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

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

January 28, 2012

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.

*Version v1.0 (last revised 2012/01/28)

Contents

1 Introduction

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.

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]

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][3]

EdNote:1
EdNote:2
EdNote:3

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 sTEX sources. The rest of the documentation pertains to the functionality introduced by `omdoc.sty`.

2.1 Package and Class Options

`noindex` `extrefs`

`omdoc.sty` has the `noindex` package option, which allows to suppress the creation of index entries. The option can be set to activate multifile support, see [Koh10c] for details.

`omdoc.cls` accepts all options of the `omdoc.sty` (see Subsection 2.1) and `article.cls` and just passes them on to these. [4]

EdNote:4

2.2 Document Structure

`document`

The top-level `document` environment is augmented with an optional key/value

[1] EdNote: talk about the advantages and give an example.
[2] EdNote: is there a way to load documents at URIs in LaTeX?
[3] EdNote: integrate with latexml's XMRef in the Math mode.
[4] EdNote: describe them

argument that 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.

`id`

`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 [DUB03]; see [Koh10a] for details of the format. The `short` allows to give a short title for the generated section.

`id`
`creators`
`contributors`
`short`

`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 `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

`showignores`

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 [Koh10d] 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{label}`, which expands to the previously stored content.

`\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. [5]

EdNote:5

[5] EdNote: make an example

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 Miscellaneous

4 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 [Ste].

1. none reported yet

5 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.

5.1 Class Options

To initialize the `omdoc` class, we declare and process the necessary options.

```
⟨*cls⟩
\DeclareOption{showmeta}{\PassOptionsToPackage{\CurrentOption}{metakeys}}
\def\omdoc@class{article}
\DeclareOption{report}{\def\omdoc@class{report}\PassOptionsToPackage{\CurrentOption}{omdoc}}
\DeclareOption{book}{\def\omdoc@class{book}\PassOptionsToPackage{\CurrentOption}{omdoc}}
\DeclareOption{chapter}{\PassOptionsToPackage{\CurrentOption}{omdoc}}
\DeclareOption{part}{\PassOptionsToPackage{\CurrentOption}{omdoc}}
\DeclareOption{showignores}{\PassOptionsToPackage{\CurrentOption}{omdoc}}
\DeclareOption{extrefs}{\PassOptionsToPackage{\CurrentOption}{sref}}
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}
\ProcessOptions
⟨/cls⟩
⟨*ltxml.cls⟩
# -*- CPERL -*-
package LaTeXML::Package::Pool;
use strict;
use LaTeXML::Package;
use LaTeXML::Util::Pathname;
use Cwd qw(cwd abs_path);
DeclareOption('report',sub {PassOptions('omdoc','sty',ToString(Digest(T_CS('\CurrentOption'))))
DeclareOption('book',sub {PassOptions('omdoc','sty',ToString(Digest(T_CS('\CurrentOption'))));
DeclareOption('chapter',sub {PassOptions('omdoc','sty',ToString(Digest(T_CS('\CurrentOption')))
DeclareOption('part',sub {PassOptions('omdoc','sty',ToString(Digest(T_CS('\CurrentOption'))));
DeclareOption('showignores',sub {PassOptions('omdoc','sty',ToString(Digest(T_CS('\CurrentOption
DeclareOption('extrefs',sub {PassOptions('sref','sty',ToString(Digest(T_CS('\CurrentOption'))))
DeclareOption(undef,sub {PassOptions('article','cls',ToString(Digest(T_CS('\CurrentOption'))));
ProcessOptions();
⟨/ltxml.cls⟩
```

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

```
⟨*cls⟩
\LoadClass{\omdoc@class}
\RequirePackage{omdoc}
```

```
⟨/cls⟩
⟨*ltxml.cls⟩
LoadClass('article');
RequirePackage('sref');
⟨/ltxml.cls⟩
```

5.2 Setting up Namespaces and Schemata for LaTeXML

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

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

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

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

EdNote:6

Then we load the `omdoc` package, which we define separately in the next section so that it can be loaded separately[6]

```
⟨*ltxml.cls⟩
RequirePackage('omdoc');
⟨/ltxml.cls⟩
```

5.3 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.

```
⟨*cls⟩
\let\orig@document=\document
\srefaddidkey{document}
```

