MathHub Support for sTeX[*]

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

November 10, 2015

Abstract

The `sref` 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).

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

Contents

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

1 Introduction

Much of the sTeX 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 ⟨*group*⟩/⟨*repo*⟩, where ⟨*group*⟩ is a MathHub-unique repository group and ⟨*repo*⟩ a repository name that is ⟨*group*⟩-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 [Hor+11]. But this structure can be hidden from the sTeX author with MathHub-enabled versions of the sTeX 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 loaded, `\importmhmodule` is the same as `\importmodule`.

Of course, neither LaTeX nor LaTeXMLknow 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 in each module, since the `\importmhmodule` macro sets the current repository automatically.

`\mhcurrentrepos`

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

`\usemhmodule`

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

`\mhinputref` `\mhinput`

2.3 `omtext-mh`: MH Variants for OMText

`\mhcgraphics`

The `\mhcgraphics` 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)

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

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

2.4 `smultiling-mh`: MH Variants for Multilinguality

EdN:1 EdN:2

[1] [2]

2.5 `structview-mh`: MH Variants for Structures and Views

EdN:3

[3]

2.6 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}}
```

[1] EdNote: needs to be documented
[2] EdNote: mhmodsig seems to be missing what happened?
[3] EdNote: needs to be documented

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}
```

2.7 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.

2.8 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}
```

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].

1. none reported yet.

4 Implementation

The `sref` package generates two files: the LaTeX package (all the code between ⟨*package⟩ and ⟨/package⟩) and the LaTeXML 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 LaTeXML binding files an the base package.

```
⟨*ltxml | modules.ltxml | omtext.ltxml | smultiling.ltxml | mikoslides.ltxml | problem.ltxml | hwexam.ltxml⟩
package LaTeXML::Package::Pool;
use strict;
use LaTeXML::Package;
⟨/ltxml | modules.ltxml | omtext.ltxml | smultiling.ltxml | mikoslides.ltxml | problem.ltxml | hwexam.ltxml⟩
⟨package⟩\ProvidesPackage{mathhub}[2015/11/04 v1.0 sTeX Support for MathHub.info]
```

```
⟨*package⟩
\DeclareOption*{}
\ProcessOptions
⟨/package⟩
⟨*ltxml⟩
DeclareOption(undef,sub {});
ProcessOptions();
⟨/ltxml⟩
```

Then we need to set up the packages by requiring the `metakeys` package [Koh15] to be loaded (in the right version).

```
⟨*package⟩
\RequirePackage{keyval}
⟨/package⟩
⟨*ltxml⟩
RequirePackage('keyval');
⟨/ltxml⟩
```

4.1 General Infrastructure

`\mhcurrentrepos`
`\@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.

```
⟨*package⟩
\newcommand\mhcurrentrepos[1]{%
  \edef\@test{#1}%
  \ifx\@test\mh@currentrepos% if new dir = old dir
    \relax% no need to change
  \else%
    \protected@write\@auxout{}{\string\@mhcurrentrepos{#1}}%
  \fi%
  \@mhcurrentrepos{#1}% define mh@currentrepos
}%
\newcommand\@mhcurrentrepos[1]{\edef\mh@currentrepos{#1}}%
```

```
⟨/package⟩
⟨*ltxml⟩
DefMacro('\mhcurrentrepos{}','\@mhcurrentrepos{#1}');
DefMacro('\@mhcurrentrepos{}','\def\mh@currentrepos{#1}\@@mhcurrentrepos{#1}');
DefConstructor('\@@mhcurrentrepos{}','',
  afterDigest => sub{ AssignValue('current_repos',ToString($_[1]->getArg(1)),'global'); } );
⟨/ltxml⟩#$
```

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

```
⟨ltxml⟩RaxTeX('
⟨*package | ltxml⟩
\def\modules@@first#1/#2;{#1}
\newcommand\libinput[1]{\def\@libfile{\MathHub{\mh@currentrepos/lib/#1}}%
\IfFileExists{\@libfile}{\input\@libfile}%
{\edef\@@group{\expandafter\modules@@first\mh@currentrepos;}
\edef\@inffile{\MathHub{\@@group/meta-inf/lib/#1}}
\IfFileExists{\@inffile}{\input{\@inffile}}%
{\PackageError{modules}
  {Library file missing, cannot input #1\MessageBreak%
    Both \@libfile.tex\MessageBreak and \@inffile.tex\MessageBreak do not exit}%
  {Check whether the file name is correct}}}}
⟨/package | ltxml⟩
⟨ltxml⟩');
```

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.

