The sTeX3 Package [*]

Michael Kohlhase, Dennis Müller
FAU Erlangen-Nürnberg
http://kwarc.info/

2022-03-14

Abstract

sTeX is a collection of LaTeX package that allow to markup documents semantically without leaving the document format, essentially turning LaTeX into a document format for mathematical knowledge management (MKM).
sTeX augments LaTeX with

- *Semantic macros* that denote and distinguish between mathematical concepts, operators, etc. independent of their notational presentation,
- A powerful *module system* that allows for authoring and importing individual fragments containing document text and/or semantic macros, independent of – and without hard coding – directory paths relative to the current document,
- A mechanism for exporting sTeX documents to (modular) XHTML, preserving all the semantic information for semantically informed knowledge management services.

This is the full documentation of sTeX. It consists of four parts:

- Part I is a general manual for the sTeX package and associated software. It is primarily directed at end-users who want to use sTeX to author semantically enriched documents.
- Part II documents the macros provided by the sTeX package. It is primarily directed at package authors who want to build on sTeX, but can also serve as a reference manual for end-users.
- Part III documents additional packages that build on sTeX, primarily its module system. These are not part of the sTeX package itself, but useful additions enabled by sTeX package functionality.
- Part IV is the detailled documentation of the sTeX package implementation.

[*]Version 3.0 (last revised 2022-03-14)

Contents

Part I

Manual

Boxes like this one contain implementation details that are mostly relevant for more advanced use cases, might be useful to know when debugging, or might be good to know to better understand how something works. They can easiyl be skipped on a first read.

Boxes like this one explain how some sTeX concept relates to the MMT/OMDoc system, philosophy or language.

Chapter 1

What is sTeX?

Formal systems for mathematics (such as interactive theorem provers) have the potential to significantly increase both the accessibility of published knowledge, as well as the confidence in its veracity, by rendering the precise semantics of statements machine actionable. This allows for a plurality of added-value services, from semantic search up to verification and automated theorem proving. Unfortunately, their usefulness is hidden behind severe barriers to accessibility; primarily related to their surface languages reminiscent of programming languages and very unlike informal standards of presentation.

sTeX minimizes this gap between informal and formal mathematics by integrating formal methods into established and widespread authoring workflows, primarily LaTeX, via non-intrusive semantic annotations of arbitrary informal document fragments. That way formal knowledge management services become available for informal documents, accessible via an IDE for authors and via generated *active* documents for readers, while remaining fully compatible with existing authoring workflows and publishing systems.

Additionally, an extensible library of reusable document fragments is being developed, that serve as reference targets for global disambiguation, intermediaries for content exchange between systems and other services.

Every component of the system is designed modularly and extensibly, and thus lay the groundwork for a potential full integration of interactive theorem proving systems into established informal document authoring workflows.

The general sTeX workflow combines functionalities provided by several pieces of software:

- The sTeX package to use semantic annotations in LaTeX documents,
- RusTeX to convert `tex` sources to (semantically enriched) `xhtml`,
- The MMT software, that extracts semantic information from the thus generated `xhtml` and provides semantically informed added value services.

Chapter 2

Quickstart

2.1 Setup

2.1.1 The sTeX IDE

TODO: VSCode Plugin

2.1.2 Manual Setup

Foregoing on the sTeX IDE, we will need several pieces of software; namely:

- **The sTeX-Package** available here.

  sTeX is also available on CTAN and in TeXLive.

- To make sure that sTeX too knows where to find its archives, we need to set a global system variable `MATHHUB`, that points to your local `MathHub`-directory (see section 3.2).

- EdN:1 **The Mmt System** available here[1]. We recommend following the setup routine documented here.

  Following the setup routine (Step 3) will entail designating a `MathHub`-directory on your local file system, where the MMT system will look for sTeX/MMT content archives.

- **sTeX Archives** If we only care about LaTeX and generating `pdfs`, we do not technically need MMT at all; however, we still need the `MATHHUB` system variable to be set. Furthermore, MMT can make downloading content archives we might want to use significantly easier, since it makes sure that all dependencies of (often highly interrelated) sTeX archives are cloned as well.

  Once set up, we can run `mmt` in a shell and download an archive along with all of its dependencies like this: `lmh install <name-of-repository>`, or a whole *group* of archives; for example, `lmh install smglom` will download all smglom archives.

- **RusTeX** The MMT system will also set up RusTeX for you, which is used to generate (semantically annotated) `xhtml` from tex sources. In lieu of using MMT, you can also download and use RusTeX directly here.

[1] EdNote: For now, we require the sTeX-branch, requiring manually compiling the MMT sources

2.2 A First sTeX Document

Having set everything up, we can write a first sTeX document. As an example, we will use the `smglom/calculus` and `smglom/arithmetics` archives, which should be present in the designated `MathHub`-folder, and write a small fragment defining the *geometric series*:

TODO: use some sTeX-archive instead of smglom, use a convergence-notion that includes the limit, mark-up the theorem properly

```latex
\documentclass{article}
\usepackage{stex,xcolor,stexthm}

\begin{document}
\begin{smodule}{GeometricSeries}
    \importmodule[smglom/calculus]{series}
    \importmodule[smglom/arithmetics]{realarith}

    \symdef{geometricSeries}[name=geometric-series]{\comp{S}}

    \begin{sdefinition}[for=geometricSeries]
        The \definame{geometricSeries} is the \symname{?series}
        \[\defeq{\geometricSeries}{\definiens{
            \infinitesum{\svar{n}}{1}{
                \realdivide[frac]{1}{
                    \realpower{2}{\svar{n}}
            }}
        }}.\]
    \end{sdefinition}

    \begin{sassertion}[name=geometricSeriesConverges,type=theorem]
    The \symname{geometricSeries} \symname{converges} towards $1$.
    \end{sassertion}
\end{smodule}
\end{document}
```

Compiling this document with `pdflatex` should yield the output

Definition 0.1. The **geometric series** is the series

$$S := \sum_{n=1}^{\infty} \frac{1}{2^n}.$$

Theorem 0.2. The geometric series converges towards 1.

Feel free to move your cursor over the various highlighted parts of the document – depending on your pdf viewer, this should yield some interesting (but possibly for now cryptic) information.

Remark 2.2.1:

Note that all of the highlighting, tooltips, coloring and the environment headers come from stexthm – by default, the amount of additional packages loaded is kept to a minimum and all the presentations can be customized, see chapter 6.

Let's investigate this document in detail now:

```
\begin{smodule}{GeometricSeries}
...
\end{smodule}
```

`smodule` First, we open a new *module* called `GeometricSeries`. This module is assigned a *globally unique* identifier (URI), which (depending on your pdf viewer) should pop up in a tooltip if you hover over the word **geometric series**.

```
\importmodule[smglom/calculus]{series}
\importmodule[smglom/arithmetics]{realarith}
```

`\importmodule` Next, we *import* two modules – `series` in the `smglom/calculus`-archive, and `realarith` in the `smglom/arithmetics`-archive. If we investigate these archives, we find the files `series.en.tex` and `realarith.en.tex` (respectively) in their respective `source`-folders, which contain the statements `\begin{smodule}{series}` and `\begin{smodule}{realarith}` (respectively).

The `\importmodule`-statements make all sTeX symbols and associated semantic macros (e.g. *`\infinitesum`*, *`\realdivide`*, *`\realpower`*) in the desired module available. Additionally, they "export" these symbols to all further modules which include the *current* module – i.e. if in some future module we would put `\importmodule{GeometricSeries}`, we would also have *`\infinitesum`* etc. at our disposal.

`\usemodule` If we only want to *use* the content of some module `Foo`, e.g. in remarks or examples, but none of the symbols in our current module actually *depend* on the content of `Foo`, we can use `\usemodule` instead – like `\importmodule`, this will make the module content available, but will *not* export it to other modules.

```
\symdef{GeometricSeries}[name=geometric-series]{\comp{S}}
```

`\symdef` Next, we introduce a new *symbol* with name `geometric-series` and assign it the semantic macro *`\geometricSeries`*. `\symdef` also immediately assigns this symbol a *notation*, namely S.

`\comp` The macro `\comp` marks the S in the notation as a *notational component*, as opposed to e.g. arguments to *`\geometricSeries`*. It is the notational components that get highlighted and associated with the corresponding symbol (i.e. in this case `geometricSeries`). Since *`\geometricSeries`* takes no arguments, we can wrap the whole notation in a `\comp`.

```
\begin{sdefinition}[for=geometricSeries]
...
\end{sdefinition}
\begin{sassertion}[name=geometricSeriesConverges,type=theorem]
...
\end{sassertion}
```

What follows are two sTeX-*statements* (e.g. definitions, theorems, examples, proofs, ...). These are semantically marked-up variants of the usual environments, which take additional optional arguments (e.g. `for=`, `type=`, `name=`). Since many LaTeX templates predefine environments like `definition` or `theorem` with different syntax, we use `sdefinition`, `sassertion`, `sexample` etc. instead. You can customize these environments to e.g. simply wrap around some predefined `theorem`-environment. That way, we can still use `sassertion` to provide semantic information, while being fully compatible with (and using the document presentation of) predefined environments.

In our case, the stexthm-package patches e.g. `\begin{sassertion}[type=theorem]` to use a `theorem`-environment defined (as usual) using amsthm.

```
The \definame{geometricSeries} is the \symname{?series}
```

`\symname`

The `\symname`-command prints the name of a symbol, highlights it (based on customizable settings) and associates the text printed with the corresponding symbol. If you hover over the word series in the pdf output, you should see a tooltip showing the full URI of the symbol used.

`\symref`

The `\symname`-command is a special case of the more general `\symref`-command, which allows customizing the precise text associated with a symbol.

`\definame`
`\definiendum`

The `sdefinition`-environment provides two additional macros, `\definame` and `\definiendum` which behave similar to `\symname` and `\symref`, but explicitly mark the symbols as *being defined* in this environment, to allow for special highlighting.

```
\[\defeq{\geometricSeries}{\definiens{
    \infinitesum{\svar{n}}{1}{
        \realdivide[frac]{1}{
            \realpower{2}{\svar{n}}
    }}
}}.\]
```

The next snippet – set in a math environment – uses several semantic macros imported from (or recursively via) `series` and `realarithmetics`, such as `\defeq`, `\infinitesum`, etc. In math mode, using a semantic macro inserts its (default) definition. A semantic macro can have several notations – in that case, we can explicitly choose a specific notation by providing its identifier as an optional argument; e.g. `\realdivide[frac]{a}{b}` will use the explicit notation named `frac` of the semantic macro `\realdivide`, which yields $\frac{a}{b}$ instead of a/b.

`\svar`

The `\svar{n}` command marks up the `n` as a variable with name `n` and notation `n`.

`\definiens`

The `sdefinition`-environment additionally provides the `\definiens`-command, which allows for explicitly marking up its argument as the *definiens* of the symbol currently being defined.

2.2.1 OMDoc/xhtml Conversion

So, if we run `pdflatex` on our document, then sTeX yields pretty colors and tooltips[1]. But sTeX becomes a lot more powerful if we additionally convert our document to `xhtml`.

TODO VSCode Plugin

Using RusTeX, we can convert the document to `xhtml` using the command `rustex -i /path/to/file.tex -o /path/to/outfile.xhtml`. Investigating the resulting file, we notice additional semantic information resulting from our usage of semantic macros, `\symref` etc. Below is the (abbreviated) snippet inside our `\definiens` block:

```
<mrow resource="" property="stex:definiens">
 <mrow resource="...?series?infinitesum" property="stex:OMBIND">
  <munderover displaystyle="true">
   <mo resource="...?series?infinitesum" property="stex:comp">Σ</mo>
   <mrow>
    <mrow resource="1" property="stex:arg">
     <mi resource="var://n" property="stex:OMV">n</mi>
    </mrow>
    <mo resource="...?series?infinitesum" property="stex:comp">=</mo>
    <mi resource="2" property="stex:arg">1</mi>
   </mrow>
   <mi resource="...?series?infinitesum" property="stex:comp">∞</mi>
  </munderover>
  <mrow resource="3" property="stex:arg">
   <mfrac resource="...?realarith?division#frac#" property="stex:OMA">
    <mi resource="1" property="stex:arg">1</mi>
    <mrow resource="2" property="stex:arg">
     <msup resource="...realarith?exponentiation" property="stex:OMA">
      <mi resource="1" property="stex:arg">2</mi>
      <mrow resource="2" property="stex:arg">
       <mi resource="var://n" property="stex:OMV">n</mi>
      </mrow>
     </msup>
    </mrow>
   </mfrac>
  </mrow>
 </mrow>
</mrow>
```

...containing all the semantic information. The Mmt system can extract from this the following OpenMath snippet:

```
<OMBIND>
  <OMID name="...?series?infinitesum"/>
  <OMV name="n"/>
  <OMLIT name="1"/>
  <OMA>
    <OMS name="...?realarith?division"/>
    <OMLIT name="1"/>
    <OMA>
      <OMS name="...realarith?exponentiation"/>
      <OMLIT name="2"/>
      <OMV name="n"/>
    </OMA>
  </OMA>
</OMBIND>
```

[1] ...and hyperlinks for symbols, and indices, and allows reusing document fragments modularly, and...

...giving us the full semantics of the snippet, allowing for a plurality of knowledge management services – in particular when serving the `xhtml`.

Remark 2.2.2:

Note that the `html` when opened in a browser will look slightly different than the `pdf` when it comes to highlighting semantic content – that is because naturally `html` allows for much more powerful features than `pdf` does. Consequently, the `html` is intended to be served by a system like MMT, which can pick up on the semantic information and offer much more powerful highlighting, linking and similar features, and being customizable by *readers* rather than being prescribed by an author.

Additionally, not all browsers (most notably Chrome) support MATHML natively, and might require additional external JavaScript libraries such as MathJax to render mathematical formulas properly.

Chapter 3

Creating sTeX Content

We can use sTeX by simply including the package with `\usepackage{stex}`, or – primarily for individual fragments to be included in other documents – by using the sTeX document class with `\documentclass{stex}` which combines the standalone document class with the stex package.

Both the stex package and document class offer the following options:

`lang` (⟨*language*⟩∗) Languages to load with the babel package.

`mathhub` (⟨*directory*⟩) MathHub folder to search for repositories – this is not necessary if the `MATHHUB` system variable is set.

`sms` (⟨*boolean*⟩) use *persisted* mode (not yet implemented).

`image` (⟨*boolean*⟩) passed on to tikzinput.

`debug` (⟨*log-prefix*⟩∗) Logs debugging information with the given prefixes to the terminal, or all if `all` is given. Largely irrelevant for the majority of users.

3.1 How Knowledge is Organized in sTeX

sTeX content is organized on multiple levels:

- sTeX **archives** (see section 3.2) contain individual `.tex`-files.
- These may contain sTeX **modules**, introduced via `\begin{smodule}{ModuleName}`.
- Modules contain sTeX **symbol declarations**, introduced via `\symdecl{symbolname}`, `\symdef{symbolname}` and some other constructions. Most symbols have a *notation* that can be used via a *semantic macro* `\symbolname` generated by symbol declarations.
- sTeX **expressions** finally are built up from usages of semantic macros.

↪M→
—M→
∼T⇝

- sTeX archives are simultaneously MMT archives, and the same directory structure is consequently used.
- sTeX modules correspond to OMDoc/MMT *theories*. `\importmodule`s (and

similar constructions) induce MMT `includes` and other *theory morphisms*, thus giving rise to a *theory graph* in the OMDoc sense.

- Symbol declarations induce OMDoc/MMT *constants*, with optional (formal) *type* and *definiens* components.
- Finally, sTeX expressions are converted to OMDoc/MMT terms, which use the syntax of OpenMath.

3.2 sTeX Archives

3.2.1 The Local MathHub-Directory

`\usemodule`, `\importmodule`, `\inputref` etc. allow for including content modularly without having to specify absolute paths, which would differ between users and machines. Instead, sTeX uses *archives* that determine the global namespaces for symbols and statements and make it possible for sTeX to find content referenced via such URIs.

All sTeX archives need to exist in the local `MathHub`-directory. sTeX knows where this folder is via one of three means:

1. If the sTeX package is loaded with the option `mathhub=/path/to/mathhub`, then sTeX will consider `/path/to/mathhub` as the local `MathHub`-directory.

2. If the `mathhub` package option is *not* set, but the macro `\mathhub` exists when the sTeX-package is loaded, then this macro is assumed to point to the local `MathHub`-directory; i.e. `\def\mathhub{/path/to/mathhub}\usepackage{stex}` will set the `MathHub`-directory as `path/to/mathhub`.

3. Otherwise, sTeX will attempt to retrieve the system variable `MATHHUB`, assuming it will point to the local `MathHub`-directory. Since this variant needs setting up only *once* and is machine-specific (rather than defined in tex code), it is compatible with collaborating and sharing tex content, and hence recommended.

3.2.2 The Structure of sTeX Archives

An sTeX archive `group/name` needs to be stored in the directory `/path/to/mathhub/group/name`; e.g. assuming your local `MathHub`-directory is set as `/user/foo/MathHub`, then in order for the `smglom/calculus`-archive to be found by the sTeX system, it needs to be in `/user/foo/MathHub/smglom/calculus`.

Each such archive needs two subdirectories:

- `/source` – this is where all your tex files go.
- `/META-INF` – a directory containing a single file `MANIFEST.MF`, the content of which we will consider shortly

An additional `lib`-directory is optional, and is where sTeX will look for files included via `\libinput`.

Additionally a *group* of archives `group/name` may have an additional archive `group/meta-inf`. If this `meta-inf`-archive has a `/lib`-subdirectory, it too will be searched by `\libinput` from all tex files in any archive in the `group/*`-group.

We recommend this additional directory structure in the `source`-folder of an sTeX archive:

- `/source/mod/` – individual sTeX modules, containing symbol declarations, notations, and `\begin{sparagraph}[type=symdoc,for=...]` environments for "encyclopedic" symbol documentations
- `/source/def/` – definitions
- `/source/ex/` – examples
- `/source/thm/` – theorems, lemmata and proofs; preferably proofs in separate files to allow for multiple proofs for the same statement
- `/source/snip/` – individual text snippets such as remarks, explanations etc.
- `/source/frag/` – individual document fragments, ideally only `\inputref`ing snippets, definitions, examples etc. in some desirable order
- `/source/tikz/` – tikz images, as individual `.tex`-files
- `/source/pic/` – image files.

3.2.3 MANIFEST.MF-Files

The `MANIFEST.MF` in the `META-INF`-directory consists of key-value-pairs, instructing sTeX (and associated software) of various properties of an archive. For example, the `MANIFEST.MF` of the `smglom/calculus`-archive looks like this:

```
id: smglom/calculus
source-base: http://mathhub.info/smglom/calculus
narration-base: http://mathhub.info/smglom/calculus
dependencies: smglom/arithmetics,smglom/sets,smglom/topology,
            smglom/mv,smglom/linear-algebra,smglom/algebra
responsible: Michael.Kohlhase@FAU.de
title: Elementary Calculus
teaser: Terminology for the mathematical study of change.
description: desc.html
```

Many of these are in fact ignored by sTeX, but some are important:

`id`: The name of the archive, including its group (e.g. `smglom/calculus`),

`source-base` or

`ns`: The namespace from which all symbol and module URIs in this repository are formed, see (TODO),

`narration-base`: The namespace from which all document URIs in this repository are formed, see (TODO),

`url-base`: The URL that is formed as a basis for *external references*, see (TODO),

`dependencies`: All archives that this archive depends on. sTeX ignores this field, but Mmt can pick up on them to resolve dependencies, e.g. for `lmh install`.

3.2.4 Using Files in sTeX Archives Directly

Several macros provided by sTeX allow for directly including files in repositories. These are:

`\mhinput`

`\mhinput[Some/Archive]{some/file}` directly inputs the file `some/file` in the `source`-folder of `Some/Archive`.

`\inputref`

`\inputref[Some/Archive]{some/file}` behaves like `\mhinput`, but wraps the input in a `\begingroup ... \endgroup`. When converting to `xhtml`, the file is not input at all, and instead an `html`-annotation is inserted that references the file.

In the majority of cases `\inputref` is likely to be preferred over `\mhinput`.

`\ifinput`

Both `\mhinput` and `\inputref` set `\ifinput` to "true" during input. This allows for selectively including e.g. bibliographies only if the current file is not being currently included in a larger document.

`\addmhbibresource`

`\addmhbibresource[Some/Archive]{some/file}` searches for a file like `\mhinput` does, but calls `\addbibresource` to the result and looks for the file in the archive root directory directly, rather than the `source` directory.

`\libinput`

`\libinput{some/file}` searches for a file `some/file` in

- the `lib`-directory of the current archive, and
- the `lib`-directory of a `meta-inf`-archive in (any of) the archive groups containing the current archive

and include all found files in reverse order; e.g. `\libinput{preamble}` in a `.tex`-file in `smglom/calculus` will *first* input `.../smglom/meta-inf/lib/preamble.tex` and then `../smglom/calculus/lib/preamble.tex`.

Will throw an error if *no* candidate for `some/file` is found.

`\libusepackage`

`\libusepackage[package-options]{some/file}` searches for a file `some/file.sty` in the same way that `\libinput` does, but will call `\usepackage[package-options]{path/to/some/file}` instead of `\input`.

Will throw an error if not *exactly one* candidate for `some/file` is found.

Remark 3.2.1:

A good practice is to have individual sTeX fragments follow basically this document frame:

```
\documentclass{stex}
\libinput{preamble}
\begin{document}
    ...
    \ifinputref \else \libinput{postamble} \fi
\end{document}
```

Then the `preamble.tex` files can take care of loading the generally required packages, setting presentation customizations etc. (per archive or archive group or both), and `postamble.tex` can e.g. print the bibliography, index etc.

3.3 Module, Symbol and Notation Declarations

3.3.1 The `smodule`-Environment

`smodule` A new module is declared using the basic syntax

`\begin{smodule}[options]{ModuleName}...\end{smodule}`.

A module is required to declare any new formal content such as symbols or notations (but not variables, which may be introduced anywhere).

The `smodule`-environment takes several optional arguments, all of which are optional:

`title` (⟨*token list*⟩) to display in customizations.

`type` (⟨*string*⟩∗) for use in customizations.

`deprecate` (⟨*module*⟩) if set, will throw a warning when loaded, urging to use ⟨*module*⟩ instead.

`id` (⟨*string*⟩) for cross-referencing.

`ns` (⟨*URI*⟩) the namespace to use. *Should not be used, unless you know precisely what you're doing.* If not explicitly set, is computed using `\stex_modules_current_namespace:`.

`lang` (⟨*language*⟩) if not set, computed from the current file name (e.g. `foo.en.tex`).

`sig` (⟨*language*⟩) if the current file is a translation of a file with the same base name but a different language suffix, setting `sig=<lang>` will preload the module from that language file. This helps ensuring that the (formal) content of both modules is (almost) identical across languages and avoids duplication.

`creators` (⟨*string*⟩∗) names of the creators.

`contributors` (⟨*string*⟩∗) names of contributors.

`srccite` (⟨*string*⟩) a source citation for the content of this module.

↪M→ —M→ ~T⇝ An sTeX module corresponds to an Mmt/OMDoc *theory*. As such it gets assigned a module URI (*universal resource identifier*) of the form `<namespace>?<module-name>`.

By default, opening a module will produce no output whatsoever, e.g.:

Example 1

Input:

```
 \begin{smodule}[title={This is Some Module}]{SomeModule}
    Hello World
\end{smodule}
```

Output:

```
Hello World
```

.

`\stexpatchmodule`

We can customize this behavior either for all modules or only for modules with a specific `type` using the command `\stexpatchmodule[optional-type]{begin-code}{end-code}`. Some optional parameters are then available in `\smodule*`-macros, specifically `\smoduletitle`, `\smoduletype` and `\smoduleid`.

For example:

Example 2

Input:

```
 \stexpatchmodule[display]
  {\textbf{Module (\smoduletitle)}\par}
  {\par\noindent\textbf{End of Module (\smoduletitle)}}

\begin{smodule}[type=display,title={Some New Module}]{SomeModule2}
   Hello World
\end{smodule}
```

Output:

Module (Some New Module)
Hello World
End of Module (Some New Module)

.

3.3.2 Declaring New Symbols and Notations

Inside an `smodule` environment, we can declare new sTeX symbols.

`\symdecl` The most basic command for doing so is using `\symdecl{symbolname}`. This introduces a new symbol with name `symbolname`, arity 0 and semantic macro *`\symbolname`*.

The starred variant `\symdecl*{symbolname}` will declare a symbol, but not introduce a semantic macro. If we don't want to supply a notation (for example to introduce concepts like "abelian", which is not something that has a notation), the starred variant is likely to be what we want.

↪M→ —M→ ~T⇝ `\symdecl` introduces a new OMDoc/Mmt constant in the current module (=OMDoc/Mmt theory). Correspondingly, they get assigned the URI `<module-URI>?<constant-name>`.

Without a semantic macro or a notation, the only meaningful way to reference a symbol is via `\symref`,`\symname` etc.

Example 3

Input:

```
\symdecl*{foo}
Given a \symname{foo}, we can...
```

Output:

Given a foo, we can...

.

Obviously, most semantic macros should take actual *arguments*, implying that the symbol we introduce is an *operator* or *function*. We can let `\symdecl` know the *arity* (i.e. number of arguments) of a symbol like this:

Example 4

Input:

```
\symdecl{binarysymbol}[args=2]
\symref{binarysymbol}{this} is a symbol taking two arguments.
```

Output:

this is a symbol taking two arguments.

.

\notation

In that case, we probably want to supply a notation as well, in which case we can finally actually use the semantic macro in math mode. We can do so using the \notation command, like this:

Example 5

Input:

```
\notation{binarysymbol}{\text{First: }#1\text{; Second: }#2}
$\binarysymbol{a}{b}$
```

Output:

First: a; Second: b

.

↪M→ Applications of semantic macros, such as \binarysymbol{a}{b} are translated to
—M→ MMT/OMDoc as OMA-terms with head <OMS name="...?binarysymbol"/>.
∼T⇝ Semantic macros with no arguments correspond to OMS directly.

\comp

Unfortunately, we have no highlighting whatsoever now. That is because we need to tell sTEX explicitly which parts of the notation are *notation components* which *should* be highlighted. We can do so with the \comp command.

We can introduce a new notation `highlight` for \binarysymbol that fixes this flaw, which we can subsequently use with \binarysymbol[highlight]:

Example 6

Input:

```
\notation{binarysymbol}[highlight]
   {\comp{\text{First: }}#1\comp{\text{; Second: }}#2}
$\binarysymbol[highlight]{a}{b}$
```

Output:

First: a; Second: b

.

Ideally, \comp would not be necessary: Everything in a notation that is *not* an argument should be a notation component. Unfortunately, it is computationally expensive to determine where an argument begins and ends, and the argument markers `#n` may themselves be nested in other macro applications or TEX groups, making it ultimately almost impossible to determine them automatically while also remaining compatible with arbitrary highlighting customizations (such as tooltips, hyperlinks, colors) that users might employ, and that are ultimately invoked by \comp.

Note that it is required that

1. the argument markers `#n` never occur inside a `\comp`, and
2. no semantic arguments may ever occur inside a notation.

Both criteria are not just required for technical reasons, but conceptionally meaningful:
The underlying principle is that the arguments to a semantic macro represent *arguments to the mathematical operation* represented by a symbol. For example, a semantic macro `\addition{a}{b}` taking two arguments would represent *the actual addition of (mathematical objects) a and b*. It should therefore be impossible for *a* or *b* to be part of a notation component of `\addition`.
Similarly, a semantic macro can not conceptually be part of the notation of `\addition`, since a semantic macro represents a *distinct mathematical concept* with *its own semantics*, whereas notations are syntactic representations of the very symbol to which the notation belongs.
If you want an argument to a semantic macro to be a purely syntactic parameter, then you are likely somewhat confused with respect to the distinction between the precise *syntax* and *semantics* of the symbol you are trying to declare (which happens quite often even to experienced sTeX users), and might want to give those another thought - quite likely, the macro you aim to implement does not actually represent a semantically maningful mathematical concept, and you will want to use `\def` and similar native LaTeX macro definitions rather than semantic macros.

`\symdef` In the vast majority of cases where a symbol declaration should come with a semantic macro, we will want to supply a notation immediately. For that reason, the `\symdef` command combines the functionality of both `\symdecl` and `\notation` with the optional arguments of both:

Example 7

Input:

```
\symdef{newbinarysymbol}[hl,args=2]
   {\comp{\text{1.: }}#1\comp{\text{; 2.: }}#2}
$\newbinarysymbol{a}{b}$
```

Output:

1.: *a*; 2.: *b*

.

We just declared a new symbol `newbinarysymbol` with `args=2` and immediately provided it with a notation with identifier `hl`. Since `hl` is the *first* (and so far, only) notation supplied for `newbinarysymbol`, using `\newbinarysymbol` without optional argument defaults to this notation.

`\setnotation`

The first notation provided will stay the default notation unless explicitly changed – this is enabled by the `\setnotation` command: `\setnotation{symbolname}{notation-id}` sets the default notation of `\symbolname` to `notation-id`, i.e. henceforth, `\symbolname` behaves like `\symbolname[notation-id]` from now on.

Often, a default notation is set right after the corresponding notation is introduced – the starred version `\notation*` for that reason introduces a new notation and immediately sets it to be the new default notation. So expressed differently, the *first* `\notation` for a symbol behaves exactly like `\notation*`, and `\notation*{foo}[bar]{...}` behaves exactly like `\notation{foo}[bar]{...}\setnotation{foo}{bar}`.

Operator Notations

Once we have a semantic macro with arguments, such as `\newbinarysymbol`, the semantic macro represents the *application* of the symbol to a list of arguments. What if we want to refer to the operator *itself*, though?

We can do so by supplying the `\notation` (or `\symdef`) with an *operator notation*, indicated with the optional argument `op=`. We can then invoke the operator notation using `\symbolname![notation-identifier]`. Since operator notations never take arguments, we do not need to use `\comp` in it, the whole notation is wrapped in a `\comp` automatically:

Example 8

Input:

```
\notation{newbinarysymbol}[ab,
 op={\text{a:}\cdot\text{; b:}\cdot}]
 {\comp{\text{a:}}#1\comp{\text{; b:}}#2}
\symname{newbinarysymbol} is also occasionally written
$\newbinarysymbol![ab]$
```

Output:

```
newbinarysymbol is also occasionally written a: · ; b:·
```

.

`\symbolname!` is translated to OMDoc/Mmt as `<OMS name="...?symbolname"/>` directly.

3.3.3 Argument Types

The notations so far used *simple* arguments which we call `i`-*type* arguments. Declaring a new symbol with `\symdecl{foo}[args=3]` is equivalent to writing `\symdecl{foo}[args=iii]`, indicating that the semantic macro takes three `i`-type arguments. However, there are three more argument types which we will investigate now, namely `b`-type, `a`-type and `B`-type arguments.

b-Type Arguments

A b-type argument represents a *variable* that is *bound* by the symbol in its application, making the symbol a *binding operator*. Typical examples of binding operators are e.g. sums $\sum$, products $\prod$, integrals $\int$, quantifiers like $\forall$ and $\exists$, that λ-operator, etc.

> ↪M→ —M→ ⇝T⇝ b-type arguments behave exactly like i-type arguments within TEX, but applications of binding operators, i.e. symbols with b-type arguments, are translated to OMBIND-terms in OMDoc/Mmt, rather than OMA.

Fo example, we can implement a summation operator binding an index variable and taking lower and upper index bounds and the expression to sum over like this:

Example 9

Input:

```
\symdef{summation}[args=biii]
{\mathop{\comp{\sum}}_{#1\comp{=}#2}^{#3}#4}
$\summation{\svar{x}}{1}{\svar{n}}{\svar{x}}^2$
```

Output:

$\sum_{x=1}^{n} x^2$

where the variable x is now *bound* by the `\summation`-symbol in the expression.

a-Type Arguments

a-type arguments represent a *flexary argument sequence*, i.e. a sequence of arguments of arbitrary length. Formally, operators that take arbitrarily many arguments don't "exist", but in informal mathematics, they are ubiquitous. a-type arguments allow us to write e.g. `\addition{a,b,c,d,e}` rather than having to write something like `\addition{a}{\addition{b}{\addition{c}{\addition{d}{e}}}}`!

`\notation` (and consequently `\symdef`, too) take one additional argument for each a-type argument that indicates how to "accumulate" a comma-separated sequence of arguments. This is best demonstrated on an example.

Let's say we want an operator representing quantification over an ascending chain of elements in some set, i.e. `\ascendingchain{S}{a,b,c,d,e}{t}` should yield $\forall a{<_S}b{<_S}c{<_S}d{<_S}e.\,t$. The "base"-notation for this operator is simply `{\comp{\forall} #2\comp{.\,}#3}`, where `#2` represents the full notation fragment *accumulated* from `{a,b,c,d,e}`.

The *additional* argument to `\notation` (or `\symdef`) takes the same arguments as the base notation and two *additional* arguments `##1` and `##2` representing successive pairs in the a-type argument, and accumulates them into `#2`, i.e. to produce $a <_S b <_S c <_S d <_S e$, we do `{##1 \comp{<}_{#1} ##2}`:

Example 10

Input:

```
\symdef{ascendingchain}[args=iai]
 {\comp{\forall} #2\comp{.\,}#3}
 {##1 \comp{<}_{#1} ##2}

Tadaa: $\ascendingchain{S}{a,b,c,d,e}{t}$
```

Output:

Tadaa: $\forall a <_S b <_S c <_S d <_S e.\, t$

.

If this seems overkill, keep in mind that you will rarely need the single-hash arguments `#1`,`#2` etc. in the `a`-notation-argument. For a much more representative and simpler example, we can introduce flexary addition via:

Example 11

Input:

```
\symdef{addition}[args=a]{#1}{##1 \comp{+} ##2}

Tadaa: $\addition{a,b,c,d,e}$
```

Output:

Tadaa: $a+b+c+d+e$

.

The `assoc`-key We mentioned earlier that "formally", flexary arguments don't really "exist". Indeed, formally, addition is usually defined as a binary operation, quantifiers bind a single variable etc.

Consequently, we can tell sTEX (or, rather, MMT/OMDOC) how to "resolve" flexary arguments by providing `\symdecl` or `\symdef` with an optional `assoc`-argument, as in `\symdecl{addition}[args=a,assoc=bin]`. The possible values for the `assoc`-key are:

`bin`: A binary, assoiative argument, e.g. as in *`\addition`*

`binl`: A binary, left-associative argument, e.g. $a^{b^{c^d}}$, which stands for $((a^b)^c)^d$

`binr`: A binary, right-associative argument, e.g. as in $A \to B \to C \to D$, which stands for $A \to (B \to (C \to D))$

`pre`: Successively prefixed, e.g. as in $\forall x, y, z.\, P$, which stands for $\forall x.\, \forall y.\, \forall z.\, P$

`conj`: Conjunctive, e.g. as in $a = b = c = d$ or $a, b, c, d \in A$, which stand for $a = d \wedge b = d \wedge c = d$ and $a \in A \wedge b \in A \wedge c \in A \wedge d \in A$, respectively

`pwconj`: Pairwise conjunctive, e.g. as in $a \neq b \neq c \neq d$, which stands for $a \neq b \wedge a \neq c \wedge a \neq d \wedge b \neq c \wedge b \neq d \wedge c \neq d$

B-Type Arguments

Finally, `B`-type arguments simply combine the functionality of both `a` and `b` - i.e. they represent an arbitrarily long sequence of variables to be bound, e.g. for implementing quantifiers:

Example 12

Input:

```
\symdef{quantforall}[args=Bi]
 {\comp{\forall}#1\comp{.}#2}
 {##1\comp,##2}

$\quantforall{\svar{x},\svar{y},\svar{z}}{P}$
```

Output:

$\forall x,y,z.P$

.

3.3.4 Type and Definiens Components

`\symdecl` and `\symdef` take two more optional arguments. TEX largely ignores them (except for special situations we will talk about later), but MMT can pick up on them for additional services. These are the `type` and `def` keys, which expect expressions in math-mode (ideally using semantic macros, of course!)

↪M→ —M→ ~T⇝ The `type` and `def` keys correspond to the `type` and `definiens` components of OMDOC/MMT constants.
Correspondingly, the name "type" should be taken with a grain of salt, since OMDOC/MMT– being foundation-independent – does not a priori implement a fixed typing system.

The `type`-key allows us to provide additional information (given the necessary STEX symbols), e.g. for addition on natural numbers:

Example 13

Input:

```
\symdef{Nat}[type=\set]{\comp{\mathbb N}}
\symdef{addition}[
    type=\funtype{\Nat,\Nat}{\Nat},
    op=+,
    args=a
]{#1}{##1 \comp+ ##2}

\symname{addition} is an operation $\funtype{\Nat,\Nat}{\Nat}$
```

Output:

addition is an operation $\mathbb{N}\times\mathbb{N}\to\mathbb{N}$

.

The `def`-key allows for declaring symbols as abbreviations:

Example 14

Input:

```
\symdef{successor}[
    type=\funtype{\Nat}{\Nat},
    def=\fun{\svar{x}}{\addition{\svar{x},1}},
    op=\mathtt{succ},
    args=1
]{\comp{\mathtt{succ(}#1\comp{)}}}

The \symname{successor} operation $\funtype{\Nat}{\Nat}$
is defined as $\fun{\svar{x}}{\addition{\svar{x},1}}$
```

Output:

The successor operation $\mathbb{N} \to \mathbb{N}$ is defined as $x \mapsto x+1$

.

3.3.5 Precedences and Automated Bracketing

Having done `\addition`, the obvious next thing to implement is `\multiplication`. This is in theory straight-forward:

Example 15

Input:

```
\symdef{multiplication}[
    type=\funtype{\Nat,\Nat}{\Nat},
    op=\cdot,
    args=a
]{#1}{##1 \comp\cdot ##2}

\symname{multiplication} is an operation $\funtype{\Nat,\Nat}{\Nat}$
```

Output:

multiplication is an operation $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$

.

However, if we *combine* `\addition` and `\multiplication`, we notice a problem:

Example 16

Input:

```
$\addition{a,\multiplication{b,\addition{c,\multiplication{d,e}}}}$
```

Output:

$a+b \cdot c+d \cdot e$

˙We all know that · binds stronger than +, so the output $a+b\cdot c+d\cdot e$ does not actually reflect the term we wrote. We can of course insert parentheses manually

Example 17

Input:

```
$\addition{a,\multiplication{b,(\addition{c,\multiplication{d,e}})}}$
```

Output:

$a+b\cdot(c+d\cdot e)$

˙but we can also do better by supplying *precedences* and have sTeX insert parentheses automatically.

For that purpose, `\notation` (and hence `\symdef`) take an optional argument `prec=<opprec>;<argprec1>x...x<argprec n>`.

We will investigate the precise meaning of `<opprec>` and the `<argprec>`s shortly – in the vast majority of cases, it is prefectly sufficient to think of `prec=` taking a single number and having that be *the* precedence of the notation, where lower precedences (somewhat counterintuitively) bind stronger than higher precedences. So fixing our notations for `\addition` and `\multiplication`, we get:

Example 18

Input:

```
\notation{multiplication}[
    op=\cdot,
    prec=50
]{#1}{##1 \comp\cdot ##2}
\notation{addition}[
    op=+,
    prec=100
]{#1}{##1 \comp+ ##2}

$\addition{a,\multiplication{b,\addition{c,\multiplication{d,e}}}}$
```

Output:

$a+b\cdot(c+d\cdot e)$

˙Note that the precise numbers used for precedences are pretty arbitrary - what matters is which precedences are higher than which other precedences when used in conjunction.

`\infprec`
`\neginfprec`

It is occasionally useful to have "infinitely" high or low precedences to enforce or forbid automated bracketing entirely – for those purposes, `\infprec` and `\neginfprec` exist (which are implemented as the maximal and minimal integer values accordingly).

More precisely, each notation takes

1. One *operator precedence* and

2. one *argument precedence* for each argument.

By default, all precedences are 0, unless the symbol takes no argument, in which case the operator precedence is `\neginfprec` (negative infinity). If we only provide a single number, this is taken as both the operator precedence and all argument precedences.
sTeX decides whether to insert parentheses by comparing operator precedences to a *downward precedence* p_d with initial value `\infprec`. When encountering a semantic macro, sTeX takes the operator precedence p_{op} of the notation used and checks whether $p_{op} > p_d$. If so, sTeX insert parentheses.
When sTeX steps into an argument of a semantic macro, it sets p_d to the respective argument precedence of the notation used.
In the example above:

1. sTeX starts out with $p_d =$`\infprec`.
2. sTeX encounters `\addition` with $p_{op} = 100$. Since $100 \not>$`\infprec`, it inserts no parentheses.
3. Next, sTeX encounters the two arguments for `\addition`. Both have no specifically provided argument precedence, so sTeX uses $p_d = p_{op} = 100$ for both and recurses.
4. Next, sTeX encounters `\multiplication{b,...}`, whose notation has $p_{op} = 50$.
5. We compare to the current downward precedence p_d set by `\addition`, arriving at $p_{op} = 50 \not> 100 = p_d$, so sTeX again inserts no parentheses.
6. Since the notation of `\multiplication` has no explicitly set argument precedences, sTeX uses the operator precedence for all arguments of `\multiplication`, hence sets $p_d = p_{op} = 50$ and recurses.
7. Next, sTeX encounters the inner `\addition{c,...}` whose notation has $p_{op} = 100$.
8. We compare to the current downward precedence p_d set by `\multiplication`, arriving at $p_{op} = 100 > 50 = p_d$ – which finally prompts sTeX to insert parentheses, and we proceed as before.

3.3.6 Variables

All symbol and notation declarations require a module with which they are associated, hence the commands `\symdecl`, `\notation`, `\symdef` etc. are disabled outside of `smodule`-environments.

Variables are different – variables are allowed everywhere, are not exported when the current module (if one exists) is imported (via `\importmodule` or `\usemodule`) and (also unlike symbol declarations) "disappear" at the end of the current TeX group.

`\svar`

So far, we have always used variables using `\svar{n}`, which marks-up n as a variable with name `n`. More generally, `\svar[foo]{<texcode>}` marks-up the arbitrary `<texcode>` as representing a variable with name `foo`.

Of course, this makes it difficult to reuse variables, or introduce "functional" variables with arities > 0, or provide them with a type or definiens.

`\vardef` For that, we can use the `\vardef` command. Its syntax is largely the same as that of `\symdef`, but unlike symbols, variables have only one notation (TODO: so far?), hence there is only `\vardef` and no `\vardecl`.

Example 19

Input:

```
    \vardef{varf}[
      name=f,
      type=\funtype{\Nat}{\Nat},
      op=f,
      args=1,
      prec=0;\neginfprec
   ]{\comp{f}#1}
   \vardef{varn}[name=n,type=\Nat]{\comp{n}}
   \vardef{varx}[name=x,type=\Nat]{\comp{x}}

   Given a function $\varf!:\funtype{\Nat}{\Nat}$,
   by $\addition{\varf!,\varn}$ we mean the function
   $\fun{\varx}{\varf{\addition{\varx,\varn}}}$
```

Output:

Given a function $f : \mathbb{N}\to\mathbb{N}$, by $f+n$ we mean the function $x\mapsto f(x+n)$

˙(of course, "lifting" addition in the way described in the previous example is an operation that deserves its own symbol rather than abusing `\addition`, but... well.)