[6] EdNote: reword

```
\renewcommand{\document}[1][]{\metasetkeys{document}{#1}\orig@document}
⟨/cls⟩
⟨*ltxml.cls⟩
sub xmlBase {
  my $baseuri = LookupValue('baseuri');
  my $baselocal = LookupValue('baselocal');
  my $cdir = abs_path(cwd());
  $cdir =~ s/^$baselocal// if $baselocal;
  #DG: Make this more robust!
  my ($d,$f,$t);
  my $srcf = LookupValue('SOURCEFILE');
  if ( $srcf =~ /^(\w+):\/\//) {
    $srcf =~ s/^(\w+):\///;
  } # TODO: Hacky, do something better
  ($d, $f,$t) = pathname_split(LookupValue('SOURCEFILE'));
  $t = '' if LookupValue('cooluri');
  Tokenize($baseuri.$cdir.'/'.$f.$t); }
DefEnvironment('{document} OptionalKeyVals:omdoc',
       "<omdoc:omdoc "
      .    "?&KeyVal(#1,'id')(xml:id='&KeyVal(#1,'id')')"
      .     "(?&Tokenize(&LookupValue('SOURCEBASE'))"
      .      "(xml:id='&Tokenize(&LookupValue('SOURCEBASE')).omdoc')()) "
      .    "?&Tokenize(&LookupValue('baseuri'))"
      .    "(xml:base='&xmlBase()')()>"
     .  "#body"
      ."</omdoc:omdoc>",
  beforeDigest=> sub { AssignValue(inPreamble=>0); },
  afterDigest=> sub { $_[0]->getGullet->flush; return; });#$
⟨/ltxml.cls⟩
```

6 Implementation: OMDoc Package

6.1 Package Options

The initial setup for LaTeXML:

```
⟨*ltxml.sty⟩
package LaTeXML::Package::Pool;
use strict;
use LaTeXML::Package;
use Cwd qw(cwd abs_path);
⟨/ltxml.sty⟩
```

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).[7]

EdNote:7

```
⟨*package⟩
\DeclareOption{showmeta}{\PassOptionsToPackage{\CurrentOption}{metakeys}}
```

[7] EdNote: need an implementation for LaTeXML

```
\newif\if@chapter\@chapterfalse
\newif\if@part\@partfalse
\newcount\section@level\section@level=3
\newif\ifshow@ignores\show@ignoresfalse
\def\omdoc@class{article}
\DeclareOption{report}{\def\omdoc@class{report}\section@level=2}
\DeclareOption{book}{\def\omdoc@class{book}\section@level=1}
\DeclareOption{chapter}{\section@level=2\@chaptertrue}
\DeclareOption{part}{\section@level=1\@chaptertrue\@parttrue}
\DeclareOption{showignores}{\show@ignorestrue}
\DeclareOption{extrefs}{\PassOptionsToPackage{\CurrentOption}{sref}}
\ProcessOptions
⟨/package⟩
⟨*ltxml.sty⟩
DeclareOption('report','');
DeclareOption('book','');
DeclareOption('chapter','');
DeclareOption('part','');
DeclareOption('showignores','');
DeclareOption('extrefs','');
⟨/ltxml.sty⟩
```

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

```
⟨*package⟩
\RequirePackage{sref}
\RequirePackage{xspace}
\RequirePackage{comment}
⟨/package⟩
⟨*ltxml.sty⟩
RequirePackage('sref');
RequirePackage('xspace');
RequirePackage('omtext');
⟨/ltxml.sty⟩
```

6.2 Document Structure

The structure of the document is given by the `omgroup` environment just like in OMDoc. The hierarchy is adjusted automatically[8]

EdNote:8

`\currentsectionlevel`

```
⟨*package⟩
\def\level@section#1{\ifcase#1\or{part}\or{chapter}\or{section}\or{subsection}\or{subsubsection
\def\Level@Section#1{\ifcase#1\or{Part}\or{Chapter}\or{Section}\or{Subsection}\or{Subsubsection
\def\in@level@section#1{\ifcase#1\or{this document}\or{part}\or{chapter}\or{section}\or{subsect
\def\In@Level@Section#1{\ifcase#1\or{This document}\or{Part}\or{Chapter}\or{Section}\or{Subsect
\def\currentsectionlevel{\in@level@section\section@level\xspace}
```

[8] EdNote: maybe define the toplevel according to a param, need to know how to detect that the chapter macro exists.

```
\def\CurrentSectionLevel{\In@Level@Section\section@level\xspace}
⟨/package⟩
⟨*ltxml.sty⟩
⟨/ltxml.sty⟩
```