```
⟨*modules⟩
\ProvidesPackage{modules-mh}[2015/11/04 v1.0 MathHub support for the sTeX modules package]
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{modules}}
\ProcessOptions
\RequirePackage{modules}
\RequirePackage{mathhub}
⟨/modules⟩
⟨*modules.ltxml⟩
DeclareOption(undef,sub{PassOptions('modules','sty',ToString(Digest(T_CS('\CurrentOption'))));
ProcessOptions();
RequirePackage('modules');
RequirePackage('mathhub');
⟨/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` comparison with an `\expandafter`, since the values may be passed on from other key bindings. Parameters will be passed to `\importmodule`.

```
⟨*modules⟩
\srefaddidkey{importmhmodule}%
\addmetakey{importmhmodule}{repos}% saves the repo’s path. E.g: smglom/numberfield
\addmetakey{importmhmodule}{path}% saves the module name. E.g: naturalnumbers
\addmetakey[sms]{importmhmodule}{ext}% saves the extension: E.g: tex
\addmetakey[false]{importmhmodule}{conservative}[true]%
\newcommand\importmhmodule[2][]{%
  \metasetkeys{importmhmodule}{#1}%
  \ifx\importmhmodule@path\@empty% if module name is not set
    \importmodule[ext=\importmhmodule@ext,id=\importmhmodule@id]{#2}%
  \else%
    \edef\mh@@repos{\mh@currentrepos}% remember so that we can reset it.
    \ifx\importmhmodule@repos\@empty% if in the same repos
      \relax% no need to change mh@currentrepos, i.e, current dirctory.
    \else%
      \mhcurrentrepos{\importmhmodule@repos}% change it.
    \fi%
    \importmodule[load=\MathHub{\mh@currentrepos/source/\importmhmodule@path},%
    ext=\importmhmodule@ext,id=\importmhmodule@id]{#2}%
    \mhcurrentrepos{\mh@@repos}% after importing, reset to old value
  \fi%
  \ignorespaces%
}%
⟨/modules⟩
⟨*modules.ltxml⟩
DefKeyVal('importmhmodule','id','Semiverbatim');
DefKeyVal('importmhmodule','repos','Semiverbatim');
DefKeyVal('importmhmodule','path','Semiverbatim');
DefKeyVal('importmhmodule','ext','Semiverbatim');
DefKeyVal('importmhmodule','conservative','Semiverbatim');
DefConstructor('\importmhmodule OptionalKeyVals:importmhmodule {}',
       "<omdoc:imports "
       . "from='?&GetKeyVal(#1,'load')(&canonical_omdoc_path(&GetKeyVal(#1,'load')))()###2'"
               . "?&defined(&GetKeyVal(#1,'conservative'))(load='&GetKeyVal(#1,'conservative')'
   afterDigest => \&importMHmoduleI);

sub importMHmoduleI {
  my ($stomach, $whatsit) = @_;
  my $keyval = $whatsit->getArg(1);
  my $id = $whatsit->getArg(2);
  if ($keyval) {
    my $repos = ToString($keyval->getValue('repos'));
    my $path = ToString($keyval->getValue('path'));
    my $current_repos = LookupValue('current_repos');
    if (!$repos) { # Use the implicit current repository
      $repos = $current_repos; }
    my $defpaths = LookupValue('defpath');
    my $load_path = ($$defpaths{MathHub}).$repos.'/source/'.$path;
    $keyval->setValue('load',$load_path);
    AssignValue('current_repos' => $repos, 'global');
    importmoduleI($stomach,$whatsit);
    AssignValue(’current_repos’ => $current_repos, ’global’); }
  else {
    importmoduleI($stomach,$whatsit);  }
  return; }

DefConstructor(’\importMHmoduleI OptionalKeyVals:importmhmodule {}’, ’’,
   afterDigest=> \&importMHmoduleI );#$
⟨/modules.ltxml⟩
```