TODO: bind=forall/exists

3.3.7 Variable Sequences

Variable *sequences* occur quite frequently in informal mathematics, hence they deserve special support. Variable sequences behave like variables in that they disappear at the end of the current TEX group and are not exported from modules, but their declaration is quite different.

`\varseq` A variable sequence is introduced via the command `\varseq`, which takes the usual optional arguments `name` and `type`. It then takes a starting index, an end index and a *notation* for the individual elements of the sequence parametric in an index.

This is best shown by example:

Example 20

Input:

```
 \vardef{varn}[name=n,type=\Nat]{\comp{n}}
\varseq{seqa}[name=a,type=\Nat]{1}{\varn}{\comp{a}_{#1}}

The $i$th index of $\seqa!$ is $\seqa{i}$.
```

Output:

The $i$th index of $a_1, \ldots, a_n$ is a_i.

.

Note that the syntax `\seqa!` now automatically generates a presentation based on the starting and ending index.

TODO: more notations for invoking sequences.

Notably, variable sequences are nicely compatible with `a`-type arguments, so we can do the following:

Example 21

Input:

```
$\addition{\seqa}$
```

Output:

$a_1 + \ldots + a_n$

.

Sequences can be *multidimensional* using the `args`-key, in which case the notation's arity increases and starting and ending indices have to be provided as a comma-separated list:

Example 22

Input:

```
\vardef{varm}[name=m,type=\Nat]{\comp{m}}
\varseq{seqa}[
    name=a,
    args=2,
    type=\Nat,
]{1,1}{\varn,\varm}{\comp{a}_{#1}^{#2}}

$\seqa!$ and $\addition{\seqa}$
```

Output:

$a_1^1, \ldots, a_n^m$ and $a_1^1 + \ldots + a_n^m$

.We can also explicitly provide a "middle" segment to be used, like such:

Example 23

Input:

```
\varseq{seqa}[
    name=a,
    type=\Nat,
    args=2,
    mid={\comp{a}_{\varn}^1,\comp{a}_1^2,\ellipses,\comp{a}_{1}^{\varm}}
]{1,1}{\varn,\varm}{\comp{a}_{#1}^{#2}}

$\seqa!$ and $\addition{\seqa}$
```

Output:

$a_1^1, \ldots, a_n^1, a_1^2, \ldots, a_1^m, \ldots, a_n^m$ and $a_1^1 + \ldots + a_n^1 + a_1^2 + \ldots + a_1^m + \ldots + a_n^m$

.

3.4 Module Inheritance and Structures

3.4.1 Multilinguality and Translations

If we load the sTeX document class or package with the option `lang=<lang>`, sTeX will load the appropriate `babel` language for you – e.g. `lang=de` will load the babel language `ngerman`. Additionally, it makes sTeX aware of the current document being set in (in this example) *german*. This matters for reasons other than mere `babel`-purposes, though:

Every *module* is assigned a language. If no sTeX package option is set that allows for inferring a language, sTeX will check whether the current file name ends in e.g. `.en.tex` (or `.de.tex` or `.fr.tex`, or...) and set the language accordingly. Alternatively, a language can be explicitly assigned via `\begin{smodule}[lang=<language>]{Foo}`.

↪M→ —M→ ⇝T⇝

Technically, each `smodule`-environment induces *two* OMDoc/Mmt theories: `\begin{smodule}[lang=<lang>]{Foo}` generates a theory `some/namespace?Foo` that only contains the "formal" part of the module – i.e. exactly the content that is exported when using `\importmodule`.
Additionally, Mmt generates a *language theory* `some/namespace/Foo?<lang>` that includes `some/namespace?Foo` and contains all the other document content – variable declarations, includes for each `\usemodule`, etc.

Notably, the language suffix in a filename is ignored for `\usemodule`, `\importmodule` and in generating/computing URIs for modules. This however allows for providing *translations* for modules between languages without needing to duplicate content:

If a module `Foo` exists in e.g. english in a file `Foo.en.tex`, we can provide a file `Foo.de.tex` right next to it, and write `\begin{smodule}[sig=en]{Foo}`. The `sig`-key then signifies, that the "signature" of the module is contained in the *english* version of the module, which is immediately imported from there, just like `\importmodule` would.

Additionally to translating the informal content of a module file to different languages, it also allows for customizing notations between languages. For example, the *least common multiple* of two numbers is often denoted as $\mathtt{lcm}(a, b)$ in english, but is called *kleinstes gemeinsames Vielfaches* in german and consequently denoted as $\mathtt{kgV}(a, b)$ there.

We can therefore imagine a german version of an lcm-module looking something like this:

```
\begin{smodule}[sig=en]{lcm}
  \notation*{lcm}[de]{\comp{\mathtt{kgV}}(#1,#2)}

  Das \symref{lcm}{kleinste gemeinsame Vielfache}
  $\lcm{a,b}$ von zwei Zahlen $a,b$ ist...
\end{smodule}
```

If we now do `\importmodule{lcm}` (or `\usemodule{lcm}`) within a *german* document, it will also load the content of the german translation, including the `de`-notation for `\lcm`.

3.4.2 Simple Inheritance and Namespaces

`\importmodule`
`\usemodule`

`\importmodule[Some/Archive]{path?ModuleName}` is only allowed within an `smodule`-environment and makes the symbols declared therein available. Additionally the content of `ModuleName` will be exported if the current module is imported somewhere else via `\importmodule`.

`\usemodule` behaves the same way, but without exporting the content of the used module.

It is worth going into some detail how exactly `\importmodule` and `\usemodule` resolve their arguments to find the desired module – which is closely related to the *namespace* generated for a module, that is used to generate its URI.

Ideally, sTeX would use arbitrary URIs for modules, with no forced relationships between the *logical* namespace of a module and the *physical* location of the file declaring the module – like Mmt does things.
Unfortunately, TeX only provides very restricted access to the file system, so we are forced to generate namespaces systematically in such a way that they reflect the physical location of the associated files, so that sTeX can resolve them accordingly. Largely, users need not concern themselves with namespaces at all, but for completenesses sake, we describe how they are constructed:

- If `\begin{smodule}{Foo}` occurs in a file `/path/to/file/Foo[.`⟨*lang*⟩`].tex` which does not belong to an archive, the namespace is `file://path/to/file`.
- If the same statement occurs in a file `/path/to/file/bar[.`⟨*lang*⟩`].tex`, the namespace is `file://path/to/file/bar`.

In other words: outside of archives, the namespace corresponds to the file URI with the filename dropped iff it is equal to the module name, and ignoring the (optional) language suffix.
If the current file is in an archive, the procedure is the same except that the initial segment of the file path up to the archive's `source`-folder is replaced by the archive's namespace URI.

Conversely, here is how namespaces/URIs and file paths are computed in import statements, examplary `\importmodule`:

- `\importmodule{Foo}` outside of an archive refers to module `Foo` in the current namespace. Consequently, `Foo` must have been declared earlier in the same document or, if not, in a file `Foo[.`⟨*lang*⟩`].tex` in the same directory.
- The same statement *within* an archive refers to either the module `Foo` declared earlier in the same document, or otherwise to the module `Foo` in the archive's top-level namespace. In the latter case, is has to be declared in a file `Foo[.`⟨*lang*⟩`].tex` directly in the archive's `source`-folder.
- Similarly, in `\importmodule{some/path?Foo}` the path `some/path` refers to either the sub-directory and relative namespace path of the current directory and namespace outside of an archive, or relative to the current archive's top-level namespace and `source`-folder, respectively.

  The module `Foo` must either be declared in the file ⟨*top-directory*⟩`/some/path/Foo[.`⟨*lang*⟩`].tex`, or in ⟨*top-directory*⟩`/some/path[.`⟨*lang*⟩`].tex` (which are checked in that order).

- Similarly, `\importmodule[Some/Archive]{some/path?Foo}` is resolved like the previous cases, but relative to the archive `Some/Archive` in the mathhub-directory.
- Finally, `\importmodule{full://uri?Foo}` naturally refers to the module `Foo` in the namespace `full://uri`. Since the file this module is declared in can not be determined directly from the URI, the module must be in memory already, e.g. by being referenced earlier in the same document.

  Since this is less compatible with a modular development, using full URIs directly is strongly discouraged, unless the module is delared in the current file directly.

`\STEXexport` `\importmodule` and `\usemodule` import all symbols, notations, semantic macros and (recursively) `\importmodule`s. If you want to additionally export e.g. convenience macros and other code from a module, you can use the command `\STEXexport{<code>}` in your module. Then `<code>` is executed (both immediately and) every time the current module is opened via `\importmodule` or `\usemodule`.

Note, that `\newcommand` defines macros *globally* and throws an error if the macro already exists, potentially leading to low-level LaTeX errors if we put a `\newcommand` in an `\STEXexport` and the `<code>` is executed more than once in a document – which can happen easily.
A safer alternative is to use macro definition principles, that are safe to use even if the macro being defined already exists, and ideally are local to the current TeX group, such as `\def` or `\let`.

3.4.3 The `mathstructure` Environment

A common occurence in mathematics is bundling several interrelated "declarations" together into *structures*. For example:

- A *monoid* is a structure $\langle M, \circ, e\rangle$ with $\circ : M \times M \to M$ and $e \in M$ such that...
- A *topological space* is a structure $\langle X, \mathcal{T}\rangle$ where X is a set and $\mathcal{T}$ is a topology on X
- A *partial order* is a structure $\langle S, \leq\rangle$ where $\leq$ is a binary relation on S such that...

This phenomenon is important and common enough to warrant special support, in particular because it requires being able to *instantiate* such structures (or, ratherer, structure *signatures*) in order to talk about (concrete or variable) *particular* monoids, topological spaces, partial orders etc.

`mathstructure` The `mathstructure` environment allows us to do exactly that. It behaves exactly like the `smodule` environment, but is itself only allowed inside an `smodule` environment, and allows for instantiation later on.

How this works is again best demonstrated by example:

Example 24

Input:

```
\begin{mathstructure}{monoid}
   \symdef{universe}[type=\set]{\comp{U}}
   \symdef{op}[
      args=2,
      type=\funtype{\universe,\universe}{\universe},
      op=\circ
   ]{#1 \comp{\circ} #2}
   \symdef{unit}[type=\universe]{\comp{e}}
\end{mathstructure}

A \symname{monoid} is...
```

Output:

A monoid is...

˙Note that the `\symname{monoid}` is appropriately highlighted and (depending on your pdf viewer) shows a URI on hovering – implying that the `mathstructure` environment has generated a *symbol* `monoid` for us. It has not generated a semantic macro though, since we can not use the `monoid`-symbol *directly*. Instead, we can instantiate it, for example for integers:

Example 25

Input:

```
\symdef{Int}[type=\set]{\comp{\mathbb Z}}
\symdef{addition}[
   type=\funtype{\Int,\Int}{\Int},
   args=2,
   op=+
]{##1 \comp{+} ##2}
\symdef{zero}[type=\Int]{\comp{0}}

$\mathstruct{\Int,\addition!,\zero}$ is a \symname{monoid}.
```

Output:

$\langle\mathbb{Z},+,0\rangle$ is a monoid.

.

So far, we have not actually instantiated `monoid`, but now that we have all the symbols to do so, we can:

Example 26

Input:

```
\instantiate{intmonoid}{
   universe = Int ,
   op = addition ,
   unit = zero
}{monoid}{\mathbb{Z}_{+,0}}

   $\intmonoid{universe}$, $\intmonoid{unit}$ and $\intmonoid{op}{a}{b}$.

   Also: $\intmonoid!$
```

Output:

$\mathbb{Z}$, 0 and $a+b$.
Also: $\mathbb{Z}_{+,0}$

.

`\instantiate`

So summarizing: `\instantiate` takes four arguments: The (macro-)name of the instance, a key-value pair assigning declarations in the corresponding `mathstructure` to symbols currently in scope, the name of the `mathstructure` to instantiate, and lastly a notation for the instance itself.

It then generates a semantic macro that takes as argument the name of a declaration in the instantiated `mathstructure` and resolves it to the corresponding instance of that particular declaration.

`\instantiate` and `mathstructure` make use of the *Theories-as-Types* paradigm: `mathstructure{<name>}` does in fact simply create a nested theory with name `<name>-structure`. The *constant* `<name>` is defined as `Mod(<name>-structure)` – a *dependent record type with manifest fields*, the fields of which are generated from (and correspond to) the constants in `<name>-structure`.
`\instantiate` appropriately generates a constant whose definiens is a record term of type `Mod(<name>-structure)`, with the fields assigned appropriately based on the key-value-list.

Notably, `\instantiate` throws an error if not *every* declaration in the instantiated `mathstructure` is being assigned.

You might consequently ask what the usefulness of `mathstructure` even is.

`\varinstantiate`

The answer is that we can also instantiate a `mathstructure` with a *variable*. The syntax of `\varianstantiate` is equivalent to that of `\instantiate`, but all of the key-value-pairs are optional, and if not explicitly assigned (to a symbol *or* a variable declared with `\vardef`) inherit their notation from the one in the `mathstructure` environment.

This allows us to do things like:

Example 27

Input:

```
\varinstantiate{varM}{}{monoid}{M}

A \symname{monoid} is a structure
$\varM!:=\mathstruct{\varM{universe},\varM{op}!,\varM{unit}}$
such that
$\varM{op}!:\funtype{\varM{universe},\varM{universe}}{\varM{universe}}$
 and...

 \varinstantiate{varMb}{universe = Int}{monoid}{M_2}

 \noindent Let $\varMb!:=\mathstruct{\varMb{universe},\varMb{op}!,\varMb{unit}}$
 a \symname{monoid} on $\Int$...
```

Output:

A monoid is a structure $M := \langle U,\circ,e\rangle$ such that $\circ : U\times U\to U$ and...
Let $M_2 := \langle \mathbb{Z},\circ,e\rangle$ a monoid on $\mathbb{Z}$...

.

We will return to this example later, when we also know how to handle the *axioms* of a monoid.

3.4.4 The `copymodule` Environment

TODO: explain

Given modules:

Example 28

Input:

```
 \begin{smodule}{magma}
    \symdef{universe}{\comp{\mathcal U}}
    \symdef{operation}[args=2,op=\circ]{#1 \comp\circ #2}
\end{smodule}
\begin{smodule}{monoid}
    \importmodule{magma}
    \symdef{unit}{\comp e}
\end{smodule}
\begin{smodule}{group}
    \importmodule{monoid}
    \symdef{inverse}[args=1]{{#1}^{\comp{-1}}}
\end{smodule}
```

Output:

.

We can form a module for *rings* by “cloning” an instance of `group` (for addition) and `monoid` (for multiplication), respectively, and “glueing them together” to ensure they share the same universe:

Example 29

Input:

```
\begin{smodule}{ring}
   \begin{copymodule}{group}{addition}
       \renamedecl[name=universe]{universe}{runiverse}
       \renamedecl[name=plus]{operation}{rplus}
       \renamedecl[name=zero]{unit}{rzero}
       \renamedecl[name=uminus]{inverse}{ruminus}
   \end{copymodule}
   \notation*{rplus}[plus,op=+,prec=60]{#1 \comp+ #2}
       \notation*{rzero}[zero]{\comp0}
       \notation*{ruminus}[uminus,op=-]{\comp- #1}
       \begin{copymodule}{monoid}{multiplication}
       \assign{universe}{\runiverse}
       \renamedecl[name=times]{operation}{rtimes}
       \renamedecl[name=one]{unit}{rone}
   \end{copymodule}
   \notation*{rtimes}[cdot,op=\cdot,prec=50]{#1 \comp\cdot #2}
       \notation*{rone}[one]{\comp1}
       Test: $\rtimes a{\rplus c{\rtimes de}}$
\end{smodule}
```

Output:

Test: $a{\cdot}c{\circ}c$

.

TODO: explain donotclone

3.4.5 The `interpretmodule` Environment

TODO: explain

Example 30

Input:

```
\begin{smodule}{int}
   \symdef{Integers}{\comp{\mathbb Z}}
   \symdef{plus}[args=2,op=+]{#1 \comp+ #2}
   \symdef{zero}{\comp0}
   \symdef{uminus}[args=1,op=-]{\comp-#1}

   \begin{interpretmodule}{group}{intisgroup}
       \assign{universe}{\Integers}
       \assign{operation}{\plus!}
       \assign{unit}{\zero}
       \assign{inverse}{\uminus!}
   \end{interpretmodule}
\end{smodule}
```

Output:

.

3.5 Primitive Symbols (The sTeX Metatheory)

TODO: metatheory documentation

Chapter 4

Using sTeX Symbols

Given a symbol declaration `\symdecl{symbolname}`, we obtain a semantic macro *`\symbolname`*. We can use this semantic macro in math mode to use its notation(s), and we can use *`\symbolname`*`!` in math mode to use its operator notation(s). What else can we do?

4.1 `\symref` and its variants

`\symref`
`\symname`

We have already seen `\symname` and `\symref`, the latter being the more general.

`\symref{<symbolname>}{<code>}` marks-up `<code>` as referencing `<symbolname>`. Since quite often, the `<code>` should be (a variant of) the name of the symbol anyway, we also have `\symname{<symbolname>}`.

Note that `\symname` uses the *name* of a symbol, not its macroname. More precisely, `\symname` will insert the name of the symbol with "`-`" replaced by spaces. If a symbol does not have an explicit `name=` given, the two are equal – but for `\symname` it often makes sense to make the two explicitly distinct. For example:

Example 31

Input:

```
\symdef{Nat}[
   name=natural-number,
   type=\set
]{\comp{\mathbb{N}}}

A \symname{Nat} is...
```

Output:

A natural number is...

.

`\symname` takes two additional optional arguments, `pre=` and `post=` that get prepended or appended respectively to the symbol name.

\Symname

Additionally, \Symname behaves exactly like \symname, but will capitalize the first letter of the name:

Example 32

Input:

```
\Symname[post=s]{Nat} are...
```

Output:

Natural numbers are...

.

This is as good a place as any other to explain how sTeX resolves a string `symbolname` to an actual symbol.
If *\symbolname* is a semantic macro, then sTeX has no trouble resolving `symbolname` to the full URI of the symbol that is being invoked.
However, especially in \symname (or if a symbol was introduced using `\symdecl*` without generating a semantic macro), we might prefer to use the *name* of a symbol directly for readability – e.g. we would want to write `A \symname{natural-number} is...` rather than `A \symname{Nat} is...`. sTeX attempts to handle this case thusly:
If `string` does *not* correspond to a semantic macro `\string`, then sTeX checks all symbols currently in scope until it finds one, whose full URI ends with `string`. This allows for disambiguating more precisely, e.g. by saying `\symname{Integers?addition}` or `\symname{RealNumbers?addition}` in the case where several `addition`s are in scope.
However, this also means that if we have symbols `foo` and e.g. `miraculous-foo`, then sTeX might resolve `\symname{foo}` to `miraculous-foo` if it finds this symbol first. It is therefore a good idea to prefix symbol names with a `?`, thus ensuring that sTeX will find the symbol `...?foo` rather than `...?miraculous-foo`.

4.2 Marking Up Text and On-the-Fly Notations

We can also use semantic macros outside of text mode though, which allows us to annotate arbitrary text fragments.

Let us assume again, that we have `\symdef{addition}[args=2]{#1 \comp+ #2}`. Then we can do

Example 33

Input:

```
\addition{\comp{The sum of} \arg{$\svar{n}$} \comp{ and }\arg{$\svar{m}$}}
is...
```

Output:

The sum of n and m is...

˙...which marks up the text fragment as representing an *application* of the `addition`-symbol to two argument n and m.

↪M→ —M→ ⇝T⇝ As expected, the above example is translated to OMDoc/Mmt as an `OMA` with `<OMS name="...?addition"/>` as head and `<OMV name="n"/>` and `<OMV name="m"/>` as arguments.

`\arg` In text mode, every semantic macro takes exactly one argument, namely the text-fragment to be annotated. The `\arg` command is only valid within the argument to a semantic macro and marks up the *individual arguments* for the symbol.

We can also use semantic macros in text mode to invoke an operator itself instead of its application, with the usuual syntax using `!`:

Example 34

Input:

```
\addition!{Addition} is...
```

Output:

Addition is...

˙

In deed, `\symbolname!{<code>}` is exactly equivalent to `\symref{symbolname}{<code>}` (the latter is in fact implemented in terms of the former).

`\arg` also allows us to switch the order of arguments around and "hide" arguments: For example, `\arg[3]{<code>}` signifies that `<code>` represents the *third* argument to the current operator, and `\arg*[i]{<code>}` signifies that `<code>` represents the ith argument, but it should not produce any output (it is exported in the `xhtml` however, so that Mmt and other systems can pick up on it)

Example 35

Input:

```
\addition{\comp{adding}
   \arg[2]{$\svar{k}$}
   \arg*{$\addition{\svar{n}}{\svar{m}}$}} yields...
```

Output:

adding k yields...

˙Note that since the second `\arg` has no explicit argument number, it automatically represents the first not-yet-given argument – i.e. in this case the first one.

The same syntax can be used in math mode, too, which allows us to spontaneously introduce new notations on the fly. We can activate it using the starred variants of semantic macros:

Example 36

Input:

```
Given $\addition{\svar{n}}{\svar{m}}$, then
$\addition*{
    \arg*{\addition{\svar{n}}{\svar{m}}}
    \comp{+}
    \arg{\svar{k}}
}$ yields...
```

Output:

Given $n+m$, then $+k$ yields...

.

4.3 Referencing Symbols and Statements

TODO: references documentation

Chapter 5

sTeX Statements

5.1 Definitions, Theorems, Examples, Paragraphs

As mentioned earlier, we can semantically mark-up *statements* such as definitions, theorems, lemmata, examples, etc.

The corresponding environments for that are:

- `sdefinition` for definitions,
- `sassertion` for assertions, i.e. propositions that are declared to be *true*, such as theorems, lemmata, axioms,
- `sexample` for examples, and
- `sparagraph` for other semantic paragraphs, such as comments, remarks, conjectures, etc.

The *presentation* of these environments can be customized to use e.g. predefined `theorem`-environments, see chapter 6 for details.

All of these environments take optional arguments in the form of `key=value`-pairs. Common to all of them are the keys `id=` (for cross-referencing, see section 4.3), `type=` for customization (see chapter 6) and additional information (e.g. definition principles, “difficulty” etc), `title=`, and `for=`.

The `for=` key expects a comma-separated list of existing symbols, allowing for e.g. things like

Example 37

Input:

```
\begin{sexample}[
   id=additionandmultiplication.ex,
   for={addition,multiplication},
   type={trivial,boring},
   title={An Example}
]
   $\addition{2,3}$ is $5$, $\multiplication{2,3}$ is $6$.
\end{sexample}
```

Output:

Example 5.1.1 (An Example)**.** 2+3 is 5, 2·3 is 6.

.

`\definiendum`
`\definame`
`\definiens`
`\Definame`

`sdefinition` (and `sparagraph` with `type=symdoc`) introduce three new macros: `definiendum` behaves like `symref` (and `definame`/`Definame` like `symname`/`Symname`, respectively), but highlights the referenced symbol as *being defined* in the current definition.

`\definiens[<optional symbolname>]{<code>}` marks up `<code>` as being the explicit *definiens* of `<optional symbolname>` (in case `for=` has multiple symbols).

The special `type=symdoc` for `sparagraph` is intended to be used for "informal definitions", or encyclopedia-style descriptions for symbols.
The MMT-system can use those (in lieu of an actual `sdefinition` in scope) to present to users, e.g. when hovering over symbols.

All four environments also take an optional parameter `name=` – if this one is given a value, the environment will generate a *symbol* by that name (but with no semantic macro). Not only does this allow for `\symref` et al, it allows us to resume our earlier example for monoids much more nicely:

Example 38

Input:

```
\begin{mathstructure}{monoid}
   \symdef{universe}[type=\set]{\comp{U}}
   \symdef{op}[
      args=2,
      type=\funtype{\universe,\universe}{\universe},
      op=\circ
   ]{#1 \comp{\circ} #2}
   \symdef{unit}[type=\universe]{\comp{e}}

   \begin{sparagraph}[type=symdoc,for=monoid]
      A \definame{monoid} is a structure
      $\mathstruct{\universe,\op!,\unit}$
      where $\op!:\funtype{\universe}{\universe}$ and
      $\inset{\unit}{\universe}$ such that

      \begin{sassertion}[name=associative,
         type=axiom,
         title=Associativity]
         $\op!$ is associative
      \end{sassertion}
      \begin{sassertion}[name=isunit,
         type=axiom,
         title=Unit]
         $\equal{\op{\svar{x}}{\unit}}{\svar{x}}$
         for all $\inset{\svar{x}}{\universe}$
      \end{sassertion}
   \end{sparagraph}
\end{mathstructure}

An example for a \symname{monoid} is...
```

Output:

A **monoid** is a structure $\langle U,\circ,e\rangle$ where $\circ : U \to U$ and $e \in U$ such that

Axiom 5.1.2 (Associativity). *$\circ$ is associative*

Axiom 5.1.3 (Unit). *$x \circ e = x$ for all $x \in U$*

An example for a monoid is...

.

Now the `mathstructure` monoid contains two additional symbols, namely the axioms for associativity and that e is a unit. Note that both symbols do not represent the mere *propositions* that e.g. $\circ$ is associative, but *the assertion that it is actually true* that $\circ$ is associative.

If we now want to instantiate `monoid` (unless with a variable, of course), we also need to assign `associative` and `neutral` to analogous assertions. So the earlier example

```
\instantiate{intmonoid}{
    universe = Int ,
    op = addition ,
    unit = zero
}{monoid}{\mathbb{Z}_{+,0}}
```

...will not work anymore. We now need to give assertions that `addition` is associative and that `zero` is a unit with respect to addition.[2]

5.2 Proofs

TODO

[2]Of course, sTeX can not check that the assertions are the "correct" ones – but if the assertions (both in `monoid` as well as those for addition and zero) are properly marked up, Mmt can. TODO: should

Chapter 6

Highlighting and Presentation Customizations

The environments starting with `s` (i.e. `smodule`, `sassertion`, `sexample`, `sdefinition`, `sparagraph` and `sproof`) by default produce no additional output whatsoever (except for the environment content of course). Instead, the document that uses them (whether directly or e.g. via `inputref`) can decide how these environments are suppposed to look like.

The stexthm defines some default customizations that can be used, but of course many existing LaTeX templates come with their own `definition`, `theorem` and similar environments that authors are supposed (or even required) to use. Their concrete syntax however is usually not compatible with all the additional arguments that sTeX allows for semantic information.

Therefore we introduced the separate environments `sdefinition` etc. instead of using `definition` directly, and allow authors to specify how these environments should be styled via the commands `stexpatch*`.

`\stexpatchmodule`
`\stexpatchdefinition`
`\stexpatchassertion`
`\stexpatchexample`
`\stexpatchparagraph`
`\stexpatchproof`

All of these commands take one optional and two proper arguments, i.e. `\stexpatch*[<type>]{<begin-code>}{<end-code>}`.

After sTeX reads and processes the optional arguments for these environments, (some of) their values are stored in the macros `\s*<field>` (i.e. `sexampleid`, `\sassertionname`, etc.). It then checks for all the values `<type>` in the `type=`-list, whether an `\stexpatch*[<type>]` for the current environment has been called. If it finds one, it uses that patches `<begin-code>` and `<end-code>` to mark up the current environment. If no patch for (any of) the type(s) is found, it checks whether and `\stexpatch*` was called without optional argument.

For example, if we want to use a predefined `theorem` environment for `sassertions` with `type=theorem`, we can do

```
\stexpatchassertion[theorem]{\begin{theorem}}{\end{theorem}}
```

...or, rather, since e.g. `theorem`-environments defined using amsthm take an optional title as argument, we can do:

```
\stexpatchassertion[theorem]
  {\ifx\sassertiontitle\@empty
      \begin{theorem}
```

```
    \else
      \begin{theorem}[\sassertiontitle]
  \fi}
  {\end{theorem}}
```

Or, if we want all sdefinitions to use a predefined `definition`-environment, we can do

```
\stexpatchdefinition
  {\ifx\sdefinitiontitle\@empty
      \begin{definition}
    \else
      \begin{definition}[\sdefinitiontitle]
  \fi}
  {\end{definition}}
```

`\compemph`
`\varemph`
`\symrefemph`
`\defemph`

Apart from the environments, we can control how sTeX highlights variables, notation components, \symrefs and \definiendums, respectively.

To do so, we simply redefine these four macros. For example, to highlight notation components (i.e. everything in a \comp) in blue, as in this document, we can do `\def\compemph#1{\textcolor{blue}{#1}}`. By default, `\compemph` et al do nothing.

`\compemph@uri`
`\varemph@uri`
`\symrefemph@uri`
`\defemph@uri`

For each of the four macros, there exists an additional macro that takes the full URI of the relevant symbol currently being highlighted as a second argument. That allows us to e.g. use pdf tooltips and links. For example, this document uses

```
\protected\def\symrefemph@uri#1#2{
  \pdftooltip{
    \srefsymuri{#2}{\symrefemph{#1}}
  }{
    URI:~\detokenize{#2}
  }
}
```

By default, `\compemph@uri` is simply defined as `\compemph{#1}` (analogously for the other three commands).

Chapter 7

Additional Packages

TODO: tikzinput documentation

7.1 Modular Document Structuring

TODO: document-structure documentation

7.2 Slides and Course Notes

TODO: notesslides documentation

7.3 Homework, Problems and Exams

TODO: problem documentation
TODO: hwexam documentation

Part II
Documentation

Chapter 8

sTeX-Basics

This sub package provides general set up code, auxiliary methods and abstractions for `xhtml` annotations.

8.1 Macros and Environments

`\sTeX`
`\stex`

Both print this sTeX logo.

`\stex_debug:nn`

`\stex_debug:nn {⟨log-prefix⟩} {⟨message⟩}`

Logs ⟨*message*⟩, if the package option `debug` contains ⟨*log-prefix*⟩.

8.1.1 HTML Annotations

`\if@latexml`

LaTeX2e conditional for LaTeXML

`\latexml_if_p: ⋆`
`\latexml_if:TF ⋆`

LaTeX3 conditionals for LaTeXML.

`\stex_if_do_html_p: ⋆`
`\stex_if_do_html:TF ⋆`

Whether to currently produce any HTML annotations (can be false in some advanced structuring environments, for example)

`\stex_suppress_html:n`

Temporarily disables HTML annotations in its argument code

We have four macros for annotating generated HTML (via LaTeXML or RusTeX) with attributes:

`\stex_annotate:nnn`
`\stex_annotate_invisible:nnn`
`\stex_annotate_invisible:n`

`\stex_annotate:nnn {⟨property⟩} {⟨resource⟩} {⟨content⟩}`

Annotates the HTML generated by ⟨*content*⟩ with

`property="stex:⟨property⟩", resource="⟨resource⟩"`.

`\stex_annotate_invisible:n` adds the attributes

`stex:visible="false", style="display:none"`.

`\stex_annotate_invisible:nnn` combines the functionality of both.

`stex_annotate_env`

```
\begin{stex_annotate_env}{⟨property⟩}{⟨resource⟩}
⟨content⟩
\end{stex_annotate_env}
```

behaves like `\stex_annotate:nnn {⟨property⟩} {⟨resource⟩} {⟨content⟩}`.

8.1.2 Babel Languages

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

Map language abbreviations to their full babel names and vice versa. e.g. `\c_stex_languages_prop{en}` yields `english`, and `\c_stex_language_abbrevs_prop{english}` yields `en`.

8.1.3 Auxiliary Methods

`\stex_deactivate_macro:Nn`
`\stex_reactivate_macro:N`

`\stex_deactivate_macro:Nn⟨cs⟩{⟨environments⟩}`

Makes the macro ⟨*cs*⟩ throw an error, indicating that it is only allowed in the context of ⟨*environments*⟩.

`\stex_reactivate_macro:N⟨cs⟩` reactivates it again, i.e. this happens ideally in the ⟨*begin*⟩-code of the associated environments.

`\ignorespacesandpars` ignores white space characters and `\par` control sequences. Expands tokens in the process.

Chapter 9

sTeX-MathHub

This sub package provides code for handling sTeX archives, files, file paths and related methods.

9.1 Macros and Environments

`\stex_kpsewhich:n`

`\stex_kpsewhich:n` executes kpsewhich and stores the return in `\l_stex_kpsewhich_return_str`. This does not require shell escaping.

9.1.1 Files, Paths, URIs

`\stex_path_from_string:Nn`

`\stex_path_from_string:Nn` ⟨*path-variable*⟩ `{`⟨*string*⟩`}`

turns the ⟨*string*⟩ into a path by splitting it at `/`-characters and stores the result in ⟨*path-variable*⟩. Also applies `\stex_path_canonicalize:N`.

`\stex_path_to_string:NN`
`\stex_path_to_string:N`

The inverse; turns a path into a string and stores it in the second argument variable, or leaves it in the input stream.

`\stex_path_canonicalize:N`

Canonicalizes the path provided; in particular, resolves `.` and `..` path segments.

`\stex_path_if_absolute_p:N` ⋆
`\stex_path_if_absolute:N`*TF* ⋆

Checks whether the path provided is *absolute*, i.e. starts with an empty segment

`\c_stex_pwd_seq`
`\c_stex_pwd_str`
`\c_stex_mainfile_seq`
`\c_stex_mainfile_str`

Store the current working directory as path-sequence and string, respectively, and the (heuristically guessed) full path to the main file, based on the PWD and `\jobname`.

`\g_stex_currentfile_seq` The file being currently processed (respecting `\input` etc.)

`\stex_filestack_push:n`
`\stex_filestack_pop:`
Push and pop (repsectively) a file path to the file stack, to keep track of the current file. Are called in hooks `file/before` and `file/after`, respectively.

9.1.2 MathHub Archives

`\mathhub`
`\c_stex_mathhub_seq`
`\c_stex_mathhub_str`
We determine the path to the local MathHub folder via one of three means, in order of precedence:

1. The `mathhub` package option, or
2. the `\mathhub`-macro, if it has been defined before the `\usepackage{stex}`-statement, or
3. the `MATHHUB` system variable.

In all three cases, `\c_stex_mathhub_seq` and `\c_stex_mathhub_str` are set accordingly.

`\l_stex_current_repository_prop`

Always points to the *current* MathHub repository (if we currently are in one). Has the following fields corresponding to the entries in the `MANIFEST.MF`-file:

`id`: The name of the archive, including its group (e.g. `smglom/calculus`),

`ns`: The content namespace (for modules and symbols),

`narr`: the narration namespace (for document references),

`docurl`: The URL that is used as a basis for *external references*,

`deps`: All archives that this archive depends on (currently not in use).

`\stex_set_current_repository:n`

Sets the current repository to the one with the provided ID. calls `\__stex_mathhub_do_manifest:n`, so works whether this repository's `MANIFEST.MF`-file has already been read or not.

`\stex_require_repository:n` Calls `\__stex_mathhub_do_manifest:n` iff the corresponding archive property list does not already exist, and adds a corresponding definition to the `.sms`-file.

`\stex_in_repository:nn` `\stex_in_repository:nn{⟨repository-name⟩}{⟨code⟩}`

Change the current repository to `{`⟨*repository-name*⟩`}` (or not, if `{`⟨*repository-name*⟩`}` is empty), and passes its ID on to `{`⟨*code*⟩`}` as `#1`. Switches back to the previous repository after executing `{`⟨*code*⟩`}`.

9.1.3 Using Content in Archives

`\mhpath` ⋆

`\mhpath{⟨archive-ID⟩}{⟨filename⟩}`

Expands to the full path of file ⟨*filename*⟩ in repository ⟨*archive-ID*⟩. Does not check whether the file or the repository exist.

`\inputref`
`\mhinput`

`\inputref[⟨archive-ID⟩]{⟨filename⟩}`

Both `\input` the file ⟨*filename*⟩ in archive ⟨*archive-ID*⟩ (relative to the `source`-subdirectory). `\mhinput` does so directly. `\inputref` does so within an `\begingroup`...`\endgroup`-block, and skips it in `html`-mode, inserting a *reference* to the file instead.

Both also set `\ifinputref` to true.

`\addmhbibresource`

`\inputref[⟨archive-ID⟩]{⟨filename⟩}`

Adds a `.bib`-file ⟨*filename*⟩ in archive ⟨*archive-ID*⟩ (relative to the top-directory of the archive!).

`\libinput`

`\libinput{⟨filename⟩}`

Inputs ⟨*filename*⟩`.tex` from the `lib` folders in the current archive and the `meta-inf`-archive of the current archive group(s) (if existent) in descending order. Throws an error if no file by that name exists in any of the relevant `lib`-folders.

`\libusepackage`

`\libusepackage[⟨args⟩]{⟨filename⟩}`

Like `\libinput`, but looks for `.sty`-files and calls `\usepackage[\meta{args}]\Arg{filename}` instead of `\input`.

Throws an error, if none or more than one suitable package file is found.

`\mhgraphics`
`\cmhgraphics`

If the graphicx package is loaded, these macros are defined at `\begin{document}`.

`\mhgraphics` takes the same arguments as `\includegraphics`, with the additional optional key `mhrepos`. It then resolves the file path in `\mhgraphics[mhrepos=Foo/Bar]{foo/bar.png}` relative to the `source`-folder of the `Foo/Bar`-archive.

`\cmhgraphics` additional wraps the image in a `center`-environment.

`\lstinputmhlisting`
`\clstinputmhlisting`

Like `\mhgraphics`, but only defined if the listings-package is loaded, and with `\lstinputlisting` instead of `\includegraphics`.

Chapter 10

STEX-References

This sub package contains code related to links and cross-references

10.1 Macros and Environments

`\STEXreftitle`

`\STEXreftitle{⟨some title⟩}`

Sets the title of the current document to ⟨*some title*⟩. A reference to the current document from *some other* document will then be displayed accordingly. e.g. if `\STEXreftitle{foo book}` is called, then referencing Definition 3.5 in this document in another document will display `Definition 3.5 in foo book`.

`\stex_get_document_uri:` Computes the current document uri from the current archive's `narr`-field and its location relative to the archive's `source`-directory. Reference targets are computed from this URI and the reference-id.

`\l_stex_current_docns_str` Stores its result in `\l_stex_current_docns_str`

`\stex_get_document_url:` Computes the current URL from the current archive's `docurl`-field and its location relative to the archive's `source`-directory. Reference targets are computed from this URL and the reference-id, if this document is only included in SMS mode.

`\l_stex_current_docurl_str` Stores its result in `\l_stex_current_docurl_str`

10.1.1 Setting Reference Targets

`\stex_ref_new_doc_target:n`

`\stex_ref_new_doc_target:n{⟨id⟩}`

Sets a new reference target with id ⟨*id*⟩.

`\stex_ref_new_sym_target:n`

`\stex_ref_new_sym_target:n{⟨uri⟩}`

Sets a new reference target for the symbol ⟨*uri*⟩.

10.1.2 Using References

`\sref` `\sref[⟨opt-args⟩]{⟨id⟩}`

References the label with if ⟨*id*⟩. Optional arguments: TODO

`\srefsym` `\srefsym[⟨opt-args⟩]{⟨symbol⟩}`

Like `\sref`, but references the *canonical label* for the provided symbol. The canonical target is the last of the following occuring in the document:

- A `\definiendum` or `\definame` for ⟨*symbol*⟩,
- The `sassertion`, `sexample` or `sparagraph` with `for=`⟨*symbol*⟩ that generated ⟨*symbol*⟩ in the first place, or
- A `\sparagraph` with `type=symdoc` and `for=`⟨*symbol*⟩.

`\srefsymuri` `\srefsymuri{⟨URI⟩}{⟨text⟩}`

A convenient short-hand for `\srefsym[linktext={text}]{URI}`, but requires the first argument to be a full URI already. Intended to be used in e.g. `\compemph@uri`, `\defemph@uri`, etc.

Chapter 11

sTeX-Modules

This sub package contains code related to Modules

11.1 Macros and Environments

The content of a module with uri ⟨*<URI>*⟩ is stored in four macros. All modifications of these macros are global:

`\c_stex_module_<URI>_prop` A property list with the following fields:

`name`	The *name* of the module,
`ns`	the *namespace* in field `ns`,
`file`	the *file* containing the module, as a sequence of path fragments
`lang`	the module's *language*,
`sig`	the language of the signature module, if the current file is a translation from some other language,
`deprecate`	if this module is deprecated, the module that replaces it,
`meta`	the metatheory of the module.

`\c_stex_module_<URI>_code` The code to execute when this module is activated (i.e. imported), e.g. to set all the semantic macros, notations, etc.

`\c_stex_module_<URI>_constants`

The names of all constants declared in the module

`\c_stex_module_<URI>_constants`

The full URIs of all modules imported in this module

`\l_stex_current_module_str`

`\l_stex_current_module_str` always contains the URI of the current module (if existent).

`\l_stex_all_modules_seq`

Stores full URIs for all modules currently in scope.

`\stex_if_in_module_p:` ⋆
`\stex_if_in_module:TF` ⋆

Conditional for whether we are currently in a module

`\stex_if_module_exists_p:n` ⋆
`\stex_if_module_exists:nTF` ⋆

Conditional for whether a module with the provided URI is already known.

`\stex_add_to_current_module:n`
`\STEXexport`

Adds the provided tokens to the `_code` control sequence of the current module.

`\stex_add_to_current_module:n` is used internally, `\STEXexport` is intended for users and additionally executes the provided code immediately.

`\stex_add_constant_to_current_module:n`

Adds the declaration with the provided name to the `_constants` control sequence of the current module.

`\stex_add_import_to_current_module:n`

Adds the module with the provided full URI to the `_imports` control sequence of the current module.

`\stex_collect_imports:n`

Iterates over all imports of the provided (full URI of a) module and stores them as a topologically sorted list – including the provided module as the last element – in `\l_stex_collect_imports_seq`

`\stex_do_up_to_module:n`

Code that is *exported* from module (such as symbol declarations) should be local *to the current module*. For that reason, ideally all symbol declarations and similar commands should be called directly in the module environment, however, that is not always feasible, e.g. in structural features or `sparapraphs`. `\stex_do_up_to_module` therefore executes the provided code repeatedly in an `\aftergroup` up until the group level is equal to that of the innermost smodule environment.

`\stex_modules_current_namespace:`

Computes the current namespace as follows:

If the current file is `.../source/sub/file.tex` in some archive with namespace `http://some.namespace/foo`, then the namespace of is `http://some.namespace/foo/sub/file`. Otherwise, the namespace is the absolute file path of the current file (i.e. starting with `file:///`).

The result is stored in `\l_stex_modules_ns_str`. Additionally, the sub path relative to the current repository is stored in `\l_stex_modules_subpath_str`.

