MathHub Support for sTEX[*]

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

October 1, 2020

Abstract

The mathhub package collection 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 MathHub file system layout, which has co-eveolved with the MathHub.info portal for active documents, but is useful for organizing collections of sTEX documents in its own right.

Contents

[*] Version ? (last revised ?)

1 Introduction

As sTeX files tend to be highly interlinked semantically one of the most important practical problems to solve for managing larger collections is the management of (relative or absolute) paths. The `mathhub` package provides an infrastructure for supporting a regular ⇝ manageable file system layout schema that has co-eveolved with the MathHub.info portal for active documents, but is useful for organizing collections of sTeX documents in its own right. In particular, since the layout scheme is supported by the `lmh` tool (local mathhub) [lmh], which automates many management tasks.

MathHub (`http://MathHub.info`), is a portal and archive for flexiformal mathematics. It hosts much of the sTeX content MathHub on GIT repositories (public and private escrow) for mathematical documentation projects. MathHub supports online and offline (via `lmh`) authoring and document development infrastructure, and a rich, interactive reading interface.

The **MathHub file system layout** has a **MathHub root folder** (e.g. `~/localmh/MathHub`) which contains all sTeX sources, which are in turn organized in **MathHub archives** [Hor+11]. These are organized in a two-level folder system that is compatible by GIT repository managers like GitHub [GH] and GitLab [GL]. Even though it is not necessary for the `mathhub` package we will assume that these are GIT repositories, which have names of the form ⟨*group*⟩`/`⟨*arch*⟩, where ⟨*group*⟩ is a MathHub-unique repository group and ⟨*arch*⟩ a MathHub archive name that is ⟨*group*⟩-unique.

The MathHub archives have a prescribed structure; see [Hor+11] for details. For our purposes, we only need two aspects:

- the sTeX sources are all in a top-level subdirectory `source`,
- there is a top-level sub-directory `META-INF` with a manifest file `MANIFEST.MF` which consists of lines of the form ⟨*key*⟩`:` ⟨*values*⟩.

For the purposes of the `mathhub` package we assume that the `MANIFEST.MF` file has at least the `id` key specified and the the value is exactly ⟨*group*⟩`/`⟨*arch*⟩. Furthermore, we assume that the `MATHHUB` environment variable is set with the system path to the MathHub root folder.

With this information the mechanics of the MathHub archive structure can be hidden from the sTeX author with MathHub-enabled versions of the sTeX macros (let's call them **mh-variants**), which are defined in the `mh`-packages of the `mathhub` bundle, which we document in this manual. The mechanics of the `mathhub` bundle is as follows: For most sTeX package ⟨*pack*⟩`.sty` there is a `mh`-variant ⟨*pack*⟩`-mh.sty`, and ⟨*pack*⟩`.sty` takes the option `mh`. When that is given (by calling `\usepackage[mh]{`⟨*pack*⟩`}), then \meta{pack}`.sty `inputs`⟨*pack*⟩`-mh.sty` from the `mathhub` bundle, which augments the ⟨*pack*⟩ package with MathHub functionality.

EdN:1 [1]

[1] EdNote: document lmh here, how to install, what it does (build system)

2 The User Interface

2.1 `mathhub.sty`: General Infrastructure

`\mhcurrentrepos`

The next information needed for referencing in the MathHub file system layout is the repository of the referencing file and that of the referenced one. The latter will be part of the mh-variants of the referencing commands introduced below, the former, can be declared by the `\mhcurrentrepos` macro relative to the MathHub root path. Thus a typical top-level sTeX file will have lines like the following in the preamble:

```
\defpath{MathHub}{/user/foo/localmh/MathHub}
\mhcurrentrepos{group/repos}
```

They declare that it resides at the path `/user/foo/localmh/MathHub/group/repos` and declares the MathHub root path. But this fixed declaration makes the sTeX files less mobile, therefore it is a better idea to externalize these declarations into an external file (usually called `localpaths.tex` that is not under GIT control since it contains system-specific path information) and `\input` that in the sTeX file instead. Indeed `lmh` can generate these files automatically, which simplifies the management significantly.

