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 \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.
1 Introduction

STEX is a version of T\(\LaTeX\)/L\(\LaTeX\) that allows to markup T\(\LaTeX\)/L\(\LaTeX\) documents semantically without leaving the document format, essentially turning T\(\LaTeX\)/L\(\LaTeX\) into a document format for mathematical knowledge management (MKM). The package supports direct translation to the OMDoc format \cite{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.\footnote{EdNote: Integrate with latexml’s XMRef in the Math mode.}

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 class accept the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>class=(name)</td>
<td>load (\langle name\rangle).cls instead of article.cls</td>
</tr>
<tr>
<td>topsect=(sect)</td>
<td>The top-level sectioning level; the default for (sect) is section</td>
</tr>
<tr>
<td>showignores</td>
<td>show the the contents of the ignore environment after all</td>
</tr>
<tr>
<td>showmeta</td>
<td>show the metadata; see metakeys.sty</td>
</tr>
<tr>
<td>showmods</td>
<td>show modules; see modules.sty</td>
</tr>
<tr>
<td>extrefs</td>
<td>allow external references; see sref.sty</td>
</tr>
<tr>
<td>defindex</td>
<td>index definienda; see statements.sty</td>
</tr>
<tr>
<td>mh</td>
<td>MathHub support; see Kohlhase:mss:ctan</td>
</tr>
</tbody>
</table>

The omdoc package accepts the same except the first two.

2.2 Document Structure

The top-level document environment can be given key/value information by the \documentkeys macro in the preamble. This can be used to give metadata about
The structure of the document is given by the \texttt{omgroup} environment just like in OMDoc. In the \LaTeX{} route, the \texttt{omgroup} environment is flexibly mapped to sectioning commands, inducing the proper sectioning level from the nesting of \texttt{omgroup} environments. Correspondingly, the \texttt{omgroup} environment takes an optional key/value argument for metadata followed by a regular argument for the (section) title of the \texttt{omgroup}. The optional metadata argument has the keys \texttt{id} for an identifier, \texttt{creators} and \texttt{contributors} for the Dublin Core metadata \cite{DCM03}; see \cite{Koh16a} for details of the format. The \texttt{short} allows to give a short title for the generated section. If the title contains semantic macros, they need to be protected by $\texttt{\protect}$, and we need to give the \texttt{loadmodules} key it needs no value. For instance we would have

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

\LaTeX{} automatically computes the sectioning level, from the nesting of \texttt{omgroup} environments. But sometimes, we want to skip levels (e.g. to use a subsection* as an introduction for a chapter). Therefore the \texttt{omdoc} package provides a variant \texttt{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 \texttt{blindomgroup} environment is useful e.g. for creating frontmatter at the correct level. Example\footnote{1} shows a typical setup for the outer document structure of a book with parts and chapters. We use two levels of \texttt{blindomgroup}:

\begin{itemize}
\item The outer one groups the introductory parts of the book (which we assume to have a sectioning hierarchy topping at the part level). This \texttt{blindomgroup} makes sure that the introductory remarks become a “chapter” instead of a “part”.
\item The inner one groups the \texttt{frontmatter}\footnote{2} and makes the preface of the book a section-level construct. Note that here the \texttt{display=flow} on the \texttt{omgroup} environment prevents numbering as is traditional for prefaces.
\end{itemize}

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

\footnote{1}{We cannot patch the document environment to accept an optional argument, since other packages we load already do; pity.}
\footnote{2}{We shied away from redefining the \texttt{frontmatter} to induce a \texttt{blindomgroup}, but this may be the “right” way to go in the future.}
2.3 Ignoring Inputs

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 \texttt{omdoc} class or \texttt{package}. But in the generated OMDoc result, the body is marked up with a \texttt{ignore} element. This is useful in two situations. For

\textbf{editing} One may want to hide unfinished or obsolete parts of a document.

\textbf{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 \texttt{[Koh16c]} we use the \texttt{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 \texttt{inlinedef}.

2.4 Structure Sharing

\texttt{\textbackslash STRlabel} The \texttt{\textbackslash STRlabel} macro takes two arguments: a label and the content and stores the content for later use by \texttt{\textbackslash STRcopy\lbrace URL\rbrace\lbrace label\rbrace}, which expands to the previously stored content. If the \texttt{\textbackslash STRlabel} macro was in a different file, then we can give a URL \texttt{(URL)} that lets \LaTeX generate the correct reference.