11.1.1 The `smodule` environment

`module` `\begin{module}[⟨options⟩]{⟨name⟩}`

Opens a new module with name ⟨*name*⟩. Options are:

`title`	(⟨*token list*⟩) to display in customizations.
`type`	(⟨*string*⟩∗) for use in customizations.
`deprecate`	(⟨*module*⟩) if set, will throw a warning when loaded, urging to use ⟨*module*⟩ instead.
`id`	(⟨*string*⟩) for cross-referencing.
`ns`	(⟨*URI*⟩) the namespace to use. *Should not be used, unless you know precisely what you're doing.* If not explicitly set, is computed using `\stex_modules_current_namespace:`.
`lang`	(⟨*language*⟩) if not set, computed from the current file name (e.g. `foo.en.tex`).
`sig`	(⟨*language*⟩) if the current file is a translation of a file with the same base name but a different language suffix, setting `sig=<lang>` will preload the module from that language file. This helps ensuring that the (formal) content of both modules is (almost) identical across languages and avoids duplication.
`creators`	(⟨*string*⟩∗) names of the creators.
`contributors`	(⟨*string*⟩∗) names of contributors.
`srccite`	(⟨*string*⟩) a source citation for the content of this module.

`\stex_module_setup:nn` `\stex_module_setup:nn{⟨params⟩}{⟨name⟩}`

Sets up a new module with name ⟨*name*⟩ and optional parameters ⟨*params*⟩. In particular, sets `\l_stex_current_module_str` appropriately.

`\stexpatchmodule` `\stexpatchmodule [⟨type⟩] {⟨begincode⟩} {⟨endcode⟩}`

Customizes the presentation for those `smodule`-environments with `type=`⟨*type*⟩, or all others if no ⟨*type*⟩ is given.

`\STEXModule` `\STEXModule {⟨fragment⟩}`

Attempts to find a module whose URI ends with ⟨*fragment*⟩ in the current scope and passes the full URI on to `\stex_invoke_module:n`.

`\stex_invoke_module:n`

Invoked by `\STEXModule`. Needs to be followed either by `!\macro` or `?{`⟨*symbolname*⟩`}`. In the first case, it stores the full URI in `\macro`; in the second case, it invokes the symbol ⟨*symbolname*⟩ in the selected module.

`\stex_activate_module:n` Activate the module with the provided URI; i.e. executes all macro code of the module's `_code`-macro (does nothing if the module is already activated in the current context) and adds the module to `\l_stex_all_modules_seq`.

Chapter 12

STEX-Module Inheritance

Code related to Module Inheritance, in particular *sms mode*.

12.1 Macros and Environments

12.1.1 SMS Mode

"SMS Mode" is used when loading modules from external tex files. It deactivates any output and ignores all TEX commands not explicitly allowed via the following lists – all of which either declare module content or are needed in order to declare module content:

`\g_stex_smsmode_allowedmacros_tl`

Macros that are executed as is; i.e. sms mode continues immediately after. These macros may not take any arguments or otherwise gobble tokens.

Initially: `\makeatletter`, `\makeatother`, `\ExplSyntaxOn`, `\ExplSyntaxOff`.

`\g_stex_smsmode_allowedmacros_escape_tl`

Macros that are executed and potentially gobble up further tokens. These macros need to make sure, that the very last token they ultimately expand to is `\stex_smsmode_do:`.

Initially: `\symdecl`, `\notation`, `\symdef`, `\importmodule`, `\STEXexport`, `\inlineass`, `\inlinedef`, `\inlineex`, `\endinput`, `\setnotation`, `\copynotation`.

`\g_stex_smsmode_allowedenvs_seq`

The names of environments that should be allowed in SMS mode. The corresponding `\begin`-statements are treated like the macros in `\g_stex_smsmode_allowedmacros_-escape_tl`, so `\stex_smsmode_do:` needs to be the last token in the `\begin`-code. Since `\end`-statements take no arguments anyway, those are called directly and sms mode continues afterwards.

Initially: `smodule`, `copymodule`, `interpretmodule`, `sdefinition`, `sexample`, `sassertion`, `sparagraph`.

`\stex_if_smsmode_p:` ⋆
`\stex_if_smsmode:TF` ⋆

Tests whether SMS mode is currently active.

`\stex_file_in_smsmode:nn` `\stex_in_smsmode:nn {⟨filename⟩} {⟨code⟩}`

Executes ⟨*code*⟩ in SMS mode, followed by the content of ⟨*filename*⟩. ⟨*code*⟩ can be used e.g. to set the current repository, and is executed within a new tex group, and the same group as the file content.

`\stex_smsmode_do:` Starts gobbling tokens until one is encountered that is allowed in SMS mode.

12.1.2 Imports and Inheritance

`\importmodule` `\importmodule[⟨archive-ID⟩]{⟨module-path⟩}`

Imports a module by reading it from a file and "activating" it. sTeX determines the module and its containing file by passing its arguments on to `\stex_import_module_-path:nn`.

`\usemodule` `\importmodule[⟨archive-ID⟩]{⟨module-path⟩}`

Like `\importmodule`, but does not export its contents; i.e. including the current module will not activate the used module

`\stex_import_module_uri:nn` `\stex_import_module_uri:nn {⟨archive-ID⟩} {⟨module-path⟩}`

Determines the URI of a module by splitting ⟨*module-path*⟩ into ⟨*path*⟩`?`⟨*name*⟩. If ⟨*module-path*⟩ does *not* contain a `?`-character, we consider it to be the ⟨*name*⟩, and ⟨*path*⟩ to be empty.

If ⟨*archive-ID*⟩ is empty, it is automatically set to the ID of the current archive (if one exists).

1. If ⟨*archive-ID*⟩ is empty:

   (a) If ⟨*path*⟩ is empty, then ⟨*name*⟩ must have been declared earlier in the same file and retrievable from `\g_stex_modules_in_file_seq`, or a file with name ⟨*name*⟩`.`⟨*lang*⟩`.tex` must exist in the same folder, containing a module ⟨*name*⟩.

   That module should have the same namespace as the current one.

   (b) If ⟨*path*⟩ is not empty, it must point to the relative path of the containing file as well as the namespace.

2. Otherwise:

   (a) If ⟨*path*⟩ is empty, then ⟨*name*⟩ must have been declared earlier in the same file and retrievable from `\g_stex_modules_in_file_seq`, or a file with name ⟨*name*⟩`.`⟨*lang*⟩`.tex` must exist in the top `source` folder of the archive, containing a module ⟨*name*⟩.

   That module should lie directly in the namespace of the archive.

   (b) If ⟨*path*⟩ is not empty, it must point to the path of the containing file as well as the namespace, relative to the namespace of the archive.

   If a module by that namespace exists, it is returned. Otherwise, we call `\stex_require_module:nn` on the `source` directory of the archive to find the file.

`\l_stex_import_name_str`
`\l_stex_import_archive_str`
`\l_stex_import_path_str`
`\l_stex_import_ns_str`

stores the result in these four variables.

`\stex_import_require_module:nnnn` `{⟨ns⟩} {⟨archive-ID⟩} {⟨path⟩} {⟨name⟩}`

Checks whether a module with URI ⟨*ns*⟩`?`⟨*name*⟩ already exists. If not, it looks for a plausible file that declares a module with that URI.

Finally, activates that module by executing its `_code`-macro.

Chapter 13

sTeX-Symbols

Code related to symbol declarations and notations

13.1 Macros and Environments

`\symdecl` `\symdecl{⟨macroname⟩}[⟨args⟩]`

Declares a new symbol with semantic macro `\macroname`. Optional arguments are:

- `name`: An (OMDoc) name. By default equal to ⟨*macroname*⟩.
- `type`: An (ideally semantic) term, representing a *type*. Not used by sTeX, but passed on to Mmt for semantic services.
- `def`: An (ideally semantic) term, representing a *definiens*. Not used by sTeX, but passed on to Mmt for semantic services.
- `local`: A boolean (by default false). If set, this declaration will not be added to the module content, i.e. importing the current module will not make this declaration available.
- `args`: Specifies the "signature" of the semantic macro. Can be either an integer $0 \leq n \leq 9$, or a (more precise) sequence of the following characters:

  `i` a "normal" argument, e.g. `\symdecl{plus}[args=ii]` allows for `\plus{2}{2}`.

  `a` an *associative* argument; i.e. a sequence of arbitrarily many arguments provided as a comma-separated list, e.g. `\symdecl{plus}[args=a]` allows for `\plus{2,2,2}`.

  `b` a *variable* argument. Is treated by sTeX like an `i`-argument, but an application is turned into an `OMBind` in OMDoc, binding the provided variable in the subsequent arguments of the operator; e.g. `\symdecl{forall}[args=bi]` allows for `\forall{x\in\Nat}{x\geq0}`.

`\stex_symdecl_do:n` Implements the core functionality of `\symdecl`, and is called by `\symdecl` and `\symdef`.

Ultimately stores the symbol ⟨*URI*⟩ in the property list `\l_stex_symdecl_`⟨*URI*⟩`_prop` with fields:

- `name` (string),
- `module` (string),
- `notations` (sequence of strings; initially empty),
- `local` (boolean),
- `type` (token list),
- `args` (string of `i`s, `a`s and `b`s),
- `arity` (integer string),
- `assocs` (integer string; number of associative arguments),

`\stex_all_symbols:n` Iterates over all currently available symbols. Requires two `\seq_map_break:` to break fully.

`\stex_get_symbol:n` Computes the full URI of a symbol from a macro argument, e.g. the macro name, the macro itself, the full URI...

`\notation` `\notation[`⟨*args*⟩`]{`⟨*symbol*⟩`}{`⟨*notations*$^+$⟩`}`

Introduces a new notation for ⟨*symbol*⟩, see `\stex_notation_do:nn`

`\stex_notation_do:nn` `\stex_notation_do:nn{`⟨*URI*⟩`}{`⟨*notations*$^+$⟩`}`

Implements the core functionality of `\notation`, and is called by `\notation` and `\symdef`.

Ultimately stores the notation in the property list `\g_stex_notation_`⟨*URI*⟩`#`⟨*variant*⟩`#`⟨*lang*⟩`_prop` with fields:

- `symbol` (URI string),
- `language` (string),
- `variant` (string),
- `opprec` (integer string),
- `argprecs` (sequence of integer strings)

`\symdef` `\symdef[`⟨*args*⟩`]{`⟨*symbol*⟩`}{`⟨*notations*$^+$⟩`}`

Combines `\symdecl` and `\notation` by introducing a new symbol and assigning a new notation for it.

Chapter 14

sTeX-Terms

Code related to symbolic expressions, typesetting notations, notation components, etc.

14.1 Macros and Environments

`\STEXsymbol`

Uses `\stex_get_symbol:n` to find the symbol denoted by the first argument and passes the result on to `\stex_invoke_symbol:n`

`\symref`

`\symref{⟨symbol⟩}{⟨text⟩}`

shortcut for `\STEXsymbol{`⟨*symbol*⟩`}![`⟨*text*⟩`]`

`\stex_invoke_symbol:n`

Executes a semantic macro. Outside of math mode or if followed by `*`, it continues to `\stex_term_custom:nn`. In math mode, it uses the default or optionally provided notation of the associated symbol.

If followed by `!`, it will invoke the symbol *itself* rather than its application (and continue to `\stex_term_custom:nn`), i.e. it allows to refer to `\plus![addition]` as an operation, rather than `\plus[addition of]{some}{terms}`.

`\_stex_term_math_oms:nnnn`
`\_stex_term_math_oma:nnnn`
`\_stex_term_math_omb:nnnn`

⟨*URI*⟩⟨*fragment*⟩⟨*precedence*⟩⟨*body*⟩

Annotates ⟨*body*⟩ as an OMDoc-term (`OMID`, `OMA` or `OMBIND`, respectively) with head symbol ⟨*URI*⟩, generated by the specific notation ⟨*fragment*⟩ with (upwards) operator precedence ⟨*precedence*⟩. Inserts parentheses according to the current downwards precedence and operator precedence.

`\_stex_term_math_arg:nnn`

`\stex_term_arg:nnn`⟨*int*⟩⟨*prec*⟩⟨*body*⟩

Annotates ⟨*body*⟩ as the ⟨*int*⟩th argument of the current `OMA` or `OMBIND`, with (downwards) argument precedence ⟨*prec*⟩.

`\_stex_term_math_assoc_arg:nnnn`

`\stex_term_arg:nnn`⟨*int*⟩⟨*prec*⟩⟨*notation*⟩⟨*body*⟩

Annotates ⟨*body*⟩ as the ⟨*int*⟩th (associative) *sequence* argument (as comma-separated list of terms) of the current `OMA` or `OMBIND`, with (downwards) argument precedence ⟨*prec*⟩ and associative notation ⟨*notation*⟩.

`\infprec`
`\neginfprec`

Maximal and minimal notation precedences.

`\dobrackets`

`\dobrackets {⟨body⟩}`

Puts ⟨*body*⟩ in parentheses; scaled if in display mode unscaled otherwise. Uses the current sTeX brackets (by default `(` and `)`), which can be changed temporarily using `\withbrackets`.

`\withbrackets`

`\withbrackets ⟨left⟩ ⟨right⟩ {⟨body⟩}`

Temporarily (i.e. within ⟨*body*⟩) sets the brackets used by sTeX for automated bracketing (by default `(` and `)`) to ⟨*left*⟩ and ⟨*right*⟩.

Note that ⟨*left*⟩ and ⟨*right*⟩ need to be allowed after `\left` and `\right` in display-mode.

`\stex_term_custom:nn`

`\stex_term_custom:nn{⟨URI⟩}{⟨args⟩}`

Implements custom one-time notation. Invoked by `\stex_invoke_symbol:n` in text mode, or if followed by `*` in math mode, or whenever followed by `!`.

`\stex_highlight_term:nn`

`\stex_highlight_term:nn{⟨URI⟩}{⟨args⟩}`

Establishes a context for `\comp`. Stores the URI in a variable so that `\comp` knows which symbol governs the current notation.

`\comp`
`\compemph`
`\compemph@uri`
`\defemph`
`\defemph@uri`
`\symrefemph`
`\symrefemph@uri`
`\varemph`
`\varemph@uri`

`\comp{⟨args⟩}`

Marks ⟨*args*⟩ as a notation component of the current symbol for highlighting, linking, etc.

The precise behavior is governed by `\@comp`, which takes as additional argument the URI of the current symbol. By default, `\@comp` adds the URI as a PDF tooltip and colors the highlighted part in blue.

`\@defemph` behaves like `\@comp`, and can be similarly redefined, but marks an expression as *definiendum* (used by `\definiendum`)

`\STEXinvisible`

Exports its argument as OMDoc (invisible), but does not produce PDF output. Useful e.g. for semantic macros that take arguments that are not part of the symbolic notation.

`\ellipses`

TODO

Chapter 15

sTeX-Structural Features

Code related to structural features

15.1 Macros and Environments

15.1.1 Structures

`mathstructure` TODO

Chapter 16

sTeX-Statements

Code related to statements, e.g. definitions, theorems

16.1 Macros and Environments

`symboldoc` `\begin{⟨symboldoc⟩}{⟨symbols⟩} ⟨text⟩ \end{⟨symboldoc⟩}`

Declares ⟨*text*⟩ to be a (natural language, encyclopaedic) description of `{`⟨*symbols*⟩`}` (a comma separated list of symbol identifiers).

Chapter 17

sTeX-Proofs: Structural Markup for Proofs

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

This package supplies macros and environment that allow to annotate the structure of mathematical proofs in sTeX files. This structure can be used by MKM systems for added-value services, either directly from the sTeX sources, or after translation.

Contents

17.1 Introduction

The `sproof` (semantic proofs) package supplies macros and environment that allow to annotate the structure of mathematical proofs in sTeX files. This structure can be used by MKM systems for added-value services, either directly from the sTeX sources, or after translation. Even though it is part of the sTeX collection, it can be used independently, like it's sister package `statements`.

sTeX is 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).

```
\begin{sproof}[id=simple-proof]
   {We prove that $\sum_{i=1}^n{2i-1}=n^{2}$ by induction over $n$}
  \begin{spfcases}{For the induction we have to consider the following cases:}
   \begin{spfcase}{$n=1$}
    \begin{spfstep}[type=inline] then we compute $1=1^2$\end{spfstep}
   \end{spfcase}
   \begin{spfcase}{$n=2$}
      \begin{sproofcomment}[type=inline]
        This case is not really necessary, but we do it for the
        fun of it (and to get more intuition).
      \end{sproofcomment}
      \begin{spfstep}[type=inline] We compute $1+3=2^{2}=4$.\end{spfstep}
   \end{spfcase}
   \begin{spfcase}{$n>1$}
      \begin{spfstep}[type=assumption,id=ind-hyp]
        Now, we assume that the assertion is true for a certain $k\geq 1$,
        i.e. $\sum_{i=1}^k{(2i-1)}=k^{2}$.
      \end{spfstep}
      \begin{sproofcomment}
        We have to show that we can derive the assertion for $n=k+1$ from
        this assumption, i.e. $\sum_{i=1}^{k+1}{(2i-1)}=(k+1)^{2}$.
      \end{sproofcomment}
      \begin{spfstep}
        We obtain $\sum_{i=1}^{k+1}{2i-1}=\sum_{i=1}^k{2i-1}+2(k+1)-1$
        \begin{justification}[method=arith:split-sum]
          by splitting the sum.
        \end{justification}
      \end{spfstep}
      \begin{spfstep}
        Thus we have $\sum_{i=1}^{k+1}{(2i-1)}=k^2+2k+1$
        \begin{justification}[method=fertilize]
          by inductive hypothesis.
        \end{justification}
      \end{spfstep}
      \begin{spfstep}[type=conclusion]
        We can \begin{justification}[method=simplify]simplify\end{justification}
        the right-hand side to ${k+1}^2$, which proves the assertion.
      \end{spfstep}
   \end{spfcase}
    \begin{spfstep}[type=conclusion]
      We have considered all the cases, so we have proven the assertion.
    \end{spfstep}
  \end{spfcases}
\end{sproof}
```

Example 1: A very explicit proof, marked up semantically

We will go over the general intuition by way of our running example (see Figure 1 for the source and Figure 2 for the formatted result).[2] EdN:2

[2] EdNote: talk a bit more about proofs and their structure,... maybe copy from OMDoc spec.

17.2 The User Interface

17.2.1 Package Options

showmeta The sproof package takes a single option: showmeta. If this is set, then the metadata keys are shown (see [**Kohlhase:metakeys**] for details and customization options).

17.2.2 Proofs and Proof steps

sproof The proof environment is the main container for proofs. It takes an optional KeyVal argument that allows to specify the id (identifier) and for (for which assertion is this a proof) keys. The regular argument of the proof environment contains an introductory comment, that may be used to announce the proof style. The proof environment contains a sequence of \step, proofcomment, and pfcases environments that are used to markup the proof steps. The proof environment has a variant Proof, which does not use the proof end marker. This is convenient, if a proof ends in a case distinction, which brings

sProof it's own proof end marker with it. The Proof environment is a variant of proof that does not mark the end of a proof with a little box; presumably, since one of the subproofs already has one and then a box supplied by the outer proof would generate an otherwise

\spfidea empty line. The \spfidea macro allows to give a one-paragraph description of the proof idea.

spfsketch For one-line proof sketches, we use the \spfsketch macro, which takes the KeyVal argument as sproof and another one: a natural language text that sketches the proof.

spfstep Regular proof steps are marked up with the step environment, which takes an optional KeyVal argument for annotations. A proof step usually contains a local assertion (the text of the step) together with some kind of evidence that this can be derived from already established assertions.

Note that both \premise and \justarg can be used with an empty second argument to mark up premises and arguments that are not explicitly mentioned in the text.

17.2.3 Justifications

justification This evidence is marked up with the justification environment in the sproof package. This environment totally invisible to the formatted result; it wraps the text in the proof step that corresponds to the evidence. The environment takes an optional KeyVal argument, which can have the method key, whose value is the name of a proof method (this will only need to mean something to the application that consumes the semantic annotations). Furthermore, the justification can contain "premises" (specifications to assertions that were used justify the step) and "arguments" (other information taken into account by the proof method).

\premise The \premise macro allows to mark up part of the text as reference to an assertion that is used in the argumentation. In the example in Figure 1 we have used the \premise macro to identify the inductive hypothesis.

\justarg The \justarg macro is very similar to \premise with the difference that it is used to mark up arguments to the proof method. Therefore the content of the first argument is interpreted as a mathematical object rather than as an identifier as in the case of \premise. In our example, we specified that the simplification should take place on the right hand side of the equation. Other examples include proof methods that instantiate. Here we would indicate the substituted object in a \justarg macro.

Proof: We prove that $\sum_{i=1}^{n} 2i - 1 = n^2$ by induction over n

1. For the induction we have to consider the following cases:

1.1. $n = 1$: then we compute $1 = 1^2$ □

1.2. $n = 2$: This case is not really necessary, but we do it for the fun of it (and to get more intuition). We compute $1 + 3 = 2^2 = 4$ □

1.3. $n > 1$:

1.3.1. Now, we assume that the assertion is true for a certain $k \geq 1$, i.e. $\sum_{i=1}^{k} (2i - 1) = k^2$.

1.3.2. We have to show that we can derive the assertion for $n = k + 1$ from this assumption, i.e. $\sum_{i=1}^{k+1} (2i - 1) = (k + 1)^2$.

1.3.3. We obtain $\sum_{i=1}^{k+1} (2i - 1) = \sum_{i=1}^{k} (2i - 1) + 2(k + 1) - 1$ by splitting the sum

1.3.4. Thus we have $\sum_{i=1}^{k+1} (2i - 1) = k^2 + 2k + 1$ by inductive hypothesis.

1.3.5. We can simplify the right-hand side to $(k + 1)^2$, which proves the assertion. □

1.4. We have considered all the cases, so we have proven the assertion.

□

Example 2: The formatted result of the proof in Figure 1

17.2.4 Proof Structure

`subproof` `method` The `pfcases` environment is used to mark up a subproof. This environment takes an optional `KeyVal` argument for semantic annotations and a second argument that allows to specify an introductory comment (just like in the `proof` environment). The `method` key can be used to give the name of the proof method executed to make this subproof.

`spfcases` The `pfcases` environment is used to mark up a proof by cases. Technically it is a variant of the `subproof` where the `method` is `by-cases`. Its contents are `spfcase` environments that mark up the cases one by one.

`spfcase` The content of a `pfcases` environment are a sequence of case proofs marked up in the `pfcase` environment, which takes an optional `KeyVal` argument for semantic annotations. The second argument is used to specify the the description of the case under consideration. The content of a `pfcase` environment is the same as that of a `proof`, i.e. `\spfcasesketch` `steps`, `proofcomments`, and `pfcases` environments. `\spfcasesketch` is a variant of the `spfcase` environment that takes the same arguments, but instead of the `spfsteps` in the body uses a third argument for a proof sketch.

`sproofcomment` The `proofcomment` environment is much like a `step`, only that it does not have an object-level assertion of its own. Rather than asserting some fact that is relevant for the proof, it is used to explain where the proof is going, what we are attempting to to, or what we have achieved so far. As such, it cannot be the target of a `\premise`.

17.2.5 Proof End Markers

Traditionally, the end of a mathematical proof is marked with a little box at the end of the last line of the proof (if there is space and on the end of the next line if there isn't), like so:

`\sproofend` The `sproof` package provides the `\sproofend` macro for this. If a different symbol for the proof end is to be used (e.g. *q.e.d*), then this can be obtained by specifying it using the `\sProofEndSymbol` `\sProofEndSymbol` configuration macro (e.g. by specifying `\sProofEndSymbol{q.e.d}`).

Some of the proof structuring macros above will insert proof end symbols for subproofs, in most cases, this is desirable to make the proof structure explicit, but sometimes this wastes space (especially, if a proof ends in a case analysis which will supply its own proof end marker). To suppress it locally, just set `proofend={}` in them or use use `\sProofEndSymbol{}`.

17.2.6 Configuration of the Presentation

Finally, we provide configuration hooks in Figure 1 for the keywords in proofs. These are mainly intended for package authors building on `statements`, e.g. for multi-language support.[3]. The proof step labels can be customized via the `\pstlabelstyle` macro: EdN:3

Environment	configuration macro	value
`sproof`	`\spf@proof@kw`	Proof
`sketchproof`	`\spf@sketchproof@kw`	Proof Sketch

Figure 1: Configuration Hooks for Semantic Proof Markup

`\pstlabelstyle`

`\pstlabelstyle{`⟨*style*⟩`}` sets the style; see Figure ?? for an overview of styles. Package writers can add additional styles by adding a macro `\pst@make@label@`⟨*style*⟩ that takes

[3]EdNote: we might want to develop an extension `sproof-babel` in the future.

two arguments: a comma-separated list of ordinals that make up the prefix and the current ordinal. Note that comma-separated lists can be conveniently iterated over by the LaTeX `\@for...:=...\do{...}` macro; see Figure ?? for examples.

17.3 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 issue tracker at [sTeX].

1. The numbering scheme of proofs cannot be changed. It is more geared for teaching proof structures (the author's main use case) and not for writing papers. reported by Tobias Pfeiffer (fixed)

2. currently proof steps are formatted by the LaTeX `description` environment. We would like to configure this, e.g. to use the `inparaenum` environment for more condensed proofs. I am just not sure what the best user interface would be I can imagine redefining an internal environment `spf@proofstep@list` or adding a key `prooflistenv` to the `proof` environment that allows to specify the environment directly. Maybe we should do both.

Chapter 18

sTeX-Metatheory

The default meta theory for an sTeX module. Contains symbols so ubiquitous, that it is virtually impossible to describe any flexiformal content without them, or that are required to annotate even the most primitive symbols with meaningful (foundation-independent) "type"-annotations, or required for basic structuring principles (theorems, definitions).

Foundations should ideally instantiate these symbols with their formal counterparts, e.g. `isa` corresponds to a typing operation in typed setting, or the $\in$-operator in set-theoretic contexts; `bind` corresponds to a universal quantifier in (nth-order) logic, or a Π in dependent type theories.

18.1 Symbols

Part III
Extensions

Chapter 19

Tikzinput

19.1 Macros and Environments

LocalWords: bibfolder jobname.dtx tikzinput.dtx usetikzlibrary Gin@ewidth Gin@eheight
LocalWords: resizebox ctikzinput mhtikzinput Gin@mhrepos mhpath

Chapter 20

document-structure: Semantic Markup for Open Mathematical Documents in LaTeX

The `document-structure` package is part of the sTeX collection, a version of TeX/LaTeX that allows to markup TeX/LaTeX documents semantically without leaving the document format, essentially turning TeX/LaTeX into a document format for mathematical knowledge management (MKM).

This package supplies an infrastructure for writing OMDoc documents in LaTeX. This includes a simple structure sharing mechanism for sTeX that allows to to move from a copy-and-paste document development model to a copy-and-reference model, which conserves space and simplifies document management. The augmented structure can be used by MKM systems for added-value services, either directly from the sTeX sources, or after translation.

20.1 Introduction

sTeX is 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 package supports direct translation to the OMDoc format [Koh06]

The `document-structure` package supplies macros and environments that allow to label document fragments and to reference them later in the same document or in other documents. In essence, this enhances the document-as-trees model to documents-as-directed-acyclic-graphs (DAG) model. This structure can be used by MKM systems for added-value services, either directly from the sTeX sources, or after translation. Currently, trans-document referencing provided by this package can only be used in the sTeX collection.

DAG models of documents allow to replace the “Copy and Paste” in the source document with a label-and-reference model where document are shared in the document

EdN:4 source and the formatter does the copying during document formatting/presentation.[4]

20.2 The User Interface

The `document-structure` package generates two files: `document-structure.cls`, and `document-structure.sty`. The OMDoc class is a minimally changed variant of the standard `article` class that includes the functionality provided by `document-structure.sty`. The rest of the documentation pertains to the functionality introduced by `document-structure.sty`.

20.2.1 Package and Class Options

The `document-strcture` class accept the following options:

`class=`⟨*name*⟩	load ⟨*name*⟩`.cls` instead of `article.cls`
`topsect=`⟨*sect*⟩	The top-level sectioning level; the default for ⟨*sect*⟩ is `section`
`showignores`	show the the contents of the `ignore` environment after all
`showmeta`	show the metadata; see `metakeys.sty`
`showmods`	show modules; see `modules.sty`
`extrefs`	allow external references; see `sref.sty`
`defindex`	index definienda; see `statements.sty`
`minimal`	for testing; do not load any sTeX packages

The `document-structure` package accepts the same except the first two.

20.2.2 Document Structure

`document` `\documentkeys` `id` The top-level `document` environment can be given key/value information by the `\documentkeys` macro in the preamble[3]. This can be used to give metadata about the document. For the moment only the `id` key is used to give an identifier to the `omdoc` element resulting from the LaTeXML transformation.

`sfragment` The structure of the document is given by the `omgroup` environment just like in OMDoc. In the LaTeX route, the `omgroup` environment is flexibly mapped to sectioning commands, inducing the proper sectioning level from the nesting of `omgroup` environments. Correspondingly, the `omgroup` environment takes an optional key/value argument for metadata followed by a regular argument for the (section) title of the omgroup. The optional metadata argument has the keys `id` for an identifier, `creators` and `contributors` for the Dublin Core metadata [DCM03]; see [Koh20a] for details of the format. The `short` allows to give a short title for the generated section. If the title contains semantic macros, they need to be protected by `\protect`, and we need to give the `loadmodules` key it needs no value. For instance we would have

`id` `creators` `contributors` `short` `loadmodules`

```
\begin{smodule}{foo}
\symdef{bar}{B^a_r}
 ...
\begin{sfragment}[id=sec.barderiv,loadmodules]{Introducing $\protect\bar$ Derivation
```

[4]EdNote: integrate with latexml's XMRef in the Math mode.

[3]We cannot patch the document environment to accept an optional argument, since other packages we load already do; pity.

sTeX automatically computes the sectioning level, from the nesting of `omgroup` environments. But sometimes, we want to skip levels (e.g. to use a subsection* as an introduction for a chapter). Therefore the `document-structure` package provides a variant `blindomgroup` that does not produce markup, but increments the sectioning level and logically groups document parts that belong together, but where traditional document markup relies on convention rather than explicit markup. The `blindomgroup` environment is useful e.g. for creating frontmatter at the correct level. Example 3 shows a typical setup for the outer document structure of a book with parts and chapters. We use two levels of `blindomgroup`:

`blindfragment`

- The outer one groups the introductory parts of the book (which we assume to have a sectioning hierarchy topping at the part level). This `blindomgroup` makes sure that the introductory remarks become a "chapter" instead of a "part".
- Th inner one groups the frontmatter[4] and makes the preface of the book a section-level construct. Note that here the `display=flow` on the `omgroup` environment prevents numbering as is traditional for prefaces.

```
\begin{document}
\begin{blindfragment}
\begin{blindfragment}
\begin{frontmatter}
\maketitle\newpage
\begin{sfragment}[display=flow]{Preface}
... <<preface>> ...
\end{sfragment}
\clearpage\setcounter{tocdepth}{4}\tableofcontents\clearpage
\end{frontmatter}
\end{blindfragment}
... <<introductory remarks>> ...
\end{blindfragment}
\begin{sfragment}{Introduction}
... <<intro>> ...
\end{sfragment}
... <<more chapters>> ...
\bibliographystyle{alpha}\bibliography{kwarc}
\end{document}
```

Example 3: A typical Document Structure of a Book

`\skipomgroup`

The `\skipomgroup` "skips an `omgroup`", i.e. it just steps the respective sectioning counter. This macro is useful, when we want to keep two documents in sync structurally, so that section numbers match up: Any section that is left out in one becomes a `\skipomgroup`.

`\currentsectionlevel`
`\CurrentSectionLevel`

The `\currentsectionlevel` macro supplies the name of the current sectioning level, e.g. "chapter", or "subsection". `\CurrentSectionLevel` is the capitalized variant. They are useful to write something like "In this `\currentsectionlevel`, we will…" in an `omgroup` environment, where we do not know which sectioning level we will end up.

[4]We shied away from redefining the `frontmatter` to induce a blindomgroup, but this may be the "right" way to go in the future.

20.2.3 Ignoring Inputs

ignore
showignores

The `ignore` environment can be used for hiding text parts from the document structure. The body of the environment is not PDF or DVI output unless the `showignores` option is given to the `document-structure` class or `package`. But in the generated OMDoc result, the body is marked up with a `ignore` element. This is useful in two situations. For

editing One may want to hide unfinished or obsolete parts of a document

narrative/content markup In sTeX we mark up narrative-structured documents. In the generated OMDoc documents we want to be able to cache content objects that are not directly visible. For instance in the `statements` package [Koh20d] we use the `\inlinedef` macro to mark up phrase-level definitions, which verbalize more formal definitions. The latter can be hidden by an ignore and referenced by the `verbalizes` key in `\inlinedef`.

\premovedstop

For prematurely stopping the formatting of a document, sTeX provides the `\prematurestop` macro. It can be used everywhere in a document and ignores all input after that – backing out of the omgroup environment as needed. After that – and before the implicit `\end{document}` it calls the internal `\afterprematurestop`, which can be customized to do additional cleanup or e.g. print the bibliography.

\afterprematurestop

`\prematurestop` is useful when one has a driver file, e.g. for a course taught multiple years and wants to generate course notes up to the current point in the lecture. Instead of commenting out the remaining parts, one can just move the `\prematurestop` macro. This is especially useful, if we need the rest of the file for processing, e.g. to generate a theory graph of the whole course with the already-covered parts marked up as an overview over the progress; see `import_graph.py` from the `lmhtools` utilities [LMH].

20.2.4 Structure Sharing

\STRlabel
\STRcopy

The `\STRlabel` macro takes two arguments: a label and the content and stores the the content for later use by `\STRcopy[`⟨*URL*⟩`]{`⟨*label*⟩`}`, which expands to the previously stored content. If the `\STRlabel` macro was in a different file, then we can give a URL ⟨*URL*⟩ that lets LaTeXML generate the correct reference.

\STRsemantics

The `\STRlabel` macro has a variant `\STRsemantics`, where the label argument is optional, and which takes a third argument, which is ignored in LaTeX. This allows to specify the meaning of the content (whatever that may mean) in cases, where the source document is not formatted for presentation, but is transformed into some content markup format.[5]

EdN:5

20.2.5 Global Variables

Text fragments and modules can be made more re-usable by the use of global variables. For instance, the admin section of a course can be made course-independent (and therefore re-usable) by using variables (actually token registers) `courseAcronym` and `courseTitle` instead of the text itself. The variables can then be set in the sTeX preamble of the course notes file. `\setSGvar{`⟨*vname*⟩`}{`⟨*text*⟩`}` to set the global variable ⟨*vname*⟩ to ⟨*text*⟩ and `\useSGvar{`⟨*vname*⟩`}` to reference it.

\setSGvar
\useSGvar
\ifSGvar

With `\ifSGvar` we can test for the contents of a global variable: the macro call

[5] EdNote: document LMID und LMXREf here if we decide to keep them.

`\ifSGvar{`⟨*vname*⟩`}{`⟨*val*⟩`}{`⟨*ctext*⟩`}` tests the content of the global variable ⟨*vname*⟩, only if (after expansion) it is equal to ⟨*val*⟩, the conditional text ⟨*ctext*⟩ is formatted.

20.2.6 Colors

`\blue` `\red` ... `\black`

For convenience, the `document-structure` package defines a couple of color macros for the `color` package: For instance `\blue` abbreviates `\textcolor{blue}`, so that `\blue{`⟨*something*⟩`}` writes ⟨*something*⟩ in blue. The macros `\red` `\green`, `\cyan`, `\magenta`, `\brown`, `\yellow`, `\orange`, `\gray`, and finally `\black` are analogous.

20.3 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. when option `book` which uses `\pagestyle{headings}` is given and semantic macros are given in the `omgroup` titles, then they sometimes are not defined by the time the heading is formatted. Need to look into how the headings are made.

Chapter 21

NotesSlides – Slides and Course Notes

We present a document class from which we can generate both course slides and course notes in a transparent way.

21.1 Introduction

The `notesslides` document class is derived from `beamer.cls` [Tana], it adds a "notes version" for course notes derived from the `omdoc` class [**Kohlhase:smomdl**] that is more suited to printing than the one supplied by `beamer.cls`.

21.2 The User Interface

The `notesslides` class takes the notion of a slide frame from Till Tantau's excellent `beamer` class and adapts its notion of frames for use in the sTEXand OMDoc. To support semantic course notes, it extends the notion of mixing frames and explanatory text, but rather than treating the frames as images (or integrating their contents into the flowing text), the `notesslides` package displays the slides as such in the course notes to give students a visual anchor into the slide presentation in the course (and to distinguish the different writing styles in slides and course notes).

In practice we want to generate two documents from the same source: the slides for presentation in the lecture and the course notes as a narrative document for home study. To achieve this, the `notesslides` class has two modes: *slides mode* and *notes mode* which are determined by the package option.

21.2.1 Package Options

EdN:6

The `notesslides` class takes a variety of class options:[6]

`slides`
`notes`

- The options `slides` and `notes` switch between slides mode and notes mode (see Section 21.2.2).

sectocframes

- If the option `sectocframes` is given, then for the `omgroups`, special frames with the `omgroup` title (and number) are generated.

showmeta

- `showmeta`. If this is set, then the metadata keys are shown (see [Koh20b] for details and customization options).

frameimages
fiboxed

- If the option `frameimages` is set, then slide mode also shows the `\frameimage`-generated frames (see section 21.2.4). If also the `fiboxed` option is given, the slides are surrounded by a box.

topsect

- `topsect=`⟨*sect*⟩ can be used to specify the top-level sectioning level; the default for ⟨*sect*⟩ is `section`.

21.2.2 Notes and Slides

frame
note

Slides are represented with the `frame` just like in the `beamer` class, see [Tanb] for details. The `notesslides` class adds the `note` environment for encapsulating the course note fragments.[5]

Note that it is essential to start and end the `notes` environment at the start of the line – in particular, there may not be leading blanks – else LaTeX becomes confused and throws error messages that are difficult to decipher.

```
\ifnotes\maketitle\else
\frame[noframenumbering]\maketitle\fi

\begin{note}
  We start this course with ...
\end{note}

\begin{frame}
  \frametitle{The first slide}
  ...
\end{frame}
\begin{note}
  ... and more explanatory text
\end{note}

\begin{frame}
  \frametitle{The second slide}
  ...
\end{frame}
...
```

Example 4: A typical Course Notes File

By interleaving the `frame` and `note` environments, we can build course notes as shown in Figure 4.

\ifnotes

Note the use of the `\ifnotes` conditional, which allows different treatment between

[6]EdNote: leaving out noproblems for the moment until we decide what to do with it.

[5]MK: it would be very nice, if we did not need this environment, and this should be possible in principle, but not without intensive LaTeX trickery. Hints to the author are welcome.

`notes` and `slides` mode – manually setting `\notestrue` or `\notesfalse` is strongly discouraged however.

⚠: We need to give the title frame the `noframenumbering` option so that the frame numbering is kept in sync between the slides and the course notes.

⚠: The `beamer` class recommends not to use the `allowframebreaks` option on frames (even though it is very convenient). This holds even more in the `notesslides` case: At least in conjunction with `\newpage`, frame numbering behaves funnily (we have tried to fix this, but who knows).

`\inputref*`

If we want to transclude a the contents of a file as a note, we can use a new variant `\inputref*` of the `\inputref` macro from [KGA20]: `\inputref*{foo}` is equivalent to `\begin{note}\inputref{foo}\end{note}`.

`nparagraph` `nfragment` `ndefinition` `nexample` `nsproof` `nassertion`

There are some environments that tend to occur at the top-level of `note` environments. We make convenience versions of these: e.g. the `nparagraph` environment is just an `sparagraph` inside a `note` environment (but looks nicer in the source, since it avoids one level of source indenting). Similarly, we have the `nomgroup`, `ndefinition`, `nexample`, `nsproof`, and `nassertion` environments.

21.2.3 Header and Footer Lines of the Slides

`\setslidelogo`

The default logo provided by the `notesslides` package is the sTeX logo it can be customized using `\setslidelogo{`⟨*logo name*⟩`}`.

`\setsource`

The default footer line of the `notesslides` package mentions copyright and licensing. In the `beamer` class, `\source` stores the author's name as the copyright holder . By default it is *Michael Kohlhase* in the `notesslides` package since he is the main user and designer of this package. `\setsource{`⟨*name*⟩`}` can change the writer's name. For licensing, we use the Creative Commons Attribuition-ShareAlike license by default to strengthen the public domain. If package `hyperref` is loaded, then we can attach a hyperlink to the license logo. `\setlicensing[`⟨*url*⟩`]{`⟨*logo name*⟩`}` is used for customization, where ⟨*url*⟩ is optional.

`\setlicensing`

21.2.4 Frame Images

`\frameimage`

Sometimes, we want to integrate slides as images after all – e.g. because we already have a PowerPoint presentation, to which we want to add sTeXnotes. In this case we can use `\frameimage[`⟨*opt*⟩`]{`⟨*path*⟩`}`, where ⟨*opt*⟩ are the options of `\includegraphics` from the `graphicx` package [CR99] and ⟨*path*⟩ is the file path (extension can be left off like in `\includegraphics`). We have added the `label` key that allows to give a frame label that can be referenced like a regular `beamer` frame.[7] EdN:7

`\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}
```

[7] EdNote: MK: the hyperref link does not seem to work yet. I wonder why but do not have the time to fix it.

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}
```

21.2.5 Colors and Highlighting

`\textwarning` The `\textwarning` macro generates a warning sign: ⚠

21.2.6 Front Matter, Titles, etc.

21.2.7 Excursions

In course notes, we sometimes want to point to an "excursion" – material that is either presupposed or tangential to the course at the moment – e.g. in an appendix. The typical setup is the following:

```
\excursion{founif}{../ex/founif}{We will cover first-order unification in}
...
\begin{appendix}\printexcursions\end{appendix}
```

`\excursion` `\activateexcursion` The `\excursion{`⟨*ref*⟩`}{`⟨*path*⟩`}{`⟨*text*⟩`}` is syntactic sugar for

```
\begin{nparagraph}[title=Excursion]
  \activateexcursion{founif}{../ex/founif}
  We will cover first-order unification in \sref{founif}.
\end{nparagraph}
```

`\activateexcursion` `\printexcursions` where `\activateexcursion{`⟨*path*⟩`}` augments the `\printexcursions` macro by a call `\inputref{`⟨*path*⟩`}`. In this way, the3 `\printexcursions` macro (usually in the appendix) will collect up all excursions that are specified in the main text.

`\excursionref` Sometimes, we want to reference – in an excursion – part of another. We can use `\excursionref{`⟨*label*⟩`}` for that.

`\excursiongroup` Finally, we usually want to put the excursions into an `omgroup` environment and add an introduction, therefore we provide the a variant of the `\printexcursions` macro: `\excursiongroup[id=`⟨*id*⟩`,intro=`⟨*path*⟩`]` is equivalent to

```
\begin{note}
\begin{sfragment}[id=<id>]{Excursions}
  \inputref{<path>}
  \printexcursions
\end{sfragment}
\end{note}
```

21.2.8 Miscellaneous

21.3 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 sTeXGitHub repository [sTeX].

1. when option `book` which uses `\pagestyle{headings}` is given and semantic macros are given in the `omgroup` titles, then they sometimes are not defined by the time the heading is formatted. Need to look into how the headings are made. This is a problem of the underlying `omdoc` package.