and now the analogs

\usemhmodule

```
⟨*modules⟩
\newcommand\usemhmodule[2][]{%
  \metasetkeys{importmhmodule}{#1}%
  \ifx\importmhmodule@path\@empty%
    \usemodule[ext=\importmhmodule@ext,id=\importmhmodule@id]{#2}%
  \else%
    \edef\mh@@repos{\mh@currentrepos}%
    \ifx\importmhmodule@repos\@empty%
    \else%
      \mhcurrentrepos{\importmhmodule@repos}%
    \fi%
    \usemodule[load=\MathHub{\mh@currentrepos/source/\importmhmodule@path},ext=\importmhmodule@
    \mhcurrentrepos\mh@@repos%
  \fi%
  \ignorespaces%
}%
⟨/modules⟩
⟨*modules.ltxml⟩
DefConstructor(’\usemhmodule OptionalKeyVals:importmhmodule {}’,
   "<omdoc:uses from=’?&GetKeyVal(#1,’load’)(&canonical_omdoc_path(&GetKeyVal(#1,’load’)))()###
   afterDigest => \&importMHmoduleI);
⟨/modules.ltxml⟩
```

\mhinputref

```
⟨modules.ltxml⟩RawTeX(’
⟨*modules | modules.ltxml⟩
\newcommand\mhinputref[2][]{%
  \def\@repos{#1}%
  \edef\mh@@repos{\mh@currentrepos}%
  \ifx\@repos\@empty%
  \else%
    \mhcurrentrepos{#1}%
  \fi%
  \inputref{\MathHub{\mh@currentrepos/source/#2}}%
  \mhcurrentrepos\mh@@repos%
  \ignorespaces%
}%
⟨/modules | modules.ltxml⟩
⟨modules.ltxml⟩');
```

`\mhinput`

```
⟨*modules⟩
\let\mhinput\mhinputref%
⟨/modules⟩
```

4.3 `omtext-mh`: MH Variants for OMText

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

```
⟨*omtext⟩
\ProvidesPackage{omtext-mh}[2015/11/04 v1.0 MathHub support for the sTeX omtext package]
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{omtext}}
\ProcessOptions
\RequirePackage{omtext}
\RequirePackage{mathhub}
⟨/omtext⟩
⟨*omtext.ltxml⟩
DeclareOption(undef,sub{PassOptions('omtext','sty',ToString(Digest(T_CS('\CurrentOption')))); }
ProcessOptions();
RequirePackage('omtext');
RequirePackage('mathhub');
⟨/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`.

```
⟨*omtext⟩
\addmetakey{Gin}{mhrepos}
\newcommand\mhgraphics[2][]{\metasetkeys{Gin}{#1}%
\edef\mh@@repos{\mh@currentrepos}%
\ifx\Gin@mhrepos\@empty\mygraphics[#1]{\MathHub{\mh@currentrepos/source/#2}}%
\else\mygraphics[#1]{\MathHub{\Gin@mhrepos/source/#2}}\fi
\def\Gin@mhrepos{}\mhcurrentrepos\mh@@repos}
\newcommand\mhcgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
\newcommand\mhbgraphics[2][]{\fbox{\mhgraphics[#1]{#2}}}
\newcommand\mhcbgraphics[2][]{\begin{center}\fbox{\mhgraphics[#1]{#2}}\end{center}}
⟨/omtext⟩
⟨*omtext.ltxml⟩
sub mhgraphics {
  my ($gullet,$keyval,$arg2) = @_;
  my $repo_path;
  if ($keyval) {
    $repo_path = ToString(GetKeyVal($keyval,'mhrepos')); }
  if (! $repo_path) {
    $repo_path = ToString(Digest(T_CS('\mh@currentrepos'))); }
  else {
    $keyval->setValue('mhrepos',undef); }
  my $mathhub_base = ToString(Digest('\MathHub{}'));
  my $finalpath = $mathhub_base.$repo_path.'/source/'.ToString($arg2);
  return Invocation(T_CS('\@includegraphicx'), $keyval, T_OTHER($finalpath)); }#$
DefKeyVal('Gin','mhrepos','Semiverbatim');
DefMacro('\mhgraphics OptionalKeyVals:Gin {}', \&mhgraphics);
DefMacro('\mhcgraphics []{}','\begin{center}\mhgraphics[#1]{#2}\end{center}');
DefMacro('\mhbgraphics []{}','\fbox{\mhgraphics[#1]{#2}}');
⟨/omtext.ltxml⟩
```

4.4 smultiling-mh: MH Variants for Multilinguality

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

