

smglom.cls/sty: Semantic Multilingual Glossary for Math

Michael Kohlhase
FAU Erlangen-Nürnberg
<http://kwarc.info/kohlhase>

March 20, 2019

Abstract

The `smglom` 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 glossary entries.

Contents

1	Introduction	2
2	The User Interface	2
2.1	Package and Class Options	2
2.2	Convenience Macros for SMGloM Modules	2
2.3	Terminological Relations	2
2.4	Namespaces and Alignments	2
3	Implementation: The SMGloM Class	4
3.1	Class Options	4
3.2	Convenience Macros for SMGloM Modules	4
3.3	Terminological Relations	6
3.4	Namespaces and Alignments	6
3.5	For Language Bindings	6
3.6	Authoring States, etc	7
3.7	Shadowing of repositories	7

1 Introduction

2 The User Interface

2.1 Package and Class Options

`smglom.cls` accepts all options of the `omdoc.cls` and `article.cls` and just passes them on to these.

2.2 Convenience Macros for SMGloM Modules

EdN:1

1

2.3 Terminological Relations

EdN:2

2

2.4 Namespaces and Alignments

EdN:3

`\symalign`

³ In SMGloM, we often want to align the content of glossary modules to formalizations, e.g. to take advantage of type declarations there. The `\symalign` macro takes two regular arguments: the first is the name symbol declared in the current module (e.g. by a `\sympi`), and the second the URI name of a symbol in an external theory in the form $\langle theory \rangle ? \langle name \rangle$.

`\namespace`

As full MMT URIs are of the form $\langle URI \rangle ? \langle theory \rangle ? \langle name \rangle$, we need a way to specify the $\langle URI \rangle$. We adopt the system of **namespaces** of in MMT: the macro declares a namespace URI. If the optional argument is given, then this is a namespace abbreviation declaration, which can be used later, for instance in `\symalign` that takes an optional first argument: the namespace of the external theory.

`\modalign`

The situation below is typical. We first declare the namespace abbreviation `sets` and then use the `\modalign` macro to specify that the external theory `sets:?ESet` is the default alignment target, i.e. any symbol that in the local `emptyset` module is aligned by default to the symbol with the same name in the external `sets:?ESet` theory.

```
\begin{modsig}[creators=miko]{emptyset}
  \gimport{set}
  \namespace[sets]{http://mathhub.info/MitM/smgloM/sets}
  \modalign[sets]{ESet}

  \symdef{eset}{\emptyset}
```

¹EDNOTE: document them

²EDNOTE: document them

³EDNOTE: MK: maybe this should go into some other module; it seems awfully foundational.

```

\symi{non-empty}
\symalign{non-empty}{ESet?non_empty}
\end{modsig}

```

The default alignment breaks down for the symbol **non-empty**, so we specify an alignment to the symbol **Eset?non_empty** via `\symalign`.

3 Implementation: The SMGloM Class

3.1 Class Options

To initialize the `smglom` class, we pass on all options to `omdoc.cls` as well as the `stex` and `smglom` packages.

```

1 <*cls>
2 \DeclareOption*{\PassOptionsToClass{\CurrentOption}{omdoc}}
3                                     \PassOptionsToPackage{\CurrentOption}{stex}
4                                     \PassOptionsToPackage{\CurrentOption}{smglom}}
5 \ProcessOptions

```

We load `omdoc.cls`, the `smglom` package that provides the SMGloM-specific functionality⁴, and the `stex` package to allow OMDoc compatibility.

```

6 \LoadClass{omdoc}
7 \RequirePackage{smglom}
8 \RequirePackage{stex}
9 \RequirePackage{amstext}
10 \RequirePackage{amsfonts}
11 </cls>

```

Now we do the same thing for the package; first the options, which we just pass on to the `stex` package.

```

12 <*sty>
13 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{statements}}
14                                     \PassOptionsToPackage{\CurrentOption}{structview}
15                                     \PassOptionsToPackage{\CurrentOption}{smultiling}}
16 \ProcessOptions

```

We load `omdoc.cls`, and the desired packages. For the L^AT_EXML bindings, we make sure the right packages are loaded.

```

17 \RequirePackage{statements}
18 \RequirePackage[langfiles]{smultiling}
19 \RequirePackage{structview}

```

3.2 Convenience Macros for SMGloM Modules

`\gimport` Just a shortcut, we have a starred and unstarred version, the first one is conservative. For example, if we execute:

```
\gimport[smglom/numberfields]{naturalnumbers}
```

First we are redirected to `\@gimport@nostar`, we store the `smglom/numberfields` (*the repo's path*) in `\@test`, then store `\mh@currentrepos` (*current directory*) in `\mh@repos`. If no repo's path is offered, that means the module to import is

⁴EDNOTE: MK:describe that above

under the same directory, so we let `repos=\mh@repos` and pass bunch of parameters to `\importmhmodule`, which is defined in `module.sty`. If there's a repo's path, then we let `repos=\langle the repo's path \rangle`. Finally we use `\mhcurrentrepos` (defined in `module.sty`) to change the `\mh@currentrepos`.

```

20 \def\gimport{\@ifstar\@gimport@star\@gimport@nostar}%
21 \newrobustcmd\@gimport@star[2] [] {%
22   \def\@test{#1}%
23   \edef\mh@@repos{\mh@currentrepos}%
24   \ifx\@test\@empty%
25     \importmhmodule[conservative, repos=\mh@@repos, ext=tex, path=#2]{#2}%
26   \else%
27     \importmhmodule[conservative, repos=#1, ext=tex, path=#2]{#2}%
28   \fi%
29   \mhcurrentrepos{\mh@@repos}%
30   \ignorespacesandpars%
31 }%
32 \newrobustcmd\@gimport@nostar[2] [] {%
33   \def\@test{#1}%
34   \edef\mh@@repos{\mh@currentrepos}%
35   \ifx\@test\@empty%
36     \importmhmodule[repos=\mh@@repos, ext=tex, path=#2]{#2}%
37   \else%
38     \importmhmodule[repos=#1, ext=tex, path=#2]{#2}%
39   \fi%
40   \mhcurrentrepos{\mh@@repos}%
41   \ignorespacesandpars%
42 }%