\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.

```
⟨*package⟩
\srefaddidkey{omgroup}
\addmetakey{omgroup}{creators}
\addmetakey{omgroup}{date}
\addmetakey{omgroup}{contributors}
\addmetakey{omgroup}{type}
\addmetakey*{omgroup}{short}
\addmetakey*{omgroup}{display}
\def\at@begin@omgroup#1{}
\newenvironment{omgroup}[2][]% keys, title
{\metasetkeys{omgroup}{#1}\sref@target%
\ifx\omgroup@display\st@flow\noindent{\Large\textbf{#2}\\[.3ex]\noindent\ignorespaces}%
\else%
\if@part\ifnum\section@level=1\part{#2}\sref@label@id{Part \thepart}\fi\fi%
\if@chapter\ifnum\section@level=2\chapter{#2}\sref@label@id{Chapter \thechapter}\fi\fi%
\ifnum\section@level=3\section{#2}\sref@label@id{Section \thesection}\fi%
\ifnum\section@level=4\subsection{#2}\sref@label@id{Subsection \thesubsection}\fi%
\ifnum\section@level=5\subsubsection{#2}\sref@label@id{Subsubsection \thesubsubsection}\fi%
\ifnum\section@level=6\paragraph{#2}\sref@label@id{this paragraph}\fi%
\ifnum\section@level=7\subparagraph{#2}\sref@label@id{this subparagraph}\fi%
\at@begin@omgroup\section@level% for customization
\advance\section@level by 1%
\fi}{\advance\section@level by -1}
⟨/package⟩
⟨*ltxml.sty⟩
DefEnvironment('{omgroup} OptionalKeyVals:omgroup {}',
               "<omdoc:omgroup layout='sectioning'"
               .   "?&KeyVal(#1,'id')(xml:id='&KeyVal(#1,'id')')()"
               .   "?&KeyVal(#1,'type')(type='&KeyVal(#1,'type')')()>\n"
          . "<dc:title>#2</dc:title>\n"
       . "#body\n"
     . "</omdoc:omgroup>");
⟨/ltxml.sty⟩
```

6.3 Front and Backmatter

Index markup is provided by the omtext package [Koh10b], so in the omdoc package we only need to supply the corresponding \printindex command, if it is not already defined

\printindex

```
⟨*package⟩
```

```
\providecommand\printindex{\IfFileExists{\jobname.ind}{\input{\jobname.ind}}{}}
⟨/package⟩
⟨*ltxml.sty⟩
DefConstructor('\printindex','<omdoc:index/>');
⟨/ltxml.sty⟩
```

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

```
⟨*ltxml.sty⟩
DefConstructor('\tableofcontents',"<omdoc:tableofcontents level='&ToString(&CounterValue('tocde
⟨/ltxml.sty⟩
```

The case of the \bibliography command is similar

\bibliography

```
⟨*ltxml.sty⟩
DefConstructor('\bibliography{}',"<omdoc:bibliography files='#1'/>");
⟨/ltxml.sty⟩
```

6.4 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
⟨/package⟩
⟨*ltxml.sty⟩
DefKeyVal('ignore','type','Semiverbatim');
DefKeyVal('ignore','comment','Semiverbatim');
DefEnvironment('{ignore} OptionalKeyVals:ignore',
               "<omdoc:ignore  %&KeyVals(#1)>#body</omdoc:ignore>");
⟨/ltxml.sty⟩
```

6.5 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.

```
⟨*package⟩
\long\def\STRlabel#1#2{\STRlabeldef{#1}{#2}{#2}}
⟨/package⟩
⟨*ltxml.sty⟩
DefConstructor('\STRlabel{}{}', sub {
```

```
  my($document,$label,$object)=@_;
  $document->absorb($object);
  $document->addAttribute('xml:id'=>ToString($label)) if $label; });
⟨/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.

```
⟨*package⟩
\def\STRcopy#1{\expandafter\ifx\csname STR@#1\endcsname\relax
\message{STR warning: reference #1 undefined!}
\else\csname STR@#1\endcsname\fi}
⟨/package⟩
⟨*ltxml.sty⟩
DefConstructor('\STRcopy{}',"<omdoc:ref xref='##1'/>");
⟨/ltxml.sty⟩
```

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

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

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

```
⟨*package⟩
\def\STRlabeldef#1{\expandafter\gdef\csname STR@#1\endcsname}
⟨/package⟩
⟨*ltxml.sty⟩
DefMacro('\STRlabeldef{}{}', "");
⟨/ltxml.sty⟩
```

6.6 Colors

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

```
⟨*package⟩
\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}}
```

243 ⟨/package⟩

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

```
244 ⟨*ltxml.sty⟩
245 sub omdocColorMacro {
246   my ($color, @args) = @_;
247   my $tok_color = TokenizeInternal($color);
248   (T_BEGIN, T_CS('\@omdoc@color'), T_BEGIN, $tok_color->unlist,
249    T_END, T_CS('\@omdoc@color@content'), T_OTHER('['), $tok_color->unlist, T_OTHER(']'),
250    T_BEGIN, $args[1]->unlist, T_END, T_END); }
251 DefMacro('\@omdoc@color{}', sub { MergeFont(color=>$_[1]->toString); return; });#$
252 ⟨/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 ?