```
⟨*smultiling⟩
\ProvidesPackage{smultiling-mh}[2015/11/04 v1.0 MathHub support for the sTeX smultiling package
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{smultiling}}
\ProcessOptions
\RequirePackage{smultiling}
\RequirePackage{mathhub}
⟨/smultiling⟩
⟨*smultiling.ltxml⟩
DeclareOption(undef,sub{PassOptions('smultiling','sty',ToString(Digest(T_CS('\CurrentOption')))
ProcessOptions();
RequirePackage('smultiling');
RequirePackage('mathhub');
⟨/smultiling.ltxml⟩
```

mhmodnl:*

```
⟨*smultiling⟩
\addmetakey{mhmodnl}{repos}
\addmetakey{mhmodnl}{path}
\addmetakey*{mhmodnl}{title}
\addmetakey*{mhmodnl}{creators}
\addmetakey*{mhmodnl}{contributors}
\addmetakey{mhmodnl}{srccite}
\addmetakey{primary}{mhmodnl}[yes]
⟨/smultiling⟩
⟨*smultiling.ltxml⟩
DefKeyVal('mhmodnl','title','Semiverbatim');
DefKeyVal('mhmodnl','repos','Semiverbatim');
DefKeyVal('mhmodnl','path','Semiverbatim');
DefKeyVal('mhmodnl','creators','Semiverbatim');
DefKeyVal('mhmodnl','contributors','Semiverbatim');
DefKeyVal('mhmodnl','primary','Semiverbatim');
⟨/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.

```
⟨*smultiling⟩
```

```
\newenvironment{mhmodnl}[3][]{\metasetkeys{mhmodnl}{#1}%
\def\@test{#1}\ifx\@test\@empty\begin{module}[id=#2.#3]\else\begin{module}[id=#2.#3,#1]\fi%
\edef\@repos{\ifx\mhmodnl@repos\@empty\mh@currentrepos\else\mhmodnl@repos}
\if@langfiles\importmhmodule[repos=\@repos,load=#2,ext=tex]{#2}\else
\ifx\mhmodnl@load\@empty\importmodule{#2}\else\importmodule[ext=tex,load=\mhmodnl@load]{#2}\fi%
\fi}
{\end{module}}
⟨/smultiling⟩
⟨*smultiling.ltxml⟩
DefEnvironment('{mhmodnl} OptionalKeyVals:mhmodnl {}{}',
        "?#excluded()(<omdoc:theory xml:id='#2.#3' >"
        .   "?&defined(&GetKeyVal(#1,'creators'))(<dc:creator>&GetKeyVal(#1,'creators')</dc:cr
        .   "?&defined(&GetKeyVal(#1,'title'))(<dc:title>&GetKeyVal(#1,'title')</dc:title>)()"
        .   "?&defined(&GetKeyVal(#1,'contributors'))(<dc:contributor>&GetKeyVal(#1,'contribut
        .   "<omdoc:imports from='?&GetKeyVal(#1,'load')(&canonical_omdoc_path(&GetKeyVal(#1,'
        .   "#body"
        . "</omdoc:theory>)",
  afterDigestBegin=>sub {
    my ($stomach, $whatsit) = @_;
    my $keyval = $whatsit->getArg(1);
    my $signature = ToString($whatsit->getArg(2));
    my $language = ToString($whatsit->getArg(3));
    my $repos = ToString(GetKeyVal($keyval,'torepos'));
    my $current_repos = LookupValue('current_repos');
    if (!$repos) { $repos = $current_repos; }
    my $defpaths = LookupValue('defpath');
    my $load_path = ($$defpaths{MathHub}).$repos.'/source/'.$signature;

    if ($keyval) {
      # If we're not given load, AND the langfiles option is in effect,
      # default to #2
      if ((! $keyval->getValue('path')) && (LookupValue('smultiling_langfiles'))) {
        $keyval->setValue('load',$load_path); }
      # Always load a TeX file
      $keyval->setValue('ext','tex');
      $keyval->setValue('id',"$signature.$language"); }
    module_afterDigestBegin(@_);
    importmoduleI(@_);
    return; },
  afterDigest=>sub {
    module_afterDigest(@_); });
⟨/smultiling.ltxml⟩%$
```