Chapter 22

`problem.sty`: An Infrastructure for formatting Problems

The `problem` package supplies an infrastructure that allows specify problems and to reuse them efficiently in multiple environments.

22.1 Introduction

The `problem` package supplies an infrastructure that allows specify problem. Problems are text fragments that come with auxiliary functions: hints, notes, and solutions[6]. Furthermore, we can specify how long the solution to a given problem is estimated to take and how many points will be awarded for a perfect solution.

Finally, the `problem` package facilitates the management of problems in small files, so that problems can be re-used in multiple environment.

22.2 The User Interface

22.2.1 Package Options

`solutions` `notes` `hints` `gnotes` `pts` `min`

The `problem` package takes the options `solutions` (should solutions be output?), `notes` (should the problem notes be presented?), `hints` (do we give the hints?), `gnotes` (do we show grading notes?), `pts` (do we display the points awarded for solving the problem?), `min` (do we display the estimated minutes for problem soling). If theses are specified, then the corresponding auxiliary parts of the problems are output, otherwise, they remain invisible.

`boxed` `test`

The `boxed` option specifies that problems should be formatted in framed boxes so that they are more visible in the text. Finally, the `test` option signifies that we are in a test situation, so this option does not show the solutions (of course), but leaves space for the students to solve them.

`mh`

The `mh` option turns on MathHub support; see [**Kohlhase:mss**].

`showmeta`

Finally, if the `showmeta` is set, then the metadata keys are shown (see [**Kohlhase:metakeys**] for details and customization options).

[6]for the moment multiple choice problems are not supported, but may well be in a future version

22.2.2 Problems and Solutions

`problem` `id` `pts` `min` `title`

The main environment provided by the `problem` package is (surprise surprise) the `problem` environment. It is used to mark up problems and exercises. The environment takes an optional KeyVal argument with the keys `id` as an identifier that can be reference later, `pts` for the points to be gained from this exercise in homework or quiz situations, `min` for the estimated minutes needed to solve the problem, and finally `title` for an informative title of the problem. For an example of a marked up problem see Figure 5 and the resulting markup see Figure 6.

```
\usepackage[solutions,hints,pts,min]{problem}
\begin{document}
  \begin{sproblem}[id=elefants,pts=10,min=2,title=Fitting Elefants]
    How many Elefants can you fit into a Volkswagen beetle?
\begin{hint}
  Think positively, this is simple!
\end{hint}
\begin{exnote}
  Justify your answer
\end{exnote}
\begin{solution}[for=elefants,height=3cm]
  Four, two in the front seats, and two in the back.
\begin{gnote}
  if they do not give the justification deduct 5 pts
\end{gnote}
\end{solution}
  \end{sproblem}
\end{document}
```

Example 5: A marked up Problem

`solution` `solutions` `id` `for` `height` `test`

The `solution` environment can be to specify a solution to a problem. If the `solutions` option is set or `\solutionstrue` is set in the text, then the solution will be presented in the output. The `solution` environment takes an optional KeyVal argument with the keys `id` for an identifier that can be reference `for` to specify which problem this is a solution for, and `height` that allows to specify the amount of space to be left in test situations (i.e. if the `test` option is set in the `\usepackage` statement).

Problem 0.1 (Fitting Elefants)
How many Elefants can you fit into a Volkswagen beetle?

Hint: Think positively, this is simple!

Note: Justify your answer

Solution: Four, two in the front seats, and two in the back.

Example 6: The Formatted Problem from Figure 5

`hint` `exnote` `gnote`

The `hint` and `exnote` environments can be used in a `problem` environment to give hints and to make notes that elaborate certain aspects of the problem.

The `gnote` (grading notes) environment can be used to document situtations that may arise in grading.

\startsolutions \stopsolutions

Sometimes we would like to locally override the `solutions` option we have given to the package. To turn on solutions we use the `\startsolutions`, to turn them off, `\stopsolutions`. These two can be used at any point in the documents.

\ifsolutions

Also, sometimes, we want content (e.g. in an exam with master solutions) conditional on whether solutions are shown. This can be done with the `\ifsolutions` conditional.

22.2.3 Multiple Choice Blocks

mcb \mcc

Multiple choice blocks can be formatted using the `mcb` environment, in which single choices are marked up with `\mcc[`⟨*keyvals*⟩`]{`⟨*text*⟩`}` macro, which takes an optional key/value argument ⟨*keyvals*⟩ for choice metadata and a required argument ⟨*text*⟩ for the proposed answer text. The following keys are supported

T F Ttext Ftext feedback

- `T` for true answers, `F` for false ones,
- `Ttext` the verdict for true answers, `Ftext` for false ones, and
- `feedback` for a short feedback text given to the student.

See Figure **??** for an example

22.2.4 Including Problems

\includeproblem title min pts

The `\includeproblem` macro can be used to include a problem from another file. It takes an optional KeyVal argument and a second argument which is a path to the file containing the problem (the macro assumes that there is only one problem in the include file). The keys `title`, `min`, and `pts` specify the problem title, the estimated minutes for solving the problem and the points to be gained, and their values (if given) overwrite the ones specified in the `problem` environment in the included file.

22.2.5 Reporting Metadata

The sum of the points and estimated minutes (that we specified in the `pts` and `min` keys to the `problem` environment or the `\includeproblem` macro) to the log file and the screen after each run. This is useful in preparing exams, where we want to make sure that the students can indeed solve the problems in an allotted time period.

The `\min` and `\pts` macros allow to specify (i.e. to print to the margin) the distribution of time and reward to parts of a problem, if the `pts` and `pts` package options are set. This allows to give students hints about the estimated time and the points to be awarded.

22.3 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 sTeXGitHub repository [sTeX].

1. none reported yet

```
\begin{sproblem}[title=Functions]
  What is the keyword to introduce a function definition in python?
  \begin{mcb}
    \mcc[T]{def}
    \mcc[F,feedback=that is for C and C++]{function}
    \mcc[F,feedback=that is for Standard ML]{fun}
    \mcc[F,Ftext=Nooooooooo,feedback=that is for Java]{public static void}
  \end{mcb}
\end{sproblem}
```

Problem 0.2 (Functions)
What is the keyword to introduce a function definition in python?

1. def
2. function
3. fun
4. public static void

Problem 0.3 (Functions)
What is the keyword to introduce a function definition in python?

1. def
   !
2. function
   that is for C and C++
3. fun
   that is for Standard ML
4. public static void
   that is for Java

Example 7: A Problem with a multiple choice block

Chapter 23

`hwexam.sty/cls`: An Infrastructure for formatting Assignments and Exams

The `hwexam` package and class allows individual course assignment sheets and compound assignment documents using problem files marked up with the `problem` package.

Contents

23.1 Introduction

The `hwexam` package and class supplies an infrastructure that allows to format nice-looking assignment sheets by simply including problems from problem files marked up with the `problem` package [**Kohlhase:problem**]. It is designed to be compatible with `problems.sty`, and inherits some of the functionality.

23.2 The User Interface

23.2.1 Package and Class Options

The `hwexam` package and class take the options `solutions`, `notes`, `hints`, `gnotes`, `pts`, `min`, and `boxed` that are just passed on to the `problems` package (cf. its documentation for a description of the intended behavior).

`showmeta` If the `showmeta` option is set, then the metadata keys are shown (see [**Kohlhase:metakeys**] for details and customization options).

The `hwexam` class additionally accepts the options `report`, `book`, `chapter`, `part`, and `showignores`, of the `omdoc` package [**Kohlhase:smomdl**] on which it is based and passes them on to that. For the `extrefs` option see [**Kohlhase:sref**].

23.2.2 Assignments

`assignment` `number` `title` `type` `given` `due` This package supplies the `assignment` environment that groups problems into assignment sheets. It takes an optional KeyVal argument with the keys `number` (for the assignment number; if none is given, 1 is assumed as the default or — in multi-assignment documents — the ordinal of the `assignment` environment), `title` (for the assignment title; this is referenced in the title of the assignment sheet), `type` (for the assignment type; e.g. “quiz”, or “homework”), `given` (for the date the assignment was given), and `due` (for the date the assignment is due).

23.2.3 Typesetting Exams

`multiple` Furthermore, the `hwexam` package takes the option `multiple` that allows to combine multiple assignment sheets into a compound document (the assignment sheets are treated as section, there is a table of contents, etc.).

`test` Finally, there is the option `test` that modifies the behavior to facilitate formatting tests. Only in `test` mode, the macros `\testspace`, `\testnewpage`, and `\testemptypage` have an effect: they generate space for the students to solve the given problems. Thus they can be left in the LaTeX source.

`\testspace` `\testnewpage` `\testemptypage` `\testspace` takes an argument that expands to a dimension, and leaves vertical space accordingly. `\testnewpage` makes a new page in `test` mode, and `\testemptypage` generates an empty page with the cautionary message that this page was intentionally left empty.

`testheading` `duration` `min` `reqpts` Finally, the `\testheading` takes an optional keyword argument where the keys `duration` specifies a string that specifies the duration of the test, `min` specifies the equivalent in number of minutes, and `reqpts` the points that are required for a perfect grade.

23.2.4 Including Assignments

`\inputassignment` The `\inputassignment` macro can be used to input an assignment from another file. It takes an optional KeyVal argument and a second argument which is a path to the file containing the problem (the macro assumes that there is only one `assignment` environment in the included file). The keys `number`, `title`, `type`, `given`, and `due` are just as for the `assignment` environment and (if given) overwrite the ones specified in the `assignment` environment in the included file.

`number`
`title`
`type`
`given`
`due`

23.3 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 sTeXGitHub repository [sTeX].

1. none reported yet.

```
\title{320101 General Computer Science (Fall 2010)}
\begin{testheading}[duration=one hour,min=60,reqpts=27]
  Good luck to all students!
\end{testheading}
```

formats to

Name: Matriculation Number:

320101 General Computer Science (Fall 2010)

2022-03-14

You have one hour (sharp) for the test;
Write the solutions to the sheet.
The estimated time for solving this exam is 58 minutes, leaving you 2 minutes for revising your exam.
You can reach 30 points if you solve all problems. You will only need 27 points for a perfect score, i.e. 3 points are bonus points.

You have ample time, so take it slow and avoid rushing to mistakes!

Different problems test different skills and knowledge, so do not get stuck on one problem.

	To be used for grading, do not write here											
prob.	0.1	0.2	0.3	1.1	2.1	2.2	2.3	3.1	3.2	3.3	Sum	grade
total				4	4	6	6	4	4	2	30	
reached												

good luck

Example 8: A generated test heading.

Part IV
Implementation

Chapter 24

sTeX -Basics Implementation

24.1 The sTeXDocument Class

The stex document class is pretty straight-forward: It largely extends the standalone package and loads the stex package, passing all provided options on to the package.

```
⟨*cls⟩

%%%%%%%%%%%%%   basics.dtx   %%%%%%%%%%%%%

\RequirePackage{expl3,l3keys2e}
\ProvidesExplClass{stex}{2022/03/03}{3.1.0}{sTeX document class}
\LoadClass[border=1px,varwidth]{standalone}
\setlength\textwidth{15cm}

\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
\ProcessOptions

\RequirePackage{stex}
⟨/cls⟩
```

24.2 Preliminaries

```
⟨*package⟩

%%%%%%%%%%%%%   basics.dtx   %%%%%%%%%%%%%

\RequirePackage{expl3,l3keys2e,ltxcmds}
\ProvidesExplPackage{stex}{2022/03/03}{3.1.0}{sTeX package}

%\RequirePackage{morewrites}
%\RequirePackage{amsmath}

```

Package options:

```
\keys_define:nn { stex } {
  debug     .clist_set:N  = \c_stex_debug_clist ,
  lang      .clist_set:N  = \c_stex_languages_clist ,
  mathhub   .tl_set_x:N   = \mathhub ,
  sms       .bool_set:N   = \c_stex_persist_mode_bool ,
  image     .bool_set:N   = \c_tikzinput_image_bool,
  unknown   .code:n       = {}
}
\ProcessKeysOptions { stex }
```

\stex
\sTeX

The sTeXlogo:

```
\protected\def\stex{
  \texorpdfstring{\raisebox{-.5ex}S\kern-.5ex\TeX}{sTeX}\xspace%
}
\let\sTeX\stex
```

(*End definition for* `\stex` *and* `\sTeX`*. These functions are documented on page 46.*)

24.3 Messages and logging

```
⟨@@=stex_log⟩
```

Warnings and error messages

```
\msg_new:nnn{stex}{error/unknownlanguage}{
  Unknown~language:~#1
}
\msg_new:nnn{stex}{warning/nomathhub}{
  MATHHUB~system~variable~not~found~and~no~
  \detokenize{\mathhub}-value~set!
}
\msg_new:nnn{stex}{error/deactivated-macro}{
  The~\detokenize{#1}~command~is~only~allowed~in~#2!
}
```

\stex_debug:nn

A simple macro issuing package messages with subpath.

```
\cs_new_protected:Nn \stex_debug:nn {
  \clist_if_in:NnTF \c_stex_debug_clist { all } {
    \msg_set:nnn{stex}{debug / #1}{
      \\Debug~#1:~#2\\
    }
    \msg_none:nn{stex}{debug / #1}
  }{
    \clist_if_in:NnT \c_stex_debug_clist { #1 } {
      \msg_set:nnn{stex}{debug / #1}{
        \\Debug~#1:~#2\\
      }
      \msg_none:nn{stex}{debug / #1}
    }
  }
}
```

(*End definition for* `\stex_debug:nn`*. This function is documented on page 46.*)

Redirecting messages:

```
\clist_if_in:NnTF \c_stex_debug_clist {all} {
    \msg_redirect_module:nnn{ stex }{ none }{ term }
}{
  \clist_map_inline:Nn \c_stex_debug_clist {
    \msg_redirect_name:nnn{ stex }{ debug / ##1 }{ term }
  }
}

\stex_debug:nn{log}{debug~mode~on}
```

24.4 HTML Annotations

```
⟨@@=stex_annotate⟩
\RequirePackage{rustex}
```

We add the namespace abbreviation `ns:stex="http://kwarc.info/ns/sTeX"` to RusTEX:

```
\rustex_add_Namespace:nn{stex}{http://kwarc.info/ns/sTeX}
\rustex_add_Namespace:nn{mmt}{http://uniformal.github.io/MMT}
```

Conditionals for LATEXML:

`\if@latexml`

```
\ifcsname if@latexml\endcsname\else
    \expandafter\newif\csname if@latexml\endcsname\@latexmlfalse
\fi
```

(*End definition for* `\if@latexml`. *This function is documented on page 46.*)

`\latexml_if_p:`
`\latexml_if:TF`

```
\prg_new_conditional:Nnn \latexml_if: {p, T, F, TF} {
  \if@latexml
    \prg_return_true:
  \else:
    \prg_return_false:
  \fi:
}
```

(*End definition for* `\latexml_if:TF`. *This function is documented on page 46.*)

`\l__stex_annotate_arg_tl`
`\c__stex_annotate_emptyarg_tl`

Used by annotation macros to ensure that the HTML output to annotate is not empty.

```
\tl_new:N \l__stex_annotate_arg_tl
\tl_const:Nx \c__stex_annotate_emptyarg_tl {
  \rustex_if:TF {
    \rustex_direct_HTML:n { \c_ampersand_str \c_hash_str 8205;  }
  }{~}
}
```

(*End definition for* `\l__stex_annotate_arg_tl` *and* `\c__stex_annotate_emptyarg_tl`.)

`\__stex_annotate_checkempty:n`

```
\cs_new_protected:Nn \__stex_annotate_checkempty:n {
  \tl_set:Nn \l__stex_annotate_arg_tl { #1 }
  \tl_if_empty:NT \l__stex_annotate_arg_tl {
    \tl_set_eq:NN \l__stex_annotate_arg_tl \c__stex_annotate_emptyarg_tl
  }
}
```

(*End definition for* `\__stex_annotate_checkempty:n`.)

`\stex_if_do_html_p:`
`\stex_if_do_html:TF`

Whether to (locally) produce HTML output

```
\bool_new:N \_stex_html_do_output_bool
\bool_set_true:N \_stex_html_do_output_bool

\prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
  \bool_if:nTF \_stex_html_do_output_bool
    \prg_return_true: \prg_return_false:
}
```

(*End definition for* `\stex_if_do_html:TF`*. This function is documented on page 46.*)

`\stex_suppress_html:n`

Whether to (locally) produce HTML output

```
\cs_new_protected:Nn \stex_suppress_html:n {
  \exp_args:Nne \use:nn {
    \bool_set_false:N \_stex_html_do_output_bool
    #1
  }{
    \stex_if_do_html:T {
      \bool_set_true:N \_stex_html_do_output_bool
    }
  }
}
```

(*End definition for* `\stex_suppress_html:n`*. This function is documented on page 46.*)

`\stex_annotate:nnn`
`\stex_annotate_env`
`\stex_annotate_invisible:n`
`\stex_annotate_invisible:nnn`

We define four macros for introducing attributes in the HTML output. The definitions depend on the "backend" used (LaTeXML, RusTeX, `pdflatex`).

The `pdflatex`-macros largely do nothing; the RusTeX-implementations are pretty clear in what they do, the LaTeXML-implementations resort to perl bindings.

```
\rustex_if:TF{
  \cs_new_protected:Nn \stex_annotate:nnn {
    \__stex_annotate_checkempty:n { #3 }
    \rustex_annotate_HTML:nn {
      property="stex:#1" ~
      resource="#2"
    } {
      \mode_if_vertical:TF{
        \tl_use:N \l__stex_annotate_arg_tl\par
      }{
        \tl_use:N \l__stex_annotate_arg_tl
      }
    }
  }
  \cs_new_protected:Nn \stex_annotate_invisible:n {
    \__stex_annotate_checkempty:n { #1 }
    \rustex_annotate_HTML:nn {
      stex:visible="false" ~
      style:display="none"
    } {
      \mode_if_vertical:TF{
        \tl_use:N \l__stex_annotate_arg_tl\par
      }{
```

```
      \tl_use:N \l__stex_annotate_arg_tl
    }
  }
}
\cs_new_protected:Nn \stex_annotate_invisible:nnn {
  \__stex_annotate_checkempty:n { #3 }
  \rustex_annotate_HTML:nn {
    property="stex:#1" ~
    resource="#2" ~
    stex:visible="false" ~
    style:display="none"
  } {
    \mode_if_vertical:TF{
      \tl_use:N \l__stex_annotate_arg_tl\par
    }{
      \tl_use:N \l__stex_annotate_arg_tl
    }
  }
}
\NewDocumentEnvironment{stex_annotate_env} { m m } {
  \par
  \rustex_annotate_HTML_begin:n {
    property="stex:#1" ~
    resource="#2"
  }
}{
  \par\rustex_annotate_HTML_end:
}
}{
\latexml_if:TF {
  \cs_new_protected:Nn \stex_annotate:nnn {
    \__stex_annotate_checkempty:n { #3 }
    \mode_if_math:TF {
      \cs:w latexml@annotate@math\cs_end:{#1}{#2}{
        \tl_use:N \l__stex_annotate_arg_tl
      }
    }{
      \cs:w latexml@annotate@text\cs_end:{#1}{#2}{
        \tl_use:N \l__stex_annotate_arg_tl
      }
    }
  }
  \cs_new_protected:Nn \stex_annotate_invisible:n {
    \__stex_annotate_checkempty:n { #1 }
    \mode_if_math:TF {
      \cs:w latexml@invisible@math\cs_end:{
        \tl_use:N \l__stex_annotate_arg_tl
      }
    } {
      \cs:w latexml@invisible@text\cs_end:{
        \tl_use:N \l__stex_annotate_arg_tl
      }
    }
  }
```

```
    \cs_new_protected:Nn \stex_annotate_invisible:nnn {
      \__stex_annotate_checkempty:n { #3 }
      \cs:w latexml@annotate@invisible\cs_end:{#1}{#2}{
        \tl_use:N \l__stex_annotate_arg_tl
      }
    }
    \NewDocumentEnvironment{stex_annotate_env} { m m } {
      \par\begin{latexml@annotateenv}{#1}{#2}
    }{
      \par\end{latexml@annotateenv}
    }
  }{
    \cs_new_protected:Nn \stex_annotate:nnn {#3}
    \cs_new_protected:Nn \stex_annotate_invisible:n {}
    \cs_new_protected:Nn \stex_annotate_invisible:nnn {}
    \NewDocumentEnvironment{stex_annotate_env} { m m } {}{}
  }
}
```

(*End definition for* `\stex_annotate:nnn`, `\stex_annotate_invisible:n`, *and* `\stex_annotate_invisible:nnn`. *These functions are documented on page 47.*)

24.5 Babel Languages

```
⟨@@=stex_language⟩
```

\c_stex_languages_prop
\c_stex_language_abbrevs_prop

We store language abbreviations in two (mutually inverse) property lists:

```
\prop_const_from_keyval:Nn \c_stex_languages_prop {
  en = english ,
  de = ngerman ,
  ar = arabic ,
  bg = bulgarian ,
  ru = russian ,
  fi = finnish ,
  ro = romanian ,
  tr = turkish ,
  fr = french
}

\prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop {
  english   = en ,
  ngerman   = de ,
  arabic    = ar ,
  bulgarian = bg ,
  russian   = ru ,
  finnish   = fi ,
  romanian  = ro ,
  turkish   = tr ,
  french    = fr
}
% todo: chinese simplified (zhs)
%       chinese traditional (zht)
```

(*End definition for* `\c_stex_languages_prop` *and* `\c_stex_language_abbrevs_prop`. *These variables are documented on page 47.*)

we use the `lang`-package option to load the corresponding babel languages:

```
\clist_if_empty:NF \c_stex_languages_clist {
  \clist_clear:N \l_tmpa_clist
  \clist_map_inline:Nn \c_stex_languages_clist {
    \prop_get:NnNTF \c_stex_languages_prop { #1 } \l_tmpa_str {
      \clist_put_right:No \l_tmpa_clist \l_tmpa_str
    } {
      \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
    }
  }
  \stex_debug:nn{lang} {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
  \RequirePackage[\clist_use:Nn \l_tmpa_clist,]{babel}
}
\AtBeginDocument{
  \bool_lazy_any:nT {
    {\rustex_if_p:}
    {\latexml_if_p:}
  } {
    \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
    \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
    \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
    \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
    \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
      \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
      \stex_debug:nn{basics} {Language~\l_tmpa_str~
        inferred~from~file~name}
      \stex_annotate_invisible:nnn{language}{ \l_tmpa_str }{}

    }
  }
}
```

24.6 Auxiliary Methods

`\stex_deactivate_macro:Nn`

```
\cs_new_protected:Nn \stex_deactivate_macro:Nn {
  \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
  \def#1{
    \msg_error:nnnn{stex}{error/deactivated-macro}{\detokenize{#1}}{#2}
  }
}
```

(*End definition for* `\stex_deactivate_macro:Nn`. *This function is documented on page 47.*)

`\stex_reactivate_macro:N`

```
\cs_new_protected:Nn \stex_reactivate_macro:N {
  \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname
}
```

(*End definition for* `\stex_reactivate_macro:N`. *This function is documented on page 47.*)

\ignorespacesandpars

```
\protected\def\ignorespacesandpars{
  \begingroup\catcode13=10\relax
  \@ifnextchar\par{
    \endgroup\expandafter\ignorespacesandpars\@gobble
  }{
    \endgroup
  }
}
```

(*End definition for* `\ignorespacesandpars`*. This function is documented on page 47.*)

\MMTrule

```
\NewDocumentCommand \MMTrule {m m}{
  \seq_set_split:Nnn \l_tmpa_seq , {#2}
  \int_zero:N \l_tmpa_int
  \stex_annotate_invisible:nnn{mmtrule}{scala://#1}{
    $\seq_map_inline:Nn \l_tmpa_seq {
      \int_incr:N \l_tmpa_int
      \stex_annotate:nnn{arg}{i\int_use:N \l_tmpa_int}{##1}
    }$
  }
}

\NewDocumentCommand \MMTinclude {m}{
  \stex_annotate_invisible:nnn{import}{#1}{}
}
⟨/package⟩
```

(*End definition for* `\MMTrule`*. This function is documented on page ??.*)

Chapter 25

sTeX -MathHub Implementation

```
⟨*package⟩

%%%%%%%%%%%%%   mathhub.dtx   %%%%%%%%%%%%%

⟨@@=stex_path⟩
```

Warnings and error messages

```
\msg_new:nnn{stex}{error/norepository}{
  No~archive~#1~found~in~#2
}
\msg_new:nnn{stex}{error/notinarchive}{
  Not~currently~in~an~archive,~but~\detokenize{#1}~
  needs~one!
}
\msg_new:nnn{stex}{error/nofile}{
  \detokenize{#1}~could~not~find~file~#2
}
\msg_new:nnn{stex}{error/twofiles}{
  \detokenize{#1}~found~two~candidates~for~#2
}
```

25.1 Generic Path Handling

We treat paths as LaTeX3-sequences (of the individual path segments, i.e. separated by a /-character) unix-style; i.e. a path is absolute if the sequence starts with an empty entry.

\stex_path_from_string:Nn

```
\cs_new_protected:Nn \stex_path_from_string:Nn {
  \str_set:Nx \l_tmpa_str { #2 }
  \str_if_empty:NTF \l_tmpa_str {
    \seq_clear:N #1
  }{
    \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
    \sys_if_platform_windows:T{
      \seq_clear:N \l_tmpa_tl
      \seq_map_inline:Nn #1 {
        \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
        \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
      }
      \seq_set_eq:NN #1 \l_tmpa_tl
    }
    \stex_path_canonicalize:N #1
  }
}

```

(*End definition for* `\stex_path_from_string:Nn`*. This function is documented on page 48.*)

`\stex_path_to_string:NN`
`\stex_path_to_string:N`

```
\cs_new_protected:Nn \stex_path_to_string:NN {
  \exp_args:NNe \str_set:Nn #2 { \seq_use:Nn #1 / }
}

\cs_new:Nn \stex_path_to_string:N {
  \seq_use:Nn #1 /
}
```

(*End definition for* `\stex_path_to_string:NN` *and* `\stex_path_to_string:N`*. These functions are documented on page 48.*)

`\c__stex_path_dot_str`
`\c__stex_path_up_str`

`.` and `..`, respectively.

```
\str_const:Nn \c__stex_path_dot_str {.}
\str_const:Nn \c__stex_path_up_str {..}
```

(*End definition for* `\c__stex_path_dot_str` *and* `\c__stex_path_up_str`*.*)

`\stex_path_canonicalize:N`

Canonicalizes the path provided; in particular, resolves `.` and `..` path segments.

```
\cs_new_protected:Nn \stex_path_canonicalize:N {
  \seq_if_empty:NF #1 {
    \seq_clear:N \l_tmpa_seq
    \seq_get_left:NN #1 \l_tmpa_tl
    \str_if_empty:NT \l_tmpa_tl {
      \seq_put_right:Nn \l_tmpa_seq {}
    }
    \seq_map_inline:Nn #1 {
      \str_set:Nn \l_tmpa_tl { ##1 }
      \str_if_eq:NNF \l_tmpa_tl \c__stex_path_dot_str {
        \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
          \seq_if_empty:NTF \l_tmpa_seq {
            \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
              \c__stex_path_up_str
            }
          }{
            \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
            \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
              \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
                \c__stex_path_up_str
              }
            }{
              \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
            }
          }
        }{
          \str_if_empty:NF \l_tmpa_tl {
            \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
          }
        }
      }
    }
    \seq_gset_eq:NN #1 \l_tmpa_seq
  }
}
```

(*End definition for* `\stex_path_canonicalize:N`. *This function is documented on page 48.*)

`\stex_path_if_absolute_p:N`
`\stex_path_if_absolute:NTF`

```
\prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
  \seq_if_empty:NTF #1 {
    \prg_return_false:
  }{
    \seq_get_left:NN #1 \l_tmpa_tl
    \sys_if_platform_windows:TF{
      \str_if_in:NnTF \l_tmpa_tl {:}{
        \prg_return_true:
      }{
        \prg_return_false:
      }
    }{
      \str_if_empty:NTF \l_tmpa_tl {
        \prg_return_true:
      }{
        \prg_return_false:
      }
    }
  }
}
```

(*End definition for* `\stex_path_if_absolute:NTF`. *This function is documented on page 48.*)

25.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```
\str_new:N\l_stex_kpsewhich_return_str
\cs_new_protected:Nn \stex_kpsewhich:n {
  \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
  \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
  \tl_trim_spaces:N \l_stex_kpsewhich_return_str
}
```

(*End definition for* `\stex_kpsewhich:n`. *This function is documented on page 48.*)

We determine the PWD

\c_stex_pwd_seq
\c_stex_pwd_str

```
\sys_if_platform_windows:TF{
  \begingroup\escapechar=-1\catcode`\\=12
  \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
  \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
  \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_
}{
  \stex_kpsewhich:n{-var-value~PWD}
}

\stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
\stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
\stex_debug:nn {mathhub} {PWD:~\str_use:N\c_stex_pwd_str}
```

(*End definition for* `\c_stex_pwd_seq` *and* `\c_stex_pwd_str`*. These variables are documented on page 48.*)

25.3 File Hooks and Tracking

```
⟨@@=stex_files⟩
```

We introduce hooks for file inputs that keep track of the absolute paths of files used. This will be useful to keep track of modules, their archives, namespaces etc.

Note that the absolute paths are only accurate in `\input`-statements for paths relative to the PWD, so they shouldn't be relied upon in any other setting than for sTeX-purposes.

\g__stex_files_stack keeps track of file changes

```
\seq_gclear_new:N\g__stex_files_stack
```

(*End definition for* `\g__stex_files_stack`*.*)

\c_stex_mainfile_seq
\c_stex_mainfile_str

```
\str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
\stex_path_from_string:Nn \c_stex_mainfile_seq
  \c_stex_mainfile_str
```

(*End definition for* `\c_stex_mainfile_seq` *and* `\c_stex_mainfile_str`*. These variables are documented on page 48.*)

\g_stex_currentfile_seq

```
\seq_gclear_new:N\g_stex_currentfile_seq
```

(*End definition for* `\g_stex_currentfile_seq`*. This variable is documented on page 49.*)

\stex_filestack_push:n

```
\cs_new_protected:Nn \stex_filestack_push:n {
  \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
  \stex_path_if_absolute:NF\g_stex_currentfile_seq{
    \stex_path_from_string:Nn\g_stex_currentfile_seq{
      \c_stex_pwd_str/#1
    }
  }
  \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
  \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
}
```

(*End definition for* `\stex_filestack_push:n`*. This function is documented on page 49.*)

`\stex_filestack_pop:`

```
\cs_new_protected:Nn \stex_filestack_pop: {
  \seq_if_empty:NF\g__stex_files_stack{
    \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
  }
  \seq_if_empty:NTF\g__stex_files_stack{
    \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
  }{
    \seq_get:NN\g__stex_files_stack\l_tmpa_seq
    \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
  }
}
```

(*End definition for* `\stex_filestack_pop:`*. This function is documented on page 49.*)

Hooks for the current file:

```
\AddToHook{file/before}{
  \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
}
\AddToHook{file/after}{
  \stex_filestack_pop:
}
```

25.4 MathHub Repositories

```
⟨@@=stex_mathhub⟩
```

`\mathhub`
`\c_stex_mathhub_seq`
`\c_stex_mathhub_str`

The path to the mathhub directory. If the `\mathhub`-macro is not set, we query `kpsewhich` for the `MATHHUB` system variable.

```
\str_if_empty:NTF\mathhub{
  \sys_if_platform_windows:TF{
    \begingroup\escapechar=-1\catcode`\\=12
    \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str MATHHUB\c_percent_str}
    \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
    \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_ste
  }{
    \stex_kpsewhich:n{-var-value~MATHHUB}
  }
  \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str

  \str_if_empty:NTF\c_stex_mathhub_str{
    \msg_warning:nn{stex}{warning/nomathhub}
  }{
    \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
    \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
  }
}{
  \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
  \stex_path_if_absolute:NF \c_stex_mathhub_seq {
    \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
      \c_stex_pwd_str/\mathhub
    }
  }
  \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
  \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
}
```

(*End definition for* `\mathhub`*,* `\c_stex_mathhub_seq`*, and* `\c_stex_mathhub_str`*. These variables are documented on page 49.*)

`\__stex_mathhub_do_manifest:n`

Checks whether the manifest for archive `#1` already exists, and if not, finds and parses the corresponding manifest file

```
\cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
  \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
    \str_set:Nx \l_tmpa_str { #1 }
    \prop_new:c { c_stex_mathhub_#1_manifest_prop }
    \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
    \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
    \__stex_mathhub_find_manifest:N \l_tmpa_seq
    \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
      \msg_error:nnxx{stex}{error/norepository}{#1}{
        \stex_path_to_string:N \c_stex_mathhub_str
      }
    } {
      \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
    }
  }
}
```

(*End definition for* `\__stex_mathhub_do_manifest:n`*.*)

`\l__stex_mathhub_manifest_file_seq`

```
\seq_new:N\l__stex_mathhub_manifest_file_seq
```

(*End definition for* `\l__stex_mathhub_manifest_file_seq`*.*)

`\__stex_mathhub_find_manifest:N`

Attempts to find the `MANIFEST.MF` in some file path and stores its path in `\l__stex_mathhub_manifest_file_seq`:

```
\cs_new_protected:Nn \__stex_mathhub_find_manifest:N {
  \seq_set_eq:NN\l_tmpa_seq #1
  \bool_set_true:N\l_tmpa_bool
  \bool_while_do:Nn \l_tmpa_bool {
    \seq_if_empty:NTF \l_tmpa_seq {
      \bool_set_false:N\l_tmpa_bool
    }{
      \file_if_exist:nTF{
        \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
      }{
        \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
        \bool_set_false:N\l_tmpa_bool
      }{
        \file_if_exist:nTF{
          \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
        }{
          \seq_put_right:Nn\l_tmpa_seq{META-INF}
          \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
          \bool_set_false:N\l_tmpa_bool
        }{
          \file_if_exist:nTF{
            \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
          }{
            \seq_put_right:Nn\l_tmpa_seq{meta-inf}
            \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
            \bool_set_false:N\l_tmpa_bool
          }{
            \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
          }
        }
      }
    }
  }
  \seq_set_eq:NN\l__stex_mathhub_manifest_file_seq\l_tmpa_seq
}
```

(*End definition for* `\__stex_mathhub_find_manifest:N`.)

`\c__stex_mathhub_manifest_ior` File variable used for `MANIFEST`-files

```
\ior_new:N \c__stex_mathhub_manifest_ior
```

(*End definition for* `\c__stex_mathhub_manifest_ior`.)

`\__stex_mathhub_parse_manifest:n` Stores the entries in manifest file in the corresponding property list:

```
\cs_new_protected:Nn \__stex_mathhub_parse_manifest:n {
  \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
  \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
  \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
    \str_set:Nn \l_tmpa_str {##1}
    \exp_args:NNoo \seq_set_split:Nnn
        \l_tmpb_seq \c_colon_str \l_tmpa_str
    \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
      \exp_args:NNe \str_set:Nn \l_tmpb_tl {
        \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
      }
      \exp_args:No \str_case:nnTF \l_tmpa_tl {
        {id} {
          \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
            { id } \l_tmpb_tl
        }
        {narration-base} {
          \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
            { narr } \l_tmpb_tl
        }
        {url-base} {
          \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
            { docurl } \l_tmpb_tl
        }
        {source-base} {
          \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
            { ns } \l_tmpb_tl
        }
        {ns} {
          \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
            { ns } \l_tmpb_tl
        }
        {dependencies} {
          \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
            { deps } \l_tmpb_tl
        }
      }{}{}
    }{}
  }
  \ior_close:N \c__stex_mathhub_manifest_ior
}
```

(*End definition for* `\__stex_mathhub_parse_manifest:n`.)

\stex_set_current_repository:n

```
\cs_new_protected:Nn \stex_set_current_repository:n {
  \stex_require_repository:n { #1 }
  \prop_set_eq:Nc \l_stex_current_repository_prop {
    c_stex_mathhub_#1_manifest_prop
  }
}
```

(*End definition for* `\stex_set_current_repository:n`. *This function is documented on page 49.*)

\stex_require_repository:n

```
\cs_new_protected:Nn \stex_require_repository:n {
  \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
    \stex_debug:nn{mathhub}{Opening~archive:~#1}
    \__stex_mathhub_do_manifest:n { #1 }
  }
}
```

(*End definition for* `\stex_require_repository:n`. *This function is documented on page 49.*)

\l_stex_current_repository_prop

Current MathHub repository

```
%\prop_new:N \l_stex_current_repository_prop

\__stex_mathhub_find_manifest:N \c_stex_pwd_seq
\seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
  \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
} {
  \__stex_mathhub_parse_manifest:n { main }
  \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
    \l_tmpa_str
  \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str _manifest_prop }
    \c_stex_mathhub_main_manifest_prop
  \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
  \stex_debug:nn{mathhub}{Current~repository:~
    \prop_item:Nn \l_stex_current_repository_prop {id}
  }
}
```

(*End definition for* `\l_stex_current_repository_prop`. *This variable is documented on page 49.*)

\stex_in_repository:nn Executes the code in the second argument in the context of the repository whose ID is provided as the first argument.

```
\cs_new_protected:Nn \stex_in_repository:nn {
  \str_set:Nx \l_tmpa_str { #1 }
  \cs_set:Npn \l_tmpa_cs ##1 { #2 }
  \str_if_empty:NTF \l_tmpa_str {
    \prop_if_exist:NTF \l_stex_current_repository_prop {
      \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
      \exp_args:Ne \l_tmpa_cs{
        \prop_item:Nn \l_stex_current_repository_prop { id }
      }
    }{
      \l_tmpa_cs{}
    }
  }{
    \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}
    \stex_require_repository:n \l_tmpa_str
    \str_set:Nx \l_tmpa_str { #1 }
    \exp_args:Nne \use:nn {
      \stex_set_current_repository:n \l_tmpa_str
      \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
    }{
      \stex_debug:nn{mathhub}{switching~back~to:~
        \prop_if_exist:NTF \l_stex_current_repository_prop {
          \prop_item:Nn \l_stex_current_repository_prop { id }:~
          \meaning\l_stex_current_repository_prop
        }{
          no~repository
        }
      }
      \prop_if_exist:NTF \l_stex_current_repository_prop {
       \stex_set_current_repository:n {
        \prop_item:Nn \l_stex_current_repository_prop { id }
       }
      }{
        \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
      }
    }
  }
}
```

(*End definition for* `\stex_in_repository:nn`*. This function is documented on page 49.*)

25.5 Using Content in Archives

\mhpath

```
\def \mhpath #1 #2 {
  \exp_args:Ne \tl_if_empty:nTF{#1}{
    \c_stex_mathhub_str /
      \prop_item:Nn \l_stex_current_repository_prop { id }
      / source / #2
  }{
    \c_stex_mathhub_str / #1 / source / #2
  }
}
```

(*End definition for* `\mhpath`. *This function is documented on page 50.*)

\inputref
\mhinput

```
\newif \ifinputref \inputreffalse

\cs_new_protected:Nn \__stex_mathhub_mhinput:nn {
  \stex_in_repository:nn {#1} {
    \ifinputref
      \input{ \c_stex_mathhub_str / ##1 / source / #2 }
    \else
      \inputreftrue
      \input{ \c_stex_mathhub_str / ##1 / source / #2 }
      \inputreffalse
    \fi
  }
}
\NewDocumentCommand \mhinput { O{} m}{
  \stex_mhinput:nn{ #1 }{ #2 }
}

\cs_new_protected:Nn \__stex_mathhub_inputref:nn {
  \stex_in_repository:nn {#1} {
    \bool_lazy_any:nTF {
      {\rustex_if_p:}
      {\latexml_if_p:}
    } {
      \str_clear:N \l_tmpa_str
      \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
        \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
      }
      \stex_annotate_invisible:nnn{inputref}{
        \l_tmpa_str / #2
      }{}
    }{
      \begingroup
        \inputreftrue
        \tl_if_empty:nTF{ ##1 }{
          \input{#2}
        }{
          \input{ \c_stex_mathhub_str / ##1 / source / #2 }
        }
      \endgroup
    }
  }
}
\NewDocumentCommand \inputref { O{} m}{
  \__stex_mathhub_inputref:nn{ #1 }{ #2 }
}
```

(*End definition for* `\inputref` *and* `\mhinput`. *These functions are documented on page 50.*)

\addmhbibresource

```
\cs_new_protected:Nn \__stex_mathhub_mhbibresource:nn {
  \stex_in_repository:nn {#1} {
    \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
  }
}
\newcommand\addmhbibresource[2][]{
  \__stex_mathhub_mhbibresource:nn{ #1 }{ #2 }
}
```

(*End definition for* `\addmhbibresource`. *This function is documented on page 50.*)

\libinput

```
\cs_new_protected:Npn \libinput #1 {
  \prop_if_exist:NF \l_stex_current_repository_prop {
    \msg_error:nnn{stex}{error/notinarchive}\libinput
  }
  \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
    \msg_error:nnn{stex}{error/notinarchive}\libinput
  }
  \seq_clear:N \l__stex_mathhub_libinput_files_seq
  \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
  \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str

  \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
    \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #1.tex}
    \IfFileExists{ \l_tmpa_str }{
      \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
    }{}
    \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
    \seq_put_right:No \l_tmpa_seq \l_tmpa_str
  }

  \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}
  \IfFileExists{ \l_tmpa_str }{
    \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
  }{}

  \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
    \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
  }{
    \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
      \input{ ##1 }
    }
  }
}
```

(*End definition for* `\libinput`. *This function is documented on page 50.*)

\libusepackage

```
\NewDocumentCommand \libusepackage {O{} m} {
  \prop_if_exist:NF \l_stex_current_repository_prop {
    \msg_error:nnn{stex}{error/notinarchive}\libusepackage
  }
  \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
    \msg_error:nnn{stex}{error/notinarchive}\libusepackage
  }
  \seq_clear:N \l__stex_mathhub_libinput_files_seq
  \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
  \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str

  \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
    \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #2}
    \IfFileExists{ \l_tmpa_str.sty }{
      \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
    }{}
    \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
    \seq_put_right:No \l_tmpa_seq \l_tmpa_str
  }

  \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2}
  \IfFileExists{ \l_tmpa_str.sty }{
    \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
  }{}

  \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
    \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
  }{
    \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
      \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
        \usepackage[#1]{ ##1 }
      }
    }{
      \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
    }
  }
}
```

(*End definition for* `\libusepackage`*. This function is documented on page 50.*)

\mhgraphics
\cmhgraphics

```

\AddToHook{begindocument}{
\ltx@ifpackageloaded{graphicx}{
    \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
    \newcommand\mhgraphics[2][]{%
      \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
      \includegraphics[#1]{\mhpath\Gin@mhrepos{#2}}}
    \newcommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
  }{}
```

(*End definition for* `\mhgraphics` *and* `\cmhgraphics`*. These functions are documented on page 50.*)

\lstinputmhlisting
\clstinputmhlisting

```
\ltx@ifpackageloaded{listings}{
    \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
    \newcommand\lstinputmhlisting[2][]{%
      \def\lst@mhrepos{}\setkeys{lst}{#1}%
      \lstinputlisting[#1]{\mhpath\lst@mhrepos{#2}}}
    \newcommand\clstinputmhlisting[2][]{\begin{center}\lstinputmhlisting[#1]{#2}\end{center}
  }{}
}

⟨/package⟩
```

(*End definition for* `\lstinputmhlisting` *and* `\clstinputmhlisting`*. These functions are documented on page 50.*)

Chapter 26

sTeX -References Implementation

```
⟨*package⟩

%%%%%%%%%%%%%   references.dtx   %%%%%%%%%%%%%

⟨@@=stex_refs⟩
```

Warnings and error messages

References are stored in the file `\jobname.sref`, to enable cross-referencing external documents.