`\libinput`

Given a systematic grouping in the MathHub file layout scheme, sTeX files in the same repository (and often even in the same group) share much of the preamble material. Thus it makes sense to centralize that in external (shared) files and situate it at the group and repository levels: at the group level. Fort the group level, the MathHub file system layout uses a specical repository ⟨*group*⟩`/meta-inf/lib` and at the repository level we use ⟨*group*⟩`/`⟨*repos*⟩`/lib` for such files. The `\libinput` macro supports this practice: `\libinput{`⟨*filename*⟩`}` macro inputs the files ⟨*group*⟩`/meta-inf/lib/`⟨*filename*⟩ and then ⟨*group*⟩`/`⟨*repos*⟩`/lib/`⟨*filename*⟩ if they exist. Thus a typical top-level sTeX file has the following lines in the preamble:

```
\input{localpaths}
\libinput{preamble}
```

`\libusepackage`

The `\libusepackage` is analogous. [2]

EdN:2

2.2 `omdoc-mh.sty`: MH Document Infrastructure

`\addmhbibresource`

The `\addmhbibresource` macro is a variant of `\addbibresource` from bibLaTeX with repository support. Concretely, `\addmhbibresource[`⟨*repos*⟩`]{`⟨*path*⟩`}` expands to `\addbibresource{`⟨*MathHub*⟩`/`⟨*reponame*⟩`/`⟨*path*⟩`}`, where ⟨*reponame*⟩ is ⟨*repo*⟩ if that is non-empty and the current repository else. Note that in contrast to the other MH variants, this does not add the `/source/` into the path, since bibTeX files are often put into the `lib` directory parallel to `source`.

[2] EdNote: explain the pre.tex and post.tex setup for sTeX modules (see the omdoc package)

2.3 modules-mh.sty: MH Variants for Modules

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

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

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

```
\importmhmodule[mhrepos=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 one of the following forms

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

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

\mhcurrentrepos 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.

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

\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.

Caveat if you want to use the MathHub support macros, then every time a module is imported or a document fragment is included from another repository, 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.

2.4 omtext-mh.sty: MH Variants for OMText

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

```
\mycgraphics{\MathHub{fooMH/bar/source/baz/foobar}}
```

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

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

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

2.5 smultiling-mh.sty: MH Variants for Multilinguality

mhmodsig
mhmodnl

The mhmodsig and mhmodnl environments are the MH variants of the modsig and modnl environments from the smultiing package. Just as in the other MH packages, mhmodnl takes additional mhrepos and path keys and combine them to load key of modnl. Instead of writing

```
\begin{modnl}[load=\MathHub{fooMH/bar/source/baz/foobar}]{foobar}{en}
```

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

```
\begin{modnl}[mhrepos=fooMH/bar,path=baz/foobar]{foobar}{en}
```

mhmodsig is just a notational variant of modsig that allows to keep the sources uniform.

2.6 structview-mh.sty: MH Variants for Structures and Views

EdN:3

[3]

2.7 mikoslides-mh.sty: Support for MiKo Slides

\mhframeimage

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

```
\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}
```

\mhinputref*

If we want to transclude a the contents of a file as a note, we can use the \mhinputref* macro: \mhinputref*[foo]{bar} is equivalent to

```
\begin{note}
\mhinputref[foo]{bar}
\end{note}
```

2.8 problem-mh.sty: Support for Problems

\includemhproblem

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

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

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

```
\includemhproblem[mhrepos=fooMH/bar,pts=7]{baz/foobar}
```

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

2.9 hwexam-mh.sty: Support for Assignments

\includemhassignment

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

```
\includeassignment[number=3]{\MathHub{fooMH/bar/source/baz/foobar}}
```

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