\texttt{\textbackslash STRcopy} The \texttt{\textbackslash STRlabel} macro has a variant \texttt{\textbackslash 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.\footnote{EdNote: document LMID und LMXRef here if we decide to keep them.}

\section{Global Variables}

Text fragments and modules can be made more re-usable by the use of global variables. For instance, the admin section of a course can be made course-independent (and therefore re-usable) by using variables (actually token registers) \texttt{courseAcronym} and \texttt{courseTitle} instead of the text itself. The variables can then be set in the \TeX\ preamble of the course notes file:\protect\footnote{\texttt{\setSGvar{⟨vname⟩}{⟨text⟩}} to set the global variable \texttt{⟨vname⟩} to \texttt{⟨text⟩} and \texttt{\useSGvar{⟨vname⟩}} to reference it.}
\begin{verbatim}
\setSGvar{⟨vname⟩}{⟨text⟩}
\end{verbatim}
\begin{verbatim}
\useSGvar{⟨vname⟩}
\end{verbatim}

With \texttt{\ifSGvar} we can test for the contents of a global variable: the macro call \texttt{\ifSGvar{⟨vname⟩}{⟨val⟩}{⟨ctext⟩}} tests the content of the global variable \texttt{⟨vname⟩}, only if (after expansion) it is equal to \texttt{⟨val⟩}, the conditional text \texttt{⟨ctext⟩} is formatted.

\section{Colors}

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

\section{Limitations}

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

1. when option \texttt{book} which uses \texttt{\pagestyle{headings}} is given and semantic macros are given in the \texttt{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 using the \texttt{kvoptions} package for key/value options handling. For \texttt{omdoc.cls} this is quite simple. We have options report and book, which set the \texttt{\omdoc\@cls@class} macro and pass on the macro to \texttt{omdoc.sty} for further processing.

\begin{verbatim}
\RequirePackage{etoolbox}
\RequirePackage{kvoptions}
\SetupKeyvalOptions{family=omdoc\@cls,prefix=omdoc\@cls@}
\DeclareStringOption*[article]{class}
\AddToKeyvalOption*[\omdoc@cls@class]{class}
\PassOptionsToPackage{\omdoc@cls@class}{omdoc}
\Declarerelationoption{report}{\omdoc@cls@class}{report}
\ClassWarning{omdoc}{the option 'report' is deprecated, use 'class=report', instead}
\Declarerelationoption{book}{\omdoc@cls@class}{book}
\ClassWarning{omdoc}{the option 'part' is deprecated, use 'class=book', instead}
\Declarerelationoption{bookpart}{\omdoc@cls@class}{book}
\PassOptionsToPackage{topsect=chapter}{omdoc}
\ClassWarning{omdoc}{the option 'bookpart' is deprecated, use 'class=book,topsect=chapter', instead}
\end{verbatim}

the rest of the options are only passed on to \texttt{omdoc.sty} and the class selected by the first options. We need to load the \texttt{etoolbox} package early for \texttt{\@xappto}.

\begin{verbatim}
\def\@omdoc@cls@docopt{}
\DeclareDefaultOption{\ifx\@omdoc@cls@docopt\@empty\else\xappto\@omdoc@cls@docopt{\CurrentOption}\fi}
\PassOptionsToPackage{\CurrentOption}{omdoc}
\PassOptionsToPackage{\CurrentOption}{stex}
\ProcessKeyvalOptions{omdoc@cls}
\end{verbatim}

We load \texttt{article.cls}, and the desired packages. For the \LaTeX XML bindings, we make sure the right packages are loaded.

\begin{verbatim}
\LoadClass{\omdoc@cls@docopt}{\omdoc@cls@class}
\RequirePackage{omdoc}
\RequirePackage{stex}
\end{verbatim}

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

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

⟨\texttt{/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 \texttt{xxx} will just set the appropriate switches to true (otherwise they stay false).

