MathHub Support for sTeX[*]

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

November 2, 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 v1.2 (last revised 2015/10/27)

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 [**HorIacJuc:cscpnrr11**]. But this structure can be hidden from the sTeX author with MathHub-enabled versions of the `modules` macros.

2 The User Interface

2.1 Package Options

none so far

2.2 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 `\mhcurrentrepos` 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.

`\usemhmodule` `\adoptmhmodule` The `\usemhmodule` and `\adoptmhmodule` macros are the analogs to `\usemodule` and `\adoptmodule`.

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.

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

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 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 file.

```
⟨*ltxml⟩
package LaTeXML::Package::Pool;
use strict;
use LaTeXML::Package;
⟨/ltxml⟩
```

4.1 Package Options

We declare some switches which will modify the behavior according to the package options. Generally, an option xxx will just set the appropriate switches to true (otherwise they stay false).[1] EdN:1

```
⟨*package⟩
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{metakeys}}
\ProcessOptions
⟨/package⟩
⟨*ltxml⟩
DeclareOption(undef,sub {PassOptions('metakeys','sty',ToString(Digest(T_CS('\CurrentOption'))))
⟨/ltxml⟩
```

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

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

4.2 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⟩
\newrobustcmd\mhcurrentrepos[1]{%
  \edef\@test{#1}%
  \ifx\@test\mh@currentrepos% if new dir = old dir
    \relax% no need to change
  \else%
```

[1] EdNote: do we need this?

```
    \protected@write\@auxout{}{\string\@mhcurrentrepos{#1}}%
  \fi%
  \@mhcurrentrepos{#1}% define mh@currentrepos
}%
\newrobustcmd\@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⟩');
```

4.3 MH Variants for Modules

`\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]%
\newrobustcmd\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%
}%
```

and now the analogs

`\usemhmodule`

```
\newrobustcmd\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%
}%
```

`\adoptmhmodule`

```
\newrobustcmd\adoptmhmodule[2][]{%
  \metasetkeys{importmhmodule}{#1}%
  \ifx\importmhmodule@path\@empty
    \adoptmodule[ext=\importmhmodule@ext,id=\importmhmodule@id]{#2}%
  \else%
    \edef\mh@@repos{\mh@currentrepos}%
    \ifx\importmhmodule@repos\@empty%
    \else%
      \mhcurrentrepos{\importmhmodule@repos}%
    \fi%
    \adoptmodule[load=\MathHub{\mh@currentrepos/source/\importmhmodule@path},ext=\importmhmodul
    \mhcurrentrepos\mh@@repos%
  \fi%
  \ignorespaces%
}%
```

`\mhinputref`

```
\newrobustcmd\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%
}%
```

`\mhinput`

```
\let\mhinput\mhinputref%
```

`importmhmodulevia`

```
\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%
}%
```

```
\srefaddidkey{mhview}
\addmetakey{mhview}{display}
\addmetakey{mhview}{creators}
\addmetakey{mhview}{contributors}
\addmetakey{mhview}{srccite}
\addmetakey*{mhview}{title}
\addmetakey{mhview}{fromrepos}
\addmetakey{mhview}{torepos}
\addmetakey{mhview}{frompath}
\addmetakey{mhview}{topath}
\addmetakey[sms]{mhview}{ext}
```

`mhview` the MathHub version

```
\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
```

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

```
\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}%
}{}%
```

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

```
\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
⟨/modules⟩
```

EdN:2 [2]

```
⟨*modules.ltxml⟩
DefKeyVal('mhview','id','Semiverbatim');
DefKeyVal('mhview','fromrepos','Semiverbatim');
DefKeyVal('mhview','torepos','Semiverbatim');
DefKeyVal('mhview','frompath','Semiverbatim');
DefKeyVal('mhview','topath','Semiverbatim');
DefKeyVal('mhview','title','Semiverbatim');
DefKeyVal('mhview','creators','Semiverbatim');
DefKeyVal('mhview','contributors','Semiverbatim');
DefKeyVal('mhview','display','Semiverbatim');
DefKeyVal('mhview','ext','Semiverbatim');
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,
```

[2] EdNote: MK: sort these into the rest.

```
    Invocation(T_CS('\begin{viewenv}'),$keyvals,$from_arg,$to_arg)->unlist
  );
});
DefMacroI('\end{mhview}',undef,'\end{viewenv}');

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}');

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);

DefConstructor('\usemhmodule OptionalKeyVals:importmhmodule {}',
   "<omdoc:uses from='?&GetKeyVal(#1,'load')(&canonical_omdoc_path(&GetKeyVal(#1,'load')))()###
   afterDigest => \&importMHmoduleI);

DefConstructor('\adoptmhmodule OptionalKeyVals:importmhmodule {}',
   "<omdoc:adopts from='?&GetKeyVal(#1,'load')(&canonical_omdoc_path(&GetKeyVal(#1,'load')))()#
   afterDigest => \&importMHmoduleI);

RawTeX('
\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}
\newcommand\mhinput[2][]{\def\@repos{#1}%
\edef\mh@@repos{\mh@currentrepos}%
\ifx\@repos\@empty\else\mhcurrentrepos{#1}\fi%
```

```
\input{\MathHub{\mh@currentrepos/source/#2}}%
\mhcurrentrepos\mh@@repos}
\newenvironment{importmhmodulevia}[3][]{\def\@repos{#1}%
\edef\mh@@repos{\mh@currentrepos}%
\ifx\@repos\@empty\else\mhcurrentrepos{#1}\fi%
\gdef\@@doit{\importmhmodule[#1]{#2}{#3}}
\begin{importmoduleenv}[load=\MathHub{\mh@currentrepos/source/#2}]{#3}}
{\end{importmoduleenv}\aftergroup\@@doit}
');
⟨/modules.ltxml⟩
```

4.4 Finale

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

```
⟨*ltxml⟩
1;
⟨/ltxml⟩
```