```
253 ⟨*ltxml.sty⟩
254 DefConstructor('\@omdoc@color@content[]{}',
255   "?#isMath(#2)(<ltx:text ?#1(style='color:#1')()>#2</ltx:text>)");
256 foreach my $color(qw(black gray blue red green cyan magenta brown yellow orange)) {
257   DefMacro("\\".$color.'{}', sub { omdocColorMacro($color, @_); }); }#$
258 ⟨/ltxml.sty⟩
```

6.7 LaTeX Commands we interpret differently

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

```
259 ⟨*ltxml.sty⟩
260 DefConstructor('\newpage','');
261 ⟨/ltxml.sty⟩
```

6.8 Providing IDs for OMDoc Elements

To provide default identifiers, we tag all OMDoc elements that allow `xml:id` attributes by executing the `numberIt` procedure below.

```
262 ⟨*ltxml.sty⟩
263 Tag('omdoc:ignore',afterOpen=>\&numberIt,afterClose=>\&locateIt);
264 Tag('omdoc:ref',afterOpen=>\&numberIt,afterClose=>\&locateIt);
265 ⟨/ltxml.sty⟩
```

6.9 Miscellaneous

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

```
266 ⟨*package⟩
267 \newcommand\hateq{\ensuremath{\hat=}\xspace}
268 \newcommand\hatequiv{\ensuremath{\hat\equiv}\xspace}
```

```
\newcommand\textleadsto{\ensuremath{\leadsto}\xspace}
⟨/package⟩
⟨*ltxml.sty⟩
DefConstructor('\hateq',"\x{2259}");
DefConstructor('\hatequiv',"\x{2A6F}");
DefConstructor('\textleadsto',"\x{219D}");
⟨/ltxml.sty⟩
```

6.10 Leftovers

```
⟨*package⟩
\newcommand{\baseURI}[2][]{}
⟨/package⟩
⟨*ltxml.sty⟩
DefMacro('\baseURI []Semiverbatim', sub {
  if (LookupValue('SOURCEFILE')!~/^(\w+):\/\//) {
    my $baselocal = ToString(Expand($_[1]));
    $baselocal = abs_path($baselocal) unless $baselocal=~/^(\w+):\/\//;
    AssignValue('baselocal'=>$baselocal);
    AssignValue('baseuri'=>ToString(Expand($_[2])));}
  else {
    AssignValue('baselocal'=>undef);
    AssignValue('baseuri'=>ToString(Expand($_[2])));
  }});
DefConstructor('\url Semiverbatim',"<omdoc:link href='#1'>#1</omdoc:link>");
DefConstructor('\href Semiverbatim {}',"<omdoc:link href='#1'>#2</omdoc:link>");
⟨/ltxml.sty⟩
```

EdNote:9

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

```
⟨ltxml.sty | ltxml.cls⟩1;
```

[9] 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

[DUB03] 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`.

[Koh10a] Michael Kohlhase. *`dcm.sty`: An Infrastructure for marking up Dublin Core Metadata in LaTeX documents*. Self-documenting LaTeX package. Comprehensive TeX Archive Network (CTAN), 2010. URL: `http://www.ctan.org/tex-archive/macros/latex/contrib/stex/dcm/dcm.pdf`.

[Koh10b] Michael Kohlhase. *`omtext`: Semantic Markup for Mathematical Text Fragments in LaTeX*. Self-documenting LaTeX package. Comprehensive TeX Archive Network (CTAN), 2010. URL: `http://www.ctan.org/tex-archive/macros/latex/contrib/stex/omtext/omtext.pdf`.

[Koh10c] Michael Kohlhase. *`sref.sty`: Semantic Crossreferencing in LaTeX*. Self-documenting LaTeX package. Comprehensive TeX Archive Network (CTAN), 2010. URL: `http://www.ctan.org/tex-archive/macros/latex/contrib/stex/sref/sref.pdf`.

[Koh10d] Michael Kohlhase. *`statements.sty`: Structural Markup for Mathematical Statements*. Self-documenting LaTeX package. Comprehensive TeX Archive Network (CTAN), 2010. URL: `http://www.ctan.org/tex-archive/macros/latex/contrib/stex/statements/statements.pdf`.

[Ste] *Semantic Markup for LaTeX*. Project Homepage. URL: `http://trac.kwarc.info/sTeX/` (visited on 02/22/2011).