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

```
⟨smultiling.ltxml⟩RawTeX('
⟨*smultiling | smultiling.ltxml⟩
\newenvironment{mhviewsig}[4][]{\def\@test{#1}\ifx\@test\@empty%
\begin{mhview}[id=#2,ext=tex]{#3}{#4}\else%
\begin{mhview}[id=#2,#1,ext=tex]{#3}{#4}\fi}
```

```
{\end{mhview}}
```

mhviewnl The mhviewnl environment is just a layer over the mhviewsketch environment with the keys and langauge suitably adapted.[4]

EdN:4

```
\newenvironment{mhviewnl}[5][]{\def\@test{#1}\ifx\@test\@empty%
\begin{mhviewsketch}[id=#2.#5,ext=tex]{#3}{#4}\else%
\begin{mhviewsketch}[id=#2.#5,#1,ext=tex]{#3}{#4}\fi}
{\end{mhviewsketch}}
⟨/smultiling | smultiling.ltxml⟩
⟨smultiling.ltxml⟩');
```

4.5 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.

```
⟨*structview⟩
\ProvidesPackage{structview-mh}[2015/11/04 v1.0 MathHub support for the sTeX structview package
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{structview}}
\ProcessOptions
\RequirePackage{structview}
\RequirePackage{mathhub}
⟨/structview⟩
⟨*structview.ltxml⟩
DeclareOption(undef,sub{PassOptions('structview','sty',ToString(Digest(T_CS('\CurrentOption')))
ProcessOptions();
RequirePackage('structview');
RequirePackage('mathhub');
⟨/structview.ltxml⟩
```

importmhmodulevia

```
⟨structview.ltxml⟩RawTeX('
⟨*structview | structview.ltxml⟩
\newenvironment{importmhmodulevia}[3][]{%
  \gdef\@@doit{\importmhmodule[#1]{#2}{#3}}%
  \ifmod@show\par\noindent importing module #2 via \@@doit\fi
}{%
  \aftergroup\@@doit\ifmod@show end import\fi%
}%
⟨/structview | structview.ltxml⟩
⟨structview.ltxml⟩');
```

```
⟨*structview⟩
\srefaddidkey{mhview}
\addmetakey{mhview}{display}
\addmetakey{mhview}{creators}
\addmetakey{mhview}{contributors}
\addmetakey{mhview}{srccite}
```

[4]EdNote: 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*⟩!

```
\addmetakey*{mhview}{title}
\addmetakey{mhview}{fromrepos}
\addmetakey{mhview}{torepos}
\addmetakey{mhview}{frompath}
\addmetakey{mhview}{topath}
\addmetakey[sms]{mhview}{ext}
⟨/structview⟩
⟨*structview.ltxml⟩
DefKeyVal('mhview','id','Semiverbatim');
DefKeyVal('mhview','display','Semiverbatim');
DefKeyVal('mhview','creators','Semiverbatim');
DefKeyVal('mhview','contributors','Semiverbatim');
DefKeyVal('mhview','srccite','Semiverbatim');
DefKeyVal('mhview','title','Semiverbatim');
DefKeyVal('mhview','fromrepos','Semiverbatim');
DefKeyVal('mhview','torepos','Semiverbatim');
DefKeyVal('mhview','frompath','Semiverbatim');
DefKeyVal('mhview','topath','Semiverbatim');
DefKeyVal('mhview','ext','Semiverbatim');
⟨/structview.ltxml⟩
```