⟨\texttt{*package}⟩
\RequirePackage{kvoptions}
\SetupKeyvalOptions{family=omdoc@sty,prefix=omdoc@sty@}
\DeclareBoolOption{mh}
\DeclareStringOption[article]{class}
\DeclareBoolOption{showignores}
\DeclareStringOption[section]{topsect}
\newcount\section@level
\DeclareDefaultOption{\PassOptionsToPackage{\CurrentOption}{sref}}
\ProcessKeyvalOptions{omdoc@sty}

Then we need to set up the packages by requiring the \texttt{sref} package to be loaded.

\ifomdoc@sty@mh\RequirePackage{omdoc-mh}\fi
\RequirePackage{sref}
\RequirePackage{xspace}
\RequirePackage{comment}
\RequirePackage{pathsuris}

\section@level

Finally, we set the \texttt{\section@level} macro that governs sectioning. The default is two (corresponding to the \texttt{article} class), then we set the defaults for the standard classes \texttt{book} and \texttt{report} and then we take care of the levels passed in via the \texttt{topsect} option.

\section@level=2
\ifdefstring{\omdoc@sty@class}{book}{\section@level=0}{}
\ifdefstring{\omdoc@sty@class}{report}{\section@level=0}{}
\ifdefstring{\omdoc@sty@topsect}{part}{\section@level=0}{}
\ifdefstring{\omdoc@sty@topsect}{chapter}{\section@level=1}{}

³\textit{EdNote}: faking \texttt{documentkeys} for now. @HANG, please implement
5.2 Document Structure

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

\begin{verbatim}
\currentsectionlevel For the \texttt{\currentsectionlevel} and \texttt{\Currentsectionlevel} macros we use an internal macro \texttt{\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 \texttt{omdoc\_currentsectionlevel}, which will be instantiated by CSS later.4

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

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

\begin{verbatim}
\newcommand{\omgroup@nonum}{\current@section@level}{\Currentsectionlevel}\
\ifx\hyper@anchor\@undefined\else\phantomsection\fi\
\addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}\
\end{verbatim}

\texttt{\omgroup@num} convenience macro: \texttt{\omgroup@num{⟨level⟩}{⟨title⟩}} makes numbered sectioning with title \texttt{⟨title⟩} at level \texttt{⟨level⟩}. We have to check the \texttt{short} key was given in the \texttt{omgroup} environment and – if it is use it. But how to do that depends on whether the \texttt{rdfmeta} package has been loaded. In the end we call \texttt{\sref@label@id} to enable crossreferencing.

\begin{verbatim}
\edef\@@ID{\sref@id}\
\ifx\omgroup@short\@empty% no short title\n\@nameuse{#1}{#2}%\n\else% we have a short title\n\@ifundefined{rdfmeta@sectioning}%\n{\@nameuse{#1}{\omgroup@short}{#2}}%\n{\@nameuse{rdfmeta@#1@old}{\omgroup@short}{#2}}%\n\fi%\n\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\@@ID\
\end{verbatim}

\begin{verbatim}
\def\@true{true}\
\def\@false{false}
\end{verbatim}

\begin{verbatim}
\edef\@@ID{\sref@id}\
\ifx\omgroup@short\@empty% no short title\n\@nameuse{#1}{#2}%\n\else% we have a short title\n\@ifundefined{rdfmeta@sectioning}%\n{\@nameuse{#1}{\omgroup@short}{#2}}%\n{\@nameuse{rdfmeta@#1@old}{\omgroup@short}{#2}}%\n\fi%\n\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\@@ID\
\end{verbatim}

\texttt{\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.

Then we define a helper macro that takes care of the sectioning magic. It comes with its own key/value interface for customization.

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.\footnote{EdNote: MK: the extension sms is hard-coded here, but should not be. This will not work in multilingual settings.}
now the \texttt{omgroup} environment itself. This takes care of the table of contents via the helper macro above and then selects the appropriate sectioning command from \texttt{article.cls}.

\begin{verbatim}
\newenvironment{omgroup} % keys, title
{\metasetkeys{omgroup}{#1}\sref@target%}
{\ifx\omgroup@display\st@flow@@numfalse\fi
 \if@mainmatter\else@@numfalse\fi
}

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

\begin{verbatim}
\ifx\omgroup@loadmodules\@true%
\omgroup@redefine@addtocontents{%\@ifundefined{mod@id}\used@modules%
 {\@ifundefined{module@mod@id @path}{\used@modules}mod@id}}%
\fi%
\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}}%
{\advance\section@level by -1}
\end{verbatim}