```
\includemhassignment[mhrepos=fooMH/bar,number=3]{baz/foobar}
```

2.10 lstmh.sty: Support for Listings

\lstinputmhlisting

The \lstinputmhlisting macro is a variant of \lstinputlisting with repository support. Instead of writing

```
\lstinputlisting[language=XML]{\MathHub{fooMH/bar/source/baz/foobar.xml}}
```

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

```
\lstinputmhlisting[mhrepos=fooMH/bar,language=XML]{baz/foobar.xml}
```

[3] EdNote: 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].

1. none reported yet.

4 Implementation

We need to set up the packages by requiring the `metakeys` package [Koh18] to be loaded (in the right version).

```
⟨*package⟩
\ProvidesPackage{mathhub}[2010/10/01 v1.2 Basic MathHub functionality]
\RequirePackage{keyval}
\RequirePackage{pathsuris}
\if@latexml\else
\RequirePackage{xparse}
\RequirePackage{expl3}
\fi
```

4.1 `mathhub.sty`: General Infrastructure

`\mhcurrentrepos`
`\@mhcurrentrepos`

`\mhcurrentrepos` is used to initialize the current repository. If the repository 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 repository.

```
\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}}%
```

`\libinput`

the `\libinput` macro inputs from the `lib` directory of the MathHub repository and then the `meta-inf/lib` repository of the group, if they exist. Since in practice nested libinputs may occur, we make sure that we stash the old values of `\mh@inffile` and `\mh@libfile` and restore them at the end.

```
\def\modules@@first#1/#2;{#1}
\newcommand\libinput[1]{%
\edef\@mh@group{\expandafter\modules@@first\mh@currentrepos;}
\let\orig@inffile\mh@inffile\let\orig@libfile\mh@libfile
\def\mh@inffile{\MathHub{\@mh@group/meta-inf/lib/#1}}
\def\mh@libfile{\MathHub{\mh@currentrepos/lib/#1}}%
\IfFileExists\mh@inffile{\input\mh@inffile}{}%
\IfFileExists\mh@inffile{}{\IfFileExists\mh@libfile{}{%
  {\PackageError{mathhub}
    {Library file missing; cannot input #1.tex\MessageBreak%
    Both \mh@libfile.tex\MessageBreak and \mh@inffile.tex\MessageBreak%
    do not exist}%
  {Check whether the file name is correct}}}}
\IfFileExists\mh@libfile{\input\mh@libfile\relax}{}
\let\mh@inffile\orig@inffile\let\mh@libfile\orig@libfile}
```

\libusepackage the \libusepackage is analogous to \libinput

```
\newcommand\libusepackage[2][]{%
\edef\@mh@group{\expandafter\modules@@first\mh@currentrepos;}
\let\orig@inffile\mh@inffile\let\orig@libfile\mh@libfile
\edef\mh@inffile{\MathHub{\@mh@group/meta-inf/lib/#2}}
\edef\mh@libfile{\MathHub{\mh@currentrepos/lib/#2}}%
\IfFileExists{\mh@inffile.sty}{\usepackage[#1]{\mh@inffile}}{}%
\IfFileExists {\mh@inffile.sty}{}{\IfFileExists{\mh@libfile.sty}{}{%
  {\PackageError{mathhub}
    {Library file missing; cannot use package #2.sty\MessageBreak%
    Both \mh@libfile.sty\MessageBreak and \mh@inffile.sty\MessageBreak%
    do not exist}%
  {Check whether the file name is correct}}}}
\IfFileExists{\@libfile.sty}{\usepackage[#1]{\@libfile}}{}}
```

Generally, the TeX formatter pdflatex needs to know the file system paths of the referenced sTeX files: usually long relative paths. The pathsuris package [KGA18] from the sTeX bundle makes this somewhat more palatable by supplying the \defpath macro, which we can use to set the MathHub root path, e.g. by \defpath{MathHub}{/user/foo/localmh/MathHub} (we will assume this setting for all examples below). Fortunately, we can compute this automatically.

The next piece of code is adapted from https://tex.stackexchange.com/questions/62010/can-i-access-system-environment-variables-from-latex-for-instance-home, check there if it breaks. We use it to parse the MATHHUB environment variable via kpsewhich (LaTeX can run this even in paranoid mode) and then set the MathMub path using \defpath.