`mhview` the MathHub version

```
⟨*structview⟩
\newenvironment{mhview}[3][]{% keys, from, to
  \metasetkeys{mhview}{#1}%
  \sref@target%
  \begin{@mhview}{#2}{#3}%
  \view@heading{#2}{#3}{\mhview@display}{\mhview@title}%
}{%
  \end{@mhview}%
  \ignorespaces%
}%
\ifmod@show\surroundwithmdframed{mhview}\fi
⟨/structview⟩
⟨*structview.ltxml⟩
DefMacroI(T_CS('\begin{mhview}'),'OptionalKeyVals:mhview {}{}', sub {
  my ($gullet, $keyvals, $from_arg, $to_arg) = @_;
  my $from = ToString(Digest($from_arg));
  my $to = ToString(Digest($to_arg));
  AssignValue(from_module => $from);
  AssignValue(to_module => $to);
  my $from_repos = ToString(GetKeyVal($keyvals,'fromrepos'));
  my $to_repos = ToString(GetKeyVal($keyvals,'torepos'));
  my $repos = LookupValue('current_repos');
  my $from_path = ToString(GetKeyVal($keyvals,'frompath'));
  my $to_path = ToString(GetKeyVal($keyvals,'topath'));
  my $ext = ToString(GetKeyVal($keyvals,'ext')) if $keyvals;
  $ext = 'sms' unless $ext;
  my $current_repos = LookupValue('current_repos');
  if (!$from_repos) { $from_repos = $current_repos; }
  if (!$to_repos) { $to_repos = $current_repos; }
  return (
    Tokenize("\\importMHmoduleI[repos=$from_repos,path=$from_path,ext=$ext]{$from}")->unlist,
    Tokenize("\\importMHmoduleI[repos=$to_repos,path=$to_path,ext=$ext]{$to}")->unlist,
    Invocation(T_CS('\begin{viewenv}'),$keyvals,$from_arg,$to_arg)->unlist
  );
});
DefMacroI('\end{mhview}',undef,'\end{viewenv}');
⟨/structview.ltxml⟩
```

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

```
⟨*structview⟩
\newenvironment{@mhview}[2]{%from, to
  \importmhmodule[repos=\mhview@fromrepos,path=\mhview@frompath,ext=\mhview@ext]{#1}%
  \importmhmodule[repos=\mhview@torepos,path=\mhview@topath,ext=\mhview@ext]{#2}%
}{}%
⟨/structview⟩
```

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

```
⟨*structview⟩
\newenvironment{mhviewsketch}[3][]{%
  \metasetkeys{mhview}{#1}%
  \sref@target%
  \begin{@mhview}{#2}{#3}%
  \view@heading{#2}{#3}{\mhview@display}{\mhview@title}%
}{%
  \end{@mhview}%
  \ignorespaces%
}%
\ifmod@show\surroundwithmdframed{mhviewsketch}\fi
⟨/structview⟩
⟨*structview.ltxml⟩
DefMacroI(T_CS('\begin{mhviewsketch}'),'OptionalKeyVals:mhview {}{}', sub {
  my ($gullet, $keyvals, $from_arg, $to_arg) = @_;
  my $from = ToString(Digest($from_arg));
  my $to = ToString(Digest($to_arg));
  my $from_repos = ToString(GetKeyVal($keyvals,'fromrepos'));
  my $to_repos = ToString(GetKeyVal($keyvals,'torepos'));
  my $repos = LookupValue('current_repos');
  my $from_path = ToString(GetKeyVal($keyvals,'frompath'));
  my $to_path = ToString(GetKeyVal($keyvals,'topath'));
  my $ext = ToString(GetKeyVal($keyvals,'ext')) if $keyvals;
  $ext = 'sms' unless $ext;
  my $current_repos = LookupValue('current_repos');
  if (!$from_repos) { $from_repos = $current_repos; }
  if (!$to_repos) { $to_repos = $current_repos; }
  return (
    Tokenize("\\importMHmoduleI[repos=$from_repos,path=$from_path,ext=$ext]{$from}")->unlist,
    Tokenize("\\importMHmoduleI[repos=$to_repos,path=$to_path,ext=$ext]{$to}")->unlist,
    Invocation(T_CS('\begin{viewsketchenv}'),$keyvals,$from_arg,$to_arg)->unlist
  );
});
DefMacroI('\end{mhviewsketch}',undef,'\end{viewsketchenv}');
⟨/structview.ltxml⟩
```

