

MathHub Support for \LaTeX^*

Michael Kohlhasse
Jacobs University, Bremen
<http://kwarc.info/kohlhasse>

November 20, 2015

Abstract

The `sref` package is part of the \LaTeX collection, a version of $\text{\TeX}/\text{\LaTeX}$ that allows to markup $\text{\TeX}/\text{\LaTeX}$ documents semantically without leaving the document format, essentially turning $\text{\TeX}/\text{\LaTeX}$ into a document format for mathematical knowledge management (MKM).

The `mathhub` packages extend \LaTeX with support for the MathHub.info portal

Contents

1	Introduction	3
2	The User Interface	3
2.1	Package Options	3
2.2	<code>modules-mh</code> : MH Variants for Modules	3
2.3	<code>omtext-mh</code> : MH Variants for OMText	4
2.4	<code>statements-mh</code> : MH Variants for Statements	4
2.5	<code>smultiling-mh</code> : MH Variants for Multilinguality	4
2.6	<code>structview-mh</code> : MH Variants for Structures and Views	4
2.7	<code>mikoslides-mh</code> : Support for MiKo Slides	5
2.8	<code>problem-mh</code> : Support for Problems	5
2.9	<code>hwexam-mh</code> : Support for Assignments	5
3	Limitations	6
4	Implementation	7
4.1	General Infrastructure	7
4.2	<code>modules-mh</code> : MH Variants for Modules	8
4.3	<code>omtext-mh</code> : MH Variants for OMText	11
4.4	<code>statements-mh</code> : MH Variants for Statements	12

*Version v1.0 (last revised 2015/11/04)

4.5	<code>smultiling-mh</code> : MH Variants for Multilinguality	13
4.6	<code>structview-mh</code> : MH Variants for Structures and Views	15
4.7	<code>mikoslides-mh</code> : Support for MiKo Slides	18
4.8	<code>problem-mh</code> : Support for Problems	19
4.9	<code>hwexam-mh</code> : Support for Assignments	20
4.10	Finale	21

1 Introduction

Much of the \LaTeX content is hosted on **MathHub** (<http://MathHub.info>), a portal and archive for flexiformal mathematics. **MathHub** offers GIT repositories (public and private escrow) for mathematical documentation projects, online and offline authoring and document development infrastructure, and a rich, interactive reading interface. The `modules` package supports repository-sensitive operations on **MathHub**.

Note that **MathHub** has two-level repository names of the form $\langle group \rangle / \langle repo \rangle$, where $\langle group \rangle$ is a **MathHub**-unique repository group and $\langle repo \rangle$ a repository name that is $\langle group \rangle$ -unique. The file and directory structure of a repository is arbitrary – except that it starts with the directory `source` because they are Math Archives in the sense of [HorIacJuc:cscpnrr11]. But this structure can be hidden from the \LaTeX author with **MathHub**-enabled versions of the \LaTeX macros, which are defined in this package.

Caveat if you want to use the **MathHub** support macros (let’s call them **mh-variants**), then every time a module is imported or a document fragment is included from another repos, the mh-variant `\importmhmodule` must be used, so that the “current repository” is set accordingly. To be exact, we only need to use mh-variants, if the imported module or included document fragment use mh-variants.

2 The User Interface

2.1 Package Options

none so far

2.2 modules-mh: MH Variants for Modules

`\importmhmodule` The `\importmhmodule` macro is a variant of `\importmodule` with repository support. Instead of writing

```
\defpath{MathHub}{/user/foo/lmh/MathHub}
\importmodule[load=\MathHub{fooMH/bar/source/baz/foobar}]{foobar}
```

we can simply write (assuming that `\MathHub` is defined as above)

```
\importmhmodule[repos=fooMH/bar,path=baz/foobar]{foobar}
```

Note that the `\importmhmodule` form is more semantic, which allows more advanced document management features in **MathHub**.

If `baz/foobar` is the “current module”, i.e. if we are on the **MathHub** path `...MathHub/fooMH/bar...`, then stating the repository in the first optional argument is redundant, so we can just use

`\importmhmodule[path=baz/foobar]{foobar}`

if no file needs to be loaded, `\importmhmodule` is the same as `\importmodule`.

`\mhcurrentrepos` Of course, neither \LaTeX nor \LaTeXML know about the repositories when they are called from a file system, so we can use the `\mhcurrentrepos` macro to tell them. But this is only needed to initialize the infrastructure in the driver file. In particular, we do not need to set it in each module, since the `\importmhmodule` macro sets the current repository automatically.

`\usemhmodule` The `\usemhmodule` is the analog to `\usemodule`.

`\mhinputref` For this, the `modules` package supplies the mh-variants `\mhinputref` and
`\mhinput` `\mhinput` of the `\inputref` macro introduced above and normal \LaTeX `\input` macro.

2.3 omtext-mh: MH Variants for OMText

`\mhgraphics` The `\mhgraphics` macro is a variant of `\mycgraphics` with repository support. Instead of writing

```
\defpath{MathHub}{/user/foo/lmh/MathHub}
\mycgraphics{MathHub{fooMH/bar/source/baz/foobar}}
```

we can simply write (assuming that `\MathHub` is defined as above)

```
\mhgraphics[fooMH/bar]{baz/foobar}
```

Note that the `\mhgraphics` form is more semantic, which allows more advanced document management features in `MathHub`.