```
\ExplSyntaxOn
\sys_get_shell:nnN{kpsewhich ~ --var-value ~ MATHHUB} { } \MATHHUB
\tl_trim_spaces:N \MATHHUB
\ifx\MATHHUB\empty\else
\def\temp_def_path#1{\defpath{MathHub}{#1}}
\expandafter\temp_def_path\expandafter{\MATHHUB}
\fi
\ExplSyntaxOff
```

Next we find the manifest of the MathHub archive: a file MANIFEST.MF up and over in the file system tree.

```
\def\findmanifest#1{
  \@cpath{#1}
  \ifx\@CanPath\@Slash
    \def\manifest@mf{}
  \else\ifx\@CanPath\@empty
      \def\manifest@mf{}
  \else
    \IfFileExists{\@CanPath/MANIFEST.MF}{
      \edef\manifest@mf{\@CanPath/MANIFEST.MF}
    }{
    \IfFileExists{\@CanPath/META-INF/MANIFEST.MF}{
      \edef\manifest@mf{\@CanPath/META-INF/MANIFEST.MF}
    }{
    \IfFileExists{\@CanPath/meta-inf/MANIFEST.MF}{
      \edef\manifest@mf{\@CanPath/meta-inf/MANIFEST.MF}
    }{
      \findmanifest{\@CanPath/..}
    }}}
  \fi\fi
}
```

the next macro is a helper function for parsing MANIFEST.MF

```
\def\split@manifest@key{
  \IfSubStr{\manifest@line}{\@Colon}{
      \StrBefore{\manifest@line}{\@Colon}[\manifest@key]
      \StrBehind{\manifest@line}{\@Colon}[\manifest@line]
      \trimstring\manifest@line
      \trimstring\manifest@key
  }{
      \def\manifest@key{}
  }
}
```

the next helper function iterates over lines in MANIFEST.MF

```
\def\parse@manifest@loop{
  \ifeof\@manifest
  \else
    \read\@manifest to \manifest@line\relax
    \split@manifest@key
    % id
    \IfStrEq\manifest@key{id}{
        \xdef\manifest@mf@id{\manifest@line}
    }{
    % narration-base
    \IfStrEq\manifest@key{narration-base}{
        \xdef\manifest@mf@narr{\manifest@line}
    }{
    % namespace
    \IfStrEq\manifest@key{source-base}{
        \xdef\manifest@mf@ns{\manifest@line}
    }{
    \IfStrEq\manifest@key{ns}{
        \xdef\manifest@mf@ns{\manifest@line}
    }{
    % dependencies
    \IfStrEq\manifest@key{dependencies}{
        \xdef\manifest@mf@deps{\manifest@line}
    }{

    }}}}}
    \parse@manifest@loop
  \fi
}
```

and finally, we find path of main file

```
\begingroup
  \edef\oldpercentcatcode{\the\catcode`\%}
  \catcode`\%=12
  \def\percent{%}
  \catcode`\%=\oldpercentcatcode
  \ExplSyntaxOn
    \edef\windowsstring{\detokenize{windows}}
    \edef\os_string{\expandafter\detokenize\expandafter{\c_sys_platform_str}}
    \ifx\os_string\windowsstring
      \edef\cmd_string{kpsewhich ~ -expand-var ~ \percent CD\percent}
    \else
      \edef\cmd_string{kpsewhich ~ -var-value ~ PWD}
    \fi
\expandafter\sys_get_shell:nnN\expandafter{\cmd_string} { } \mainfile_dir
     \tl_trim_spaces:N \mainfile_dir
\xdef\mainfile_dir{\expandafter\detokenize\expandafter{\mainfile_dir}}
  \ExplSyntaxOff
\endgroup
\expandafter\let\expandafter\mathhub@maindir\csname mainfile_dir\endcsname
```

We read in and parse MANIFEST.MF