4.6 **mikoslides-mh:** Support for MiKo Slides

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

```
⟨*mikoslides⟩
\ProvidesPackage{mikoslides-mh}[2015/11/04 v1.0 MathHub support for the sTeX mikoslides package
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{mikoslides}}
\ProcessOptions
\RequirePackage{mikoslides}
\RequirePackage{mathhub}
⟨/mikoslides⟩
⟨*mikoslides.ltxml⟩
DeclareOption(undef,sub{PassOptions('mikoslides','sty',ToString(Digest(T_CS('\CurrentOption')))
ProcessOptions();
RequirePackage('mikoslides');
RequirePackage('mathhub');
⟨/mikoslides.ltxml⟩
```

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

```
⟨mikoslides⟩\addmetakey{Gin}{mhrepos}
⟨mikoslides.ltxml⟩DefKeyVal('Gin','mhrepos','Semiverbatim');
⟨mikoslides.ltxml⟩RawTeX('
⟨*mikoslides.ltxml | mikoslides⟩
\newcommand\mhframeimage[2][]{%
  \metasetkeys{Gin}{#1}%
  \edef\mh@@repos{\mh@currentrepos}%
  \ifx\Gin@mhrepos\@empty%
    \frameimage[#1]{\MathHub{\mh@currentrepos/source/#2}}%
  \else%
    \frameimage[#1]{\MathHub{\Gin@mhrepos/source/#2}}%
  \fi%
}%
⟨/mikoslides.ltxml | mikoslides⟩
⟨mikoslides.ltxml⟩');
```

4.7 **problem-mh:** Support for Problems

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

```
⟨*problem⟩
\ProvidesPackage{problem-mh}[2015/11/04 v1.0 MathHub support for the sTeX problem package]
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
```

```
\ProcessOptions
\RequirePackage{problem}
\RequirePackage{mathhub}
⟨/problem⟩
⟨*problem.ltxml⟩
DeclareOption(undef,sub{PassOptions('problem','sty',ToString(Digest(T_CS('\CurrentOption'))));
ProcessOptions();
RequirePackage('problem');
RequirePackage('mathhub');
⟨/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`.

```
⟨*problem⟩
\newcommand\includemhproblem[2][]{\metasetkeys{inclprob}{#1}%
\edef\mh@@repos{\mh@currentrepos}%
\ifx\inclprob@mhrepos\@empty\else\mhcurrentrepos\inclprob@mhrepos\fi%
\input{\MathHub{\mh@currentrepos/source/#2}}%
\mhcurrentrepos\mh@@repos\clear@inclprob@keys}
⟨/problem⟩
⟨*problem.ltxml⟩
sub includemhproblem {
  my ($gullet,$keyval,$arg2) = @_;
  my $repo_path;
  if ($keyval) {
    $repo_path = ToString(GetKeyVal($keyval,'mhrepos')); }
  if (! $repo_path) {
    $repo_path = ToString(Digest(T_CS('\mh@currentrepos'))); }
  else {
    $keyval->setValue('mhrepos',undef); }
  my $mathhub_base = ToString(Digest('\MathHub{}'));
  my $finalpath = $mathhub_base.$repo_path.'/source/'.ToString($arg2);
  return Invocation(T_CS('\includeproblem'), $keyval, T_OTHER($finalpath)); }#$
DefKeyVal('inclprob','mhrepos','Semiverbatim');
DefMacro('\includemhproblem OptionalKeyVals:inclprob {}', \&includemhproblem);
⟨/problem.ltxml⟩
```

4.8 hwexam-mh: Support for Assignments

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

```
⟨*hwexam⟩
\ProvidesPackage{hwexam-mh}[2015/11/04 v1.0 MathHub support for the sTeX hwexam package]
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{hwexam}}
\ProcessOptions
\RequirePackage{hwexam}
\RequirePackage{mathhub}
```