```

guse just a shortcut

```

43 \newrobustcmd\guse[2] [] {\def\@test{#1}%
44   \edef\mh@@repos{\mh@currentrepos}%
45   \ifx\@test\@empty%
46     \usemhmodule[repos=\mh@@repos, ext=tex, path=#2]{#2}%
47   \else%
48     \usemhmodule[repos=#1, ext=tex, path=#2]{#2}%
49   \fi%
50   \mhcurrentrepos{\mh@@repos}%
51   \ignorespacesandpars%
52 }%

```

gstructure we essentially copy over the definition of `mhstructure`, but adapt it to the SM-GloM situation.

```

53 \newenvironment{gstructure}[3] [] {\def\@test{#1}%
54   \xdef\mh@@@repos{\mh@currentrepos}%
55   \ifx\@test\@empty%
56     \gdef\@@doit{\importmhmodule[repos=\mh@@@repos, path=#3, ext=tex]{#3}}%
57   \else%
58     \gdef\@@doit{\importmhmodule[repos=#1, path=#3, ext=tex]{#3}}%
59   \fi%

```

```

60 \ifmod@show\par\noindent structure import "#2" from module #3 \@@doit\fi%
61 \ignorespacesandpars}
62 {\aftergroup\@@doit\ifmod@show end import\fi%
63 \ignorespacesandparsafterend}

```

3.3 Terminological Relations

*nym

```

64 \newrobustcmd\hypernym[3] [] {\if@importing\else\par\noindent #2 is a hypernym of #3\fi}%
65 \newrobustcmd\hyponym[3] [] {\if@importing\else\par\noindent #2 is a hyponym of #3\fi}%
66 \newrobustcmd\meronym[3] [] {\if@importing\else\par\noindent #2 is a meronym of #3\fi}%

```

EdN:5

\MSC to define the Math Subject Classification,⁵

```

67 \newrobustcmd\MSC[1] {\if@importing\else MSC: #1\fi\ignorespacesandpars}%

```

3.4 Namespaces and Alignments

\namespace

```

68 \newcommand\namespace[2] [] {\ignorespaces}

```

\modalign

```

69 \newcommand\modalign[2] [] {\ignorespaces}

```

\symalign

```

70 \newcommand\symalign[3] [] {\ignorespaces}

```

3.5 For Language Bindings

Here we adapt the `smultiling` functionality to the special situation, where the module and file names are identical by design.

gviewsig The `gviewsig` environment is just a layer over the `mhviewsig` environment with the keys suitably adapted.

```

71 \newenvironment{gviewsig}[4] [] {% keys, id, from, to
72 \def\test{#1}%
73 \ifx\@test\@empty%
74 \begin{mhviewsig}[frompath=#3,topath=#4]{#2}{#3}{#4}%
75 \else%
76 \begin{mhviewsig}[frompath=#3,topath=#4,#1]{#2}{#3}{#4}%
77 \fi%
78 \ignorespacesandpars%
79 }{%
80 \end{mhviewsig}%
81 \ignorespacesandparsafterend%
82 }%

```

⁵EdNOTE: MK: what to do for the LaTeXML side?

`gviewnl` The `gviewnl` environment is just a layer over the `mhviewnl` environment with the keys suitably adapted.

```
83 \newenvironment{gviewnl}[5][{}]{% keys, id, lang, from, to
84   \def\@test{#1}\ifx\@test\@empty%
85     \begin{mhviewnl}[frompath=#4,topath=#5]{#2}{#3}{#4}{#5}%
86   \else%
87     \begin{mhviewnl}[frompath=#4,topath=#5,#1]{#2}{#3}{#4}{#5}%
88   \fi%
89 \ignorespacesandpars%
90 }{%
91   \end{mhviewnl}%
92 \ignorespacesandparsafterend%
93 }%
```

EdN:6

```
\gincludeview 6
94 \newcommand\gincludeview[2][{}]{\ignorespacesandpars}%
```

3.6 Authoring States, etc

We add a key to the module environment.

```
95 \addmetakey{module}{state}%
```

3.7 Shadowing of repositories

`\repos@macro` `\repos@macro` parses a GitLab repository name $\langle group \rangle / \langle name \rangle$ and creates an internal macro name from that, which will be used

```
96 \def\repos@macro#1/#2;{#1@shadows@#2}%
```

`\shadow` `\shadow{\langle orig \rangle}{\langle fork \rangle}` declares a that the private repository $\langle fork \rangle$ shadows the MathHub repository $\langle orig \rangle$. Internally, it simply defines an internal macro with the shadowing information.

```
97 \def\shadow#1#2{\@namedef{\repos@macro#1;}{#2}}%
```

`\MathHubPath` `\MathHubPath{\langle repos \rangle}` computes the path of the fork that shadows the MathHub repository $\langle repos \rangle$ according to the current `\shadow` specification. The computed path can be used for loading modules from the private version of $\langle repos \rangle$.

```
98 \def\MathHubPath#1{\@ifundefined{\repos@macro#1;}{#1}{\@nameuse{\repos@macro#1;}}}%
99 \</sty>
```

⁶EDNOTE: This is fake for now, needs to be implemented and documented