```
\findmanifest{\mathhub@maindir}
\begingroup
  \gdef\manifest@mf@id{}
  \gdef\manifest@mf@narr{}
  \gdef\manifest@mf@ns{}
  \gdef\manifest@mf@deps{}
  \newread\@manifest
  \openin\@manifest\manifest@mf
  \parse@manifest@loop
  \closein\@manifest
\endgroup
```

Finally – and that is the ultimate goal of all of the above, we set the current repos.

```
\mhcurrentrepos{\manifest@mf@id}
⟨/package⟩
```

4.2 omdoc--mh.sty: MH Document Infrastructure

```
⟨*omdoc⟩
\ProvidesPackage{omdoc-mh}[2019/03/20 v1.1 MathHub support for OMDoc Documents]
\RequirePackage{mathhub}
```

\addmhbibresource

```
\newcommand\addmhbibresource[2][]{%
  \def\@repos{#1}%
  \ifx\@repos\@empty%
  \addbibresource{\MathHub{\mh@currentrepos/#2}}%
  \else
  \addbibresource{\MathHub{\@repos/#2}}%
  \fi%
  \ignorespacesandpars}%
⟨/omdoc⟩
```

4.3 modules-mh.sty: MH Variants for Modules

```
⟨*modules⟩
\ProvidesPackage{modules-mh}[2019/03/20 v1.1 MathHub support for the sTeX modules package]
\RequirePackage{mathhub}
```

\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.

```
\srefaddidkey{importmhmodule}%
\addmetakey{importmhmodule}{mhrepos}%
\addmetakey{importmhmodule}{path}%
\addmetakey{importmhmodule}{dir}%
\addmetakey[sms]{importmhmodule}{ext}%
\addmetakey[false]{importmhmodule}{conservative}[true]%
\newcommand\importmhmodule[2][]{%
\metasetkeys{importmhmodule}{#1}%
\ifx\importmhmodule@dir\@empty%
\edef\@path{\importmhmodule@path}%
\else\edef\@path{\importmhmodule@dir/#2}\fi%
\ifx\@path\@empty% if module name is not set
\importmodule[id=\importmhmodule@id]{#2}%
\else%
\edef\mh@@repos{\mh@currentrepos}% remember so that we can reset it.
\ifx\importmhmodule@mhrepos\@empty% if in the same repos
\relax% no need to change mh@currentrepos, i.e, current directory.
\else%
\mhcurrentrepos{\importmhmodule@mhrepos}% change it.
\fi%
\importmodule[load=\MathHub{\mh@currentrepos/source/\@path},
                         ext=\importmhmodule@ext,id=\importmhmodule@id]{#2}%
\mhcurrentrepos{\mh@@repos}% after importing, reset to old value
\fi%
\ignorespacesandpars}
```

and now the analogs

\usemhmodule

```
\newcommand\usemhmodule[2][]{%
\metasetkeys{importmhmodule}{#1}%
\ifx\importmhmodule@dir\@empty%
\edef\@path{\importmhmodule@path}%
\else\edef\@path{\importmhmodule@dir/#2}\fi%
\ifx\@path\@empty%
\usemodule[id=\importmhmodule@id]{#2}%
\else%
\edef\mh@@repos{\mh@currentrepos}%
\ifx\importmhmodule@mhrepos\@empty%
\else\mhcurrentrepos{\importmhmodule@mhrepos}\fi%
\usemodule[load=\MathHub{\mh@currentrepos/source/\@path},
                      ext=\importmhmodule@ext,id=\importmhmodule@id]{#2}%
\mhcurrentrepos\mh@@repos%
\fi%
\ignorespacesandpars}
```

\mhinputref

