Kohlhase. *statements.sty: Structural Markup for Mathematical Statements*. Tech. rep. Comprehensive TeX Archive Network (CTAN), 2016. URL: http://mirror.ctan.org/macros/latex/contrib/stex/sty/statements/statements.pdf.

[sTeX] *KWARC/sTeX*. URL: https://github.com/KWARC/sTeX (visited on 05/15/2015).