`smglom.cls/sty`: Semantic Multilingual Glossary for Math

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

October 11, 2019

Abstract

The `smglom` package and class are 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

We use STEX as the surface language for the SMGLoM (Semantic Multilingual Glossary of Mathematics), see [Gin+16; Koh14; SMG]. The `smglom` package and class provides some infrastructure to make this more convenient.

2 The User Interface

The `smglom` package provides convenience macros on top of the STEX infrastructure to simplify writing SMGLoM glossary modules and make them more concise for reading.The `smglom` class just sets up the necessary STEX packages and loads the `smglom` package.

2.1 Package and Class Options

`smglom.sty` accepts all options of the STEX package and passes them along to `stex.sty`. `smglom.cls` also does that for the casses `omdoc.cls` and `article.cls`.

2.2 Convenience Macros for SMGLoM Modules

The SMGLoM source files are more regular than arbitrary STEX files. In particular,
- make heavy use of the `smultiling` package for multilingual STEX,
- use the `mathhub` extensions to STEX for file system organization,
- enforce the one-module-one-file convention and make sure that the module name must be the same as the (base name) of the file.

This allows use to abbreviate e.g.

  `\importmhmodule[mhrepos=lib/archive,path=current/modfile]{modname}`

`\gimport`
`\guse`

by`\gimport[lib/archive]{modname}` and analogously for `\guse`. [1]

EdN: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 `\symi`), and the second the URI name of a symbol in an external theory in the form $\langle theory\rangle?\langle name\rangle$.

---

[1]EdNote: document them
[2]EdNote: document them
[3]EdNote: MK: maybe this should go into some other module; it seems awfully foundational.

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 `\namespace` 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.

The situation below is typical. We first declare the namespace abbreviation `\modalign` `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}
  \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 functionality[4], 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}{dcm}
15                             \PassOptionsToPackage{\CurrentOption}{cmath}
16                             \PassOptionsToPackage{\CurrentOption}{structview}
17                             \PassOptionsToPackage{\CurrentOption}{smultiling}}
18 \ProcessOptions
```

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

```
19 \RequirePackage{statements}
20 \RequirePackage[langfiles]{smultiling}
21 \RequirePackage{structview}
22 \RequirePackage{dcm}
23 \RequirePackage{cmath}
```

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}
```

---

[4] EDNOTE: MK:describe that above

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 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=⟨*the repo's path*⟩. Finally we use `\mhcurrentrepos`(defined in `module.sty`) to change the `\mh@currentrepos`.

```
24 \def\gimport{\@ifstar\@gimport@star\@gimport@nostar}%
25 \newrobustcmd\@gimport@star[2][]{%
26   \def\@test{#1}%
27   \edef\mh@@repos{\mh@currentrepos}%
28   \ifx\@test\@empty%
29     \importmhmodule[conservative,repos=\mh@@repos,ext=tex,path=#2]{#2}%
30   \else%
31     \importmhmodule[conservative,repos=#1,ext=tex,path=#2]{#2}%
32   \fi%
33   \mhcurrentrepos{\mh@@repos}%
34   \ignorespacesandpars%
35 }%
36 \newrobustcmd\@gimport@nostar[2][]{%
37   \def\@test{#1}%
38   \edef\mh@@repos{\mh@currentrepos}%
39   \ifx\@test\@empty%
40     \importmhmodule[repos=\mh@@repos,ext=tex,path=#2]{#2}%
41   \else%
42     \importmhmodule[repos=#1,ext=tex,path=#2]{#2}%
43   \fi%
44   \mhcurrentrepos{\mh@@repos}%
45   \ignorespacesandpars%
46 }%
```

guse just a shortcut

```
47 \newrobustcmd\guse[2][]{\def\@test{#1}%
48   \edef\mh@@repos{\mh@currentrepos}%
49   \ifx\@test\@empty%
50     \usemhmodule[repos=\mh@@repos,ext=tex,path=#2]{#2}%
51   \else%
52     \usemhmodule[repos=#1,ext=tex,path=#2]{#2}%
53   \fi%
54   \mhcurrentrepos{\mh@@repos}%
55   \ignorespacesandpars%
56 }%
```