```
\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%
\ignorespacesandpars}
```

\mhinput

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

4.4 omtext-mh.sty: MH Variants for OMText

```
⟨*omtext⟩
\ProvidesPackage{omtext-mh}[2019/03/20 v1.1 MathHub support for the sTeX omtext package]
\RequirePackage{mathhub}
```

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

```
\def\Gin@mhrepos{}
\define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
\newcommand\mhgraphics[2][]{\setkeys{Gin}{#1}%
\edef\mh@@repos{\mh@currentrepos}%
\ifx\Gin@mhrepos\@empty\includegraphics[#1]{\MathHub{\mh@currentrepos/source/#2}}%
\else\includegraphics[#1]{\MathHub{\Gin@mhrepos/source/#2}}\fi
\def\Gin@mhrepos{}\mhcurrentrepos\mh@@repos}
\newcommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
```

The following macros are deprecated.

```
\newcommand\mhcgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}
  \PackageWarning{omtext-mh}{\protect\mhcgraphics\space is deprecated, use \protect\cmhgraphics
\newcommand\mhbgraphics[2][]{\fbox{\mhgraphics[#1]{#2}}
  \PackageWarning{omtext-mh}{\protect\mhbgraphics\space is deprecated, use
    \protect\mhgraphics\space and {center} instead}}
\newcommand\mhcbgraphics[2][]{\begin{center}\fbox{\mhgraphics[#1]{#2}}\end{center}
  \PackageWarning{omtext-mh}{\protect\mhcbgraphics\space is deprecated, use
    \protect\mhgraphics,\space {center}, and \protect\fbox\space instead}}
⟨/omtext⟩
```

4.5 smultiling-mh.sty: MH Variants for Multilinguality

```
⟨*smultiling⟩
\ProvidesPackage{smultiling-mh}[2019/03/20 v1.1 MathHub support for the sTeX smultiling package
\RequirePackage{mathhub}
```

mhmodsig

```
\newenvironment{mhmodsig}{\begin{modsig}}{\end{modsig}}
```

mhmodnl:*

```
\addmetakey{mhmodnl}{mhrepos}
\addmetakey{mhmodnl}{path}
\addmetakey*{mhmodnl}{title}
\addmetakey*{mhmodnl}{creators}
\addmetakey*{mhmodnl}{contributors}
\addmetakey{mhmodnl}{srccite}
\addmetakey{primary}{mhmodnl}[yes]
```

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

```
\newenvironment{mhmodnl}[3][]{\metasetkeys{mhmodnl}{#1}\def\@test{#1}%
\edef\@repos{\ifx\mhmodnl@mhrepos\@empty\mh@currentrepos\else\mhmodnl@mhrepos\fi}%
\edef\@load{\MathHub{\@repos/source/\ifx\mhmodnl@path\@empty #2\else\mhmodnl@path\fi}}%
\ifx\@test\@empty\begin{modnl}[load=\@load]{#2}{#3}\else\begin{modnl}[load=\@load,#1]{#2}{#3}\f
\ignorespacesandpars}
{\end{modnl}\ignorespacesandparsafterend}
```

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

```
\newenvironment{mhviewsig}[4][]{% keys, id, from, to
\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%
\ignorespacesandpars}
{\end{mhview}\ignorespacesandparsafterend}
```

mhviewnl The mhviewnl environment is just a layer over the mhview environment with the keys and language suitably adapted.[4] EdN:4

[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*⟩!