```
%\iow_new:N \c__stex_refs_refs_iow
\AddToHook{begindocument}{
%  \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
}
\AddToHook{enddocument}{
%  \iow_close:N \c__stex_refs_refs_iow
}
```

`\STEXreftitle`

```
\str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}

\NewDocumentCommand \STEXreftitle { m } {
  \tl_gset:Nx \g__stex_refs_title_tl { #1 }
}
```

(*End definition for* `\STEXreftitle`. *This function is documented on page 51.*)

26.1 Document URIs and URLs

`\l_stex_current_docns_str`

```
\str_new:N \l_stex_current_docns_str
```

(*End definition for* `\l_stex_current_docns_str`. *This variable is documented on page 51.*)

\stex_get_document_uri:

```
\cs_new_protected:Nn \stex_get_document_uri: {
  \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
  \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
  \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
  \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
  \seq_put_right:No \l_tmpa_seq \l_tmpb_str

  \str_clear:N \l_tmpa_str
  \prop_if_exist:NT \l_stex_current_repository_prop {
    \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
      \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
    }
  }

  \str_if_empty:NTF \l_tmpa_str {
    \str_set:Nx \l_stex_current_docns_str {
      file:/\stex_path_to_string:N \l_tmpa_seq
    }
  }{
    \bool_set_true:N \l_tmpa_bool
    \bool_while_do:Nn \l_tmpa_bool {
      \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
      \exp_args:No \str_case:nnTF { \l_tmpb_str } {
        {source} { \bool_set_false:N \l_tmpa_bool }
      }{}{
        \seq_if_empty:NT \l_tmpa_seq {
          \bool_set_false:N \l_tmpa_bool
        }
      }
    }

    \seq_if_empty:NTF \l_tmpa_seq {
      \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
    }{
      \str_set:Nx \l_stex_current_docns_str {
        \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
      }
    }
  }
}
```

(*End definition for* `\stex_get_document_uri:`. *This function is documented on page 51.*)

\l_stex_current_docurl_str

```
\str_new:N \l_stex_current_docurl_str
```

(*End definition for* `\l_stex_current_docurl_str`. *This variable is documented on page 51.*)

\stex_get_document_url:

```
\cs_new_protected:Nn \stex_get_document_url: {
  \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
  \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
  \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
  \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
  \seq_put_right:No \l_tmpa_seq \l_tmpb_str

  \str_clear:N \l_tmpa_str
  \prop_if_exist:NT \l_stex_current_repository_prop {
    \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
      \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
        \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
      }
    }
  }

  \str_if_empty:NTF \l_tmpa_str {
    \str_set:Nx \l_stex_current_docurl_str {
      file:/\stex_path_to_string:N \l_tmpa_seq
    }
  }{
    \bool_set_true:N \l_tmpa_bool
    \bool_while_do:Nn \l_tmpa_bool {
      \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
      \exp_args:No \str_case:nnTF { \l_tmpb_str } {
        {source} { \bool_set_false:N \l_tmpa_bool }
      }{}{
        \seq_if_empty:NT \l_tmpa_seq {
          \bool_set_false:N \l_tmpa_bool
        }
      }
    }

    \seq_if_empty:NTF \l_tmpa_seq {
      \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
    }{
      \str_set:Nx \l_stex_current_docurl_str {
        \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
      }
    }
  }
}
```

(*End definition for* `\stex_get_document_url:`. *This function is documented on page 51.*)

26.2 Setting Reference Targets

```
\str_const:Nn \c__stex_refs_url_str{URL}
\str_const:Nn \c__stex_refs_ref_str{REF}
\str_new:N \l__stex_refs_curr_label_str
% @currentlabel -> number
% @currentlabelname -> title
% @currentHref -> name.number <- id of some kind
% \theH# -> \arabic{section}
% \the#  -> number
% \hyper@makecurrent{#}
\int_new:N \l__stex_refs_unnamed_counter_int
```

\stex_ref_new_doc_target:n

```
\cs_new_protected:Nn \stex_ref_new_doc_target:n {
  \stex_get_document_uri:
  \str_clear:N \l__stex_refs_curr_label_str
  \str_set:Nx \l_tmpa_str { #1 }
  \str_if_empty:NT \l_tmpa_str {
    \int_incr:N \l__stex_refs_unnamed_counter_int
    \str_set:Nx \l_tmpa_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
  }
  \str_set:Nx \l__stex_refs_curr_label_str {
    \l_stex_current_docns_str?\l_tmpa_str
  }
  \seq_if_exist:cF{g__stex_refs_labels_\l_tmpa_str _seq}{
    \seq_new:c {g__stex_refs_labels_\l_tmpa_str _seq}
  }
  \seq_if_in:coF{g__stex_refs_labels_\l_tmpa_str _seq}\l__stex_refs_curr_label_str {
    \seq_gput_right:co{g__stex_refs_labels_\l_tmpa_str _seq}\l__stex_refs_curr_label_str
  }
  \stex_if_smsmode:TF {
    \stex_get_document_url:
    \str_gset_eq:cN {sref_url_\l__stex_refs_curr_label_str _str}\l_stex_current_docurl_str
    \str_gset_eq:cN {sref_\l__stex_refs_curr_label_str _type}\c__stex_refs_url_str
  }{
    %\iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=~\expandafter\unexpanded\expandafter{
    \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
    \immediate\write\@auxout{\stexauxadddocref{\l_stex_current_docns_str}{\l_tmpa_str}}
    \str_gset:cx {sref_\l__stex_refs_curr_label_str _type}\c__stex_refs_ref_str
  }
}
```

(*End definition for* `\stex_ref_new_doc_target:n`. *This function is documented on page 51.*)

The following is used to set the necessary macros in the `.aux`-file.

```
\cs_new_protected:Npn \stexauxadddocref #1 #2 {
  \str_set:Nn \l_tmpa_str {#1?#2}
  \str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
  \seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
    \seq_new:c {g__stex_refs_labels_#2_seq}
  }
  \seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
    \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
  }
}
```

To avoid resetting the same macros when the `.aux`-file is read at the end of the document:

```
\AtEndDocument{
  \def\stexauxadddocref#1 #2 {}{}
}
```

\stex_ref_new_sym_target:n

```
\cs_new_protected:Nn \stex_ref_new_sym_target:n {
  \stex_if_smsmode:TF {
    \str_if_exist:cF{sref_sym_#1_type}{
      \stex_get_document_url:
      \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str
      \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
    }
  }{
    \str_if_empty:NF \l__stex_refs_curr_label_str {
      \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
      \immediate\write\@auxout{
        \exp_not:N\expandafter\def\exp_not:N\csname \exp_not:N\detokenize{sref_sym_#1_label_
            \l__stex_refs_curr_label_str
          }
      }
    }
  }
}
```

(*End definition for* `\stex_ref_new_sym_target:n`*. This function is documented on page 51.*)

26.3 Using References

```
\str_new:N \l__stex_refs_indocument_str
```

`\sref` Optional arguments:

```

\keys_define:nn { stex / sref } {
  linktext      .tl_set:N  = \l__stex_refs_linktext_tl ,
  fallback      .tl_set:N  = \l__stex_refs_fallback_tl ,
  pre           .tl_set:N  = \l__stex_refs_pre_tl ,
  post          .tl_set:N  = \l__stex_refs_post_tl ,
}
\cs_new_protected:Nn \__stex_refs_args:n {
  \tl_clear:N \l__stex_refs_linktext_tl
  \tl_clear:N \l__stex_refs_fallback_tl
  \tl_clear:N \l__stex_refs_pre_tl
  \tl_clear:N \l__stex_refs_post_tl
  \str_clear:N \l__stex_refs_repo_str
  \keys_set:nn { stex / sref } { #1 }
}
```

The actual macro:

```
\NewDocumentCommand \sref { O{} m}{
  \__stex_refs_args:n { #1 }
  \str_if_empty:NTF \l__stex_refs_indocument_str {
    \str_set:Nx \l_tmpa_str { #2 }
    \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
    \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
      \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
        \seq_get_left:cNF {g__stex_refs_labels_\l_tmpa_str _seq} \l_tmpa_str {
          \str_clear:N \l_tmpa_str
        }
      }{
        \str_clear:N \l_tmpa_str
      }
    }{
      \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
      \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
      \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
      \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
        \str_set_eq:NN \l_tmpc_str \l_tmpa_str
        \str_clear:N \l_tmpa_str
        \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str _seq} {
          \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
            \str_range:nnn { ##1 }{ -\l_tmpa_int}{ -1 }
          }{
            \seq_map_break:n {
              \str_set:Nn \l_tmpa_str { ##1 }
            }
          }
        }
      }{
        \str_clear:N \l_tmpa_str
      }
    }
    \str_if_empty:NTF \l_tmpa_str {
      \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_lin
    }{
      \str_if_eq:cNTF {sref_\l_tmpa_str _type} \c__stex_refs_ref_str {
        \tl_if_empty:NTF \l__stex_refs_linktext_tl {
          \cs_if_exist:cTF{autoref}{
            \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
          }{
            \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
          }
        }{
          \ltx@ifpackageloaded{hyperref}{
            \hyperref[sref_\l_tmpa_str]\l__stex_refs_linktext_tl
          }{
            \l__stex_refs_linktext_tl
          }
        }
      }{
        \ltx@ifpackageloaded{hyperref}{
          \href{\use:c{sref_url_\l_tmpa_str _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_t
        }{
          \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs
        }
      }
    }
  }{
    % TODO
  }
}
```

(*End definition for* `\sref`. *This function is documented on page 52.*)

`\srefsym`

```
\NewDocumentCommand \srefsym { O{} m}{
  \stex_get_symbol:n { #2 }
  \__stex_refs_sym_aux:nn{#1}{\l_stex_get_symbol_uri_str}
}
```

```

\cs_new_protected:Nn \__stex_refs_sym_aux:nn {
  \str_if_exist:cTF {sref_sym_#2 _label_str }{
    \sref[#1]{\use:c{sref_sym_#2 _label_str}}
  }{
    \__stex_refs_args:n { #1 }
    \str_if_empty:NTF \l__stex_refs_indocument_str {
      \tl_if_exist:cTF{sref_sym_#2 _type}{
        % doc uri in \l_tmpb_str
        \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
        \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
          % reference
          \tl_if_empty:NTF \l__stex_refs_linktext_tl {
            \cs_if_exist:cTF{autoref}{
              \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
            }{
              \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
            }
          }{
            \ltx@ifpackageloaded{hyperref}{
              \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
            }{
              \l__stex_refs_linktext_tl
            }
          }
        }{
          % URL
          \ltx@ifpackageloaded{hyperref}{
            \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl \
          }{
            \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_re
          }
        }
      }{
        \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_l
      }
    }{
      % TODO
    }
  }
}
```

(*End definition for* `\srefsym`. *This function is documented on page 52.*)

`\srefsymuri`

```
\cs_new_protected:Npn \srefsymuri #1 #2 {
  \__stex_refs_sym_aux:nn{linktext={#2}}{#1}
}
```

(*End definition for* `\srefsymuri`. *This function is documented on page 52.*)

```
⟨/package⟩
```

Chapter 27

sTeX -Modules Implementation

```
⟨*package⟩

%%%%%%%%%%%%%   modules.dtx   %%%%%%%%%%%%%

⟨@@=stex_modules⟩
```

Warnings and error messages

```
\msg_new:nnn{stex}{error/unknownmodule}{
  No~module~#1~found
}
\msg_new:nnn{stex}{error/syntax}{
  Syntax~error:~#1
}
\msg_new:nnn{stex}{error/siglanguage}{
  Module~#1~declares~signature~#2,~but~does~not~
  declare~its~language
}
\msg_new:nnn{stex}{warning/deprecated}{
  #1~is~deprecated;~please~use~#2~instead!
}

\msg_new:nnn{stex}{error/conflictingmodules}{
  Conflicting~imports~for~module~#1
}
```

\l_stex_current_module_str The current module:

```
\str_new:N \l_stex_current_module_str
```

(*End definition for* \l_stex_current_module_str*. This variable is documented on page 54.*)

\l_stex_all_modules_seq Stores all available modules

```
\seq_new:N \l_stex_all_modules_seq
```

(*End definition for* \l_stex_all_modules_seq*. This variable is documented on page 54.*)

\stex_if_in_module_p:
\stex_if_in_module:*TF*

```
\prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
  \str_if_empty:NTF \l_stex_current_module_str
    \prg_return_false: \prg_return_true:
}
```

(*End definition for* `\stex_if_in_module:TF`*. This function is documented on page 54.*)

\stex_if_module_exists_p:n
\stex_if_module_exists:n*TF*

```
\prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
  \prop_if_exist:cTF { c_stex_module_#1_prop }
    \prg_return_true: \prg_return_false:
}
```

(*End definition for* `\stex_if_module_exists:nTF`*. This function is documented on page 54.*)

\stex_add_to_current_module:n
\STEXexport

Only allowed within modules:

```
\cs_new_protected:Nn \stex_add_to_current_module:n {
  \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
}
\cs_new_protected:Npn \STEXexport {
  \begingroup
  \newlinechar=-1\relax
  \endlinechar=-1\relax
  %\catcode`\ = 9\relax
  \expandafter\endgroup\__stex_modules_export:n
}
\cs_new_protected:Nn \__stex_modules_export:n {
  \ignorespaces #1
  \stex_add_to_current_module:n { \ignorespaces #1 }
  \stex_smsmode_do:
}
\stex_deactivate_macro:Nn \STEXexport {module~environments}
```

(*End definition for* `\stex_add_to_current_module:n` *and* `\STEXexport`*. These functions are documented on page 54.*)

\stex_add_constant_to_current_module:n

```
\cs_new_protected:Nn \stex_add_constant_to_current_module:n {
  \str_set:Nx \l_tmpa_str { #1 }
  \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
}
```

(*End definition for* `\stex_add_constant_to_current_module:n`*. This function is documented on page 54.*)

\stex_add_import_to_current_module:n

```
\cs_new_protected:Nn \stex_add_import_to_current_module:n {
  \str_set:Nx \l_tmpa_str { #1 }
  \exp_args:Nno
  \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
    \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
  }
}
```

(*End definition for* `\stex_add_import_to_current_module:n`. *This function is documented on page 54.*)

`\stex_collect_imports:n`

```
\cs_new_protected:Nn \stex_collect_imports:n {
  \seq_clear:N \l_stex_collect_imports_seq
  \__stex_modules_collect_imports:n {#1}
}
\cs_new_protected:Nn \__stex_modules_collect_imports:n {
  \seq_map_inline:cn {c_stex_module_#1_imports} {
    \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
      \__stex_modules_collect_imports:n { ##1 }
    }
  }
  \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
    \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
  }
}
```

(*End definition for* `\stex_collect_imports:n`. *This function is documented on page 54.*)

`\stex_do_up_to_module:n`

```
\int_new:N \l__stex_modules_group_depth_int
\tl_new:N \l__stex_modules_aftergroup_tl
\cs_new_protected:Nn \stex_do_up_to_module:n {
  \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
    #1
  }{
    #1
    \expandafter \tl_gset:Nn \expandafter \l__stex_modules_aftergroup_tl \expandafter { \l__
    \aftergroup\__stex_modules_aftergroup_do:
  }
}
\cs_new_protected:Nn \__stex_modules_aftergroup_do: {
  \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
    \l__stex_modules_aftergroup_tl
    \tl_clear:N \l__stex_modules_aftergroup_tl
  }{
    \l__stex_modules_aftergroup_tl
    \aftergroup\__stex_modules_aftergroup_do:
  }
}
\cs_new_protected:Nn \_stex_reset_up_to_module: {

  \tl_gset_eq:NN \l__stex_modules_aftergroup_tl \l__stex_modules_aftergroup_outer_tl
}
```

(*End definition for* `\stex_do_up_to_module:n`. *This function is documented on page 54.*)

`\stex_modules_compute_namespace:nN` Computes the appropriate namespace from the top-level namespace of a repository (`#1`) and a file path (`#2`).

(*End definition for* `\stex_modules_compute_namespace:nN`. *This function is documented on page ??.*)

\stex_modules_current_namespace: Computes the current namespace based on the current MathHub repository (if existent) and the current file.

```
\str_new:N \l_stex_modules_ns_str
\str_new:N \l_stex_modules_subpath_str
\cs_new_protected:Nn \__stex_modules_compute_namespace:nN {
  \str_set:Nx \l_tmpa_str { #1 }
  \seq_set_eq:NN \l_tmpa_seq #2
  % split off file extension
  \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
  \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
  \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
  \seq_put_right:No \l_tmpa_seq \l_tmpb_str

  \bool_set_true:N \l_tmpa_bool
  \bool_while_do:Nn \l_tmpa_bool {
    \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
    \exp_args:No \str_case:nnTF { \l_tmpb_str } {
      {source} { \bool_set_false:N \l_tmpa_bool }
    }{}{
      \seq_if_empty:NT \l_tmpa_seq {
        \bool_set_false:N \l_tmpa_bool
      }
    }
  }

  \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
  \str_if_empty:NTF \l_stex_modules_subpath_str {
    \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
  }{
    \str_set:Nx \l_stex_modules_ns_str {
      \l_tmpa_str/\l_stex_modules_subpath_str
    }
  }
}

\cs_new_protected:Nn \stex_modules_current_namespace: {
  \str_clear:N \l_stex_modules_subpath_str
  \prop_if_exist:NTF \l_stex_current_repository_prop {
    \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
    \__stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
  }{
    % split off file extension
    \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
    \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
    \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
    \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
    \seq_put_right:No \l_tmpa_seq \l_tmpb_str
    \str_set:Nx \l_stex_modules_ns_str {
      file:/\stex_path_to_string:N \l_tmpa_seq
    }
  }
}
```

(*End definition for* `\stex_modules_current_namespace:`*. This function is documented on page 55.*)

27.1 The smodule environment

smodule arguments:

```
\keys_define:nn { stex / module } {
  title         .tl_set:N     = \smoduletitle ,
  type          .str_set_x:N  = \smoduletype ,
  id            .str_set_x:N  = \smoduleid ,
  deprecate     .str_set_x:N  = \l_stex_module_deprecate_str ,
  ns            .str_set_x:N  = \l_stex_module_ns_str ,
  lang          .str_set_x:N  = \l_stex_module_lang_str ,
  sig           .str_set_x:N  = \l_stex_module_sig_str ,
  creators      .str_set_x:N  = \l_stex_module_creators_str ,
  contributors  .str_set_x:N  = \l_stex_module_contributors_str ,
  meta          .str_set_x:N  = \l_stex_module_meta_str ,
  srccite       .str_set_x:N  = \l_stex_module_srccite_str
}

\cs_new_protected:Nn \__stex_modules_args:n {
  \str_clear:N \smoduletitle
  \str_clear:N \smoduletype
  \str_clear:N \smoduleid
  \str_clear:N \l_stex_module_ns_str
  \str_clear:N \l_stex_module_deprecate_str
  \str_clear:N \l_stex_module_lang_str
  \str_clear:N \l_stex_module_sig_str
  \str_clear:N \l_stex_module_creators_str
  \str_clear:N \l_stex_module_contributors_str
  \str_clear:N \l_stex_module_meta_str
  \str_clear:N \l_stex_module_srccite_str
  \keys_set:nn { stex / module } { #1 }
}

% module parameters here? In the body?

```

\stex_module_setup:nn

Sets up a new module property list:

```
\cs_new_protected:Nn \stex_module_setup:nn {
  \tl_gset_eq:NN \l__stex_modules_aftergroup_outer_tl \l__stex_modules_aftergroup_tl
  \tl_clear:N \l__stex_modules_aftergroup_tl
  \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
  \str_set:Nx \l_stex_module_name_str { #2 }
  \__stex_modules_args:n { #1 }
```

First, we set up the name and namespace of the module.
Are we in a nested module?

```
  \stex_if_in_module:TF {
    % Nested module
    \prop_get:cnN {c_stex_module_\l_stex_current_module_str _prop}
      { ns } \l_stex_module_ns_str
    \str_set:Nx \l_stex_module_name_str {
      \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
        { name } / \l_stex_module_name_str
    }
  }{
    % not nested:
    \str_if_empty:NT \l_stex_module_ns_str {
      \stex_modules_current_namespace:
      \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
      \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
          / {\l_stex_module_ns_str}
      \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
      \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
        \str_set:Nx \l_stex_module_ns_str {
          \stex_path_to_string:N \l_tmpa_seq
        }
      }
    }
  }
```

Next, we determine the language of the module:

```
  \str_if_empty:NT \l_stex_module_lang_str {
    \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
    \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
    \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
    \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
    \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
      \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
        inferred~from~file~name}
      \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
    }
  }

  \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
    \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
      \l_tmpa_str {
        \ltx@ifpackageloaded{babel}{
          \exp_args:Nx \selectlanguage { \l_tmpa_str }
        }{}
      } {
        \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
      }
  }}
```

We check if we need to extend a signature module, and set `\l_stex_current_-module_prop` accordingly:

```
  \str_if_empty:NTF \l_stex_module_sig_str {
    \exp_args:Nnx \prop_gset_from_keyval:cn {
      c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
    } {
      name      = \l_stex_module_name_str ,
      ns        = \l_stex_module_ns_str ,
      file      = \exp_not:o { \g_stex_currentfile_seq } ,
      lang      = \l_stex_module_lang_str ,
      sig       = \l_stex_module_sig_str ,
      deprecate = \l_stex_module_deprecate_str ,
      meta      = \l_stex_module_meta_str
    }
    \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
    \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
    \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _copymodules}
    \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
    \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
```

We load the metatheory:

```
    \str_if_empty:NT \l_stex_module_meta_str {
      \str_set:Nx \l_stex_module_meta_str {
        \c_stex_metatheory_ns_str ? Metatheory
      }
    }
    \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
      \bool_set_true:N \l_stex_in_meta_bool
      \exp_args:Nx \stex_add_to_current_module:n {
        \bool_set_true:N \l_stex_in_meta_bool
        \stex_activate_module:n {\l_stex_module_meta_str}
        \bool_set_false:N \l_stex_in_meta_bool
      }
      \stex_activate_module:n {\l_stex_module_meta_str}
      \bool_set_false:N \l_stex_in_meta_bool
    }
  }{
    \str_if_empty:NT \l_stex_module_lang_str {
      \msg_error:nnxx{stex}{error/siglanguage}{
        \l_stex_module_ns_str?\l_stex_module_name_str
      }{\l_stex_module_sig_str}
    }

    \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
    \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
    \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
    \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
    \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
    \str_set:Nx \l_tmpa_str {
      \stex_path_to_string:N \l_tmpa_seq /
      \l_tmpa_str . \l_stex_module_sig_str .tex
    }
    \IfFileExists \l_tmpa_str {
      \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
        \str_clear:N \l_stex_current_module_str
        \seq_clear:N \l_stex_all_modules_seq
        \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
      }
    }{
      \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
    }
    \stex_if_smsmode:F {
      \stex_activate_module:n {
        \l_stex_module_ns_str ? \l_stex_module_name_str
      }
    }
    \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
  }
  \str_if_empty:NF \l_stex_module_deprecate_str {
    \msg_warning:nnxx{stex}{warning/deprecated}{
      Module~\l_stex_current_module_str
    }{
      \l_stex_module_deprecate_str
    }
  }
  \seq_put_right:Nx \l_stex_all_modules_seq {
    \l_stex_module_ns_str ? \l_stex_module_name_str
  }
}
```

(*End definition for* `\stex_module_setup:nn`. *This function is documented on page 55.*)

smodule The `module` environment.

`\__stex_modules_begin_module:` implements `\begin{smodule}`

```
\cs_new_protected:Nn \__stex_modules_begin_module: {
  \stex_reactivate_macro:N \STEXexport
  \stex_reactivate_macro:N \importmodule
  \stex_reactivate_macro:N \symdecl
  \stex_reactivate_macro:N \notation
  \stex_reactivate_macro:N \symdef

  \stex_debug:nn{modules}{
    New~module:\\
    Namespace:~\l_stex_module_ns_str\\
    Name:~\l_stex_module_name_str\\
    Language:~\l_stex_module_lang_str\\
    Signature:~\l_stex_module_sig_str\\
    Metatheory:~\l_stex_module_meta_str\\
    File:~\stex_path_to_string:N \g_stex_currentfile_seq
  }

  \stex_if_smsmode:F{
    \begin{stex_annotate_env} {theory} {
      \l_stex_module_ns_str ? \l_stex_module_name_str
    }

    \stex_annotate_invisible:nnn{header}{} {
      \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
      \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
      \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
        \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
      }
      \str_if_empty:NF \smoduletype {
        \stex_annotate:nnn{type}{\smoduletype}{}
      }
    }
  }
  % TODO: Inherit metatheory for nested modules?
}
\iffalse \end{stex_annotate_env} \fi %^^A make syntax highlighting work again
```

(*End definition for* `\__stex_modules_begin_module:`.)

__stex_modules_end_module: implements \end{module}

```
\cs_new_protected:Nn \__stex_modules_end_module: {
  \stex_debug:nn{modules}{Closing~module~\prop_item:cn {c_stex_module_\l_stex_current_module
}
```

(*End definition for* `\__stex_modules_end_module:`.)

The core environment

```
\iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
\NewDocumentEnvironment { smodule } { O{} m } {
  \stex_module_setup:nn{#1}{#2}
  \par
  \stex_if_smsmode:F{
    \tl_clear:N \l_tmpa_tl
    \clist_map_inline:Nn \smoduletype {
      \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
        \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
      }
    }
    \tl_if_empty:NTF \l_tmpa_tl {
      \__stex_modules_smodule_start:
    }{
      \l_tmpa_tl
    }
  }
  \__stex_modules_begin_module:
  \str_if_empty:NF \smoduleid {
    \stex_ref_new_doc_target:n \smoduleid
  }
  \stex_smsmode_do:
} {
  \__stex_modules_end_module:
  \stex_if_smsmode:F {
    \end{stex_annotate_env}
    \clist_set:No \l_tmpa_clist \smoduletype
    \tl_clear:N \l_tmpa_tl
    \clist_map_inline:Nn \l_tmpa_clist {
      \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
        \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
      }
    }
    \tl_if_empty:NTF \l_tmpa_tl {
      \__stex_modules_smodule_end:
    }{
      \l_tmpa_tl
    }
  }
}
```

\stexpatchmodule

```
\cs_new_protected:Nn \__stex_modules_smodule_start: {}
\cs_new_protected:Nn \__stex_modules_smodule_end: {}

\newcommand\stexpatchmodule[3][] {
    \str_set:Nx \l_tmpa_str{ #1 }
    \str_if_empty:NTF \l_tmpa_str {
      \tl_set:Nn \__stex_modules_smodule_start: { #2 }
      \tl_set:Nn \__stex_modules_smodule_end: { #3 }
    }{
      \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
      \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
    }
}
```

(*End definition for* `\stexpatchmodule`. *This function is documented on page 55.*)

27.2 Invoking modules

`\STEXModule`
`\stex_invoke_module:n`

```
\NewDocumentCommand \STEXModule { m } {
  \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
  \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
  \tl_set:Nn \l_tmpa_tl {
    \msg_error:nnx{stex}{error/unknownmodule}{#1}
  }
  \seq_map_inline:Nn \l_stex_all_modules_seq {
    \str_set:Nn \l_tmpb_str { ##1 }
    \str_if_eq:eeT { \l_tmpa_str } {
      \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
    } {
      \seq_map_break:n {
        \tl_set:Nn \l_tmpa_tl {
          \stex_invoke_module:n { ##1 }
        }
      }
    }
  }
  \l_tmpa_tl
}

\cs_new_protected:Nn \stex_invoke_module:n {
  \stex_debug:nn{modules}{Invoking~module~#1}
  \peek_charcode_remove:NTF ! {
    \__stex_modules_invoke_uri:nN { #1 }
  } {
    \peek_charcode_remove:NTF ? {
      \__stex_modules_invoke_symbol:nn { #1 }
    } {
      \msg_error:nnx{stex}{error/syntax}{
        ?~or~!~expected~after~
        \c_backslash_str STEXModule{#1}
      }
    }
  }
}

\cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
  \str_set:Nn #2 { #1 }
}

\cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
  \stex_invoke_symbol:n{#1?#2}
}
```

(*End definition for* `\STEXModule` *and* `\stex_invoke_module:n`*. These functions are documented on page 55.*)

`\stex_activate_module:n`

```
\bool_new:N \l_stex_in_meta_bool
\bool_set_false:N \l_stex_in_meta_bool
\cs_new_protected:Nn \stex_activate_module:n {
  \stex_debug:nn{modules}{Activating~module~#1}
  \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
    \msg_error:nnn{stex}{error/conflictingmodules}{ #1 }
  }
  \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
    \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
    \use:c{ c_stex_module_#1_code }
  }
}
```

(*End definition for* `\stex_activate_module:n`*. This function is documented on page 56.*)

```
⟨/package⟩
```

Chapter 28

sTeX -Module Inheritance Implementation

```
⟨*package⟩

%%%%%%%%%%%%%   inheritance.dtx   %%%%%%%%%%%%%

```

28.1 SMS Mode

```
⟨@@=stex_smsmode⟩
```

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

```
\tl_new:N \g_stex_smsmode_allowedmacros_tl
\tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
\seq_new:N \g_stex_smsmode_allowedenvs_seq

\tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
  \makeatletter
  \makeatother
  \ExplSyntaxOn
  \ExplSyntaxOff
  \rustexBREAK
}

\tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
  \symdef
  \importmodule
  \notation
  \symdecl
  \STEXexport
  \inlineass
  \inlinedef
  \inlineex
  \endinput
  \setnotation
  \copynotation
  \assign
  \renamedecl
  \donotcopy
  \instantiate
}

\exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
  \tl_to_str:n {
    smodule,
    copymodule,
    interpretmodule,
    sdefinition,
    sexample,
    sassertion,
    sparagraph,
    mathstructure
  }
}
```

(*End definition for* `\g_stex_smsmode_allowedmacros_tl` *,* `\g_stex_smsmode_allowedmacros_escape_tl` *, and* `\g_stex_smsmode_allowedenvs_seq`*. These variables are documented on page 57.*)

`\stex_if_smsmode_p:`
`\stex_if_smsmode:TF`

```
\bool_new:N \g__stex_smsmode_bool
\bool_set_false:N \g__stex_smsmode_bool
\prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
  \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
}
```

(*End definition for* `\stex_if_smsmode:TF`*. This function is documented on page 57.*)

`\__stex_smsmode_in_smsmode:nn`

```
\cs_new_protected:Nn \__stex_smsmode_in_smsmode:nn {
  \vbox_set:Nn \l_tmpa_box {
    \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
    \bool_gset_true:N \g__stex_smsmode_bool
    #2
    \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
  }
  \box_clear:N \l_tmpa_box
}
```

(*End definition for* `\__stex_smsmode_in_smsmode:nn`*.*)

`\stex_file_in_smsmode:nn`

```
\quark_new:N \q__stex_smsmode_break

\NewDocumentCommand \__stex_smsmode_importmodule: { O{} m} {
  \seq_gput_right:Nn \l__stex_smsmode_importmodules_seq {{#1}{#2}}
  \stex_smsmode_do:
}

\cs_new_protected:Nn \stex_file_in_smsmode:nn {
  \stex_filestack_push:n{#1}
  \seq_gclear:N \l__stex_smsmode_importmodules_seq
  % ----- new -----------------------------
  \__stex_smsmode_in_smsmode:nn{#1}{
    \let\importmodule\__stex_smsmode_importmodule:
    \seq_clear:N \g_stex_smsmode_allowedenvs_seq
    \tl_clear:N \g_stex_smsmode_allowedmacros_tl
    \tl_clear:N \g_stex_smsmode_allowedmacros_escape_tl
    \tl_put_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\importmodule}
    \everyeof{\q__stex_smsmode_break\noexpand}
    \expandafter\expandafter\expandafter
    \stex_smsmode_do:
    \csname @ @ input\endcsname "#1"\relax
  }
  % ----- new -----------------------------
  \__stex_smsmode_in_smsmode:nn{#1} {
    #2
    % ----- new -----------------------------
    \begingroup
    \stex_debug:nn{smsmode}{Here:~\seq_use:Nn\l__stex_smsmode_importmodules_seq, }
    \seq_map_inline:Nn \l__stex_smsmode_importmodules_seq {
      \stex_import_module_uri:nn ##1
      \stex_import_require_module:nnnn
        \l_stex_import_ns_str
        \l_stex_import_archive_str
        \l_stex_import_path_str
        \l_stex_import_name_str
    }
    \endgroup
    \stex_debug:nn{smsmode}{Actually~loading~file~#1}
    % ----- new -----------------------------
    \everyeof{\q__stex_smsmode_break\noexpand}
    \expandafter\expandafter\expandafter
    \stex_smsmode_do:
    \csname @ @ input\endcsname "#1"\relax
  }
  \stex_filestack_pop:
}
```

(*End definition for* `\stex_file_in_smsmode:nn`. *This function is documented on page 58.*)

`\stex_smsmode_do:` is executed on encountering \ in smsmode. It checks whether the corresponding command is allowed and executes or ignores it accordingly:

```
\cs_new_protected:Npn \stex_smsmode_do: {
  \stex_if_smsmode:T {
    \__stex_smsmode_do:w
  }
}
\cs_new_protected:Npn \__stex_smsmode_do:w #1 {
  \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 }}{
    \expandafter\if\expandafter\relax\noexpand#1
      \expandafter\__stex_smsmode_do_aux:N\expandafter#1
    \else\expandafter\__stex_smsmode_do:w\fi
  }{
    \__stex_smsmode_do:w %#1
  }
}
\cs_new_protected:Nn \__stex_smsmode_do_aux:N {
  \cs_if_eq:NNF #1 \q__stex_smsmode_break {
    \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
      #1\__stex_smsmode_do:w
    }{
      \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
        #1
      }{
        \cs_if_eq:NNTF \begin #1 {
          \__stex_smsmode_check_begin:n
        }{
          \cs_if_eq:NNTF \end #1 {
            \__stex_smsmode_check_end:n
          }{
            \__stex_smsmode_do:w
          }
        }
      }
    }
  }
}

\cs_new_protected:Nn \__stex_smsmode_check_begin:n {
  \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
    \begin{#1}
  }{
    \__stex_smsmode_do:w
  }
}
\cs_new_protected:Nn \__stex_smsmode_check_end:n {
  \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
    \end{#1}\__stex_smsmode_do:w
  }{
    \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
  }
}
```

(*End definition for* `\stex_smsmode_do:`. *This function is documented on page 58.*)

28.2 Inheritance

```
⟨@@=stex_importmodule⟩
```

`\stex_import_module_uri:nn`

```
\cs_new_protected:Nn \stex_import_module_uri:nn {
  \str_set:Nx \l_stex_import_archive_str { #1 }
  \str_set:Nn \l_stex_import_path_str { #2 }

  \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
  \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
  \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }

  \stex_modules_current_namespace:
  \bool_lazy_all:nTF {
    {\str_if_empty_p:N \l_stex_import_archive_str}
    {\str_if_empty_p:N \l_stex_import_path_str}
    {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
  }{
    \str_set_eq:NN \l_stex_import_path_str \l_stex_modules_subpath_str
    \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
  }{
    \str_if_empty:NT \l_stex_import_archive_str {
      \prop_if_exist:NT \l_stex_current_repository_prop {
        \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
      }
    }
    \str_if_empty:NTF \l_stex_import_archive_str {
      \str_if_empty:NF \l_stex_import_path_str {
        \str_set:Nx \l_stex_import_ns_str {
          \l_stex_module_ns_str / \l_stex_import_path_str
        }
      }
    }{
      \stex_require_repository:n \l_stex_import_archive_str
      \prop_get:cnN { c_stex_mathhub_\l_stex_import_archive_str _manifest_prop } { ns }
        \l_stex_import_ns_str
      \str_if_empty:NF \l_stex_import_path_str {
        \str_set:Nx \l_stex_import_ns_str {
          \l_stex_import_ns_str / \l_stex_import_path_str
        }
      }
    }
  }
}
```

(*End definition for* `\stex_import_module_uri:nn`. *This function is documented on page 59.*)

`\l_stex_import_name_str`
`\l_stex_import_archive_str`
`\l_stex_import_path_str`
`\l_stex_import_ns_str`

Store the return values of `\stex_import_module_uri:nn`.