2.4 statements-mh: MH Variants for Statements

this only provides `\usemhvocab` a variant of `\usevocab` (which might go away at some time)

2.5 smultiling-mh: MH Variants for Multilinguality

1 2

2.6 structview-mh: MH Variants for Structures and Views

3

EdN:1
EdN:2

EdN:3

2.7 mikoslides-mh: Support for MiKo Slides

`\mhframeimage` The `\mhframeimage` macro is a variant of `\frameimage` with repository support. Instead of writing

```
\defpath{MathHub}{/user/foo/lmh/MathHub}
\frameimage{\MathHub{fooMH/bar/source/baz/foobar}}
```

we can simply write (assuming that `\MathHub` is defined as above)

```
\mhframeimage[fooMH/bar]{baz/foobar}
```

Note that the `\mhframeimage` form is more semantic, which allows more advanced document management features in `MathHub`.

If `baz/foobar` is the “current module”, i.e. if we are on the `MathHub` path `...MathHub/fooMH/bar...`, then stating the repository in the first optional argument is redundant, so we can just use

```
\mhframeimage{baz/foobar}
```

2.8 problem-mh: Support for Problems

`\includemhproblem` The `\includemhproblem` macro is a variant of `\importmodule` with repository support. Instead of writing

```
\defpath{MathHub}{/user/foo/lmh/MathHub}
\includeproblem[pts=7]{\MathHub{fooMH/bar/source/baz/foobar}}
```

we can simply write (assuming that `\MathHub` is defined as above)

```
\includemhproblem[fooMH/bar]{baz/foobar}
```

Note that the `\importmhproblem` form is more semantic, which allows more advanced document management features in `MathHub`.

2.9 hwexam-mh: Support for Assignments

`\includemhassignment` The `\includemhassignment` macro is a variant of `\includeassignment` with repository support. Instead of writing

```
\defpath{MathHub}{/user/foo/lmh/MathHub}
\includeassignment[pts=7]{\MathHub{fooMH/bar/source/baz/foobar}}
```

we can simply write (assuming that `\MathHub` is defined as above)

```
\includemhassignment[fooMH/bar]{baz/foobar}
```

¹EDNOTE: needs to be documented

²EDNOTE: mhmodsig seems to be missing what happened?

³EDNOTE: needs to be documented

3 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` GitHub repository [`sTeX:github:on`].

1. none reported yet.

4 Implementation

The `sref` package generates two files: the \LaTeX package (all the code between `<*package>` and `</package>`) and the \LaTeX XML bindings (between `<*ltxml>` and `</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.

We first set up header information for the \LaTeX XML binding files in the base package.

```

1 <*ltxml | modules.ltxml | structview.ltxml | omtex.ltxml | statements.ltxml | smultiling.ltxml | mikosides.ltxml | problem
2 # -*- PERL -*-
3 package LaTeXML::Package::Pool;
4 use strict;
5 use LaTeXML::Package;
6 </ltxml | modules.ltxml | structview.ltxml | omtex.ltxml | statements.ltxml | smultiling.ltxml | mikosides.ltxml | problem
7 <package>\ProvidesPackage{mathhub}[2015/11/04 v1.0 sTeX Support for MathHub.info]

8 <*package>
9 \DeclareOption*{}
10 \ProcessOptions
11 </package>
12 <*ltxml>
13 use LaTeXML::Util::Pathname;
14 \DeclareOption(undef,sub {});
15 \ProcessOptions();
16 </ltxml>

```

Then we need to set up the packages by requiring the `metakeys` package [Kohlhase:metakeys:ctan] to be loaded (in the right version).

```

17 <*package>
18 \RequirePackage{keyval}
19 </package>
20 <*ltxml>
21 \RequirePackage('keyval');
22 </ltxml>

```

4.1 General Infrastructure

`\mhcurrentrepos` `\mhcurrentrepos` is used to initialize the current repository. If the repos has changed, it writes a call to the internal macro `\@mhcurrentrepos` for the aux file and calls it. So that the `\importmodule` calls there work with the correct repos.

```

23 <*package>
24 \newcommand\mhcurrentrepos[1]{%
25   \edef\@test{#1}%
26   \ifx\@test\mh@currentrepos% if new dir = old dir
27     \relax% no need to change
28   \else%
29     \protected@write\@auxout{}{\string\@mhcurrentrepos{#1}}%
30   \fi%
31   \@mhcurrentrepos{#1}% define mh@currentrepos

```

```

32 }%
33 \newcommand\mhcurrentrepos[1]{\edef\mh@currentrepos{#1}}%
34 \end{package}
35 \end{xml}
36 DefMacro('mhcurrentrepos{', '\mhcurrentrepos{#1}');
37 DefMacro('mh@currentrepos{', '\def\mh@currentrepos{#1}\@mhcurrentrepos{#1}');
38 DefConstructor('mh@currentrepos{', '\def\mh@currentrepos{#1}\@mhcurrentrepos{#1}');
39 afterDigest => sub{ AssignValue('current_repos', ToString($_[1]->getArg(1)), 'global'); } );
40 \end{xml}#