```
\newenvironment{mhviewnl}[5][]{% keys, id, lang, from, to
\def\@test{#1}\ifx\@test\@empty%
\begin{mhview}[id=#2.#3,ext=tex]{#4}{#5}\else%
\begin{mhview}[id=#2.#3,#1,ext=tex]{#4}{#5}\fi%
\ignorespacesandpars}
{\end{mhview}\ignorespacesandparsafterend}
⟨/smultiling⟩
```

4.6 structview-mh.sty: MH Variants for Structures and Views

```
⟨*structview⟩
\ProvidesPackage{structview-mh}[2019/03/20 v1.1 MathHub support for the sTeX structview package
\RequirePackage{mathhub}
```

mhstructure

```
\newenvironment{mhstructure}[3][]{%
  \gdef\@@doit{\importmhmodule[#1]{#3}}%
  \ifmod@show\par\noindent structure import "#2" from module #3 \@@doit\fi%
  \ignorespacesandpars}
{\aftergroup\@@doit\ifmod@show end import\fi%
  \ignorespacesandparsafterend}
```

importmhmodulevia this is now deprecated, we give an error

```
\newenvironment{importmhmodulevia}[2][]%
{\PackageError{structview-mh}%
  {The {importmhmodulevia} environment is deprecated}{use the {mhstructure} instead!}%
  \begin{mhstructure}[#1]{missing}{#2}}
{\end{mhstructure}}
```

```
\srefaddidkey{mhview}
\addmetakey{mhview}{display}
\addmetakey{mhview}{creators}
\addmetakey{mhview}{contributors}
\addmetakey{mhview}{srccite}
\addmetakey*{mhview}{title}
\addmetakey{mhview}{type}
\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}%
  \ignorespacesandpars}
```

```
{\end{@mhview}\ignorespacesandparsafterend}
\ifmod@show\surroundwithmdframed{mhview}\fi
```

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

```
\newenvironment{@mhview}[2]{%from, to
  \importmhmodule[mhrepos=\mhview@fromrepos,path=\mhview@frompath,ext=\mhview@ext]{#1}%
  \importmhmodule[mhrepos=\mhview@torepos,path=\mhview@topath,ext=\mhview@ext]{#2}%
}{}%
```

mhviewsketch The mhviewsketch environment is deprecated, we give an error

```
\newenvironment{mhviewsketch}[3][]%
{\PackageError{structview}%
  {The {mhviewsketch} environment is deprecated}{use the {mhview} instead!}%
  \begin{mhview}[#1]{#2}{#3}}
{\end{mhview}}
```

mhinlineView Analogous modification to inlineView

```
\newenvironment{mhinlineView}[2][]% keys, source
{\metasetkeys{mhview}{#1}\sref@target%
  \importmhmodule[mhrepos=\mhview@fromrepos,path=\mhview@frompath,ext=\mhview@ext]{#2}%
  \ignorespacesandpars}
{\ignorespacesandpars}
```

mhinlineview

```
\newcommand\mhinlineview[3][]{\begin{mhinlineView}[#1]{#2}{\mod@id}#3\end{mhinlineView}}
⟨/structview⟩
```

4.7 mikoslides-mh.sty: Support for MiKo Slides

```
⟨*mikoslides⟩
\ProvidesPackage{mikoslides-mh}[2019/03/20 v1.1 MathHub support for the sTeX mikoslides package
\RequirePackage{mathhub}
```

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

```
\def\Gin@mhrepos{}
\define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
\newcommand\mhframeimage[2][]{%
  \setkeys{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%
}%
```

\mhinputref*

```
\let\orig@mhinputref\mhinputref
```

```
\def\mhinputref{\@ifstar\nmhinputref\orig@mhinputref}
\newcommand\nmhinputref[2][]{\ifnotes\orig@mhinputref[#1]{#2}\fi}
⟨/mikoslides⟩
```

4.8 problem-mh.sty: Support for Problems

```
⟨*problem⟩
\ProvidesPackage{problem-mh}[2019/03/20 v1.1 MathHub support for the sTeX problem package]
\RequirePackage{mathhub}
```

\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.