5.3 Front and Backmatter

Index markup is provided by the \texttt{omtext} package \cite{Koh16b}, so in the \texttt{omdoc} package we only need to supply the corresponding \texttt{\printindex} command, if it is not already defined

\begin{verbatim}
\printindex
\providecommand\printindex{\IfFileExists{\jobname.ind}{\input{\jobname.ind}}{}}
\end{verbatim}

\verbatimائلpoint{frontmatter} some classes (e.g. \texttt{book.cls}) already have \texttt{frontmatter}, \texttt{mainmatter}, and \texttt{backmatter} macros. As we want to define \texttt{frontmatter} and \texttt{backmatter} environments, we save their behavior (possibly defining it) in \texttt{orig@*matter} macros and make them undefined (so that we can define the environments).

\begin{verbatim}
\verbatimائلpoint{(*cls)}
\ifsodef{frontmatter} to redefine if necessary
{\cslet{orig@frontmatter}{{frontmatter}\cslet{frontmatter}{\relax}}
 {\cslet{orig@frontmatter}{{clearpage}\@mainmatterfalse\pagenumbering{roman}}}
\ifsodef{backmatter} to redefine if necessary
\end{verbatim}

now we can define the environments

\newenvironment{frontmatter}{\orig@frontmatter}{\ifcsdef{mainmatter}{}{\clearpage\@mainmattertrue\pagenumbering{arabic}}}\\
\newenvironment{backmatter}{\orig@backmatter}{\ifcsdef{mainmatter}{}{\clearpage\@mainmattertrue\pagenumbering{arabic}}}\\
finally, we make sure that page numbering is arabic.

\pagenumbering{arabic}\\

5.4 Ignoring Inputs

\begin{tabbing}
\\ignore\end{tabbing}

5.5 Structure Sharing

\begin{tabbing}
\\STRLab\end{tabbing}

\begin{tabbing}
\\STRCopy\end{tabbing}

\footnote{\textbf{EdNote}: The following is simply copied over from the \texttt{latexml} package, which we eliminated, we should integrate better.}

\footnote{\textbf{EdNote}: MK: we need to do something about the ref!}
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}

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

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

5.6 Global Variables

\setSGvar set a global variable
\newcommand{\setSGvar}[1]{\@namedef{sTeX@Gvar@#1}}

\useSGvar use a global variable
\newcommand{\useSGvar}[1]{\@nameuse{sTeX@Gvar@#1}}

\ifSGvar set a global variable
\newcommand{\ifSGvar}[3]{\def\@test{#2}\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}

5.7 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}}
\def{\magenta#1\textcolor{magenta}{#1}}
\def{\brown#1\textcolor{brown}{#1}}
\def{\yellow#1\textcolor{yellow}{#1}}
\def{\orange#1\textcolor{orange}{#1}}
### Change History

<table>
<thead>
<tr>
<th>Version</th>
<th>Change Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>v0.1</td>
<td>General: First Version</td>
<td>1</td>
</tr>
<tr>
<td>v0.2</td>
<td>General: added OMDoc class</td>
<td>1</td>
</tr>
<tr>
<td>v0.3</td>
<td>General: moved omtext and friends here from the statements package</td>
<td>1</td>
</tr>
<tr>
<td>v0.4</td>
<td>General: added quotes</td>
<td>1</td>
</tr>
<tr>
<td>v0.5</td>
<td>General: more package/class options</td>
<td>1</td>
</tr>
<tr>
<td>v0.7</td>
<td>General: giving keyval arguments to the document environment</td>
<td>1</td>
</tr>
<tr>
<td>v1.0</td>
<td>General: separated out \texttt{omtext.dtx}</td>
<td>1</td>
</tr>
<tr>
<td>v1.1</td>
<td>General: integrated \texttt{etoolbox} package</td>
<td>1</td>
</tr>
<tr>
<td>v1.2</td>
<td>General: front/backmatter</td>
<td>1</td>
</tr>
<tr>
<td>v1.3</td>
<td>General: changed to keyval class/package options, allowed arbitrary classes</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>global variables and switches</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>removing keyval arg from document in favor of \texttt{documentkeys} macro</td>
<td>1</td>
</tr>
</tbody>
</table>
References


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