```

`\libinput` the `\libinput` macro inputs from the `lib` directory of the MathHub repository or the `meta-inf/lib` repos of the group.

```

41 \end{package}
42 \def\modules@@first#1/#2;{#1}
43 \newcommand\libinput[1]{\def\@libfile{\MathHub{\mh@currentrepos/lib/#1}}%
44 \IfFileExists{\@libfile}{\input\@libfile}%
45 {\edef\@group{\expandafter\modules@@first\mh@currentrepos;}
46 \edef\@inffile{\MathHub{\@group/meta-inf/lib/#1}}
47 \IfFileExists{\@inffile}{\input\@inffile}%
48 {\PackageError{modules}
49 {Library file missing, cannot input #1\MessageBreak%
50 Both \@libfile.tex\MessageBreak and \@inffile.tex\MessageBreak do not exist}%
51 {Check whether the file name is correct}}}%
52 \end{package}
53 \end{xml}
54 DefMacro('modules@@first#1/#2;', '\#1');
55 DefMacro('libinput {', sub{
56 my ($gullet, $name) = @_;
57 my $mathhub_base = ToString(Digest('\MathHub{'}));
58 my $repos = LookupValue('current_repos');
59 # file name to search for
60 $name = ToString($name);
61 #Relative paths for recursive search
62 my $reponame = substr($repos, 0, index($repos, '/'));
63 my $FIRSTLIB = $mathhub_base . $repos . '/lib' ;
64 my $SECONDLIB = $mathhub_base . $reponame . '/meta-inf/lib';
65 my $file = pathname_find($name, types => ['tex'], paths => [$FIRSTLIB]);
66 $file = pathname_find($name, types=>['tex'], paths=>[$SECONDLIB]) unless $file;
67 # Singal error if the file cannot be found
68 LaTeXML::Package::InputContent($file, noerror=>1); });
69 \end{xml}

```

4.2 modules-mh: MH Variants for Modules

We set up package options and pass them on to the `modules` package, which we also load.

```

70 \end{modules}
71 \ProvidesPackage{modules-mh}[2015/11/04 v1.0 MathHub support for the sTeX modules package]
72 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{modules}}

```



```

73 \ProcessOptions
74 \RequirePackage{modules}
75 \RequirePackage{mathhub}
76 \</modules>
77 \<*modules.ltxml>
78 DeclareOption(undef,sub{PassOptions('modules','sty',ToString(Digest(T_CS('\CurrentOption'))));
79 ProcessOptions();
80 RequirePackage('modules');
81 RequirePackage('mathhub');
82 \</modules.ltxml>

\importmhmodule The \importmhmodule[<key=value list>]{module} saves the current value of
\mh@currentrepos in a local macro \mh@@repos, resets \mh@currentrepos to
the new value if one is given in the optional argument, and after importing resets
\mh@currentrepos to the old value in \mh@@repos. We do all the \ifx compar-
ison with an \expandafter, since the values may be passed on from other key
bindings. Parameters will be passed to \importmodule.

83 \<*modules>
84 \srefaddidkey{importmhmodule}%
85 \addmetakey{importmhmodule}{repos}% saves the repo's path. E.g: smglom/numberfield
86 \addmetakey{importmhmodule}{path}% saves the module name. E.g: naturalnumbers
87 \addmetakey[sms]{importmhmodule}{ext}% saves the extension: E.g: tex
88 \addmetakey[false]{importmhmodule}{conservative}[true]%
89 \newcommand\importmhmodule[2][]{%
90   \metasetkeys{importmhmodule}{#1}%
91   \ifx\importmhmodule@path@empty% if module name is not set
92     \importmodule[ext=\importmhmodule@ext,id=\importmhmodule@id]{#2}%
93   \else%
94     \edef\mh@@repos{\mh@currentrepos}% remember so that we can reset it.
95     \ifx\importmhmodule@repos@empty% if in the same repos
96       \relax% no need to change mh@currentrepos, i.e, current dirctory.
97     \else%
98       \mhcurrentrepos{\importmhmodule@repos}% change it.
99     \fi%
100     \importmodule[load=\MathHub{\mh@currentrepos/source/\importmhmodule@path},%
101       ext=\importmhmodule@ext,id=\importmhmodule@id]{#2}%
102     \mhcurrentrepos{\mh@@repos}% after importing, reset to old value
103   \fi%
104   \ignorespaces%
105 }%
106 \</modules>
107 \<*modules.ltxml>
108 DefKeyVal('importmhmodule','id','Semiverbatim');
109 DefKeyVal('importmhmodule','repos','Semiverbatim');
110 DefKeyVal('importmhmodule','path','Semiverbatim');
111 DefKeyVal('importmhmodule','ext','Semiverbatim');
112 DefKeyVal('importmhmodule','conservative','Semiverbatim');
113 DefConstructor('\importmhmodule OptionalKeyVals:importmhmodule {}',
114   "<omdoc:imports "

```

```

115 . "from='?&GetKeyVal(#1,'load')(&canonical_omdoc_path(&GetKeyVal(#1,'load')))(###2'"
116 . "?&defined(&GetKeyVal(#1,'conservative'))(load='&GetKeyVal(#1,'conservative'))'"
117 afterDigest => \&importMHmoduleI);
118
119 sub importMHmoduleI {
120 my ($stomach, $whatsit) = @_;
121 my $keyval = $whatsit->getArg(1);
122 my $id = $whatsit->getArg(2);
123 if ($keyval) {
124 my $repos = ToString($keyval->getValue('repos'));
125 my $path = ToString($keyval->getValue('path'));
126 my $current_repos = LookupValue('current_repos');
127 if (!$repos) { # Use the implicit current repository
128 $repos = $current_repos; }
129 my $defpaths = LookupValue('defpath');
130 my $load_path = ($$defpaths{MathHub}).$repos.'/source/'.$path;
131 $keyval->setValue('load',$load_path);
132 AssignValue('current_repos' => $repos, 'global');
133 importmoduleI($stomach,$whatsit);
134 AssignValue('current_repos' => $current_repos, 'global'); }
135 else {
136 importmoduleI($stomach,$whatsit); }
137 return; }
138
139 DefConstructor('\importMHmoduleI OptionalKeyVals:importmhmodule {}', '',
140 afterDigest=> \&importMHmoduleI );#$
141 </modules.ltxml>