```
⟨/hwexam⟩
⟨*hwexam.ltxml⟩
DeclareOption(undef,sub{PassOptions('hwexam','sty',ToString(Digest(T_CS('\CurrentOption')))); }
ProcessOptions();
RequirePackage('hwexam');
RequirePackage('mathhub');
⟨/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`.

```
⟨*package⟩
\newcommand\includemhassignment[2][]{\metasetkeys{inclassig}{#1}%
\edef\mh@@repos{\mh@currentrepos}%
\ifx\inclassig@mhrepos\@empty\else\mhcurrentrepos\inclassig@mhrepos\fi%
\includeassignment[#1]{\MathHub{\mh@currentrepos/source/#2}}%
\mhcurrentrepos\mh@@repos\clear@inclassig@keys}
⟨/package⟩
⟨*ltxml⟩
sub includemhassignment {
  my ($gullet,$keyval,$arg2) = @_;
  my $repo_path;
  if ($keyval) {
    $repo_path = ToString(GetKeyVal($keyval,'mhrepos')); }
  if (! $repo_path) {
    $repo_path = ToString(Digest(T_CS('\mh@currentrepos'))); }
  else {
    $keyval->setValue('mhrepos',undef); }
  my $mathhub_base = ToString(Digest('\MathHub{}'));
  my $finalpath = $mathhub_base.$repo_path.'/source/'.ToString($arg2);
  return Invocation(T_CS('\includeassignment'), $keyval, T_OTHER($finalpath)); }#$
DefKeyVal('inclprob','mhrepos','Semiverbatim');
DefMacro('\includemhassignment OptionalKeyVals:inclprob {}', \&includemhassignment);
⟨/ltxml⟩
```

`\inputmhassignment` analogous

```
⟨*package⟩
\newcommand\inputmhassignment[2][]{\metasetkeys{inclassig}{#1}%
\edef\mh@@repos{\mh@currentrepos}%
\ifx\inclassig@mhrepos\@empty\else\mhcurrentrepos\inclassig@mhrepos\fi%
\inputassignment[#1]{\MathHub{\mh@currentrepos/source/#2}}%
\mhcurrentrepos\mh@@repos\clear@inclassig@keys}
⟨/package⟩
⟨*ltxml⟩
sub inputmhassignment {
  my ($gullet,$keyval,$arg2) = @_;
  my $repo_path;
  if ($keyval) {
```

```
  $repo_path = ToString(GetKeyVal($keyval,'mhrepos')); }
 if (! $repo_path) {
  $repo_path = ToString(Digest(T_CS('\mh@currentrepos'))); }
 else {
  $keyval->setValue('mhrepos',undef); }
 my $mathhub_base = ToString(Digest('\MathHub{}'));
 my $finalpath = $mathhub_base.$repo_path.'/source/'.ToString($arg2);
 return Invocation(T_CS('\inputassignment'), $keyval, T_OTHER($finalpath)); }#$
DefMacro('\inputmhassignment OptionalKeyVals:inclprob {}', \&inputmhassignment);
⟨/ltxml⟩
```

4.9 Finale

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

```
⟨*ltxml⟩
1;
⟨/ltxml⟩
```

References

[Hor+11] Fulya Horozal et al. "Combining Source, Content, Presentation, Narration, and Relational Representation". In: *Intelligent Computer Mathematics*. Ed. by James Davenport et al. LNAI 6824. Springer Verlag, 2011, pp. 212–227. ISBN: 978-3-642-22672-4. URL: `http://kwarc.info/frabe/Research/HIJKR_dimensions_11.pdf`.

[Koh15] Michael Kohlhase. *`metakeys.sty`: A generic framework for extensible Metadata in LaTeX*. Tech. rep. Comprehensive TeX Archive Network (CTAN), 2015. URL: `http://www.ctan.org/tex-archive/macros/latex/contrib/stex/metakeys/metakeys.pdf`.

[sTeX] *KWARC/sTeX*. URL: `https://svn.kwarc.info/repos/stex` (visited on 05/15/2015).