```
\addmetakey{inclprob}{mhrepos}
\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⟩
```

4.9 hwexam-mh.sty: Support for Assignments

```
⟨*hwexam⟩
\ProvidesPackage{hwexam-mh}[2019/03/20 v1.1 MathHub support for the sTeX hwexam package]
\RequirePackage{mathhub}
```

\inputmhassignment

The \inputmhassignment 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.

```
\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}
\newcommand\includemhassignment[2][]{\newpage\inputmhassignment[#1]{#2}}
⟨/hwexam⟩
```

4.10 tikzinput-mh.sty: Support for Assignments

```
⟨*tikzinput⟩
\ProvidesPackage{tikzinput-mh}[2019/03/20 v1.1 MathHub support for the sTeX tikzinput package]
\RequirePackage{mathhub}
\RequirePackage{pathsuris}

\define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
\newcommand\mhtikzinput[2][]{\def\Gin@mhrepos{}\setkeys{Gin}{#1}%
\edef\mh@@repos{\mh@currentrepos}%
\ifx\Gin@mhrepos\@empty\tikzinput[#1]{\MathHub{\mh@currentrepos/source/#2}}%
\else\mhcurrentrepos\Gin@mhrepos\tikzinput[#1]{\MathHub{\Gin@mhrepos/source/#2}}\fi
\def\Gin@mhrepos{}\mhcurrentrepos\mh@@repos}
\newcommand\cmhtikzinput[2][]{\begin{center}\mhtikzinput[#1]{#2}\end{center}}
⟨/tikzinput⟩
```

4.11 lstmh.sty: Support for Listings

```
⟨*lst⟩
\ProvidesPackage{lstmh}[2019/03/20 v1.1 MathHub support for the listings package]
\RequirePackage{mathhub}
\RequirePackage{pathsuris}
\RequirePackage{listings}
```

```
\define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
\newcommand\lstinputmhlisting[2][]{\def\lst@mhrepos{}\setkeys{lst}{#1}%
\edef\mh@@repos{\mh@currentrepos}%
\ifx\lst@mhrepos\@empty\lstinputlisting[#1]{\MathHub{\mh@currentrepos/source/#2}}%
\else\lstinputlisting[#1]{\MathHub{\lst@mhrepos/source/#2}}\fi
\def\lst@mhrepos{}\mhcurrentrepos\mh@@repos}
\newcommand\clstinputmhlisting[2][]{\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
⟨/lst⟩
```

Change History

v1.0

General: Deprecated `mhviewsketch` 1

moved all MH functionality into one DTX file 1

v1.1

General: Changed the semantics of `libinput` to input first the `meta-inf`-level and then repos-level file; this allows more sharing and does not break MathHub content (only one of them currently exists) . . 1

References

[GH] *GitHub: Build software better, together.* URL: `http://github.com` (visited on 02/24/2014).

[GL] *The first single application for the entire DevOps lifecycle – GitLab.* URL: `http://gitlab.com` (visited on 01/12/2019).

[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`.

[KGA18] Michael Kohlhase, Deyan Ginev, and Rares Ambrus. *`modules.sty`: Semantic Macros and Module Scoping in sTeX.* Tech. rep. 2018. URL: `https://github.com/sLaTeX/sTeX/raw/master/sty/pathsuris/pathsuris.pdf`.

[Koh18] Michael Kohlhase. *`metakeys.sty`: A generic framework for extensible Metadata in LaTeX.* Tech. rep. 2018. URL: `https://github.com/sLaTeX/sTeX/raw/master/sty/metakeys/metakeys.pdf`.

[lmh] *`lmh`: A cross-repository administration tool for the local authoring in MathHub.info.* URL: `https://github.com/KWARC/localmh` (visited on 02/05/2017).

[sTeX] *sTeX: A semantic Extension of TeX/LaTeX.* URL: `https://github.com/sLaTeX/sTeX` (visited on 05/15/2015).