```

and now the analogs

\usemhmodule

```

142 <*modules>
143 \newcommand\usemhmodule[2] [] {%
144 \metasetkeys{importmhmodule}{#1}%
145 \ifx\importmhmodule@path\empty%
146 \usemodule[ext=\importmhmodule@ext,id=\importmhmodule@id]{#2}%
147 \else%
148 \edef\mh@@@repos{\mh@currentrepos}%
149 \ifx\importmhmodule@repos\empty%
150 \else%
151 \mhcurrentrepos{\importmhmodule@repos}%
152 \fi%
153 \usemodule[load=MathHub{\mh@currentrepos/source/\importmhmodule@path},ext=\importmhmodule@
154 \mhcurrentrepos\mh@@@repos%
155 \fi%
156 \ignorespaces%
157 }%
158 </modules>
159 <*modules.ltxml>
160 DefConstructor('\usemhmodule OptionalKeyVals:importmhmodule {}',

```

```

161   "<omdoc:uses from='?'&GetKeyVal{#1,'load'}(&canonical_omdoc_path(&GetKeyVal{#1,'load'}))>()###
162   afterDigest => \&importMHmoduleI);
163 </modules.ltxml>

\mhinputref
164 <modules.ltxml>RawTeX( '
165 <*modules | modules.ltxml>
166 \newcommand\mhinputref[2] []{%
167   \def\@repos{#1}%
168   \edef\mh@@repos{\mh@currentrepos}%
169   \ifx\@repos\empty%
170     \else%
171       \mhcurrentrepos{#1}%
172     \fi%
173   \inputref{\MathHub{\mh@currentrepos/source/#2}}%
174   \mhcurrentrepos\mh@@repos%
175   \ignorespaces%
176 }%
177 </modules | modules.ltxml>
178 <modules.ltxml>' );

\mhinput
179 <*modules>
180 \let\mhinput\mhinputref%
181 </modules>

```

4.3 omtex-mh: MH Variants for OMTex

We set up package options and pass them on to the `omtext` package, which we also load.

```

182 <*omtext>
183 \ProvidesPackage{omtext-mh}[2015/11/04 v1.0 MathHub support for the sTeX omtex package]
184 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{omtext}}
185 \ProcessOptions
186 \RequirePackage{mathhub}
187 \RequirePackage{omtext}
188 \RequirePackage{modules-mh}
189 </omtext>
190 <*omtext.ltxml>
191 \DeclareOption{undef,sub{PassOptions('omtext','sty',ToString(Digest(T_CS('\CurrentOption')))); }
192 \ProcessOptions();
193 \RequirePackage('mathhub');
194 \RequirePackage('omtext');
195 \RequirePackage('modules-mh');
196 </omtext.ltxml>

\mh*graphics Use the current value of \mh@currentrepos or the value of the mhrepos key if it
is given in \my*graphics.

197 <*omtext>

```

```

198 \addmetakey{Gin}{mhrepos}
199 \newcommand\mhgraphics[2] [] {\metasetkeys{Gin}{#1}%
200 \edef\mh@@repos{\mh@currentrepos}%
201 \ifx\Gin@mhrepos\@empty\mygraphics[#1]{\MathHub{\mh@currentrepos/source/#2}}%
202 \else\mygraphics[#1]{\MathHub{\Gin@mhrepos/source/#2}}\fi
203 \def\Gin@mhrepos{\mhcurrentrepos\mh@@repos}
204 \newcommand\mhgraphics[2] [] {\begin{center}\mhgraphics[#1]{#2}\end{center}}
205 \newcommand\mhgraphics[2] [] {\fbox{\mhgraphics[#1]{#2}}}
206 \newcommand\mhgraphics[2] [] {\begin{center}\fbox{\mhgraphics[#1]{#2}\end{center}}
207 \end{omtext}
208 \end{omtext.ltxml}
209 sub mhgraphics {
210   my ($gullet,$keyval,$arg2) = @_;
211   my $repo_path;
212   if ($keyval) {
213     $repo_path = ToString(GetKeyVal($keyval,'mhrepos')); }
214   if (! $repo_path) {
215     $repo_path = ToString(Digest(T_CS('\mh@currentrepos'))); }
216   else {
217     $keyval->setValue('mhrepos',undef); }
218   my $mathhub_base = ToString(Digest('\MathHub{'}));
219   my $finalpath = $mathhub_base.$repo_path.'/source/'.ToString($arg2);
220   return Invocation(T_CS('@includegraphicx'), $keyval, T_OTHER($finalpath)); }#
221 DefKeyVal('Gin','mhrepos','Semiverbatim');
222 DefMacro('\mhgraphics OptionalKeyVals:Gin {}', \&mhgraphics);
223 DefMacro('\mhgraphics [] {}', '\begin{center}\mhgraphics[#1]{#2}\end{center}');
224 DefMacro('\mhgraphics [] {}', '\fbox{\mhgraphics[#1]{#2}}');
225 \end{omtext.ltxml}

```

4.4 statements-mh: MH Variants for Statements

We set up package options and pass them on to the `statements` package, which we also load.

```

226 \begin{statements}
227 \ProvidesPackage{statements-mh}[2015/11/04 v1.0 MathHub support for the sTeX statements package]
228 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{statements}}
229 \ProcessOptions
230 \RequirePackage{mathhub}
231 \RequirePackage{statements}
232 \RequirePackage{omtext-mh}
233 \end{statements}
234 \end{statements.ltxml}
235 \DeclareOption(undef,sub{PassOptions('statements','sty',ToString(Digest(T_CS('\CurrentOption'))))}
236 \ProcessOptions();
237 \RequirePackage('mathhub');
238 \RequirePackage('statements');
239 \RequirePackage('omtext-mh');
240 \end{statements.ltxml}
241 \begin{statements}