```
\str_new:N \l_stex_import_name_str
\str_new:N \l_stex_import_archive_str
\str_new:N \l_stex_import_path_str
\str_new:N \l_stex_import_ns_str
```

(*End definition for* `\l_stex_import_name_str` *and others. These variables are documented on page 59.*)

`\stex_import_require_module:nnnn` {⟨*ns*⟩} {⟨*archive-ID*⟩} {⟨*path*⟩} {⟨*name*⟩}

```
\cs_new_protected:Nn \stex_import_require_module:nnnn {
  \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {

    % archive
    \str_set:Nx \l_tmpa_str { #2 }
    \str_if_empty:NTF \l_tmpa_str {
      \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
    } {
      \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
      \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
      \seq_put_right:Nn \l_tmpa_seq { source }
    }

    % path
    \str_set:Nx \l_tmpb_str { #3 }
    \str_if_empty:NTF \l_tmpb_str {
      \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }

      \ltx@ifpackageloaded{babel} {
        \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
            { \languagename } \l_tmpb_str {
              \msg_error:nnx{stex}{error/unknownlanguage}{\languagename}
            }
      } {
        \str_clear:N \l_tmpb_str
      }

      \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
      \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
        \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
      }{
        \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
        \IfFileExists{ \l_tmpa_str.tex }{
          \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
        }{
          % try english as default
          \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
          \IfFileExists{ \l_tmpa_str.en.tex }{
            \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
          }{
            \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
          }
        }
      }

    } {
      \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
      \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq

      \ltx@ifpackageloaded{babel} {
        \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
            { \languagename } \l_tmpb_str {
              \msg_error:nnx{stex}{error/unknownlanguage}{\languagename}
            }
      } {
        \str_clear:N \l_tmpb_str
      }

      \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str

      \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
      \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
        \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
      }{
        \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
        \IfFileExists{ \l_tmpa_str/#4.tex }{
          \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
        }{
          % try english as default
          \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
          \IfFileExists{ \l_tmpa_str/#4.en.tex }{
            \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
          }{
            \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
            \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
              \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
            }{
              \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
              \IfFileExists{ \l_tmpa_str.tex }{
                \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
              }{
                % try english as default
                \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
                \IfFileExists{ \l_tmpa_str.en.tex }{
                  \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
                }{
                  \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
                }
              }
            }
          }
        }
      }
    }

    \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
      \seq_clear:N \l_stex_all_modules_seq
      \str_clear:N \l_stex_current_module_str
      \str_set:Nx \l_tmpb_str { #2 }
      \str_if_empty:NF \l_tmpb_str {
        \stex_set_current_repository:n { #2 }
      }
      \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
    }

    \stex_if_module_exists:nF { #1 ? #4 } {
      \msg_error:nnx{stex}{error/unknownmodule}{
        #1?#4~(in~file~\g__stex_importmodule_file_str)
      }
    }
  }
  \stex_activate_module:n { #1 ? #4 }
}
```

(*End definition for* `\stex_import_require_module:nnnn`*. This function is documented on page 59.*)

`\importmodule`

```
\NewDocumentCommand \importmodule { O{} m } {
  \stex_import_module_uri:nn { #1 } { #2 }
  \stex_debug:nn{modules}{Importing~module:~
    \l_stex_import_ns_str ? \l_stex_import_name_str
  }
  \stex_import_require_module:nnnn
  { \l_stex_import_ns_str } { \l_stex_import_archive_str }
  { \l_stex_import_path_str } { \l_stex_import_name_str }
  \stex_if_smsmode:F {
    \stex_annotate_invisible:nnn
      {import} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
  }
  \exp_args:Nx \stex_add_to_current_module:n {
    \stex_import_require_module:nnnn
    { \l_stex_import_ns_str } { \l_stex_import_archive_str }
    { \l_stex_import_path_str } { \l_stex_import_name_str }
  }
  \exp_args:Nx \stex_add_import_to_current_module:n {
    \l_stex_import_ns_str ? \l_stex_import_name_str
  }
  \stex_smsmode_do:
  \ignorespacesandpars
}
\stex_deactivate_macro:Nn \importmodule {module~environments}
```

(*End definition for* `\importmodule`. *This function is documented on page 58.*)

`\usemodule`

```
\NewDocumentCommand \usemodule { O{} m } {
  \stex_if_smsmode:F {
    \stex_import_module_uri:nn { #1 } { #2 }
    \stex_import_require_module:nnnn
    { \l_stex_import_ns_str } { \l_stex_import_archive_str }
    { \l_stex_import_path_str } { \l_stex_import_name_str }
    \stex_annotate_invisible:nnn
      {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
  }
  \stex_smsmode_do:
  \ignorespacesandpars
}
```

(*End definition for* `\usemodule`. *This function is documented on page 58.*)

```
⟨/package⟩
```

Chapter 29

sTeX -Symbols Implementation

```
⟨*package⟩

%%%%%%%%%%%%%   symbols.dtx   %%%%%%%%%%%%%

```

Warnings and error messages

```
\msg_new:nnn{stex}{error/wrongargs}{
  args~value~in~symbol~declaration~for~#1~
  needs~to~be~i,~a,~b~or~B,~but~#2~given
}
\msg_new:nnn{stex}{error/unknownsymbol}{
  No~symbol~#1~found!
}
\msg_new:nnn{stex}{error/seqlength}{
  Expected~#1~arguments;~got~#2!
}
```

29.1 Symbol Declarations

```
⟨@@=stex_symdecl⟩
```

\stex_all_symbols:n

Map over all available symbols

```
\cs_new_protected:Nn \stex_all_symbols:n {
  \def \__stex_symdecl_all_symbols_cs ##1 {#1}
  \seq_map_inline:Nn \l_stex_all_modules_seq {
    \seq_map_inline:cn{c_stex_module_##1_constants}{
      \__stex_symdecl_all_symbols_cs{##1?####1}
    }
  }
}
```

(*End definition for* `\stex_all_symbols:n`. *This function is documented on page 61.*)

\STEXsymbol

```
\NewDocumentCommand \STEXsymbol { m } {
  \stex_get_symbol:n { #1 }
  \exp_args:No
  \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
}
```

(*End definition for* `\STEXsymbol`*. This function is documented on page 62.*)

`symdecl` arguments:

```
\keys_define:nn { stex / symdecl } {
  name        .str_set_x:N  = \l_stex_symdecl_name_str ,
  local       .bool_set:N   = \l_stex_symdecl_local_bool ,
  args        .str_set_x:N  = \l_stex_symdecl_args_str ,
  type        .tl_set:N     = \l_stex_symdecl_type_tl ,
  deprecate   .str_set_x:N  = \l_stex_symdecl_deprecate_str ,
  align       .str_set:N    = \l_stex_symdecl_align_str , % TODO(?)
  gfc         .str_set:N    = \l_stex_symdecl_gfc_str , % TODO(?)
  specializes .str_set:N    = \l_stex_symdecl_specializes_str , % TODO(?)
  def         .tl_set:N     = \l_stex_symdecl_definiens_tl ,
  assoc       .choices:nn   =
      {bin,binl,binr,pre,conj,pwconj}
      {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}
}

\bool_new:N \l_stex_symdecl_make_macro_bool

\cs_new_protected:Nn \__stex_symdecl_args:n {
  \str_clear:N \l_stex_symdecl_name_str
  \str_clear:N \l_stex_symdecl_args_str
  \str_clear:N \l_stex_symdecl_deprecate_str
  \str_clear:N \l_stex_symdecl_assoctype_str
  \bool_set_false:N \l_stex_symdecl_local_bool
  \tl_clear:N \l_stex_symdecl_type_tl
  \tl_clear:N \l_stex_symdecl_definiens_tl

  \keys_set:nn { stex / symdecl } { #1 }
}
```

`\symdecl` Parses the optional arguments and passes them on to `\stex_symdecl_do:` (so that `\symdef` can do the same)

```

\NewDocumentCommand \symdecl { s m O{}} {
  \__stex_symdecl_args:n { #3 }
  \IfBooleanTF #1 {
    \bool_set_false:N \l_stex_symdecl_make_macro_bool
  } {
    \bool_set_true:N \l_stex_symdecl_make_macro_bool
  }
  \stex_symdecl_do:n { #2 }
  \stex_smsmode_do:
}

\cs_new_protected:Nn \stex_symdecl_do:nn {
  \__stex_symdecl_args:n{#1}
  \bool_set_false:N \l_stex_symdecl_make_macro_bool
  \stex_symdecl_do:n{#2}
}
```

```
\stex_deactivate_macro:Nn \symdecl {module~environments}
```

(*End definition for* `\symdecl`*. This function is documented on page 60.*)

`\stex_symdecl_do:n`

```
\cs_new_protected:Nn \stex_symdecl_do:n {
  \stex_if_in_module:F {
    % TODO throw error? some default namespace?
  }

  \str_if_empty:NT \l_stex_symdecl_name_str {
    \str_set:Nx \l_stex_symdecl_name_str { #1 }
  }

  \prop_if_exist:cT { l_stex_symdecl_
      \l_stex_current_module_str ?
      \l_stex_symdecl_name_str
    _prop
  }{
    % TODO throw error (beware of circular dependencies)
  }

  \prop_clear:N \l_tmpa_prop
  \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
  \seq_clear:N \l_tmpa_seq
  \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
  \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl

  \str_if_empty:NT \l_stex_symdecl_deprecate_str {
    \str_if_empty:NF \l_stex_module_deprecate_str {
      \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
    }
  }
  \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str

  \exp_args:No \stex_add_constant_to_current_module:n {
    \l_stex_symdecl_name_str
  }

  % arity/args
  \int_zero:N \l_tmpb_int

  \bool_set_true:N \l_tmpa_bool
  \str_map_inline:Nn \l_stex_symdecl_args_str {
    \token_case_meaning:NnF ##1 {
      0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
      {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
      {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
      {\tl_to_str:n a} {
        \bool_set_false:N \l_tmpa_bool
        \int_incr:N \l_tmpb_int
      }
      {\tl_to_str:n B} {
```

```
      \bool_set_false:N \l_tmpa_bool
      \int_incr:N \l_tmpb_int
    }
  }{
    \msg_error:nnxx{stex}{error/wrongargs}{
      \l_stex_current_module_str ?
      \l_stex_symdecl_name_str
    }{##1}
  }
}
\bool_if:NTF \l_tmpa_bool {
  % possibly numeric
  \str_if_empty:NTF \l_stex_symdecl_args_str {
    \prop_put:Nnn \l_tmpa_prop { args } {}
    \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
  }{
    \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
    \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
    \str_clear:N \l_tmpa_str
    \int_step_inline:nn \l_tmpa_int {
      \str_put_right:Nn \l_tmpa_str i
    }
    \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
  }
} {
  \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
  \prop_put:Nnx \l_tmpa_prop { arity }
    { \str_count:N \l_stex_symdecl_args_str }
}
\prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }

\tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
  \prop_put:Nnx \l_tmpa_prop { defined }{ false }
}{
  \prop_put:Nnx \l_tmpa_prop { defined }{ true }
}

% semantic macro

\bool_if:NT \l_stex_symdecl_make_macro_bool {
  \exp_args:Nx \stex_do_up_to_module:n {
    \tl_set:cn { #1 } { \stex_invoke_symbol:n {
      \l_stex_current_module_str ? \l_stex_symdecl_name_str
    }}
  }

  \bool_if:NF \l_stex_symdecl_local_bool {
    \exp_args:Nx \stex_add_to_current_module:n {
      \tl_set:cn { #1 } { \stex_invoke_symbol:n {
        \l_stex_current_module_str ? \l_stex_symdecl_name_str
      } }
    }
  }
}
```

```

\stex_debug:nn{symbols}{New~symbol:~
  \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
  Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
  Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
  Definiens:~\exp_not:o {\l_stex_symdecl_definiens_tl}
}

% circular dependencies require this:

\prop_if_exist:cF {
  l_stex_symdecl_
  \l_stex_current_module_str ? \l_stex_symdecl_name_str
  _prop
} {
  \exp_args:Nx \stex_do_up_to_module:n {
    \prop_set_from_keyval:cn {
      l_stex_symdecl_
      \l_stex_current_module_str ? \l_stex_symdecl_name_str
      _prop
    } {\prop_to_keyval:N \l_tmpa_prop}
    \seq_clear:c {
      l_stex_symdecl_
      \l_stex_current_module_str ? \l_stex_symdecl_name_str
      _notations
    }
  }
}

\bool_if:NF \l_stex_symdecl_local_bool {
  \exp_args:Nx
  \stex_add_to_current_module:n {
    \seq_clear:c {
      l_stex_symdecl_
      \l_stex_current_module_str ? \l_stex_symdecl_name_str
      _notations
    }
    \prop_set_from_keyval:cn {
      l_stex_symdecl_
      \l_stex_current_module_str ? \l_stex_symdecl_name_str
      _prop
    } {
      name      = \prop_item:Nn \l_tmpa_prop { name }       ,
      module    = \prop_item:Nn \l_tmpa_prop { module }     ,
      type      = \prop_item:Nn \l_tmpa_prop { type }       ,
      args      = \prop_item:Nn \l_tmpa_prop { args }       ,
      arity     = \prop_item:Nn \l_tmpa_prop { arity }      ,
      assocs    = \prop_item:Nn \l_tmpa_prop { assocs }     ,
      defined   = \prop_item:Nn \l_tmpa_prop { defined }
    }
  }
}

\stex_if_smsmode:F {
```

```
%    \exp_args:Nx \stex_do_up_to_module:n {
%        \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
%        \l_stex_current_module_str ? \l_stex_symdecl_name_str
%      }
%    }
    \stex_if_do_html:T {
      \stex_annotate_invisible:nnn {symdecl} {
        \l_stex_current_module_str ? \l_stex_symdecl_name_str
      } {
        \tl_if_empty:NF \l_stex_symdecl_type_tl {
          \stex_annotate_invisible:nnn{type}{}{$\l_stex_symdecl_type_tl$}
        }
        \stex_annotate_invisible:nnn{args}{}{
          \prop_item:Nn \l_tmpa_prop { args }
        }
        \stex_annotate_invisible:nnn{macroname}{#1}{}
        \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
          \stex_annotate_invisible:nnn{definiens}{}
            {$\l_stex_symdecl_definiens_tl$}
        }
        \str_if_empty:NF \l_stex_symdecl_assoctype_str {
          \stex_annotate_invisible:nnn{assoctype}{\l_stex_symdecl_assoctype_str}{}
        }
      }
    }
  }
}
```

(*End definition for* `\stex_symdecl_do:n`*. This function is documented on page 61.*)

`\stex_get_symbol:n`

```
\str_new:N \l_stex_get_symbol_uri_str

\cs_new_protected:Nn \stex_get_symbol:n {
  \tl_if_head_eq_catcode:nNTF { #1 } \relax {
    \tl_set:Nn \l_tmpa_tl { #1 }
    \__stex_symdecl_get_symbol_from_cs:
  }{
    % argument is a string
    % is it a command name?
    \cs_if_exist:cTF { #1 }{
      \cs_set_eq:Nc \l_tmpa_tl { #1 }
      \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
      \str_if_empty:NTF \l_tmpa_str {
        \exp_args:Nx \cs_if_eq:NNTF {
          \tl_head:N \l_tmpa_tl
        } \stex_invoke_symbol:n {
          \__stex_symdecl_get_symbol_from_cs:
        }{
          \__stex_symdecl_get_symbol_from_string:n { #1 }
        }
      } {
        \__stex_symdecl_get_symbol_from_string:n { #1 }
      }
    }{
      % argument is not a command name
      \__stex_symdecl_get_symbol_from_string:n { #1 }
      % \l_stex_all_symbols_seq
    }
  }
  \str_if_eq:eeF {
    \prop_item:cn {
      l_stex_symdecl_\l_stex_get_symbol_uri_str _prop
    }{ deprecate }
  }{}{
    \msg_warning:nnxx{stex}{warning/deprecated}{
      Symbol~\l_stex_get_symbol_uri_str
    }{
      \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{ deprecate }
    }
  }
}

\cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
  \tl_set:Nn \l_tmpa_tl {
    \msg_error:nnn{stex}{error/unknownsymbol}{#1}
  }
  \str_set:Nn \l_tmpa_str { #1 }
  \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }

  \stex_all_symbols:n {
    \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
      \seq_map_break:n{\seq_map_break:n{
        \tl_set:Nn \l_tmpa_tl {
          \str_set:Nn \l_stex_get_symbol_uri_str { ##1 }
        }
      }}
    }
  }

  \l_tmpa_tl
}

\cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
  \exp_args:NNx \tl_set:Nn \l_tmpa_tl
    { \tl_tail:N \l_tmpa_tl }
  \tl_if_single:NTF \l_tmpa_tl {
    \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
      \exp_after:wN \str_set:Nn \exp_after:wN
        \l_stex_get_symbol_uri_str \l_tmpa_tl
    }{
      % TODO
      % tail is not a single group
    }
  }{
    % TODO
    % tail is not a single group
  }
```

```
}
```

(*End definition for* `\stex_get_symbol:n`. *This function is documented on page 61.*)

29.2 Notations

```
⟨@@=stex_notation⟩
```

`notation` arguments:

```
\keys_define:nn { stex / notation } {
  lang    .tl_set_x:N  = \l__stex_notation_lang_str ,
  variant .tl_set_x:N  = \l__stex_notation_variant_str ,
  prec    .str_set_x:N = \l__stex_notation_prec_str ,
  op      .tl_set:N    = \l__stex_notation_op_tl ,
  primary .bool_set:N  = \l__stex_notation_primary_bool ,
  primary .default:n   = {true} ,
  unknown .code:n      = \str_set:Nx
      \l__stex_notation_variant_str \l_keys_key_str
}

\cs_new_protected:Nn \_stex_notation_args:n {
  \str_clear:N \l__stex_notation_lang_str
  \str_clear:N \l__stex_notation_variant_str
  \str_clear:N \l__stex_notation_prec_str
  \tl_clear:N \l__stex_notation_op_tl
  \bool_set_false:N \l__stex_notation_primary_bool

  \keys_set:nn { stex / notation } { #1 }
}
```

\notation

```
\NewDocumentCommand \notation { s m O{}} {
  \_stex_notation_args:n { #3 }
  \tl_clear:N \l_stex_symdecl_definiens_tl
  \stex_get_symbol:n { #2 }
  \tl_set:Nn \l_stex_notation_after_do_tl {
    \__stex_notation_final:
    \IfBooleanTF#1{
      \stex_setnotation:n {\l_stex_get_symbol_uri_str}
    }{}
    \stex_smsmode_do:\ignorespacesandpars
  }
  \stex_notation_do:nnnnn
    { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } }
    { \prop_item:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } }
    { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
    { \l__stex_notation_prec_str}
}
\stex_deactivate_macro:Nn \notation {module~environments}
```

(*End definition for* `\notation`. *This function is documented on page 61.*)

\stex_notation_do:nnnnn

```
\seq_new:N \l__stex_notation_precedences_seq
```

```
\tl_new:N \l__stex_notation_opprec_tl
\int_new:N \l__stex_notation_currarg_int
\tl_new:N \stex_symbol_after_invokation_tl

\cs_new_protected:Nn \stex_notation_do:nnnnn {
  \let\l_stex_current_symbol_str\relax
  \seq_clear:N \l__stex_notation_precedences_seq
  \tl_clear:N \l__stex_notation_opprec_tl
  \str_set:Nx \l__stex_notation_args_str { #1 }
  \str_set:Nx \l__stex_notation_arity_str { #2 }
  \str_set:Nx \l__stex_notation_suffix_str { #3 }
  \str_set:Nx \l__stex_notation_prec_str { #4 }

  % precedences
  \str_if_empty:NTF \l__stex_notation_prec_str {
    \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
      \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
    }{
      \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
    }
  } {
    \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
      \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
      \int_step_inline:nn { \l__stex_notation_arity_str } {
        \exp_args:NNo
        \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
      }
    }{
      \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
      \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
        \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
        \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
          \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
            \l_tmpa_seq {\tl_to_str:n{x} } { \l_tmpa_str }
          \seq_map_inline:Nn \l_tmpa_seq {
            \seq_put_right:Nn \l_tmpb_seq { ##1 }
          }
        }
      }{
        \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
          \tl_set:No \l__stex_notation_opprec_tl { \infprec }
        }{
          \tl_set:No \l__stex_notation_opprec_tl { 0 }
        }
      }
    }
  }

  \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
  \int_step_inline:nn { \l__stex_notation_arity_str } {
    \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
      \exp_args:NNo
      \seq_put_right:No \l__stex_notation_precedences_seq {
        \l__stex_notation_opprec_tl
      }
    }
  }
  \tl_clear:N \l_stex_notation_dummyargs_tl

  \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
    \exp_args:NNe
    \cs_set:Npn \l_stex_notation_macrocode_cs {
      \_stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
        { \l__stex_notation_suffix_str }
        { \l__stex_notation_opprec_tl }
        { \exp_not:n { #5 } }
    }
    \l_stex_notation_after_do_tl
  }{
    \str_if_in:NnTF \l__stex_notation_args_str b {
      \exp_args:Nne \use:nn
      {
      \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
      \cs_set:Npn \l__stex_notation_arity_str } { {
        \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
          { \l__stex_notation_suffix_str }
          { \l__stex_notation_opprec_tl }
          { \exp_not:n { #5 } }
      }}
    }{
      \str_if_in:NnTF \l__stex_notation_args_str B {
        \exp_args:Nne \use:nn
        {
        \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
        \cs_set:Npn \l__stex_notation_arity_str } { {
          \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
            { \l__stex_notation_suffix_str }
            { \l__stex_notation_opprec_tl }
            { \exp_not:n { #5 } }
        } }
      }{
        \exp_args:Nne \use:nn
        {
        \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
        \cs_set:Npn \l__stex_notation_arity_str } { {
          \_stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
            { \l__stex_notation_suffix_str }
            { \l__stex_notation_opprec_tl }
            { \exp_not:n { #5 } }
        } }
      }
    }

    \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
    \int_zero:N \l__stex_notation_currarg_int
    \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
    \__stex_notation_arguments:
  }
```

```
}
```

(*End definition for* `\stex_notation_do:nnnnn`*. This function is documented on page* **??**.)

`\__stex_notation_arguments:` Takes care of annotating the arguments in a notation macro

```
\cs_new_protected:Nn \__stex_notation_arguments: {
  \int_incr:N \l__stex_notation_currarg_int
  \str_if_empty:NTF \l__stex_notation_remaining_args_str {
    \l_stex_notation_after_do_tl
  }{
    \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
    \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
    \str_if_eq:VnTF \l_tmpa_str a {
      \__stex_notation_argument_assoc:nn{a}
    }{
      \str_if_eq:VnTF \l_tmpa_str B {
        \__stex_notation_argument_assoc:nn{B}
      }{
        \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpb_str
        \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
          { \_stex_term_math_arg:nnn
            { \l_tmpa_str\int_use:N \l__stex_notation_currarg_int }
            { \l_tmpb_str }
            { ####\int_use:N \l__stex_notation_currarg_int }
          }
        }
        \__stex_notation_arguments:
      }
    }
  }
}
```

(*End definition for* `\__stex_notation_arguments:`.)

`\__stex_notation_argument_assoc:nn`

```
\cs_new_protected:Nn \__stex_notation_argument_assoc:nn {

  \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
    {\l__stex_notation_arity_str}{
    #2
  }
  \int_zero:N \l_tmpa_int
  \tl_clear:N \l_tmpa_tl
  \str_map_inline:Nn \l__stex_notation_args_str {
    \int_incr:N \l_tmpa_int
    \tl_put_right:Nx \l_tmpa_tl {
      \str_if_eq:nnTF {##1}{a}{ {} }{
        \str_if_eq:nnTF {##1}{B}{ {} }{
          {\_stex_term_arg:nn{##1\int_use:N \l_tmpa_int}{################ \int_use:N \l_tmpa
        }
      }
    }
  }
  \exp_after:wN\exp_after:wN\exp_after:wN \def
  \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
  \exp_after:wN\exp_after:wN\exp_after:wN ##
  \exp_after:wN\exp_after:wN\exp_after:wN 1
  \exp_after:wN\exp_after:wN\exp_after:wN ##
  \exp_after:wN\exp_after:wN\exp_after:wN 2
  \exp_after:wN\exp_after:wN\exp_after:wN {
    \exp_after:wN \exp_after:wN \exp_after:wN
    \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
      \exp_after:wN \l_tmpa_cs \l_tmpa_tl
    }
  }

  \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
  \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
    \_stex_term_math_assoc_arg:nnnn
      { #1\int_use:N \l__stex_notation_currarg_int }
      { \l_tmpa_str }
      { ####\int_use:N \l__stex_notation_currarg_int }
      { \l_tmpa_cs {####1} {####2} }
  } }
  \__stex_notation_arguments:
}
```

(*End definition for* `\__stex_notation_argument_assoc:nn`.)

`\__stex_notation_final:` Called after processing all notation arguments

```
\cs_new_protected:Nn \__stex_notation_final: {
%  \exp_args:Nne \use:nn
%  {
%  \cs_generate_from_arg_count:cNnn {
%      stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
%      \l__stex_notation_suffix_str
%      _cs
%    }
%    \cs_set:Npn \l__stex_notation_arity_str } { {
%      \exp_after:wN \exp_after:wN \exp_after:wN
%      \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
%      { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sym
%  } }

%  \tl_if_empty:NF \l__stex_notation_op_tl {
%    \cs_set:cpx {
%      stex_op_notation_ \l_stex_get_symbol_uri_str \c_hash_str
%      \l__stex_notation_suffix_str
%      _cs
%    } { \exp_not:N \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
%  }

  \exp_args:Nx \stex_do_up_to_module:n {
    \cs_generate_from_arg_count:cNnn {
      stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
      \l__stex_notation_suffix_str
      _cs
    } \cs_set:Npn {\l__stex_notation_arity_str} {
```

```
    \exp_after:wN \exp_after:wN \exp_after:wN
    \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
    { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
  }
  \tl_if_empty:NF \l__stex_notation_op_tl {
    \cs_set:cpn {
      stex_op_notation_\l_stex_get_symbol_uri_str \c_hash_str
      \l__stex_notation_suffix_str
      _cs
    } { \exp_not:N \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
  }
}

\exp_args:Ne
\stex_add_to_current_module:n {
  \cs_generate_from_arg_count:cNnn {
    stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
    \l__stex_notation_suffix_str
    _cs
  } \cs_set:Npn {\l__stex_notation_arity_str} {
    \exp_after:wN \exp_after:wN \exp_after:wN
    \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
    { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
  }
  \tl_if_empty:NF \l__stex_notation_op_tl {
    \cs_set:cpn {
      stex_op_notation_\l_stex_get_symbol_uri_str \c_hash_str
      \l__stex_notation_suffix_str
      _cs
    } { \exp_not:N \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
  }
}

\stex_debug:nn{symbols}{
  Notation~\l__stex_notation_suffix_str
  ~for~\l_stex_get_symbol_uri_str^^J
  Operator~precedence:~\l__stex_notation_opprec_tl^^J
  Argument~precedences:~
    \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
  Notation: \cs_meaning:c {
    stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
    \l__stex_notation_suffix_str
    _cs
  }
}

\exp_args:Nx
\stex_do_up_to_module:n {
  \seq_put_right:cx {
    l_stex_symdecl_ \l_stex_get_symbol_uri_str
    _notations
  } {
    \l__stex_notation_suffix_str
  }
```

```
}
\exp_args:Ne
\stex_add_to_current_module:n {
  \seq_put_right:cn {
    l_stex_symdecl_\l_stex_get_symbol_uri_str
    _notations
  } { \l__stex_notation_suffix_str }
}

\stex_if_smsmode:F {

  % HTML annotations
  \stex_if_do_html:T {
    \stex_annotate_invisible:nnn { notation }
    { \l_stex_get_symbol_uri_str } {
      \stex_annotate_invisible:nnn { notationfragment }
        { \l__stex_notation_suffix_str }{}
      \stex_annotate_invisible:nnn { precedence }
        { \l__stex_notation_prec_str }{}

      \int_zero:N \l_tmpa_int
      \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
      \tl_clear:N \l_tmpa_tl
      \int_step_inline:nn { \l__stex_notation_arity_str }{
        \int_incr:N \l_tmpa_int
        \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
        \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_r
        \str_if_eq:VnTF \l_tmpb_str a {
          \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
            \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
            \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
          } }
        }{
          \str_if_eq:VnTF \l_tmpb_str B {
            \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
              \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
              \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
            } }
          }{
            \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
              \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
            } }
          }
        }
      }
      \stex_annotate_invisible:nnn { notationcomp }{}{
        \str_set:Nx \l_stex_current_symbol_str {\l_stex_get_symbol_uri_str }
        $ \exp_args:Nno \use:nn { \use:c {
          stex_notation_ \l_stex_current_symbol_str
          \c_hash_str \l__stex_notation_suffix_str _cs
        } } { \l_tmpa_tl } $
      }
    }
  }
  }
}
```

(*End definition for* `\__stex_notation_final:`.)

\setnotation

```
\keys_define:nn { stex / setnotation } {
  lang    .tl_set_x:N  = \l__stex_notation_lang_str ,
  variant .tl_set_x:N  = \l__stex_notation_variant_str ,
  unknown .code:n      = \str_set:Nx
      \l__stex_notation_variant_str \l_keys_key_str
}

\cs_new_protected:Nn \_stex_setnotation_args:n {
  \str_clear:N \l__stex_notation_lang_str
  \str_clear:N \l__stex_notation_variant_str
  \keys_set:nn { stex / setnotation } { #1 }
}

\cs_new_protected:Nn \stex_setnotation:n {
  \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1 _notations }
    { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
      \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
        { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
      \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
        { \c_hash_str }
      \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl_#1 _notations }
        { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
      \exp_args:Nx \stex_add_to_current_module:n {
        \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
          { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
        \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
          { \c_hash_str }
        \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl_#1 _notations }
          { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
      }
      \stex_debug:nn {notations}{
        Setting~default~notation~
        {\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str}~for~
        #1 \\
        \expandafter\meaning\csname
        l_stex_symdecl_#1 _notations\endcsname
      }
    }{
      % todo throw error
    }
}

\NewDocumentCommand \setnotation {m m} {
  \stex_get_symbol:n { #1 }
  \_stex_setnotation_args:n { #2 }
  \stex_setnotation:n{\l_stex_get_symbol_uri_str}
  \stex_smsmode_do:\ignorespacesandpars
}
```

```

\cs_new_protected:Nn \stex_copy_notations:nn {
  \stex_debug:nn {notations}{
    Copying~notations~from~#2~to~#1\\
    \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
  }
  \tl_clear:N \l_tmpa_tl
  \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
    \tl_put_right:Nn \l_tmpa_tl { {## ##1} }
  }
  \seq_map_inline:cn {l_stex_symdecl_#2_notations}{
    \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
    \edef \l_tmpa_tl {
      \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
      \exp_after:wN\exp_after:wN\exp_after:wN {
        \exp_after:wN \l_tmpa_cs \l_tmpa_tl
      }
    }
    \exp_args:Nx
    \stex_do_up_to_module:n {
      \seq_put_right:cn{l_stex_symdecl_#1_notations}{##1}
      \cs_generate_from_arg_count:cNnn {
        stex_notation_ #1 \c_hash_str ##1 _cs
      } \cs_set:Npn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }{
        \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl}
      }
    }
  }
}

\NewDocumentCommand \copynotation {m m} {
  \stex_get_symbol:n { #1 }
  \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
  \stex_get_symbol:n { #2 }
  \exp_args:Noo
  \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
  \exp_args:Nx \stex_add_import_to_current_module:n{
    \stex_copy_notations:nn {\l_tmpa_str} {\l_stex_get_symbol_uri_str}
  }
  \stex_smsmode_do:\ignorespacesandpars
}

```

(*End definition for* `\setnotation`. *This function is documented on page 18.*)

`\symdef`

```
\keys_define:nn { stex / symdef } {
  name     .str_set_x:N = \l_stex_symdecl_name_str ,
  local    .bool_set:N  = \l_stex_symdecl_local_bool ,
  args     .str_set_x:N = \l_stex_symdecl_args_str ,
  type     .tl_set:N    = \l_stex_symdecl_type_tl ,
  def      .tl_set:N    = \l_stex_symdecl_definiens_tl ,
  op       .tl_set:N    = \l__stex_notation_op_tl ,
  lang     .str_set_x:N = \l__stex_notation_lang_str ,
  variant .str_set_x:N = \l__stex_notation_variant_str ,
  prec    .str_set_x:N = \l__stex_notation_prec_str ,
  assoc   .choices:nn  =
      {bin,binl,binr,pre,conj,pwconj}
      {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
  unknown .code:n      = \str_set:Nx
      \l__stex_notation_variant_str \l_keys_key_str
}

\cs_new_protected:Nn \__stex_notation_symdef_args:n {
  \str_clear:N \l_stex_symdecl_name_str
  \str_clear:N \l_stex_symdecl_args_str
  \str_clear:N \l_stex_symdecl_assoctype_str
  \bool_set_false:N \l_stex_symdecl_local_bool
  \tl_clear:N \l_stex_symdecl_type_tl
  \tl_clear:N \l_stex_symdecl_definiens_tl
  \str_clear:N \l__stex_notation_lang_str
  \str_clear:N \l__stex_notation_variant_str
  \str_clear:N \l__stex_notation_prec_str
  \tl_clear:N \l__stex_notation_op_tl

  \keys_set:nn { stex / symdef } { #1 }
}

\NewDocumentCommand \symdef { m O{} } {
  \__stex_notation_symdef_args:n { #2 }
  \bool_set_true:N \l_stex_symdecl_make_macro_bool
  \stex_symdecl_do:n { #1 }
  \tl_set:Nn \l_stex_notation_after_do_tl {
    \__stex_notation_final:
    \stex_smsmode_do:\ignorespacesandpars
  }
  \str_set:Nx \l_stex_get_symbol_uri_str {
    \l_stex_current_module_str ? \l_stex_symdecl_name_str
  }
  \exp_args:Nx \stex_notation_do:nnnnn
    { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } }
    { \prop_item:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } }
    { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
    { \l__stex_notation_prec_str}
}
\stex_deactivate_macro:Nn \symdef {module~environments}
```

(*End definition for* `\symdef`. *This function is documented on page 61.*)

29.3 Variables

```
⟨@@=stex_variables⟩

\keys_define:nn { stex / vardef } {
  name    .str_set_x:N  = \l__stex_variables_name_str ,
  args    .str_set_x:N  = \l__stex_variables_args_str ,
  type    .tl_set:N     = \l__stex_variables_type_tl ,
  def     .tl_set:N     = \l__stex_variables_def_tl ,
  op      .tl_set:N     = \l__stex_variables_op_tl ,
  prec    .str_set_x:N  = \l__stex_variables_prec_str ,
  assoc   .choices:nn   =
      {bin,binl,binr,pre,conj,pwconj}
      {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
  bind    .choices:nn   =
      {forall,exists}
      {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
}

\cs_new_protected:Nn \__stex_variables_args:n {
  \str_clear:N \l__stex_variables_name_str
  \str_clear:N \l__stex_variables_args_str
  \str_clear:N \l__stex_variables_prec_str
  \str_clear:N \l__stex_variables_assoctype_str
  \str_clear:N \l__stex_variables_bind_str
  \tl_clear:N \l__stex_variables_type_tl
  \tl_clear:N \l__stex_variables_def_tl
  \tl_clear:N \l__stex_variables_op_tl

  \keys_set:nn { stex / vardef } { #1 }
}

\NewDocumentCommand \__stex_variables_do_simple:nnn { m O{}} {
  \__stex_variables_args:n {#2}
  \str_if_empty:NT \l__stex_variables_name_str {
    \str_set:Nx \l__stex_variables_name_str { #1 }
  }
  \prop_clear:N \l_tmpa_prop
  \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str

  \int_zero:N \l_tmpb_int
  \bool_set_true:N \l_tmpa_bool
  \str_map_inline:Nn \l__stex_variables_args_str {
    \token_case_meaning:NnF ##1 {
      0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
      {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
      {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
      {\tl_to_str:n a} {
        \bool_set_false:N \l_tmpa_bool
        \int_incr:N \l_tmpb_int
      }
      {\tl_to_str:n B} {
        \bool_set_false:N \l_tmpa_bool
        \int_incr:N \l_tmpb_int
      }
    }{
      \msg_error:nnxx{stex}{error/wrongargs}{
        variable~\l__stex_variables_name_str
      }{##1}
    }
  }
  \bool_if:NTF \l_tmpa_bool {
    % possibly numeric
    \str_if_empty:NTF \l__stex_variables_args_str {
      \prop_put:Nnn \l_tmpa_prop { args } {}
      \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
    }{
      \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
      \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
      \str_clear:N \l_tmpa_str
      \int_step_inline:nn \l_tmpa_int {
        \str_put_right:Nn \l_tmpa_str i
      }
      \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
      \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
    }
  } {
    \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
    \prop_put:Nnx \l_tmpa_prop { arity }
      { \str_count:N \l__stex_variables_args_str }
  }
  \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
  \tl_set:cx { #1 }{ \stex_invoke_variable:n { \l__stex_variables_name_str } }

  \prop_set_eq:cN { l_stex_variable_\l__stex_variables_name_str _prop} \l_tmpa_prop

  \tl_if_empty:NF \l__stex_variables_op_tl {
    \cs_set:cpx {
      stex_var_op_notation_ \l__stex_variables_name_str _cs
    } { \exp_not:N\comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
  }

  \tl_set:Nn \l_stex_notation_after_do_tl {
    \exp_args:Nne \use:nn {
      \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str _cs }
        \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
    } {{
      \exp_after:wN \exp_after:wN \exp_after:wN
      \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
      { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_symb
    }}
    \stex_if_do_html:T {
      \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
        \stex_annotate_invisible:nnn { precedence }
          { \l__stex_variables_prec_str }{}
        \tl_if_empty:NF \l__stex_variables_type_tl {\stex_annotate_invisible:nnn{type}{}{$\l
        \stex_annotate_invisible:nnn{args}{}{ \l__stex_variables_args_str }
        \stex_annotate_invisible:nnn{macroname}{#1}{}
        \tl_if_empty:NF \l__stex_variables_def_tl {
          \stex_annotate_invisible:nnn{definiens}{}
            {$\l__stex_variables_def_tl$}
        }
        \str_if_empty:NF \l__stex_variables_assoctype_str {
          \stex_annotate_invisible:nnn{assoctype}{\l__stex_variables_assoctype_str}{}
        }
        \str_if_empty:NF \l__stex_variables_bind_str {
          \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{}
        }
        \int_zero:N \l_tmpa_int
        \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
        \tl_clear:N \l_tmpa_tl
        \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
          \int_incr:N \l_tmpa_int
          \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
          \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables
          \str_if_eq:VnTF \l_tmpb_str a {
            \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
              \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
              \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
            } }
          }{
            \str_if_eq:VnTF \l_tmpb_str B {
              \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
                \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
                \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
              } }
            }{
              \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
                \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
              } }
            }
          }
        }
        \stex_annotate_invisible:nnn { notationcomp }{}{
          \str_set:Nx \l_stex_current_symbol_str {var://\l__stex_variables_name_str }
          $ \exp_args:Nno \use:nn { \use:c {
            stex_var_notation_\l__stex_variables_name_str _cs
          } } { \l_tmpa_tl } $
        }
      }
    }\ignorespacesandpars
  }

  \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
}

\cs_new:Nn \_stex_reset:N {
  \tl_if_exist:NTF #1 {
    \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
  }{
    \let \exp_not:N #1 \exp_not:N \undefined
  }
}

\NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
  \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
  \exp_args:Nnx \use:nn {
    % TODO
    \stex_annotate_invisible:nnn {vardecl}{\clist_use:Nn\l__stex_variables_names,}{
      #2
    }
  }{
    \_stex_reset:N \varnot
    \_stex_reset:N \vartype
    \_stex_reset:N \vardefi
  }
}

\NewDocumentCommand \vardef { s } {
  \IfBooleanTF#1 {
    \__stex_variables_do_complex:nn
  }{
    \__stex_variables_do_simple:nnn
  }
}

\NewDocumentCommand \svar { O{} m }{
  \tl_if_empty:nTF {#1}{
    \str_set:Nn \l_tmpa_str { #2 }
  }{
    \str_set:Nn \l_tmpa_str { #1 }
  }
  \_stex_term_omv:nn {
    var://\l_tmpa_str
  }{
    \exp_args:Nnx \use:nn {
      \def\comp{\_varcomp}
      \str_set:Nx \l_stex_current_symbol_str { var://\l_tmpa_str }
      \comp{ #2 }
    }{
      \_stex_reset:N \comp
      \_stex_reset:N \l_stex_current_symbol_str
    }
  }
}



\keys_define:nn { stex / varseq } {
  name    .str_set_x:N  = \l__stex_variables_name_str ,
  args    .int_set:N    = \l__stex_variables_args_int ,
  type    .tl_set:N     = \l__stex_variables_type_tl  ,
  mid     .tl_set:N     = \l__stex_variables_mid_tl   ,
  bind    .choices:nn   =
      {forall,exists}
      {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
}

\cs_new_protected:Nn \__stex_variables_seq_args:n {
  \str_clear:N \l__stex_variables_name_str
  \int_set:Nn \l__stex_variables_args_int 1
  \tl_clear:N \l__stex_variables_type_tl
  \str_clear:N \l__stex_variables_bind_str

  \keys_set:nn { stex / varseq } { #1 }
```

```
}

\NewDocumentCommand \varseq {m O{} m m m}{
  \__stex_variables_seq_args:n { #2 }
  \str_if_empty:NT \l__stex_variables_name_str {
    \str_set:Nx \l__stex_variables_name_str { #1 }
  }
  \prop_clear:N \l_tmpa_prop
  \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}

  \seq_set_from_clist:Nn \l_tmpa_seq {#3}
  \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
    \msg_error:nnxx{stex}{error/seqlength}
      {\int_use:N \l__stex_variables_args_int}
      {\seq_count:N \l_tmpa_seq}
  }
  \seq_set_from_clist:Nn \l_tmpb_seq {#4}
  \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
    \msg_error:nnxx{stex}{error/seqlength}
      {\int_use:N \l__stex_variables_args_int}
      {\seq_count:N \l_tmpb_seq}
  }
  \prop_put:Nnn \l_tmpa_prop {starts} {#3}
  \prop_put:Nnn \l_tmpa_prop {ends} {#4}

  \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
    \cs_set:Npn  {\int_use:N \l__stex_variables_args_int} { #5 }

  \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
  \int_step_inline:nn \l__stex_variables_args_int {
    \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
  }
  \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
  \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
  \tl_if_empty:NF \l__stex_variables_mid_tl {
    \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
    \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
  }
  \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
  \int_step_inline:nn \l__stex_variables_args_int {
    \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
  }
  \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
  \tl_put_right:No \l_tmpa_tl \l_tmpb_tl


  \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl

  \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}

  \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}

  \int_step_inline:nn \l__stex_variables_args_int {
    \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
      \_stex_term_math_arg:nnn{i##1}{0}{\exp_not:n{####}##1}
    }}
  }

  \tl_set:Nx \l_tmpa_tl {
    \_stex_term_math_oma:nnnn { varseq://\l__stex_variables_name_str}{}{0}{
      \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
    }
  }

  \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \stex_symbol_after_invokation_tl} }

  \exp_args:Nno \use:nn {
  \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
    \cs_set:Npn {\int_use:N \l__stex_variables_args_int}}{\l_tmpa_tl}

  \stex_debug:nn{sequences}{New~Sequence:~
    \expandafter\meaning\csname stex_varseq_\l__stex_variables_name_str _cs\endcsname\\~\\
    \prop_to_keyval:N \l_tmpa_prop
  }
  \stex_if_do_html:T{\stex_annotate_invisible:nnn{varseq}{\l__stex_variables_name_str}{
    \tl_if_empty:NF \l__stex_variables_type_tl {
      \stex_annotate:nnn {type}{}{$\seqtype\l__stex_variables_type_tl$}
    }
    \stex_annotate:nnn {args}{\int_use:N \l__stex_variables_args_int}{}
    \str_if_empty:NF \l__stex_variables_bind_str {
      \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{}
    }
  }}

  \prop_set_eq:cN {stex_varseq_\l__stex_variables_name_str _prop}\l_tmpa_prop
  \ignorespacesandpars
}

⟨/package⟩
```

Chapter 30

sTeX -Terms Implementation

```
⟨*package⟩

%%%%%%%%%%%%%   terms.dtx   %%%%%%%%%%%%%

⟨@@=stex_terms⟩
```

Warnings and error messages

```
\msg_new:nnn{stex}{error/nonotation}{
  Symbol~#1~invoked,~but~has~no~notation#2!
}
\msg_new:nnn{stex}{error/notationarg}{
  Error~in~parsing~notation~#1
}
\msg_new:nnn{stex}{error/noop}{
  Symbol~#1~has~no~operator~notation~for~notation~#2
}
\msg_new:nnn{stex}{error/notallowed}{
  Symbol~invokation~#1~not~allowed~in~notation~component~of~#2
}
\msg_new:nnn{stex}{error/doubleargument}{
  Argument~#1~of~symbol~#2~already~assigned
}
\msg_new:nnn{stex}{error/overarity}{
  Argument~#1~invalid~for~symbol~#2~with~arity~#3
}

```

30.1 Symbol Invocations

\stex_invoke_symbol:n Invokes a semantic macro

```


\bool_new:N \l_stex_allow_semantic_bool
\bool_set_true:N \l_stex_allow_semantic_bool

```

```
\cs_new_protected:Nn \stex_invoke_symbol:n {
  \bool_if:NTF \l_stex_allow_semantic_bool {
    \str_if_eq:eeF {
      \prop_item:cn {
        l_stex_symdecl_#1_prop
      }{ deprecate }
    }{}{
      \msg_warning:nnxx{stex}{warning/deprecated}{
        Symbol~#1
      }{
        \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
      }
    }
    \if_mode_math:
      \exp_after:wN \__stex_terms_invoke_math:n
    \else:
      \exp_after:wN \__stex_terms_invoke_text:n
    \fi: { #1 }
  }{
    \msg_error:nnxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
  }
}

\cs_new_protected:Nn \__stex_terms_invoke_text:n {
  \peek_charcode_remove:NTF ! {
    \__stex_terms_invoke_op_custom:nn {#1}
  }{
    \__stex_terms_invoke_custom:nn {#1}
  }
}

\cs_new_protected:Nn \__stex_terms_invoke_math:n {
  \peek_charcode_remove:NTF ! {
    % operator
    \peek_charcode_remove:NTF * {
      % custom op
      \__stex_terms_invoke_op_custom:nn {#1}
    }{
      % op notation
      \peek_charcode:NTF [ {
        \__stex_terms_invoke_op_notation:nw {#1}
      }{
        \__stex_terms_invoke_op_notation:nw {#1}[]
      }
    }
  }{
    \peek_charcode_remove:NTF * {
      \__stex_terms_invoke_custom:nn {#1}
      % custom
    }{
      % normal
      \peek_charcode:NTF [ {
        \__stex_terms_invoke_notation:nw {#1}
      }{
        \__stex_terms_invoke_notation:nw {#1}[]
      }
    }
  }
}


\cs_new_protected:Nn \__stex_terms_invoke_op_custom:nn {
  \exp_args:Nnx \use:nn {
    \def\comp{\_comp}
    \str_set:Nn \l_stex_current_symbol_str { #1 }
    \bool_set_false:N \l_stex_allow_semantic_bool
    \_stex_term_oms:nnn {#1}{#1 \c_hash_str CUSTOM-}{
      \comp{ #2 }
    }
  }{
    \_stex_reset:N \comp
    \_stex_reset:N \l_stex_current_symbol_str
    \bool_set_true:N \l_stex_allow_semantic_bool
  }
}

\keys_define:nn { stex / terms } {
  lang    .tl_set_x:N = \l_stex_notation_lang_str ,
  variant .tl_set_x:N = \l_stex_notation_variant_str ,
  unknown .code:n     = \str_set:Nx
      \l_stex_notation_variant_str \l_keys_key_str
}

\cs_new_protected:Nn \__stex_terms_args:n {
  \str_clear:N \l_stex_notation_lang_str
  \str_clear:N \l_stex_notation_variant_str

  \keys_set:nn { stex / terms } { #1 }
}

\cs_new_protected:Nn \stex_find_notation:nn {
  \__stex_terms_args:n { #2 }
  \seq_if_empty:cTF {
    l_stex_symdecl_ #1 _notations
  } {
    \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
  } {
    \bool_lazy_all:nTF {
      {\str_if_empty_p:N \l_stex_notation_variant_str}
      {\str_if_empty_p:N \l_stex_notation_lang_str}
    }{
      \seq_get_left:cN {l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
    }{
      \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
        \l_stex_notation_variant_str \c_hash_str \l_stex_notation_lang_str
      }{
        \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
      }{
        \msg_error:nnxx{stex}{error/nonotation}{#1}{
          ~\l_stex_notation_variant_str \c_hash_str \l_stex_notation_lang_str
        }
      }
    }
  }
}

\cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
  \exp_args:Nnx \use:nn {
    \def\comp{\_comp}
    \str_set:Nn \l_stex_current_symbol_str { #1 }
    \stex_find_notation:nn { #1 }{ #2 }
    \bool_set_false:N \l_stex_allow_semantic_bool
    \cs_if_exist:cTF {
      stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
    }{
      \_stex_term_oms:nnn { #1 }{
        #1 \c_hash_str \l_stex_notation_variant_str
      }{
        \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
      }
    }{
      \int_compare:nNnTF {\prop_item:cn {l_stex_symdecl_#1_prop}{arity}} = 0{
        \cs_if_exist:cTF {
          stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
        }{
          \tl_set:Nx \stex_symbol_after_invokation_tl {
            \_stex_reset:N \comp
            \_stex_reset:N \stex_symbol_after_invokation_tl
            \_stex_reset:N \l_stex_current_symbol_str
            \bool_set_true:N \l_stex_allow_semantic_bool
          }
          \def\comp{\_comp}
          \str_set:Nn \l_stex_current_symbol_str { #1 }
          \bool_set_false:N \l_stex_allow_semantic_bool
          \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
        }{
          \msg_error:nnxx{stex}{error/nonotation}{#1}{
            ~\l_stex_notation_variant_str
          }
        }
      }{
        \msg_error:nnxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
      }
    }
  }{
    \_stex_reset:N \comp
    \_stex_reset:N \l_stex_current_symbol_str
    \bool_set_true:N \l_stex_allow_semantic_bool
  }
}

\cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {
  \stex_find_notation:nn { #1 }{ #2 }
  \cs_if_exist:cTF {
    stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
  }{
    \tl_set:Nx \stex_symbol_after_invokation_tl {
      \_stex_reset:N \comp
      \_stex_reset:N \stex_symbol_after_invokation_tl
      \_stex_reset:N \l_stex_current_symbol_str
      \bool_set_true:N \l_stex_allow_semantic_bool
    }
    \def\comp{\_comp}
    \str_set:Nn \l_stex_current_symbol_str { #1 }
    \bool_set_false:N \l_stex_allow_semantic_bool
    \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
  }{
    \msg_error:nnxx{stex}{error/nonotation}{#1}{
      ~\l_stex_notation_variant_str
    }
  }
}

\prop_new:N \l__stex_terms_custom_args_prop

\cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
  \exp_args:Nnx \use:nn {
    \bool_set_false:N \l_stex_allow_semantic_bool
    \def\comp{\_comp}
    \str_set:Nn \l_stex_current_symbol_str { #1 }
    \prop_clear:N \l__stex_terms_custom_args_prop
    \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
    \prop_get:cnN {
      l_stex_symdecl_#1 _prop
    }{ args } \l_tmpa_str
    \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
    \tl_set:Nn \arg { \__stex_terms_arg: }
    \str_if_empty:NTF \l_tmpa_str {
      \_stex_term_oms:nnn {#1}{#1\c_hash_str CUSTOM-}{#2}
    }{
      \str_if_in:NnTF \l_tmpa_str b {
        \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
      }{
        \str_if_in:NnTF \l_tmpa_str B {
          \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
        }{
          \_stex_term_oma:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
        }
      }
    }
    % TODO check that all arguments exist
  }{
    \_stex_reset:N \l_stex_current_symbol_str
    \_stex_reset:N \arg
    \_stex_reset:N \comp
    \_stex_reset:N \l__stex_terms_custom_args_prop
    \bool_set_true:N \l_stex_allow_semantic_bool
  }
}

\NewDocumentCommand \__stex_terms_arg: { s O{} m}{
  \tl_if_empty:nTF {#2}{
    \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
    \bool_set_true:N \l_tmpa_bool
    \bool_do_while:Nn \l_tmpa_bool {
      \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int
        \int_incr:N \l_tmpa_int
      }{
        \bool_set_false:N \l_tmpa_bool
      }
    }
  }{
    \int_set:Nn \l_tmpa_int { #2 }
  }
  \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
  \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
    \msg_error:nnxxx{stex}{error/overarity}
      {\int_use:N \l_tmpa_int}
      {\l_stex_current_symbol_str}
      {\str_count:N \l_tmpa_str}
  }
  \str_set:Nx \l_tmpa_str {\str_item:Nn \l_tmpa_str \l_tmpa_int}
  \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
    \bool_lazy_any:nF {
      {\str_if_eq_p:Vn \l_tmpa_str {a}}
      {\str_if_eq_p:Vn \l_tmpa_str {B}}
    }{
      \msg_error:nnxx{stex}{error/doubleargument}
        {\int_use:N \l_tmpa_int}
        {\l_stex_current_symbol_str}
    }
  }
  \exp_args:NNx \prop_put:Nnn \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {#3}
  \bool_set_true:N \l_stex_allow_semantic_bool
  \IfBooleanTF#1{
    \stex_annotate_invisible:n { %TODO
      \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{#3}
    }
  }{ %TODO
    \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{#3}
  }
  \bool_set_false:N \l_stex_allow_semantic_bool
}


\cs_new_protected:Nn \_stex_term_arg:nn {
  \bool_set_true:N \l_stex_allow_semantic_bool
  \stex_annotate:nnn{ arg }{ #1 }{ #2 }
  \bool_set_false:N \l_stex_allow_semantic_bool
}
```

```
\cs_new_protected:Nn \_stex_term_math_arg:nnn {
  \exp_args:Nnx \use:nn
    { \int_set:Nn \l__stex_terms_downprec { #2 }
        \_stex_term_arg:nn { #1 }{ #3 }
    }
    { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
}
```

(*End definition for* `\stex_invoke_symbol:n`*. This function is documented on page 62.*)

_stex_term_math_assoc_arg:nnnn

```
\cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
  \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
  \tl_set:Nn \l_tmpb_tl {\_stex_term_math_arg:nnn{#1}{#2}}
  \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 }}{
    \expandafter\if\expandafter\relax\noexpand#3
      \expandafter\__stex_terms_math_assoc_arg_maybe_sequence:N\expandafter#3
    \else\expandafter\__stex_terms_math_assoc_arg_simple:nn
    \expandafter{\expandafter}\expandafter#3\fi
  }{
    \__stex_terms_math_assoc_arg_simple:nn{#1}{#3}
  }
}

\cs_new_protected:Nn \__stex_terms_math_assoc_arg_maybe_sequence:N {
  \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
  \str_if_empty:NTF \l_tmpa_str {
    \exp_args:Nx \cs_if_eq:NNTF {
      \tl_head:N #1
    } \stex_invoke_sequence:n {
      \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
      \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
      \tl_set:Nx \l_tmpa_tl {\prop_item:cn {stex_varseq_\l_tmpa_str _prop}{notation}}
      \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
      \tl_set:Nx \l_tmpa_tl {{\exp_not:N \exp_not:n{
        \exp_not:n{\exp_args:Nnx \use:nn} {
          \exp_not:n {
            \def\comp{\_varcomp}
            \str_set:Nn \l_stex_current_symbol_str
          } {varseq://\l_tmpa_str}
          \exp_not:n{ ##1 }
        }{
          \exp_not:n {
            \_stex_reset:N \comp
            \_stex_reset:N \l_stex_current_symbol_str
          }
        }
      }}}
      \exp_args:Nno \use:nn {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
      \seq_reverse:N \l_tmpa_seq
      \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
      \seq_map_inline:Nn \l_tmpa_seq {
        \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
          \exp_args:Nno
          \l_tmpa_cs { ##1 } \l_tmpa_tl
        }
      }
      \tl_set:Nx \l_tmpa_tl {
        \_stex_term_omv:nn {varseq://\l_tmpa_str}{
          \exp_args:No \exp_not:n \l_tmpa_tl
        }
      }
      \exp_args:No\l_tmpb_tl\l_tmpa_tl
    }{
      \__stex_terms_math_assoc_arg_simple:nn{} { #1 }
    }
  } {
    \__stex_terms_math_assoc_arg_simple:nn{} { #1 }
  }

}

\cs_new_protected:Nn \__stex_terms_math_assoc_arg_simple:nn {
  \clist_set:Nn \l_tmpa_clist{ #2 }
  \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
    \tl_set:Nn \l_tmpa_tl { #2 }
  }{
    \clist_reverse:N \l_tmpa_clist
    \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
    \tl_set:Nx \l_tmpa_tl { \_stex_term_arg:nn{A#1}{
      \exp_args:No \exp_not:n \l_tmpa_tl
    }}
    \clist_map_inline:Nn \l_tmpa_clist {
      \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
        \exp_args:Nno
        \l_tmpa_cs { \_stex_term_arg:nn{A#1}{##1} } \l_tmpa_tl
      }
    }
  }
  \exp_args:No\l_tmpb_tl\l_tmpa_tl
}
```

(*End definition for* `\_stex_term_math_assoc_arg:nnnn`*. This function is documented on page 62.*)

30.2 Terms

Precedences:

`\infprec`
`\neginfprec`
`\l__stex_terms_downprec`

```
\tl_const:Nx \infprec {\int_use:N \c_max_int}
\tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
\int_new:N \l__stex_terms_downprec
\int_set_eq:NN \l__stex_terms_downprec \infprec
```

(*End definition for* `\infprec`*,* `\neginfprec`*, and* `\l__stex_terms_downprec`*. These variables are documented on page 63.*)

Bracketing:

\l__stex_terms_left_bracket_str
\l__stex_terms_right_bracket_str

```
\tl_set:Nn \l__stex_terms_left_bracket_str (
\tl_set:Nn \l__stex_terms_right_bracket_str )
```

(*End definition for* `\l__stex_terms_left_bracket_str` *and* `\l__stex_terms_right_bracket_str`*.*)

__stex_terms_maybe_brackets:nn

Compares precedences and insert brackets accordingly

```
\cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
  \bool_if:NTF \l__stex_terms_brackets_done_bool {
    \bool_set_false:N \l__stex_terms_brackets_done_bool
    #2
  } {
    \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
      \bool_if:NTF \l_stex_inparray_bool { #2 }{
        \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
        \dobrackets { #2 }
      }
    }{ #2 }
  }
}
```

(*End definition for* `\__stex_terms_maybe_brackets:nn`*.*)

\dobrackets

```
\bool_new:N \l__stex_terms_brackets_done_bool
%\RequirePackage{scalerel}
\cs_new_protected:Npn \dobrackets #1 {
  %\ThisStyle{\if D\m@switch
  %    \exp_args:Nnx \use:nn
  %    { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
  %    { \exp_not:N\right\l__stex_terms_right_bracket_str }
  %  \else
      \exp_args:Nnx \use:nn
      {
        \bool_set_true:N \l__stex_terms_brackets_done_bool
        \int_set:Nn \l__stex_terms_downprec \infprec
        \l__stex_terms_left_bracket_str
        #1
      }
      {
        \bool_set_false:N \l__stex_terms_brackets_done_bool
        \l__stex_terms_right_bracket_str
        \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
      }
  %\fi}
}
```

(*End definition for* `\dobrackets`*. This function is documented on page 63.*)

\withbrackets

```
\cs_new_protected:Npn \withbrackets #1 #2 #3 {
  \exp_args:Nnx \use:nn
  {
    \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
    \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
    #3
  }
  {
    \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
      {\l__stex_terms_left_bracket_str}
    \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
      {\l__stex_terms_right_bracket_str}
  }
}
```

(*End definition for* `\withbrackets`*. This function is documented on page 63.*)

`\STEXinvisible`

```
\cs_new_protected:Npn \STEXinvisible #1 {
  \stex_annotate_invisible:n { #1 }
}
```

(*End definition for* `\STEXinvisible`*. This function is documented on page 63.*)

OMDoc terms:

`\_stex_term_math_oms:nnnn`

```
\cs_new_protected:Nn \_stex_term_oms:nnn {
  \stex_annotate:nnn{ OMID }{ #2 }{
    \stex_highlight_term:nn { #1 } { #3 }
  }
}

\cs_new_protected:Nn \_stex_term_math_oms:nnnn {
  \__stex_terms_maybe_brackets:nn { #3 }{
    \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
  }
}
```

(*End definition for* `\_stex_term_math_oms:nnnn`*. This function is documented on page 62.*)

`\_stex_term_math_omv:nn`

```
\cs_new_protected:Nn \_stex_term_omv:nn {
  \stex_annotate:nnn{ OMV }{ #1 }{
    \stex_highlight_term:nn { #1 } { #2 }
  }
}
```

(*End definition for* `\_stex_term_math_omv:nn`*. This function is documented on page* **??***.)*

`\_stex_term_math_oma:nnnn`

```
\cs_new_protected:Nn \_stex_term_oma:nnn {
  \stex_annotate:nnn{ OMA }{ #2 }{
    \stex_highlight_term:nn { #1 } { #3 }
  }
}

\cs_new_protected:Nn \_stex_term_math_oma:nnnn {
  \__stex_terms_maybe_brackets:nn { #3 }{
    \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
  }
}
```

(*End definition for* `\_stex_term_math_oma:nnnn`. *This function is documented on page 62.*)

`\_stex_term_math_omb:nnnn`

```
\cs_new_protected:Nn \_stex_term_ombind:nnn {
  \stex_annotate:nnn{ OMBIND }{ #2 }{
    \stex_highlight_term:nn { #1 } { #3 }
  }
}

\cs_new_protected:Nn \_stex_term_math_omb:nnnn {
  \__stex_terms_maybe_brackets:nn { #3 }{
    \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
  }
}
```

(*End definition for* `\_stex_term_math_omb:nnnn`. *This function is documented on page 62.*)

`\symref`
`\symname`

```
\cs_new:Nn \stex_capitalize:n { \uppercase{#1} }

\keys_define:nn { stex / symname } {
  pre     .tl_set_x:N    = \l__stex_terms_pre_tl ,
  post    .tl_set_x:N    = \l__stex_terms_post_tl ,
  root    .tl_set_x:N    = \l__stex_terms_root_tl
}

\cs_new_protected:Nn \stex_symname_args:n {
  \tl_clear:N \l__stex_terms_post_tl
  \tl_clear:N \l__stex_terms_pre_tl
  \tl_clear:N \l__stex_terms_root_str
  \keys_set:nn { stex / symname } { #1 }
}

\NewDocumentCommand \symref { m m }{
  \let\compemph_uri_prev:\compemph@uri
  \let\compemph@uri\symrefemph@uri
  \STEXsymbol{#1}!{ #2 }
  \let\compemph@uri\compemph_uri_prev:
}

\NewDocumentCommand \synonym { O{} m m}{
  \stex_symname_args:n { #1 }
  \let\compemph_uri_prev:\compemph@uri
  \let\compemph@uri\symrefemph@uri
  % TODO
  \STEXsymbol{#2}!{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
  \let\compemph@uri\compemph_uri_prev:
}

\NewDocumentCommand \symname { O{} m }{
  \stex_symname_args:n { #1 }
  \stex_get_symbol:n { #2 }
  \str_set:Nx \l_tmpa_str {
    \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
  }
  \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}

  \let\compemph_uri_prev:\compemph@uri
  \let\compemph@uri\symrefemph@uri
  \exp_args:NNx \use:nn
  \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!{
    \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
   } }
  \let\compemph@uri\compemph_uri_prev:
}

\NewDocumentCommand \Symname { O{} m }{
  \stex_symname_args:n { #1 }
  \stex_get_symbol:n { #2 }
  \str_set:Nx \l_tmpa_str {
    \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
  }
  \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
  \let\compemph_uri_prev:\compemph@uri
  \let\compemph@uri\symrefemph@uri
  \exp_args:NNx \use:nn
  \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!{
    \exp_after:wN \stex_capitalize:n \l_tmpa_str
      \l__stex_terms_post_tl
   } }
  \let\compemph@uri\compemph_uri_prev:
}
```

(*End definition for* `\symref` *and* `\symname`. *These functions are documented on page 62.*)

30.3 Notation Components

```
⟨@@=stex_notationcomps⟩
```

`\stex_highlight_term:nn`

```
\cs_new_protected:Nn \stex_highlight_term:nn {
  #2
}

\cs_new_protected:Nn \stex_unhighlight_term:n {
%  \latexml_if:TF {
%    #1
%  } {
%    \rustex_if:TF {
%      #1
%    } {
       #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
%    }
%  }
}
```

(*End definition for* `\stex_highlight_term:nn`. *This function is documented on page 63.*)

\comp
\compemph@uri
\compemph
\defemph
\defemph@uri
\symrefemph
\symrefemph@uri
\varemph
\varemph@uri

```
\cs_new_protected:Npn \_comp #1 {
  \str_if_empty:NF \l_stex_current_symbol_str {
    \rustex_if:TF {
      \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
    }{
      \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
    }
  }
}

\cs_new_protected:Npn \_varcomp #1 {
  \str_if_empty:NF \l_stex_current_symbol_str {
    \rustex_if:TF {
      \stex_annotate:nnn { varcomp }{ \l_stex_current_symbol_str }{ #1 }
    }{
      \exp_args:Nnx \varemph@uri { #1 } { \l_stex_current_symbol_str }
    }
  }
}

\def\comp{\_comp}

\cs_new_protected:Npn \compemph@uri #1 #2 {
    \compemph{ #1 }
}


\cs_new_protected:Npn \compemph #1 {
    #1
}

\cs_new_protected:Npn \defemph@uri #1 #2 {
    \defemph{#1}
}

\cs_new_protected:Npn \defemph #1 {
    \textbf{#1}
}

\cs_new_protected:Npn \symrefemph@uri #1 #2 {
    \symrefemph{#1}
}

\cs_new_protected:Npn \symrefemph #1 {
    \textbf{#1}
}

\cs_new_protected:Npn \varemph@uri #1 #2 {
    \varemph{#1}
}

```

```
\cs_new_protected:Npn \varemph #1 {
    #1
}
```

(*End definition for* `\comp` *and others. These functions are documented on page 63.*)

`\ellipses`

```
\NewDocumentCommand \ellipses {} { \ldots }
```

(*End definition for* `\ellipses`*. This function is documented on page 63.*)

`\parray`
`\prmatrix`
`\parrayline`
`\parraylineh`
`\parraycell`

```
\bool_new:N \l_stex_inparray_bool
\bool_set_false:N \l_stex_inparray_bool
\NewDocumentCommand \parray { m m } {
  \begingroup
  \bool_set_true:N \l_stex_inparray_bool
  \begin{array}{#1}
    #2
  \end{array}
  \endgroup
}

\NewDocumentCommand \prmatrix { m } {
  \begingroup
  \bool_set_true:N \l_stex_inparray_bool
  \begin{matrix}
    #1
  \end{matrix}
  \endgroup
}

\def \maybephline {
  \bool_if:NT \l_stex_inparray_bool {\hline}
}

\def \parrayline #1 #2 {
  #1 #2 \bool_if:NT \l_stex_inparray_bool {\\}
}

\def \pmrow #1 { \parrayline{}{ #1 } }

\def \parraylineh #1 #2 {
  #1 #2 \bool_if:NT \l_stex_inparray_bool {\\\hline}
}

\def \parraycell #1 {
  #1 \bool_if:NT \l_stex_inparray_bool {&}
}
```

(*End definition for* `\parray` *and others. These functions are documented on page* ??.)

30.4 Variables

```
⟨@@=stex_variables⟩
```

\stex_invoke_variable:n Invokes a variable

```
\cs_new_protected:Nn \stex_invoke_variable:n {
  \if_mode_math:
    \exp_after:wN \__stex_variables_invoke_math:n
  \else:
    \exp_after:wN \__stex_variables_invoke_text:n
  \fi: {#1}
}

\cs_new_protected:Nn \__stex_variables_invoke_text:n {
  %TODO
}


\cs_new_protected:Nn \__stex_variables_invoke_math:n {
  \peek_charcode_remove:NTF ! {
    \peek_charcode_remove:NTF ! {
      \peek_charcode:NTF [ {
        \__stex_variables_invoke_op_custom:nw
      }{
        % TODO throw error
      }
    }{
      \__stex_variables_invoke_op:n { #1 }
    }
  }{
    \peek_charcode_remove:NTF * {
      \__stex_variables_invoke_text:n { #1 }
    }{
      \__stex_variables_invoke_math_ii:n { #1 }
    }
  }
}

\cs_new_protected:Nn \__stex_variables_invoke_op:n {
  \cs_if_exist:cTF {
    stex_var_op_notation_ #1 _cs
  }{
    \exp_args:Nnx \use:nn {
      \def\comp{\_varcomp}
      \str_set:Nn \l_stex_current_symbol_str { var://#1 }
      \_stex_term_omv:nn { var://#1 }{
        \use:c{stex_var_op_notation_ #1 _cs }
      }
    }{
      \_stex_reset:N \comp
      \_stex_reset:N \l_stex_current_symbol_str
    }
  }{
    \int_compare:nNnTF {\prop_item:cn {l_stex_variable_#1_prop}{arity}} = 0{
      \__stex_variables_invoke_math_ii:n {#1}
    }{
      \msg_error:nnxx{stex}{error/noop}{variable~#1}{}
    }
  }
}

\cs_new_protected:Npn \__stex_variables_invoke_math_ii:n  #1 {
  \cs_if_exist:cTF {
    stex_var_notation_#1_cs
  }{
    \tl_set:Nx \stex_symbol_after_invokation_tl {
      \_stex_reset:N \comp
      \_stex_reset:N \stex_symbol_after_invokation_tl
      \_stex_reset:N \l_stex_current_symbol_str
      \bool_set_true:N \l_stex_allow_semantic_bool
    }
    \def\comp{\_varcomp}
    \str_set:Nn \l_stex_current_symbol_str { var://#1 }
    \bool_set_false:N \l_stex_allow_semantic_bool
    \use:c{stex_var_notation_#1_cs}
  }{
    \msg_error:nnxx{stex}{error/nonotation}{variable~#1}{s}
  }
}
```

(*End definition for* `\stex_invoke_variable:n`*. This function is documented on page* **??**.)

30.5 Sequences

```
⟨@@=stex_sequences⟩

\cs_new_protected:Nn \stex_invoke_sequence:n {
  \peek_charcode_remove:NTF ! {
    \_stex_term_omv:nn {varseq://#1}{
      \exp_args:Nnx \use:nn {
        \def\comp{\_varcomp}
        \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
        \prop_item:cn{stex_varseq_#1_prop}{notation}
      }{
        \_stex_reset:N \comp
        \_stex_reset:N \l_stex_current_symbol_str
      }
    }
  }{
    \bool_set_false:N \l_stex_allow_semantic_bool
    \def\comp{\_varcomp}
    \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
    \tl_set:Nx \stex_symbol_after_invokation_tl {
      \_stex_reset:N \comp
      \_stex_reset:N \stex_symbol_after_invokation_tl
      \_stex_reset:N \l_stex_current_symbol_str
      \bool_set_true:N \l_stex_allow_semantic_bool
    }
    \use:c { stex_varseq_#1_cs }
  }
}
⟨/package⟩
```

Chapter 31

sTeX -Structural Features Implementation

```
⟨*package⟩

%%%%%%%%%%%%%   features.dtx   %%%%%%%%%%%%%

```

Warnings and error messages

```
\msg_new:nnn{stex}{error/copymodule/notallowed}{
  Symbol~#1~can~not~be~assigned~in~copymodule~#2
}
\msg_new:nnn{stex}{error/interpretmodule/nodefiniens}{
  Symbol~#1~not~assigned~in~interpretmodule~#2
}

\msg_new:nnn{stex}{error/unknownstructure}{
  No~structure~#1~found!
}

\msg_new:nnn{stex}{error/unknownfield}{
  No~field~#1~in~instance~#2~found!\\#3
}

\msg_new:nnn{stex}{error/keyval}{
  Invalid~key=value~pair:#1
}
\msg_new:nnn{stex}{error/instantiate/missing}{
  Assignments~missing~in~instantiate:~#1
}
\msg_new:nnn{stex}{error/incompatible}{
  Incompatible~signature:~#1~(#2)~and~#3~(#4)
}

```

31.1 Imports with modification

```
⟨@@=stex_copymodule⟩
\cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
  \tl_if_head_eq_catcode:nNTF { #1 } \relax {
    \tl_set:Nn \l_tmpa_tl { #1 }
    \__stex_copymodule_get_symbol_from_cs:
  }{
    % argument is a string
    % is it a command name?
    \cs_if_exist:cTF { #1 }{
      \cs_set_eq:Nc \l_tmpa_tl { #1 }
      \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
      \str_if_empty:NTF \l_tmpa_str {
        \exp_args:Nx \cs_if_eq:NNTF {
          \tl_head:N \l_tmpa_tl
        } \stex_invoke_symbol:n {
          \__stex_copymodule_get_symbol_from_cs:n{ #2 }
        }{
          \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
        }
      } {
        \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
      }
    }{
      % argument is not a command name
      \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
      % \l_stex_all_symbols_seq
    }
  }
}

\cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
  \str_set:Nn \l_tmpa_str { #1 }
  \bool_set_false:N \l_tmpa_bool
  \bool_if:NF \l_tmpa_bool {
    \tl_set:Nn \l_tmpa_tl {
      \msg_error:nnn{stex}{error/unknownsymbol}{#1}
    }
    \str_set:Nn \l_tmpa_str { #1 }
    \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
    \seq_map_inline:Nn #2 {
      \str_set:Nn \l_tmpb_str { ##1 }
      \str_if_eq:eeT { \l_tmpa_str } {
        \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
      } {
        \seq_map_break:n {
          \tl_set:Nn \l_tmpa_tl {
            \str_set:Nn \l_stex_get_symbol_uri_str {
              ##1
            }
          }
        }
      }
    }
    \l_tmpa_tl
  }
}

\cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
  \exp_args:NNx \tl_set:Nn \l_tmpa_tl
    { \tl_tail:N \l_tmpa_tl }
  \tl_if_single:NTF \l_tmpa_tl {
    \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
      \exp_after:wN \str_set:Nn \exp_after:wN
        \l_stex_get_symbol_uri_str \l_tmpa_tl
      \__stex_copymodule_get_symbol_check:n { #1 }
    }{
      % TODO
      % tail is not a single group
    }
  }{
    % TODO
    % tail is not a single group
  }
}

\cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
  \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
    \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
      :~\seq_use:Nn #1 {,~}
    }
  }
}

\cs_new_protected:Nn \stex_copymodule_start:nnnn {
  \stex_import_module_uri:nn { #1 } { #2 }
  \str_set:Nx \l_stex_current_copymodule_name_str {#3}
  \stex_import_require_module:nnnn
    { \l_stex_import_ns_str } { \l_stex_import_archive_str }
    { \l_stex_import_path_str } { \l_stex_import_name_str }
  \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
  \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
  \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
  \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
    \seq_map_inline:cn {c_stex_module_##1_constants}{
      \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
        ##1 ? ####1
      }
    }
  }
  \seq_clear:N \l_tmpa_seq
  \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
    name      = \l_stex_current_copymodule_name_str ,
    module    = \l_stex_current_module_str ,
    from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
    includes  = \l_tmpa_seq ,
    fields    = \l_tmpa_seq
  }
  \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
    as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
    \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_se
  \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,
  \stex_if_smsmode:F {
    \begin{stex_annotate_env} {#4} {
      \l_stex_current_module_str?\l_stex_current_copymodule_name_str
    }
    \stex_annotate_invisible:nnn{domain}{\l_stex_import_ns_str ?\l_stex_import_name_str}{}
  }
  %\bool_set_eq:NN \l__stex_copymodule_oldhtml_bool \_stex_html_do_output_bool
  %\bool_set_false:N \_stex_html_do_output_bool
}
\cs_new_protected:Nn \stex_copymodule_end:n {
  \def \l_tmpa_cs ##1 ##2 {#1}
  %\bool_set_eq:NN \_stex_html_do_output_bool \l__stex_copymodule_oldhtml_bool
  \tl_clear:N \l_tmpa_tl
  \tl_clear:N \l_tmpb_tl
  \prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
  \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
    \seq_map_inline:cn {c_stex_module_##1_constants}{
      \tl_clear:N \l_tmpc_tl
      \l_tmpa_cs{##1}{####1}
      \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
        \tl_put_right:Nx \l_tmpa_tl {
          \prop_set_from_keyval:cn {
            l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule
          }{
            \exp_after:wN \prop_to_keyval:N \csname
              l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodu
            \endcsname
          }
          \seq_clear:c {
            l_stex_symdecl_
            \l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_name
            _notations
          }
        }
        \tl_put_right:Nx \l_tmpc_tl {
          \stex_copy_notations:nn {\l_stex_current_module_str ? \use:c{l__stex_copymodule_co
          \stex_if_smsmode:F{\stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_c
        }
        \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \use:c{l__stex_copymodul
        \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
          \tl_put_right:Nx \l_tmpc_tl {
            \stex_if_smsmode:F{\stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymo
          }
          \tl_put_right:Nx \l_tmpa_tl {
            \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
              \stex_invoke_symbol:n {
                \l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_
              }
            }
          }
        }
      }{
        \tl_put_right:Nx \l_tmpc_tl {
          \stex_copy_notations:nn {\l_stex_current_module_str ? \l_stex_current_copymodule_n
        }
        \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
        \prop_put:Nnx \l_tmpa_prop { name }{ \l_stex_current_copymodule_name_str / ####1 }
        \prop_put:Nnx \l_tmpa_prop { module }{ \l_stex_current_module_str }
        \tl_put_right:Nx \l_tmpa_tl {
          \prop_set_from_keyval:cn {
            l_stex_symdecl_\l_stex_current_module_str ? \l_stex_current_copymodule_name_str
          }{
            \prop_to_keyval:N \l_tmpa_prop
          }
          \seq_clear:c {
            l_stex_symdecl_
            \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
            _notations
          }
        }
        \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \l_stex_current_copymodu
        \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
          \tl_put_right:Nx \l_tmpc_tl {
            \stex_if_smsmode:F{\stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymo
          }
          \tl_put_right:Nx \l_tmpa_tl {
            \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
              \stex_invoke_symbol:n {
                \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
              }
            }
          }
        }
      }
      \stex_if_smsmode:F{
        \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
          \tl_put_right:Nx \l_tmpc_tl {
            $\stex_annotate_invisible:nnn{definiens}{}{\exp_after:wN \exp_not:N\csname l__st
          }
        }
        \tl_put_right:Nx \l_tmpb_tl {
          \stex_annotate:nnn{assignment} {##1?####1} { \exp_after:wN \exp_not:n \exp_after:w
        }
      }
    }
  }
  \prop_put:Nno \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
  \tl_put_left:Nx \l_tmpa_tl {
    \prop_set_from_keyval:cn {
      l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _pro
    }{
      \prop_to_keyval:N \l_stex_current_copymodule_prop
    }
  }
  \seq_gput_right:cx{c_stex_module_\l_stex_current_module_str _copymodules}{
    \l_stex_current_module_str?\l_stex_current_copymodule_name_str
  }
  \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl
  \stex_debug:nn{copymodule}{result:\meaning \l_tmpa_tl}
  \exp_args:Nx \stex_do_up_to_module:n {
      \exp_args:No \exp_not:n \l_tmpa_tl
  }
  \stex_debug:nn{copymodule}{output:\meaning \l_tmpb_tl}
  \l_tmpb_tl
  \stex_if_smsmode:F {
    \end{stex_annotate_env}
  }
}

\NewDocumentEnvironment {copymodule} { O{} m m}{
  \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ copymodule }
  \stex_deactivate_macro:Nn \symdecl {module~environments}
  \stex_deactivate_macro:Nn \symdef {module~environments}
  \stex_deactivate_macro:Nn \notation {module~environments}
  \stex_reactivate_macro:N \assign
  \stex_reactivate_macro:N \renamedecl
  \stex_reactivate_macro:N \donotcopy
  \stex_smsmode_do:
}{
  \stex_copymodule_end:n {}
}

\NewDocumentEnvironment {interpretmodule} { O{} m m}{
  \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ interpretmodule }
  \stex_deactivate_macro:Nn \symdecl {module~environments}
  \stex_deactivate_macro:Nn \symdef {module~environments}
  \stex_deactivate_macro:Nn \notation {module~environments}
  \stex_reactivate_macro:N \assign
  \stex_reactivate_macro:N \renamedecl
  \stex_reactivate_macro:N \donotcopy
  \stex_smsmode_do:
}{
  \stex_copymodule_end:n {
    \tl_if_exist:cF {
      l__stex_copymodule_copymodule_##1?##2_def_tl
    }{
      \str_if_eq:eeF {
        \prop_item:cn{
          l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
      }{ true }{
        \msg_error:nnxx{stex}{error/interpretmodule/nodefiniens}{
          ##1?##2
        }{\l_stex_current_copymodule_name_str}
      }
    }
  }
}
```

```

\NewDocumentCommand \donotcopy { m }{
  \str_clear:N \l_stex_import_name_str
  \str_set:Nn \l_tmpa_str { #1 }
  \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
  \seq_map_inline:Nn \l_stex_all_modules_seq {
    \str_set:Nn \l_tmpb_str { ##1 }
    \str_if_eq:eeT { \l_tmpa_str } {
      \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
    } {
      \seq_map_break:n {
        \stex_if_do_html:T {
          \stex_if_smsmode:F {
            \stex_annotate_invisible:nnn{donotcopy}{##1}{
              \stex_annotate:nnn{domain}{##1}{}
            }
          }
        }
        \str_set_eq:NN \l_stex_import_name_str \l_tmpb_str
      }
    }
    \seq_map_inline:cn {c_stex_module_##1_copymodules}{
      \str_set:Nn \l_tmpb_str { ####1 }
      \str_if_eq:eeT { \l_tmpa_str } {
        \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
      } {
        \seq_map_break:n {\seq_map_break:n {
          \stex_if_do_html:T {
            \stex_if_smsmode:F {
              \stex_annotate_invisible:nnn{donotcopy}{####1}{
                \stex_annotate:nnn{domain}{
                  \prop_item:cn {l_stex_copymodule_ ####1 _prop}{module}
                }{}
              }
            }
          }
          \str_set:Nx \l_stex_import_name_str {
            \prop_item:cn {l_stex_copymodule_ ####1 _prop}{module}
          }
        }}
      }
    }
  }
  \str_if_empty:NTF \l_stex_import_name_str {
    % TODO throw error
  }{
    \stex_collect_imports:n {\l_stex_import_name_str }
    \seq_map_inline:Nn \l_stex_collect_imports_seq {
      \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
      \seq_map_inline:cn {c_stex_module_##1_constants}{
        \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? ####1 }
        \bool_lazy_any:nT {
          { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_name_str}}
          { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_macroname_str}}
          { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_def_tl}}
        }{
          % TODO throw error
        }
      }
    }
    \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
    \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_name_str }
    \prop_put:Nno \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
   }
  \stex_smsmode_do:
 }

\NewDocumentCommand \assign { m m }{
  \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
  \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
  \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _def_tl}{#2}
  \stex_smsmode_do:
}

\keys_define:nn { stex / renamedecl } {
  name        .str_set_x:N  = \l_stex_renamedecl_name_str
}
\cs_new_protected:Nn \__stex_copymodule_renamedecl_args:n {
  \str_clear:N \l_stex_renamedecl_name_str
  \keys_set:nn { stex / renamedecl } { #1 }
}

\NewDocumentCommand \renamedecl { O{} m m}{
  \__stex_copymodule_renamedecl_args:n { #1 }
  \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
  \stex_debug:nn{renamedecl}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
  \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
  \str_if_empty:NTF \l_stex_renamedecl_name_str {
    \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
      \l_stex_get_symbol_uri_str
    } }
  } {
    \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex
    \stex_debug:nn{renamedecl}{@~\l_stex_current_module_str ? \l_stex_renamedecl_name_str}
    \prop_set_eq:cc {l_stex_symdecl_
      \l_stex_current_module_str ? \l_stex_renamedecl_name_str
      _prop
    }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
    \seq_set_eq:cc {l_stex_symdecl_
      \l_stex_current_module_str ? \l_stex_renamedecl_name_str
      _notations
    }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
    \prop_put:cnx {l_stex_symdecl_
      \l_stex_current_module_str ? \l_stex_renamedecl_name_str
      _prop
    }{ name }{ \l_stex_renamedecl_name_str }
    \prop_put:cnx {l_stex_symdecl_
      \l_stex_current_module_str ? \l_stex_renamedecl_name_str
```

```
      _prop
    }{ module }{ \l_stex_current_module_str }
    \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
      \l_stex_current_module_str ? \l_stex_renamedecl_name_str
    }
    \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
      \l_stex_current_module_str ? \l_stex_renamedecl_name_str
    } }
  }
  \stex_smsmode_do:
}

\stex_deactivate_macro:Nn \assign {copymodules}
\stex_deactivate_macro:Nn \renamedecl {copymodules}
\stex_deactivate_macro:Nn \donotcopy {copymodules}


\seq_new:N \l_stex_implicit_morphisms_seq
\NewDocumentCommand \implicitmorphism { O{} m m}{
  \stex_import_module_uri:nn { #1 } { #2 }
  \stex_debug:nn{implicits}{
    Implicit~morphism:~
    \l_stex_module_ns_str ? \l__stex_copymodule_name_str
  }
  \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
    \l_stex_module_ns_str ? \l__stex_copymodule_name_str
  }{
    \msg_error:nnn{stex}{error/conflictingmodules}{
      \l_stex_module_ns_str ? \l__stex_copymodule_name_str
    }
  }

  % TODO



  \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
    \l_stex_module_ns_str ? \l__stex_copymodule_name_str
  }
}

```

31.2 The feature environment

structural@feature

```
⟨@@=stex_features⟩

\NewDocumentEnvironment{structural_feature_module}{ m m m }{
  \stex_if_in_module:F {
    \msg_set:nnn{stex}{error/nomodule}{
      Structural~Feature~has~to~occur~in~a~module:\\
      Feature~#2~of~type~#1\\
      In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
    }
    \msg_error:nn{stex}{error/nomodule}
  }

  \str_set_eq:NN \l_tmpa_str \l_stex_current_module_str

  \stex_module_setup:nn{meta=NONE}{#2 - #1}

  \stex_if_smsmode:F {
    \begin{stex_annotate_env}{ feature:#1 }{\l_tmpa_str ? #2 - #1}
      \stex_annotate_invisible:nnn{header}{}{ #3 }
  }
}{
  \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
  \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
  \stex_debug:nn{features}{
    Feature: \l_stex_last_feature_str
  }
  \stex_if_smsmode:F {
    \end{stex_annotate_env}
  }
}
```

31.3 Structure

structure

```
⟨@@=stex_structures⟩
\cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
  \prop_if_exist:cF {c_stex_module_\l_stex_current_module_str _structures}{
    \prop_new:c {c_stex_module_\l_stex_current_module_str _structures}
  }
  \prop_gput:cxx{c_stex_module_\l_stex_current_module_str _structures}
    {#1}{#2}
}

\keys_define:nn { stex / features / structure } {
  name         .str_set_x:N  = \l__stex_structures_name_str ,
}

\cs_new_protected:Nn \__stex_structures_structure_args:n {
  \str_clear:N \l__stex_structures_name_str
  \keys_set:nn { stex / features / structure } { #1 }
}

\NewDocumentEnvironment{mathstructure}{m O{}}{
  \__stex_structures_structure_args:n { #2 }
  \str_if_empty:NT \l__stex_structures_name_str {
    \str_set:Nx \l__stex_structures_name_str { #1 }
  }
  \stex_suppress_html:n {
    \exp_args:Nx \stex_symdecl_do:nn {
      name = \l__stex_structures_name_str ,
      def  = {\STEXsymbol{module-type}{
        \_stex_term_math_oms:nnnn {
          \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
            { ns } ?
            \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
              { name } / \l__stex_structures_name_str - structure
         }{}{0}{}
      }}
    }{ #1 }
  }
  \exp_args:Nnnx
  \begin{structural_feature_module}{ structure }
    { \l__stex_structures_name_str }{}
  \stex_smsmode_do:
}{
  \end{structural_feature_module}
  \_stex_reset_up_to_module:
  \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
  \seq_clear:N \l_tmpa_seq
  \seq_map_inline:Nn \l_stex_collect_imports_seq {
    \seq_map_inline:cn{c_stex_module_##1_constants}{
      \seq_put_right:Nn \l_tmpa_seq { ##1 ? ####1 }
    }
  }
  \exp_args:Nnno
  \prop_gput:cnn {c_stex_module_ \l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
  \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
  \stex_add_structure_to_current_module:nn
    \l__stex_structures_name_str
    \l_stex_last_feature_str
  \exp_args:Nx
  \stex_add_to_current_module:n {
    \tl_set:cn { #1 }{
      \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structure
    }
  }
  \exp_args:Nx
  \stex_do_up_to_module:n {
    \tl_set:cn { #1 }{
      \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structure
    }
  }
}

\cs_new:Nn \stex_invoke_structure:nn {
  \stex_invoke_symbol:n { #1?#2 }
}

\cs_new_protected:Nn \stex_get_structure:n {
  \tl_if_head_eq_catcode:nNTF { #1 } \relax {
    \tl_set:Nn \l_tmpa_tl { #1 }
    \__stex_structures_get_from_cs:
  }{
    \cs_if_exist:cTF { #1 }{
      \cs_set_eq:Nc \l_tmpa_cs { #1 }
      \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
      \str_if_empty:NTF \l_tmpa_str {
        \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs} \stex_invoke_structure:nn {
          \__stex_structures_get_from_cs:
        }{
          \__stex_structures_get_from_string:n { #1 }
        }
      }{
        \__stex_structures_get_from_string:n { #1 }
      }
    }{
      \__stex_structures_get_from_string:n { #1 }
    }
  }
}

\cs_new_protected:Nn \__stex_structures_get_from_cs: {
  \exp_args:NNx \tl_set:Nn \l_tmpa_tl
    { \tl_tail:N \l_tmpa_tl }
  \str_set:Nx \l_tmpa_str {
    \exp_after:wN \use_i:nn \l_tmpa_tl
  }
  \str_set:Nx \l_tmpb_str {
    \exp_after:wN \use_ii:nn \l_tmpa_tl
  }
  \str_set:Nx \l_stex_get_structure_str {
    \l_tmpa_str ? \l_tmpb_str
  }
  \str_set:Nx \l_stex_get_structure_module_str {
    \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
  }
}

\cs_new_protected:Nn \__stex_structures_get_from_string:n {
  \tl_set:Nn \l_tmpa_tl {
    \msg_error:nnn{stex}{error/unknownstructure}{#1}
  }
  \str_set:Nn \l_tmpa_str { #1 }
  \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }

  \seq_map_inline:Nn \l_stex_all_modules_seq {
    \prop_if_exist:cT {c_stex_module_##1_structures} {
      \prop_map_inline:cn {c_stex_module_##1_structures} {
        \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
          \prop_map_break:n{\seq_map_break:n{
            \tl_set:Nn \l_tmpa_tl {
              \str_set:Nn \l_stex_get_structure_str {##1?####1}
              \str_set:Nn \l_stex_get_structure_module_str {####2}
            }
          }}
        }
      }
    }
  }
  \l_tmpa_tl
}
```

\instantiate

```

\keys_define:nn { stex / instantiate } {
  name        .str_set_x:N  = \l__stex_structures_name_str
}
\cs_new_protected:Nn \__stex_structures_instantiate_args:n {
  \str_clear:N \l__stex_structures_name_str
  \keys_set:nn { stex / instantiate } { #1 }
}

