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

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

October 9, 2017

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 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

[1]

2.3 Terminological Relations

[2]

---

[1]EDNOTE: document them
[2]EDNOTE: document them

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[3], 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 LaTeXML 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:

$$\text{\gimport[smglom/numberfields]{naturalnumbers}}$$

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

---

[3]EDNOTE: 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=⟨*the repo's path*⟩. 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:4   \MSC   to define the Math Subject Classification, [4]

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

3.4 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.

```
68 \newenvironment{gviewsig}[4][]{%
69   \def\test{#1}%
70   \ifx\@test\@empty%
71     \begin{mhviewsig}[frompath=#3,topath=#4]{#2}{#3}{#4}%
72   \else%
73     \begin{mhviewsig}[frompath=#3,topath=#4,#1]{#2}{#3}{#4}%
74   \fi%
75   \ignorespacesandpars%
76 }{%
77   \end{mhviewsig}%
78   \ignorespacesandparsafterend%
79 }%
```

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

```
80 \newenvironment{gviewnl}[5][]{%
81   \def\@test{#1}\ifx\@test\@empty%
82     \begin{mhviewnl}[frompath=#4,topath=#5]{#2}{#3}{#4}{#5}%
83   \else%
84     \begin{mhviewnl}[frompath=#4,topath=#5,#1]{#2}{#3}{#4}{#5}%
85   \fi%
86 \ignorespacesandpars%
87 }{%
88   \end{mhviewnl}%
89   \ignorespacesandparsafterend%
90 }%
```

---

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

\gincludeview   ⁵

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

3.5   Authoring States, etc

We add a key to the module environment.

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

3.6   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

93 `\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.

94 `\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 $\langle repos \rangle$.

95 `\def\MathHubPath#1{\@ifundefined{\repos@macro#1;}{#1}{\@nameuse{\repos@macro#1;}}}%`
96 $\langle$/sty$\rangle$

---

⁵EDNOTE: This is fake for now, needs to be implemented and documented