```

```

242 \let\usemhvocab=\usemhmodule
243 \</statements>
244 \<statements.ltxml>
245 DefMacro('usemhvocab','usemhmodule');
246 \</statements.ltxml>

```

4.5 smultiling-mh: MH Variants for Multilinguality

We set up package options and pass them on to the `smultiling` package, which we also load.

```

247 \<smultiling>
248 \ProvidesPackage{smultiling-mh}[2015/11/04 v1.0 MathHub support for the sTeX smultiling package]
249 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{smultiling}}
250 \ProcessOptions
251 \RequirePackage{mathhub}
252 \RequirePackage{smultiling}
253 \RequirePackage{structview-mh}
254 \</smultiling>
255 \<smultiling.ltxml>
256 DeclareOption(undef,sub{PassOptions('smultiling','sty',ToString(Digest(T_CS('\CurrentOption'))))}
257 ProcessOptions();
258 RequirePackage('mathhub');
259 RequirePackage('smultiling');
260 RequirePackage('structview-mh');
261 \</smultiling.ltxml>

```

`mhmodnl:`

```

262 \<smultiling>
263 \addmetakey{mhmodnl}{repos}
264 \addmetakey{mhmodnl}{path}
265 \addmetakey*{mhmodnl}{title}
266 \addmetakey*{mhmodnl}{creators}
267 \addmetakey*{mhmodnl}{contributors}
268 \addmetakey{mhmodnl}{srccite}
269 \addmetakey{primary}{mhmodnl}[yes]
270 \</smultiling>
271 \<smultiling.ltxml>
272 DefKeyVal('mhmodnl','title','Semiverbatim');
273 DefKeyVal('mhmodnl','repos','Semiverbatim');
274 DefKeyVal('mhmodnl','path','Semiverbatim');
275 DefKeyVal('mhmodnl','creators','Semiverbatim');
276 DefKeyVal('mhmodnl','contributors','Semiverbatim');
277 DefKeyVal('mhmodnl','primary','Semiverbatim');
278 \</smultiling.ltxml>

```

`mhmodnl` The `mhmodnl` environment is just a layer over the `module` environment and the `\importmhmodule` macro with the keys and language suitably adapted.

```

279 \<smultiling>
280 \newenvironment{mhmodnl}[3][\metasetkeys{mhmodnl}{#1}%

```

```

281 \def\@test{#1}\ifx\@test\@empty\begin{module}[id=#2.#3]\else\begin{module}[id=#2.#3,#1]\fi%
282 \edef\@repos{\ifx\mhmodnl@repos\@empty\mh@currentrepos\else\mhmodnl@repos}
283 \if@langfiles\importmhmodule[repos=\@repos,load=#2,ext=tex]{#2}\else
284 \ifx\mhmodnl@load\@empty\importmodule{#2}\else\importmodule[ext=tex,load=\mhmodnl@load]{#2}\fi%
285 \fi}
286 {\end{module}}
287 \</smultiling>
288 \<*smultiling.ltxml>
289 DefEnvironment('{mhmodnl} OptionalKeyVals:mhmodnl {}{}',
290     "?#excluded()(<omdoc:theory xml:id='#2.#3' >"
291     . "?&defined(&GetKeyVal(#1,'creators'))(<dc:creator>&GetKeyVal(#1,'creators')</dc:cr
292     . "?&defined(&GetKeyVal(#1,'title'))(<dc:title>&GetKeyVal(#1,'title')</dc:title>())"
293     . "?&defined(&GetKeyVal(#1,'contributors'))(<dc:contributor>&GetKeyVal(#1,'contribut
294     . "<omdoc:imports from='?'&GetKeyVal(#1,'load')(&canonical_omdoc_path(&GetKeyVal(#1,'
295     . "#body"
296     . "</omdoc:theory>)",
297 afterDigestBegin=>sub {
298     my ($stomach, $whatsit) = @_;
299     my $keyval = $whatsit->getArg(1);
300     my $signature = ToString($whatsit->getArg(2));
301     my $language = ToString($whatsit->getArg(3));
302     my $repos = ToString(GetKeyVal($keyval,'torepos'));
303     my $current_repos = LookupValue('current_repos');
304     if (!$repos) { $repos = $current_repos; }
305     my $defpaths = LookupValue('defpath');
306     my $load_path = ($$defpaths[MathHub]).$repos.'/source/'. $signature;
307
308     if ($keyval) {
309         # If we're not given load, AND the langfiles option is in effect,
310         # default to #2
311         if ((! $keyval->getValue('path')) && (LookupValue('smultiling_langfiles')) {
312             $keyval->setValue('load',$load_path); }
313         # Always load a TeX file
314         $keyval->setValue('ext','tex');
315         $keyval->setValue('id',"$signature.$language"); }
316     module_afterDigestBegin(@_);
317     importmoduleI(@_);
318     return; },
319 afterDigest=>sub {
320     module_afterDigest(@_); });
321 \</smultiling.ltxml>%$

```

mhviewsig The **mhviewsig** environment is just a layer over the **mhview** environment with the keys suitably adapted.

```

322 \<smultiling.ltxml>RawTeX(
323 \<*smultiling | smultiling.ltxml>
324 \newenvironment{mhviewsig}[4][\def\@test{#1}\ifx\@test\@empty%
325 \begin{mhview}[id=#2,ext=tex]{#3}{#4}\else%
326 \begin{mhview}[id=#2,#1,ext=tex]{#3}{#4}\fi}
327 {\end{mhview}}