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

```
57 \newenvironment{gstructure}[3][]{\def\@test{#1}%
58   \xdef\mh@@@repos{\mh@currentrepos}%
59   \ifx\@test\@empty%
60     \gdef\@@doit{\importmhmodule[repos=\mh@@@repos,path=#3,ext=tex]{#3}}%
```

```
61    \else%
62      \gdef\@@doit{\importmhmodule[repos=#1,path=#3,ext=tex]{#3}}%
63    \fi%
64    \ifmod@show\par\noindent structure import "#2" from module #3 \@@doit\fi%
65    \ignorespacesandpars}
66  {\aftergroup\@@doit\ifmod@show end import\fi%
67    \ignorespacesandparsafterend}
```

3.3   Terminological Relations

```
68 \newrobustcmd\hypernym[3][]{\if@importing\else\par\noindent #2 is a hypernym of #3\fi}%
69 \newrobustcmd\hyponym[3][]{\if@importing\else\par\noindent #2 is a hyponym of  #3\fi}%
70 \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]

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

3.4   Namespaces and Alignments

\namespace

```
72 \newcommand\namespace[2][]{\ignorespaces}
```

\modalign

```
73 \newcommand\modalign[2][]{\ignorespaces}
```

\symalign

```
74 \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.

```
75 \newenvironment{gviewsig}[4][]{% keys, id, from, to
76    \def\test{#1}%
77    \ifx\@test\@empty%
78      \begin{mhviewsig}[frompath=#3,topath=#4]{#2}{#3}{#4}%
79    \else%
80      \begin{mhviewsig}[frompath=#3,topath=#4,#1]{#2}{#3}{#4}%
81    \fi%
82    \ignorespacesandpars%
83 }{%
84    \end{mhviewsig}%
```

---

[5]EDNOTE: MK: what to do for the LaTeXML side?

```
85    \ignorespacesandparsafterend%
86 }%
```

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

```
87 \newenvironment{gviewnl}[5][]{% keys, id, lang, from, to
88    \def\@test{#1}\ifx\@test\@empty%
89      \begin{mhviewnl}[frompath=#4,topath=#5]{#2}{#3}{#4}{#5}%
90    \else%
91      \begin{mhviewnl}[frompath=#4,topath=#5,#1]{#2}{#3}{#4}{#5}%
92    \fi%
93 \ignorespacesandpars%
94 }{%
95    \end{mhviewnl}%
96    \ignorespacesandparsafterend%
97 }%
```

\gincludeview   [6]

```
98 \newcommand\gincludeview[2][]{\ignorespacesandpars}%
```

3.6   Authoring States, etc

We add a key to the module environment.

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

3.7   Shadowing of repositories

\repos@macro   \repos@macro parses a GitLab repository name ⟨*group*⟩/⟨*name*⟩ and creates an internal macro name from that, which will be used

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

\shadow   \shadow{⟨*orig*⟩}{⟨*fork*⟩} declares a that the private repository ⟨*fork*⟩ shadows the MathHub repository ⟨*orig*⟩. Internally, it simply defines an internal macro with the shadowing information.

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

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

```
102 \def\MathHubPath#1{\@ifundefined{\repos@macro#1;}{#1}{\@nameuse{\repos@macro#1;}}}%
103 ⟨/sty⟩
```

---

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

References

[Gin+16] Deyan Ginev et al. "The SMGloM Project and System. Towards a Terminology and Ontology for Mathematics". In: *Mathematical Software - ICMS 2016 - 5th International Congress*. Ed. by Gert-Martin Greuel et al. Vol. 9725. LNCS. Springer, 2016. DOI: `10.1007/978-3-319-42432-3`. URL: `http://kwarc.info/kohlhase/papers/icms16-smglom.pdf`.

[Koh14] Michael Kohlhase. "A Data Model and Encoding for a Semantic, Multilingual Terminology of Mathematics". In: *Intelligent Computer Mathematics*. Conferences on Intelligent Computer Mathematics (Coimbra, Portugal, July 7–11, 2014). Ed. by Stephan Watt et al. LNCS 8543. Springer, 2014, pp. 169–183. ISBN: 978-3-319-08433-6. URL: `http://kwarc.info/kohlhase/papers/cicm14-smglom.pdf`.

[SMG] *SMGloM Git Repository*. URL: `http://gl.mathhub.info/smglom/smglom` (visited on 07/10/2013).