```

EdN:4

`mhviewnl` The `mhviewnl` environment is just a layer over the `mhviewsketch` environment with the keys and language suitably adapted.⁴

```
328 \newenvironment{mhviewnl}[5][]{\def@test{#1}\ifx@test\@empty%
329 \begin{mhviewsketch}[id=#2.#5,ext=tex]{#3}{#4}\else%
330 \begin{mhviewsketch}[id=#2.#5,#1,ext=tex]{#3}{#4}\fi}
331 {\end{mhviewsketch}}
332 \smultiling | smultiling.ltxml>
333 \smultiling.ltxml)';
```

4.6 structview-mh: MH Variants for Structures and Views

We set up package options and pass them on to the `structview` package, which we also load.

```
334 \*structview>
335 \ProvidesPackage{structview-mh}[2015/11/04 v1.0 MathHub support for the sTeX structview package]
336 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{structview}}
337 \ProcessOptions
338 \RequirePackage{mathhub}
339 \RequirePackage{structview}
340 \RequirePackage{modules-mh}
341 \</structview>
342 \*structview.ltxml>
343 \DeclareOption{undef,sub{PassOptions('structview','sty',ToString(Digest(T_CS('\CurrentOption'))))}}
344 \ProcessOptions();
345 \RequirePackage('mathhub');
346 \RequirePackage('structview');
347 \RequirePackage('modules-mh');
348 \</structview.ltxml>
```

`importmhmodulevia`

```
349 \<structview.ltxml>RawTeX('
350 \*structview | structview.ltxml>
351 \newenvironment{importmhmodulevia}[3][]{%
352 \gdef\@@doit{\importmhmodule[#1]{#2}{#3}}%
353 \ifmod@show\par\noindent importing module #2 via \@@doit\fi
354 }{%
355 \aftergroup\@@doit\ifmod@show end import\fi%
356 }%
357 \</structview | structview.ltxml>
358 \<structview.ltxml>');

359 \*structview>
360 \srefaddidkey{mhview}
361 \addmetakey{mhview}{display}
362 \addmetakey{mhview}{creators}
363 \addmetakey{mhview}{contributors}
```

⁴EdNOTE: MK: we have to do something about the `if@langfiles` situation here. But this is non-trivial, since we do not know the current path, to which we could append `.(lang)`!

```

364 \addmetakey{mhview}{srccite}
365 \addmetakey*{mhview}{title}
366 \addmetakey{mhview}{fromrepos}
367 \addmetakey{mhview}{torepos}
368 \addmetakey{mhview}{frompath}
369 \addmetakey{mhview}{topath}
370 \addmetakey[sms]{mhview}{ext}
371 \end{structview}
372 \end{structview.ltxml}
373 DefKeyVal('mhview','id','Semiverbatim');
374 DefKeyVal('mhview','display','Semiverbatim');
375 DefKeyVal('mhview','creators','Semiverbatim');
376 DefKeyVal('mhview','contributors','Semiverbatim');
377 DefKeyVal('mhview','srccite','Semiverbatim');
378 DefKeyVal('mhview','title','Semiverbatim');
379 DefKeyVal('mhview','fromrepos','Semiverbatim');
380 DefKeyVal('mhview','torepos','Semiverbatim');
381 DefKeyVal('mhview','frompath','Semiverbatim');
382 DefKeyVal('mhview','topath','Semiverbatim');
383 DefKeyVal('mhview','ext','Semiverbatim');
384 \end{structview.ltxml}

```

mhview the MathHub version

```

385 \begin{structview}
386 \newenvironment{mhview}[3][{}]{% keys, from, to
387 \metasetkeys{mhview}{#1}%
388 \sref@target%
389 \begin{@mhview}{#2}{#3}%
390 \view@heading{#2}{#3}{\mhview@display}{\mhview@title}%
391 }{%
392 \end{@mhview}%
393 \ignorespaces%
394 }%
395 \ifmod@show\surroundwithmdframed{mhview}\fi
396 \end{structview}
397 \end{structview.ltxml}
398 DefMacroI(T_CS(' \begin{mhview} '), 'OptionalKeyVals: mhview {}{}', sub {
399 my ($gullet, $keyvals, $from_arg, $to_arg) = @_;
400 my $from = ToString(Digest($from_arg));
401 my $to = ToString(Digest($to_arg));
402 AssignValue(from_module => $from);
403 AssignValue(to_module => $to);
404 my $from_repos = ToString(GetKeyVal($keyvals, 'fromrepos'));
405 my $to_repos = ToString(GetKeyVal($keyvals, 'torepos'));
406 my $repos = LookupValue('current_repos');
407 my $from_path = ToString(GetKeyVal($keyvals, 'frompath'));
408 my $to_path = ToString(GetKeyVal($keyvals, 'topath'));
409 my $ext = ToString(GetKeyVal($keyvals, 'ext')) if $keyvals;
410 $ext = 'sms' unless $ext;
411 my $current_repos = LookupValue('current_repos');

```



```

412 if (!$from_repos) { $from_repos = $current_repos; }
413 if (!$to_repos) { $to_repos = $current_repos; }
414 return (
415   Tokenize("\importMHmoduleI[repos=$from_repos,path=$from_path,ext=$ext]{$from}")->unlist,
416   Tokenize("\importMHmoduleI[repos=$to_repos,path=$to_path,ext=$ext]{$to}")->unlist,
417   Invocation(T_CS('\begin{viewenv}'),$keyvals,$from_arg,$to_arg)->unlist
418 );
419 });
420 DefMacroI('\end{mhview}',undef,'\end{viewenv}');
421 \</structview.ltxml>

```

@mhview The @mhview does the actual bookkeeping at the module level.

```

422 \<structview>
423 \newenvironment{@mhview}[2]{%from, to
424   \importmhmodule[repos=\mhview@fromrepos,path=\mhview@frompath,ext=\mhview@ext]{#1}%
425   \importmhmodule[repos=\mhview@torepos,path=\mhview@topath,ext=\mhview@ext]{#2}%
426 }{}%
427 \</structview>

```

mhviewsketch The mhviewsketch environment behaves like mhview, but only has text contents.

```

428 \<structview>
429 \newenvironment{mhviewsketch}[3][]{%
430   \metasetkeys{mhview}{#1}%
431   \sref@target%
432   \begin{@mhview}{#2}{#3}%
433   \view@heading{#2}{#3}{\mhview@display}{\mhview@title}%
434 }{}%
435   \end{@mhview}%
436   \ignorespaces%
437 }%
438 \ifmod@show\surroundwithmdframed{mhviewsketch}\fi
439 \</structview>
440 \<structview.ltxml>
441 DefMacroI(T_CS('\begin{mhviewsketch}'),'OptionalKeyVals:mhview {}{}', sub {
442   my ($gullet, $keyvals, $from_arg, $to_arg) = @_;
443   my $from = ToString(Digest($from_arg));
444   my $to = ToString(Digest($to_arg));
445   my $from_repos = ToString(GetKeyVal($keyvals,'fromrepos'));
446   my $to_repos = ToString(GetKeyVal($keyvals,'torepos'));
447   my $repos = LookupValue('current_repos');
448   my $from_path = ToString(GetKeyVal($keyvals,'frompath'));
449   my $to_path = ToString(GetKeyVal($keyvals,'topath'));
450   my $ext = ToString(GetKeyVal($keyvals,'ext')) if $keyvals;
451   $ext = 'sms' unless $ext;
452   my $current_repos = LookupValue('current_repos');
453   if (!$from_repos) { $from_repos = $current_repos; }
454   if (!$to_repos) { $to_repos = $current_repos; }
455   return (
456     Tokenize("\importMHmoduleI[repos=$from_repos,path=$from_path,ext=$ext]{$from}")->unlist,
457     Tokenize("\importMHmoduleI[repos=$to_repos,path=$to_path,ext=$ext]{$to}")->unlist,

```

```

458     Invocation(T_CS('\begin{viewsketchenv}'),$keyvals,$from_arg,$to_arg)->unlist
459   );
460 });
461 DefMacroI('\end{mhviewsketch}',undef,'\end{viewsketchenv}');
462 </structview.ltxml>

```

4.7 mikoslides-mh: Support for MiKo Slides

We set up package options and pass them on to the mikoslides package, which we also load.

```

463 <*mikoslides>
464 \ProvidesPackage{mikoslides-mh}[2015/11/04 v1.0 MathHub support for the sTeX mikoslides package]
465 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{mikoslides}}
466 \ProcessOptions
467 \RequirePackage{mathhub}
468 \RequirePackage{mikoslides}
469 \RequirePackage{statements-mh}
470 </mikoslides>
471 <*mikoslides.ltxml>
472 DeclareOption(undef,sub{PassOptions('mikoslides','sty',ToString(Digest(T_CS('\CurrentOption'))))}
473 ProcessOptions();
474 RequirePackage('mathhub');
475 RequirePackage('mikoslides');
476 RequirePackage('statements-mh');
477 </mikoslides.ltxml>

```

`\mhframeimage` Use the current value of `\mh@currentrepos` or the value of the `mhrepos` key if it is given in `\frameimage`.

```

478 <mikoslides>\addmetakey{Gin}{mhrepos}
479 <mikoslides.ltxml>DefKeyVal('Gin','mhrepos','Semiverbatim');
480 <mikoslides.ltxml>RawTeX('
481 <*mikoslides.ltxml | mikoslides>
482 \newcommand\mhframeimage[2][{}]{%
483   \metasetkeys{Gin}{#1}%
484   \edef\mh@crepos{\mh@currentrepos}%
485   \ifx\Gin@mhrepos\empty%
486     \frameimage[#1]{\MathHub{\mh@currentrepos/source/#2}}%
487   \else%
488     \frameimage[#1]{\MathHub{\Gin@mhrepos/source/#2}}%
489   \fi%
490 }%
491 </mikoslides.ltxml | mikoslides>
492 <mikoslides.ltxml>');

```

4.8 problem-mh: Support for Problems

We set up package options and pass them on to the problem package, which we also load.

```

493 <*problem>
494 \ProvidesPackage{problem-mh}[2015/11/04 v1.0 MathHub support for the sTeX problem package]
495 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
496 \ProcessOptions
497 \RequirePackage{mathhub}
498 \RequirePackage{problem}
499 \RequirePackage{omtext-mh}
500 </problem>
501 <*problem.ltxml>
502 DeclareOption(undef,sub{PassOptions('problem','sty',ToString(Digest(T_CS('\CurrentOption'))));
503 ProcessOptions();
504 RequirePackage('mathhub');
505 RequirePackage('problem');
506 RequirePackage('omtext-mh');
507 </problem.ltxml>

```

`\includemhproblem` The `\includemhproblem` saves the current value of `\mh@currentrepos` in a local macro `\mh@@repos`, resets `\mh@currentrepos` to the new value if one is given in the optional argument, and after importing resets `\mh@currentrepos` to the old value in `\mh@@repos`.

```

508 <*problem>
509 \newcommand\includemhproblem[2][\metasetkeys{inclprob}]{#1}%
510 \edef\mh@@repos{\mh@currentrepos}%
511 \ifx\inclprob\mhrepos\empty\else\mhcurrentrepos\inclprob\mhrepos\fi%
512 \input{\MathHub{\mh@currentrepos/source/#2}}%
513 \mhcurrentrepos\mh@@repos\clear@inclprob@keys}
514 </problem>
515 <*problem.ltxml>
516 sub includemhproblem {
517   my ($gullet,$keyval,$arg2) = @_ ;
518   my $repo_path;
519   if ($keyval) {
520     $repo_path = ToString(GetKeyVal($keyval,'mhrepos')); }
521   if (! $repo_path) {
522     $repo_path = ToString(Digest(T_CS('\mh@currentrepos'))); }
523   else {
524     $keyval->setValue('mhrepos',undef); }
525   my $mathhub_base = ToString(Digest('\MathHub{'}));
526   my $finalpath = $mathhub_base.$repo_path.'/source/'.ToString($arg2);
527   return Invocation(T_CS('\includeproblem'), $keyval, T_OTHER($finalpath)); }#
528 DefKeyVal('inclprob','mhrepos','Semiverbatim');
529 DefMacro('\includemhproblem OptionalKeyVals:inclprob {}', \&includemhproblem);
530 </problem.ltxml>

```

4.9 hwexam-mh: Support for Assignments

We set up package options and pass them on to the `hwexam` package, which we also load.

```

531 <*hwexam>

```

```

532 \ProvidesPackage{hwexam-mh}[2015/11/04 v1.0 MathHub support for the sTeX hwexam package]
533 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{hwexam}}
534 \ProcessOptions
535 \RequirePackage{mathhub}
536 \RequirePackage{hwexam}
537 \RequirePackage{problem-mh}
538 \</hwexam>
539 \<*hwexam.ltxml>
540 DeclareOption(undef,sub{PassOptions('hwexam','sty',ToString(Digest(T_CS('\CurrentOption')))); }
541 ProcessOptions();
542 RequirePackage('mathhub');
543 RequirePackage('hwexam');
544 RequirePackage('problem-mh');
545 \</hwexam.ltxml>

```

`\includemhassignment` The `\includemhassignment` saves the current value of `\mh@currentrepos` in a local macro `\mh@@repos`, resets `\mh@currentrepos` to the new value if one is given in the optional argument, and after importing resets `\mh@currentrepos` to the old value in `\mh@@repos`.

```

546 \<*hwexam>
547 \newcommand\includemhassignment[2][\metasetkeys{inclassig}{#1}%
548 \edef\mh@@repos{\mh@currentrepos}%
549 \ifx\inclassig@mhrepos\empty\else\mhcurrentrepos\inclassig@mhrepos\fi%
550 \includeassignment[#1]{\MathHub{\mh@currentrepos/source/#2}}%
551 \mhcurrentrepos\mh@@repos\clear@inclassig@keys}
552 \</hwexam>
553 \<*hwexam.ltxml>
554 sub includemhassignment {
555   my ($gullet,$keyval,$arg2) = @_ ;
556   my $repo_path;
557   if ($keyval) {
558     $repo_path = ToString(GetKeyVal($keyval,'mhrepos')); }
559   if (! $repo_path) {
560     $repo_path = ToString(Digest(T_CS('\mh@currentrepos'))); }
561   else {
562     $keyval->setValue('mhrepos',undef); }
563   my $mathhub_base = ToString(Digest('\MathHub{ }'));
564   my $finalpath = $mathhub_base.$repo_path.'/source/'.ToString($arg2);
565   return Invocation(T_CS('\includeassignment'), $keyval, T_OTHER($finalpath)); }#
566 DefKeyVal('inclprob','mhrepos','Semiverbatim');
567 DefMacro('\includemhassignment OptionalKeyVals:inclprob {}', \&includemhassignment);
568 \</hwexam.ltxml>

```

`\inputmhassignment` analogous

```

569 \<*hwexam>
570 \newcommand\inputmhassignment[2][\metasetkeys{inclassig}{#1}%
571 \edef\mh@@repos{\mh@currentrepos}%
572 \ifx\inclassig@mhrepos\empty\else\mhcurrentrepos\inclassig@mhrepos\fi%
573 \inputassignment[#1]{\MathHub{\mh@currentrepos/source/#2}}%

```

```

574 \mhcurrentrepos\mh@@repos\clear@inclassig@keys}
575 </hwexam>
576 <*hwexam.ltxml>
577 sub inputmhassignment {
578   my ($gullet,$keyval,$arg2) = @_;
579   my $repo_path;
580   if ($keyval) {
581     $repo_path = ToString(GetKeyVal($keyval,'mhrepos')); }
582   if (! $repo_path) {
583     $repo_path = ToString(Digest(T_CS('\mh@currentrepos'))); }
584   else {
585     $keyval->setValue('mhrepos',undef); }
586   my $mathhub_base = ToString(Digest('\MathHub{'}));
587   my $finalpath = $mathhub_base.$repo_path.'/source/'.ToString($arg2);
588   return Invocation(T_CS('\inputassignment'), $keyval, T_OTHER($finalpath)); }#$
589 DefMacro('\inputmhassignment OptionalKeyVals:inclprob {}', \&inputmhassignment);
590 </hwexam.ltxml>

```

4.10 Finale

Finally, we need to terminate the file with a success mark for perl.

```

591 <ltxml | modules.ltxml | structview.ltxml | omtex.ltxml | statements.ltxml | smultiling.ltxml | mikoslides.ltxml | problem.

```