\NewDocumentCommand \instantiate {m O{} m m m}{
  \begingroup
    \stex_get_structure:n {#4}
    \__stex_structures_instantiate_args:n { #2 }
    \str_if_empty:NT \l__stex_structures_name_str {
      \str_set:Nn \l__stex_structures_name_str { #1 }
    }
    \seq_clear:N \l__stex_structures_fields_seq
    \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
    \seq_map_inline:Nn \l_stex_collect_imports_seq {
      \seq_map_inline:cn {c_stex_module_##1_constants}{
        \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
      }
    }
    \seq_set_split:Nnn \l_tmpa_seq , {#3}
    \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
    \prop_clear:N \l_tmpa_prop
    \seq_map_inline:Nn \l_tmpa_seq {
      \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
      \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
        \msg_error:nnn{stex}{error/keyval}{##1}
      }
      \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_structur
      \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
      \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_uri
      \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}
      \exp_args:Nxx \str_if_eq:nnF
        {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
        {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
        \msg_error:nnxxxx{stex}{error/incompatible}
          {\l__stex_structures_dom_str}
          {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
          {\l_stex_get_symbol_uri_str}
          {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
      }
      \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str
    }
    \seq_if_empty:NF \l__stex_structures_fields_seq {
      \msg_error:nnx{stex}{error/instantiate/missing}{\seq_use:Nn\l__stex_structures_fields_
    }
    \exp_args:Nx
    \stex_add_to_current_module:n {
      \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
        domain = \l_stex_get_structure_module_str ,
        \prop_to_keyval:N \l_tmpa_prop
      }
      \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
    }
    \exp_args:Nx
    \stex_do_up_to_module:n {
      \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
        domain = \l_stex_get_structure_module_str ,
        \prop_to_keyval:N \l_tmpa_prop
      }
      \tl_set:cn{ #1 }{\stex_invoke_instance:n{\l_stex_current_module_str?\l__stex_structure
    }
    \stex_debug:nn{instantiate}{
      Instance~\l_stex_current_module_str?\l__stex_structures_name_str \\
      \prop_to_keyval:N \l_tmpa_prop
    }
    \exp_args:Nxx \stex_symdecl_do:nn {
      type={\STEXsymbol{module-type}{
        \_stex_term_math_oms:nnnn {
          \l_stex_get_structure_module_str
        }{}{0}{}
      }}
    }{\l__stex_structures_name_str}
    \exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}
  \endgroup
  \stex_smsmode_do:\ignorespacesandpars
}

\cs_new_protected:Nn \stex_symbol_or_var:n {
  \cs_if_exist:cTF{#1}{
    \cs_set_eq:Nc \l_tmpa_tl { #1 }
    \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
    \str_if_empty:NTF \l_tmpa_str {
      \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
        \stex_invoke_variable:n {
          \bool_set_true:N \l_stex_symbol_or_var_bool
          \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
          \str_set:Nx \l_stex_get_symbol_uri_str {
            \exp_after:wN \use:n \l_tmpa_tl
          }
        }{
          \bool_set_false:N \l_stex_symbol_or_var_bool
          \stex_get_symbol:n{#1}
        }
    }{
      \__stex_structures_symbolorvar_from_string:n{ #1 }
    }
  }{
    \__stex_structures_symbolorvar_from_string:n{ #1 }
  }
}
```

```

\cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
  \prop_if_exist:cTF {l_stex_variable_#1 _prop}{
    \bool_set_true:N \l_stex_symbol_or_var_bool
    \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
  }{
    \bool_set_false:N \l_stex_symbol_or_var_bool
    \stex_get_symbol:n{#1}
  }
}

\keys_define:nn { stex / varinstantiate } {
  name        .str_set_x:N  = \l__stex_structures_name_str,
  bind        .choices:nn   =
      {forall,exists}
      {\str_set:Nx \l__stex_structures_bind_str {\l_keys_choice_tl}}

}
\cs_new_protected:Nn \__stex_structures_varinstantiate_args:n {
  \str_clear:N \l__stex_structures_name_str
  \str_clear:N \l__stex_structures_bind_str
  \keys_set:nn { stex / varinstantiate } { #1 }
}

\NewDocumentCommand \varinstantiate {m O{} m m m}{
  \begingroup
    \stex_get_structure:n {#4}
    \__stex_structures_varinstantiate_args:n { #2 }
    \str_if_empty:NT \l__stex_structures_name_str {
      \str_set:Nn \l__stex_structures_name_str { #1 }
    }
    \stex_if_do_html:TF{
      \stex_annotate:nnn{varinstance}{\l__stex_structures_name_str}
    }{\use:n}
    {
      \stex_if_do_html:T{
        \stex_annotate:nnn{domain}{\l_stex_get_structure_module_str}{}
      }
      \seq_clear:N \l__stex_structures_fields_seq
      \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
      \seq_map_inline:Nn \l_stex_collect_imports_seq {
        \seq_map_inline:cn {c_stex_module_##1_constants}{
          \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
        }
      }
      \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
      \prop_clear:N \l_tmpa_prop
      \tl_if_empty:nF {#3} {
        \seq_set_split:Nnn \l_tmpa_seq , {#3}
        \seq_map_inline:Nn \l_tmpa_seq {
          \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
          \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
            \msg_error:nnn{stex}{error/keyval}{##1}
          }
          \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_stru
          \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
          \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol
          \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
          \stex_if_do_html:T{
            \stex_annotate:nnn{assign}{\l__stex_structures_dom_str,\l_stex_get_symbol_uri_st
          }
          \bool_if:NTF \l_stex_symbol_or_var_bool {
            \exp_args:Nxx \str_if_eq:nnF
              {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
              {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}{
              \msg_error:nnxxxx{stex}{error/incompatible}
                {\l__stex_structures_dom_str}
                {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
                {\l_stex_get_symbol_uri_str}
                {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}
            }
            \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n
          }{
            \exp_args:Nxx \str_if_eq:nnF
              {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
              {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
              \msg_error:nnxxxx{stex}{error/incompatible}
                {\l__stex_structures_dom_str}
                {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
                {\l_stex_get_symbol_uri_str}
                {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
            }
            \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n {
          }
        }
      }
      \tl_gclear:N \g__stex_structures_aftergroup_tl
      \seq_map_inline:Nn \l__stex_structures_fields_seq {
        \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdec
        \stex_debug:nn{varinstantiate}{Field~\l_tmpa_str :~##1}
        \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
          \stex_find_notation:nn{##1}{}
          \cs_gset_eq:cc{g__stex_structures_tmpa_\l_tmpa_str _cs}
            {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
          \stex_debug:nn{varinstantiate}{Notation:~\cs_meaning:c{g__stex_structures_tmpa_\l_
          \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
            \cs_gset_eq:cc {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
              {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
              \stex_debug:nn{varinstantiate}{Operator~Notation:~\cs_meaning:c{g__stex_struct
          }
        }

        \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
          \prop_set_from_keyval:cn { l_stex_variable_ \l_tmpa_str _prop}{
            name   = \l_tmpa_str ,
            args   = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
            arity  = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
            assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
          }
          \cs_set_eq:cc {stex_var_notation_\l_tmpa_str _cs}
            {g__stex_structures_tmpa_\l_tmpa_str _cs}
          \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str _cs}
            {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
        }
        \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\stex_inv
      }
      \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
        \prop_set_from_keyval:cn {l_stex_varinstance_\l__stex_structures_name_str _prop }{
          domain = \l_stex_get_structure_module_str ,
          \prop_to_keyval:N \l_tmpa_prop
        }
        \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
        \tl_set:cn {l_stex_varinstance_\l__stex_structures_name_str _op_tl}{
          \exp_args:Nnx \exp_not:N \use:nn {
            \str_set:Nn \exp_not:N \l_stex_current_symbol_str {var://\l__stex_structures_nam
            \_stex_term_omv:nn {var://\l__stex_structures_name_str}{
              \exp_not:n{
                \_varcomp{#5}
              }
            }
          }{
            \exp_not:n{\_stex_reset:N \l_stex_current_symbol_str}
          }
        }
      }
    }
    \stex_debug:nn{varinstantiate}{\expandafter\detokenize\expandafter{\g__stex_structures_a
    \aftergroup\g__stex_structures_aftergroup_tl
  \endgroup
  \stex_smsmode_do:\ignorespacesandpars
}

\cs_new_protected:Nn \stex_invoke_instance:n {
  \peek_charcode_remove:NTF ! {
    \stex_invoke_symbol:n{#1}
  }{
    \_stex_invoke_instance:nn {#1}
  }
}


\cs_new_protected:Nn \stex_invoke_varinstance:n {
  \peek_charcode_remove:NTF ! {
    \exp_args:Nnx \use:nn {
      \def\comp{\_varcomp}
      \use:c{l_stex_varinstance_#1_op_tl}
    }{
      \_stex_reset:N \comp
    }
  }{
    \_stex_invoke_varinstance:nn {#1}
  }
```

```
}

\cs_new_protected:Nn \_stex_invoke_instance:nn {
  \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
    \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
  }{
    \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
    \msg_error:nnnnn{stex}{error/unknownfield}{#2}{#1}{
      \prop_to_keyval:N \l_tmpa_prop
    }
  }
}

\cs_new_protected:Nn \_stex_invoke_varinstance:nn {
  \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
    \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
    \l_tmpa_tl
  }{
    \msg_error:nnnnn{stex}{error/unknownfield}{#2}{#1}{}
  }
}
```

(*End definition for* `\instantiate`. *This function is documented on page 31.*)

`\stex_invoke_structure:nnn`

```
% #1: URI of the instance
% #2: URI of the instantiated module
\cs_new_protected:Nn \stex_invoke_structure:nnn {
  \tl_if_empty:nTF{ #3 }{
    \prop_set_eq:Nc \l__stex_structures_structure_prop {
      c_stex_feature_ #2 _prop
    }
    \tl_clear:N \l_tmpa_tl
    \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
    \seq_map_inline:Nn \l_tmpa_seq {
      \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
      \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
      \cs_if_exist:cT {
        stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
      }{
        \tl_if_empty:NF \l_tmpa_tl {
          \tl_put_right:Nn \l_tmpa_tl {,}
        }
        \tl_put_right:Nx \l_tmpa_tl {
          \stex_invoke_symbol:n {#1/\l_tmpa_str}!
        }
      }
    }
    \exp_args:No \mathstruct \l_tmpa_tl
  }{
    \stex_invoke_symbol:n{#1/#3}
  }
}
```

(*End definition for* `\stex_invoke_structure:nnn`*. This function is documented on page* **??***.*)

```
⟨/package⟩
```

Chapter 32

sTeX -Statements Implementation

```
⟨*package⟩

%%%%%%%%%%%%%   features.dtx   %%%%%%%%%%%%%

⟨@@=stex_statements⟩
```

Warnings and error messages

\titleemph

```
\def\titleemph#1{\textbf{#1}}
```

(*End definition for* `\titleemph`. *This function is documented on page* ??.)

32.1 Definitions

definiendum

```
\keys_define:nn {stex / definiendum }{
  pre     .tl_set:N     = \l__stex_statements_definiendum_pre_tl,
  post    .tl_set:N     = \l__stex_statements_definiendum_post_tl,
  root    .str_set_x:N  = \l__stex_statements_definiendum_root_str,
  gfa     .str_set_x:N  = \l__stex_statements_definiendum_gfa_str
}
\cs_new_protected:Nn \__stex_statements_definiendum_args:n {
  \str_clear:N \l__stex_statements_definiendum_root_str
  \tl_clear:N \l__stex_statements_definiendum_post_tl
  \str_clear:N \l__stex_statements_definiendum_gfa_str
  \keys_set:nn { stex / definiendum }{ #1 }
}
\NewDocumentCommand \definiendum { O{} m m} {
  \__stex_statements_definiendum_args:n { #1 }
  \stex_get_symbol:n { #2 }
  \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
  \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
    \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
      \tl_set:Nn \l_tmpa_tl { #3 }
    } {
      \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
      \tl_set:Nn \l_tmpa_tl {
        \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
       }
    }
  } {
    \tl_set:Nn \l_tmpa_tl { #3 }
  }

  % TODO root
  \rustex_if:TF {
    \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
  } {
    \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
  }
}
\stex_deactivate_macro:Nn \definiendum {definition~environments}
```

(*End definition for* `definiendum`. *This function is documented on page 40.*)

definame

```

\NewDocumentCommand \definame { O{} m } {
  \__stex_statements_definiendum_args:n { #1 }
  % TODO: root
  \stex_get_symbol:n { #2 }
  \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
  \str_set:Nx \l_tmpa_str {
    \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
  }
  \str_replace_all:Nnn \l_tmpa_str {-} {~}
  \rustex_if:TF {
    \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
      \l_tmpa_str\l__stex_statements_definiendum_post_tl
      }
  } {
    \exp_args:Nnx \defemph@uri {
      \l_tmpa_str\l__stex_statements_definiendum_post_tl
    } { \l_stex_get_symbol_uri_str }
  }
}
\stex_deactivate_macro:Nn \definame {definition~environments}

\NewDocumentCommand \Definame { O{} m } {
  \__stex_statements_definiendum_args:n { #1 }
  \stex_get_symbol:n { #2 }
  \str_set:Nx \l_tmpa_str {
    \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
  }
  \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
  \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
  \rustex_if:TF {
    \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
      \l_tmpa_str\l__stex_statements_definiendum_post_tl
     }
  } {
    \exp_args:Nnx \defemph@uri {
      \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
    } { \l_stex_get_symbol_uri_str }
  }
}
\stex_deactivate_macro:Nn \Definame {definition~environments}

\NewDocumentCommand \premise { m }{
  \stex_annotate:nnn{ premise }{}{ #1 }
}
\NewDocumentCommand \conclusion { m }{
  \stex_annotate:nnn{ conclusion }{}{ #1 }
}
\NewDocumentCommand \definiens { O{} m }{
  \str_clear:N \l_stex_get_symbol_uri_str
  \tl_if_empty:nF {#1} {
    \stex_get_symbol:n { #1 }
  }
  \str_if_empty:NT \l_stex_get_symbol_uri_str {
    \int_compare:nNnTF {\clist_count:N \l__stex_statements_sdefinition_for_clist} = 1 {
      \str_set:Nx \l_stex_get_symbol_uri_str {\clist_item:Nn \l__stex_statements_sdefinition
    }{
      % TODO throw error
    }
  }
  \str_if_eq:eeT {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{module}}
    {\l_stex_current_module_str}{
      \str_if_eq:eeF {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defin
      {true}{
        \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
        \exp_args:Nx \stex_add_to_current_module:n {
          \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
        }
      }
  }
  \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
}

\stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
\stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
\stex_deactivate_macro:Nn \definiens {definition~environments}

```

(*End definition for* `definame`. *This function is documented on page 40.*)

`sdefinition`

```

\keys_define:nn {stex / sdefinition }{
  type    .str_set_x:N  = \sdefinitiontype,
  id      .str_set_x:N  = \sdefinitionid,
  name    .str_set_x:N  = \sdefinitionname,
  for     .clist_set:N  = \l__stex_statements_sdefinition_for_clist ,
  title   .tl_set:N     = \sdefinitiontitle
}
\cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
  \str_clear:N \sdefinitiontype
  \str_clear:N \sdefinitionid
  \str_clear:N \sdefinitionname
  \clist_clear:N \l__stex_statements_sdefinition_for_clist
  \tl_clear:N \sdefinitiontitle
  \keys_set:nn { stex / sdefinition }{ #1 }
}

\NewDocumentEnvironment{sdefinition}{O{}}{
  \__stex_statements_sdefinition_args:n{ #1 }
  \stex_reactivate_macro:N \definiendum
  \stex_reactivate_macro:N \definame
  \stex_reactivate_macro:N \Definame
  \stex_reactivate_macro:N \premise
  \stex_reactivate_macro:N \definiens
  \stex_if_smsmode:F{
    \seq_clear:N \l_tmpa_seq
    \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
      \tl_if_empty:nF{ ##1 }{
        \stex_get_symbol:n { ##1 }
        \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
          \l_stex_get_symbol_uri_str
        }
      }
    }
    \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpa_seq
    \exp_args:Nnnx
    \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {,}}
    \str_if_empty:NF \sdefinitiontype {
      \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
    }
    \str_if_empty:NF \sdefinitionname {
      \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
    }
    \clist_set:No \l_tmpa_clist \sdefinitiontype
    \tl_clear:N \l_tmpa_tl
    \clist_map_inline:Nn \l_tmpa_clist {
      \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
        \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
      }
    }
    \tl_if_empty:NTF \l_tmpa_tl {
      \__stex_statements_sdefinition_start:
    }{
      \l_tmpa_tl
    }
  }
  \stex_ref_new_doc_target:n \sdefinitionid
  \stex_smsmode_do:
}{
  \stex_suppress_html:n {
    \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
  }
  \stex_if_smsmode:F {
    \clist_set:No \l_tmpa_clist \sdefinitiontype
    \tl_clear:N \l_tmpa_tl
    \clist_map_inline:Nn \l_tmpa_clist {
      \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
        \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
      }
    }
    \tl_if_empty:NTF \l_tmpa_tl {
      \__stex_statements_sdefinition_end:
    }{
      \l_tmpa_tl
    }
    \end{stex_annotate_env}
  }
}
```

\stexpatchdefinition

```
\cs_new_protected:Nn \__stex_statements_sdefinition_start: {
  \par\noindent\titleemph{Definition\tl_if_empty:NF \sdefinitiontitle {
    ~(\sdefinitiontitle)
  }~}
}
\cs_new_protected:Nn \__stex_statements_sdefinition_end: {\par\medskip}

\newcommand\stexpatchdefinition[3][] {
    \str_set:Nx \l_tmpa_str{ #1 }
    \str_if_empty:NTF \l_tmpa_str {
      \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
      \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
    }{
      \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2
      \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
    }
}
```

(*End definition for* `\stexpatchdefinition`*. This function is documented on page 42.*)

`\inlinedef` inline:

```
\keys_define:nn {stex / inlinedef }{
  type     .str_set_x:N  = \sdefinitiontype,
  id       .str_set_x:N  = \sdefinitionid,
  for      .clist_set:N  = \l__stex_statements_sdefinition_for_clist ,
  name     .str_set_x:N  = \sdefinitionname
}
\cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
  \str_clear:N \sdefinitiontype
  \str_clear:N \sdefinitionid
  \str_clear:N \sdefinitionname
  \clist_clear:N \l__stex_statements_sdefinition_for_clist
  \keys_set:nn { stex / inlinedef }{ #1 }
}
\NewDocumentCommand \inlinedef { O{} m } {
  \begingroup
  \__stex_statements_inlinedef_args:n{ #1 }
  \stex_reactivate_macro:N \definiendum
  \stex_reactivate_macro:N \definame
  \stex_reactivate_macro:N \Definame
  \stex_reactivate_macro:N \premise
  \stex_reactivate_macro:N \definiens
  \stex_ref_new_doc_target:n \sdefinitionid
  \stex_if_smsmode:TF{\stex_suppress_html:n {
    \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
  }}{
    \seq_clear:N \l_tmpa_seq
    \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
      \tl_if_empty:nF{ ##1 }{
        \stex_get_symbol:n { ##1 }
        \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
          \l_stex_get_symbol_uri_str
        }
      }
    }
    \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpa_seq
    \exp_args:Nnx
    \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
      \str_if_empty:NF \sdefinitiontype {
        \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
      }
      #2
      \str_if_empty:NF \sdefinitionname {
        \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sdefinitionname}}
        \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
      }
    }
  }
  \endgroup
  \stex_smsmode_do:
}
```

(*End definition for* `\inlinedef`. *This function is documented on page* **??**.)

32.2 Assertions

`sassertion`

```

\keys_define:nn {stex / sassertion }{
  type    .str_set_x:N  = \sassertiontype,
  id      .str_set_x:N  = \sassertionid,
  title   .tl_set:N     = \sassertiontitle ,
  for     .clist_set:N  = \l__stex_statements_sassertion_for_clist ,
  name    .str_set_x:N  = \sassertionname
}
```

```
\cs_new_protected:Nn \__stex_statements_sassertion_args:n {
  \str_clear:N \sassertiontype
  \str_clear:N \sassertionid
  \str_clear:N \sassertionname
  \clist_clear:N \l__stex_statements_sassertion_for_clist
  \tl_clear:N \sassertiontitle
  \keys_set:nn { stex / sassertion }{ #1 }
}

%\tl_new:N \g__stex_statements_aftergroup_tl

\NewDocumentEnvironment{sassertion}{O{}}{
  \__stex_statements_sassertion_args:n{ #1 }
  \stex_reactivate_macro:N \premise
  \stex_reactivate_macro:N \conclusion
  \stex_if_smsmode:F {
    \seq_clear:N \l_tmpa_seq
    \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
      \tl_if_empty:nF{ ##1 }{
        \stex_get_symbol:n { ##1 }
        \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
          \l_stex_get_symbol_uri_str
        }
      }
    }
    \exp_args:Nnnx
    \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}
    \str_if_empty:NF \sassertiontype {
      \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
    }
    \str_if_empty:NF \sassertionname {
      \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
    }
    \clist_set:No \l_tmpa_clist \sassertiontype
    \tl_clear:N \l_tmpa_tl
    \clist_map_inline:Nn \l_tmpa_clist {
      \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
        \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
      }
    }
    \tl_if_empty:NTF \l_tmpa_tl {
      \__stex_statements_sassertion_start:
    }{
      \l_tmpa_tl
    }
  }
  \str_if_empty:NTF \sassertionid {
    \str_if_empty:NF \sassertionname {
      \stex_ref_new_doc_target:n {}
    }
  } {
    \stex_ref_new_doc_target:n \sassertionid
  }
  \stex_smsmode_do:
}{
  \str_if_empty:NF \sassertionname {
    \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
    \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
  }
  \stex_if_smsmode:F {
    \clist_set:No \l_tmpa_clist \sassertiontype
    \tl_clear:N \l_tmpa_tl
    \clist_map_inline:Nn \l_tmpa_clist {
      \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
        \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
      }
    }
    \tl_if_empty:NTF \l_tmpa_tl {
      \__stex_statements_sassertion_end:
    }{
      \l_tmpa_tl
    }
    \end{stex_annotate_env}
  }
}
```

\stexpatchassertion

```

\cs_new_protected:Nn \__stex_statements_sassertion_start: {
  \par\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {
    (\sassertiontitle)
  }~}
}
\cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}

\newcommand\stexpatchassertion[3][] {
    \str_set:Nx \l_tmpa_str{ #1 }
    \str_if_empty:NTF \l_tmpa_str {
      \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
      \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
    }{
      \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
      \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
    }
}
```

(*End definition for* `\stexpatchassertion`. *This function is documented on page 42.*)

\inlineass inline:

```
\keys_define:nn {stex / inlineass }{
  type    .str_set_x:N  = \sassertiontype,
  id      .str_set_x:N  = \sassertionid,
  for     .clist_set:N  = \l__stex_statements_sassertion_for_clist ,
  name    .str_set_x:N  = \sassertionname
}
\cs_new_protected:Nn \__stex_statements_inlineass_args:n {
  \str_clear:N \sassertiontype
  \str_clear:N \sassertionid
  \str_clear:N \sassertionname
  \clist_clear:N \l__stex_statements_sassertion_for_clist
  \keys_set:nn { stex / inlineass }{ #1 }
}
\NewDocumentCommand \inlineass { O{} m } {
  \begingroup
  \stex_reactivate_macro:N \premise
  \stex_reactivate_macro:N \conclusion
  \__stex_statements_inlineass_args:n{ #1 }
  \str_if_empty:NTF \sassertionid {
    \str_if_empty:NF \sassertionname {
      \stex_ref_new_doc_target:n {}
    }
  } {
    \stex_ref_new_doc_target:n \sassertionid
  }

  \stex_if_smsmode:TF{
    \str_if_empty:NF \sassertionname {
      \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
      \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
    }
  }{
    \seq_clear:N \l_tmpa_seq
    \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
      \tl_if_empty:nF{ ##1 }{
        \stex_get_symbol:n { ##1 }
        \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
          \l_stex_get_symbol_uri_str
        }
      }
    }
    \exp_args:Nnx
    \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{
      \str_if_empty:NF \sassertiontype {
        \stex_annotate_invisible:nnn{typestrings}{\sassertiontype}{}
      }
      #2
      \str_if_empty:NF \sassertionname {
        \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
        \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
        \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
      }
    }
  }
  \endgroup
  \stex_smsmode_do:
}
```

(*End definition for* `\inlineass`*. This function is documented on page* **??**.)

32.3 Examples

sexample

```

\keys_define:nn {stex / sexample }{
  type    .str_set_x:N  = \exampletype,
  id      .str_set_x:N  = \sexampleid,
  title   .tl_set:N     = \sexampletitle,
  name    .str_set_x:N  = \sexamplename ,
  for     .clist_set:N  = \l__stex_statements_sexample_for_clist,
}
\cs_new_protected:Nn \__stex_statements_sexample_args:n {
  \str_clear:N \sexampletype
  \str_clear:N \sexampleid
  \str_clear:N \sexamplename
  \tl_clear:N \sexampletitle
  \clist_clear:N \l__stex_statements_sexample_for_clist
  \keys_set:nn { stex / sexample }{ #1 }
}

\NewDocumentEnvironment{sexample}{O{}}{
  \__stex_statements_sexample_args:n{ #1 }
  \stex_reactivate_macro:N \premise
  \stex_reactivate_macro:N \conclusion
  \stex_if_smsmode:F {
    \seq_clear:N \l_tmpa_seq
    \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
      \tl_if_empty:nF{ ##1 }{
        \stex_get_symbol:n { ##1 }
        \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
          \l_stex_get_symbol_uri_str
        }
      }
    }
    \exp_args:Nnnx
    \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
    \str_if_empty:NF \sexampletype {
      \stex_annotate_invisible:nnn{typestrings}{\sexampletype}{}
    }
    \str_if_empty:NF \sexamplename {
      \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}
    }
    \clist_set:No \l_tmpa_clist \sexampletype
    \tl_clear:N \l_tmpa_tl
    \clist_map_inline:Nn \l_tmpa_clist {
      \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
        \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
      }
    }
    \tl_if_empty:NTF \l_tmpa_tl {
      \__stex_statements_sexample_start:
    }{
      \l_tmpa_tl
    }
  }
  \str_if_empty:NF \sexampleid {
    \stex_ref_new_doc_target:n \sexampleid
  }
  \stex_smsmode_do:
}{
  \str_if_empty:NF \sexamplename {
    \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
  }
  \stex_if_smsmode:F {
    \clist_set:No \l_tmpa_clist \sexampletype
    \tl_clear:N \l_tmpa_tl
    \clist_map_inline:Nn \l_tmpa_clist {
      \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
        \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
      }
    }
    \tl_if_empty:NTF \l_tmpa_tl {
      \__stex_statements_sexample_end:
    }{
      \l_tmpa_tl
    }
    \end{stex_annotate_env}
  }
}
```

\stexpatchexample

```

\cs_new_protected:Nn \__stex_statements_sexample_start: {
  \par\noindent\titleemph{Example~\tl_if_empty:NF \sexampletitle {
    (\sexampletitle)
  }~}
}
\cs_new_protected:Nn \__stex_statements_sexample_end: {\par\medskip}

\newcommand\stexpatchexample[3][] {
    \str_set:Nx \l_tmpa_str{ #1 }
    \str_if_empty:NTF \l_tmpa_str {
      \tl_set:Nn \__stex_statements_sexample_start: { #2 }
      \tl_set:Nn \__stex_statements_sexample_end: { #3 }
    }{
      \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
      \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
    }
}
```

(*End definition for* `\stexpatchexample`*. This function is documented on page 42.*)

\inlineex inline:

```
\keys_define:nn {stex / inlineex }{
  type     .str_set_x:N  = \sexampletype,
  id       .str_set_x:N  = \sexampleid,
  for      .clist_set:N  = \l__stex_statements_sexample_for_clist ,
  name     .str_set_x:N  = \sexamplename
}
\cs_new_protected:Nn \__stex_statements_inlineex_args:n {
  \str_clear:N \sexampletype
  \str_clear:N \sexampleid
  \str_clear:N \sexamplename
  \clist_clear:N \l__stex_statements_sexample_for_clist
  \keys_set:nn { stex / inlineex }{ #1 }
}
\NewDocumentCommand \inlineex { O{} m } {
  \begingroup
  \stex_reactivate_macro:N \premise
  \stex_reactivate_macro:N \conclusion
  \__stex_statements_inlineex_args:n{ #1 }
  \str_if_empty:NF \sexampleid {
    \stex_ref_new_doc_target:n \sexampleid
  }
  \stex_if_smsmode:TF{
    \str_if_empty:NF \sexamplename {
      \stex_suppress_html:n{\stex_symdecl_do:nn{}{\examplename}}
    }
  }{
    \seq_clear:N \l_tmpa_seq
    \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
      \tl_if_empty:nF{ ##1 }{
        \stex_get_symbol:n { ##1 }
        \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
          \l_stex_get_symbol_uri_str
        }
      }
    }
    \exp_args:Nnx
    \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpa_seq {,}}{
      \str_if_empty:NF \sexampletype {
        \stex_annotate_invisible:nnn{typestrings}{\sexampletype}{}
      }
      #2
      \str_if_empty:NF \sexamplename {
        \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
        \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}
      }
    }
  }
  \endgroup
  \stex_smsmode_do:
}
```

(*End definition for* `\inlineex`*. This function is documented on page* **??**.)

32.4 Logical Paragraphs

`sparagraph`

```
\keys_define:nn { stex / sparagraph} {
  id      .str_set_x:N   = \sparagraphid ,
  title   .tl_set:N      = \l_stex_sparagraph_title_tl ,
  type    .str_set_x:N   = \sparagraphtype ,
  for     .clist_set:N   = \l__stex_statements_sparagraph_for_clist ,
  from    .tl_set:N      = \sparagraphfrom ,
  to      .tl_set:N      = \sparagraphto ,
  start   .tl_set:N      = \l_stex_sparagraph_start_tl ,
  name    .str_set:N     = \sparagraphname
}

\cs_new_protected:Nn \stex_sparagraph_args:n {
  \tl_clear:N \l_stex_sparagraph_title_tl
  \tl_clear:N \sparagraphfrom
  \tl_clear:N \sparagraphto
  \tl_clear:N \l_stex_sparagraph_start_tl
  \str_clear:N \sparagraphid
  \str_clear:N \sparagraphtype
  \clist_clear:N \l__stex_statements_sparagraph_for_clist
  \str_clear:N \sparagraphname
  \keys_set:nn { stex / sparagraph }{ #1 }
}
\newif\if@in@omtext\@in@omtextfalse

\NewDocumentEnvironment {sparagraph} { O{} } {
  \stex_sparagraph_args:n { #1 }
  \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
    \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
  }{
    \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
  }
  \@in@omtexttrue
  \stex_if_smsmode:F {
    \seq_clear:N \l_tmpa_seq
    \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
      \tl_if_empty:nF{ ##1 }{
        \stex_get_symbol:n { ##1 }
        \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
          \l_stex_get_symbol_uri_str
        }
      }
    }
    \exp_args:Nnnx
    \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
    \str_if_empty:NF \sparagraphtype {
      \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
    }
    \str_if_empty:NF \sparagraphfrom {
      \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
    }
    \str_if_empty:NF \sparagraphto {
      \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
    }
    \str_if_empty:NF \sparagraphname {
      \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
    }
    \clist_set:No \l_tmpa_clist \sparagraphtype
    \tl_clear:N \l_tmpa_tl
    \clist_map_inline:Nn \sparagraphtype {
      \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
        \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
      }
    }
    \tl_if_empty:NTF \l_tmpa_tl {
      \__stex_statements_sparagraph_start:
    }{
      \l_tmpa_tl
    }
  }
  \clist_set:No \l_tmpa_clist \sparagraphtype
  \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}
  {
    \stex_reactivate_macro:N \definiendum
    \stex_reactivate_macro:N \definame
    \stex_reactivate_macro:N \Definame
    \stex_reactivate_macro:N \premise
    \stex_reactivate_macro:N \definiens
  }
  \str_if_empty:NTF \sparagraphid {
    \str_if_empty:NTF \sparagraphname {
      \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
        \stex_ref_new_doc_target:n {}
      }
    } {
      \stex_ref_new_doc_target:n {}
    }
  } {
    \stex_ref_new_doc_target:n \sparagraphid
  }
  \exp_args:NNx
  \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
    \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
      \tl_if_empty:nF{ ##1 }{
        \stex_get_symbol:n { ##1 }
        \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
      }
    }
  }
  \stex_smsmode_do:
  \ignorespacesandpars
}{
  \str_if_empty:NF \sparagraphname {
    \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
    \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
  }
  \stex_if_smsmode:F {
    \clist_set:No \l_tmpa_clist \sparagraphtype
    \tl_clear:N \l_tmpa_tl
    \clist_map_inline:Nn \l_tmpa_clist {
      \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
        \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
      }
    }
    \tl_if_empty:NTF \l_tmpa_tl {
      \__stex_statements_sparagraph_end:
    }{
      \l_tmpa_tl
    }
    \end{stex_annotate_env}
  }
}
```

\stexpatchparagraph

```

\cs_new_protected:Nn \__stex_statements_sparagraph_start: {
  \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
    \tl_if_empty:NF \l_stex_sparagraph_title_tl {
      \titleemph{\l_stex_sparagraph_title_tl}:~
    }
  }{
    \titleemph{\l_stex_sparagraph_start_tl}~
  }
}
\cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}

\newcommand\stexpatchparagraph[3][] {
    \str_set:Nx \l_tmpa_str{ #1 }
    \str_if_empty:NTF \l_tmpa_str {
      \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
      \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
    }{
      \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
      \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
    }
}

\keys_define:nn { stex / inlinepara} {
  id      .str_set_x:N   = \sparagraphid ,
  type    .str_set_x:N   = \sparagraphtype ,
  for     .clist_set:N   = \l__stex_statements_sparagraph_for_clist ,
  from    .tl_set:N      = \sparagraphfrom ,
  to      .tl_set:N      = \sparagraphto ,
  name    .str_set:N     = \sparagraphname
}
\cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
  \tl_clear:N \sparagraphfrom
  \tl_clear:N \sparagraphto
  \str_clear:N \sparagraphid
  \str_clear:N \sparagraphtype
  \clist_clear:N \l__stex_statements_sparagraph_for_clist
  \str_clear:N \sparagraphname
  \keys_set:nn { stex / inlinepara }{ #1 }
}
\NewDocumentCommand \inlinepara { O{} m } {
```

```
\begingroup
\__stex_statements_inlinepara_args:n{ #1 }
\clist_set:No \l_tmpa_clist \sparagraphtype
\str_if_empty:NTF \sparagraphid {
  \str_if_empty:NTF \sparagraphname {
    \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
      \stex_ref_new_doc_target:n {}
    }
  } {
    \stex_ref_new_doc_target:n {}
  }
} {
  \stex_ref_new_doc_target:n \sparagraphid
}
\stex_if_smsmode:TF{
  \str_if_empty:NF \sparagraphname {
    \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
    \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
  }
}{
  \seq_clear:N \l_tmpa_seq
  \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
    \tl_if_empty:nF{ ##1 }{
      \stex_get_symbol:n { ##1 }
      \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
        \l_stex_get_symbol_uri_str
      }
    }
  }
  \exp_args:Nnx
  \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}{
    \str_if_empty:NF \sparagraphtype {
      \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
    }
    \str_if_empty:NF \sparagraphfrom {
      \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
    }
    \str_if_empty:NF \sparagraphto {
      \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
    }
    \str_if_empty:NF \sparagraphname {
      \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
      \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
      \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
    }
    \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
      \clist_map_inline:Nn \l_tmpa_seq {
        \stex_ref_new_sym_target:n {##1}
      }
    }
    #2
  }
}
\endgroup
```

```
  \stex_smsmode_do:
}

```

(*End definition for* `\stexpatchparagraph`*. This function is documented on page 42.*)

```
⟨/package⟩
```

Chapter 33

The Implementation

33.1 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).[8] EdN:8

```
⟨*package⟩
⟨@@=stex_sproof⟩

%%%%%%%%%%%%%   sproof.dtx   %%%%%%%%%%%%%

```

33.2 Proofs

We first define some keys for the proof environment.

```
\keys_define:nn { stex / spf } {
  id          .str_set_x:N  = \spfid,
  for         .clist_set:N  = \l__stex_sproof_spf_for_clist ,
  from        .tl_set:N     = \l__stex_sproof_spf_from_tl ,
  proofend    .tl_set:N     = \l__stex_sproof_spf_proofend_tl,
  type        .str_set_x:N  = \spftype,
  title       .tl_set:N     = \spftitle,
  continues   .tl_set:N     = \l__stex_sproof_spf_continues_tl,
  functions   .tl_set:N     = \l__stex_sproof_spf_functions_tl,
  method      .tl_set:N     = \l__stex_sproof_spf_method_tl
}
\cs_new_protected:Nn \__stex_sproof_spf_args:n {
\str_clear:N \spfid
\tl_clear:N \l__stex_sproof_spf_for_tl
\tl_clear:N \l__stex_sproof_spf_from_tl
\tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
\str_clear:N \spftype
\tl_clear:N \spftitle
\tl_clear:N \l__stex_sproof_spf_continues_tl
\tl_clear:N \l__stex_sproof_spf_functions_tl
```

[8]EdNote: need an implementation for LaTeXML

```
\tl_clear:N \l__stex_sproof_spf_method_tl
  \bool_set_false:N \l__stex_sproof_inc_counter_bool
\keys_set:nn { stex / spf }{ #1 }
}
```

\c__stex_sproof_flow_str We define this macro, so that we can test whether the `display` key has the value `flow`

```
\str_set:Nn\c__stex_sproof_flow_str{inline}
```

(*End definition for* `\c__stex_sproof_flow_str`*.*)

For proofs, we will have to have deeply nested structures of enumerated list-like environments. However, LaTeX only allows `enumerate` environments up to nesting depth 4 and general list environments up to listing depth 6. This is not enough for us. Therefore we have decided to go along the route proposed by Leslie Lamport to use a single top-level list with dotted sequences of numbers to identify the position in the proof tree. Unfortunately, we could not use his `pf.sty` package directly, since it does not do automatic numbering, and we have to add keyword arguments all over the place, to accomodate semantic information.

pst@with@label This environment manages[7] the path labeling of the proof steps in the description environment of the outermost `proof` environment. The argument is the label prefix up to now; which we cache in `\pst@label` (we need evaluate it first, since are in the right place now!). Then we increment the proof depth which is stored in `\count10` (lower counters are used by TeX for page numbering) and initialize the next level counter `\count\count10` with 1. In the end call for this environment, we just decrease the proof depth counter by 1 again.

```
\intarray_new:Nn\l__stex_sproof_counter_intarray{50}
\cs_new_protected:Npn \sproofnumber {
  \int_set:Nn \l_tmpa_int {1}
  \bool_while_do:nn {
    \int_compare_p:nNn {
      \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
    } > 0
  }{
    \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
    \int_incr:N \l_tmpa_int
  }
}
\cs_new_protected:Npn \__stex_sproof_inc_counter: {
  \int_set:Nn \l_tmpa_int {1}
  \bool_while_do:nn {
    \int_compare_p:nNn {
      \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
    } > 0
  }{
    \int_incr:N \l_tmpa_int
  }
  \int_compare:nNnF \l_tmpa_int = 1 {
    \int_decr:N \l_tmpa_int
  }
  \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
    \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1
```

[7]This gets the labeling right but only works 8 levels deep

```
  }
}

\cs_new_protected:Npn \__stex_sproof_add_counter: {
  \int_set:Nn \l_tmpa_int {1}
  \bool_while_do:nn {
    \int_compare_p:nNn {
      \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
    } > 0
  }{
    \int_incr:N \l_tmpa_int
  }
  \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 1 }
}

\cs_new_protected:Npn \__stex_sproof_remove_counter: {
  \int_set:Nn \l_tmpa_int {1}
  \bool_while_do:nn {
    \int_compare_p:nNn {
      \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
    } > 0
  }{
    \int_incr:N \l_tmpa_int
  }
  \int_decr:N \l_tmpa_int
  \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 0 }
}
```

`\sproofend` This macro places a little box at the end of the line if there is space, or at the end of the next line if there isn't

```
\def\sproof@box{
  \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
}
\def\sproofend{
  \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
    \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
  }
}
```

(*End definition for* `\sproofend`*. This function is documented on page* **??**.)

`spf@*@kw`

```
\def\spf@proofsketch@kw{Proof~Sketch}
\def\spf@proof@kw{Proof}
\def\spf@step@kw{Step}
```

(*End definition for* `spf@*@kw`*. This function is documented on page* **??**.)

For the other languages, we set up triggers

```
\AddToHook{begindocument}{
  \ltx@ifpackageloaded{babel}{
    \makeatletter
    \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
    \clist_if_in:NnT \l_tmpa_clist {ngerman}{
      \input{sproof-ngerman.ldf}
    }
    \clist_if_in:NnT \l_tmpa_clist {finnish}{
      \input{sproof-finnish.ldf}
    }
    \clist_if_in:NnT \l_tmpa_clist {french}{
      \input{sproof-french.ldf}
    }
    \clist_if_in:NnT \l_tmpa_clist {russian}{
      \input{sproof-russian.ldf}
    }
    \makeatother
  }{}
}
```

spfsketch

```
\newcommand\spfsketch[2][]{
  \begingroup
  \let \premise \stex_proof_premise:
  \__stex_sproof_spf_args:n{#1}
  \stex_if_smsmode:TF {
    \str_if_empty:NF \spfid {
      \stex_ref_new_doc_target:n \spfid
    }
  }{
    \seq_clear:N \l_tmpa_seq
    \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
      \tl_if_empty:nF{ ##1 }{
        \stex_get_symbol:n { ##1 }
        \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
          \l_stex_get_symbol_uri_str
        }
      }
    }
    \exp_args:Nnx
    \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
      \str_if_empty:NF \spftype {
        \stex_annotate_invisible:nnn{type}{\spftype}{}
      }
      \clist_set:No \l_tmpa_clist \spftype
      \tl_set:Nn \l_tmpa_tl {
        \titleemph{
          \tl_if_empty:NTF \spftitle {
            \spf@proofsketch@kw
          }{
            \spftitle
          }
        }:~
      }
      \clist_map_inline:Nn \l_tmpa_clist {
        \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
          \tl_clear:N \l_tmpa_tl
        }
      }
      \str_if_empty:NF \spfid {
        \stex_ref_new_doc_target:n \spfid
      }
      \l_tmpa_tl #2 \sproofend
    }
  }
  \endgroup
  \stex_smsmode_do:
}

```

(*End definition for* `spfsketch`*. This function is documented on page* **??**.)

EdN:9
EdN:10

`spfeq` This is very similar to `\spfsketch`, but uses a computation array[9][10]

```
\newenvironment{spfeq}[2][]{
  \__stex_sproof_spf_args:n{#1}
  \let \premise \stex_proof_premise:
  \stex_if_smsmode:TF {
    \str_if_empty:NF \spfid {
      \stex_ref_new_doc_target:n \spfid
    }
  }{
    \seq_clear:N \l_tmpa_seq
    \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
      \tl_if_empty:nF{ ##1 }{
        \stex_get_symbol:n { ##1 }
        \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
          \l_stex_get_symbol_uri_str
        }
      }
    }
    \exp_args:Nnnx
    \begin{stex_annotate_env}{spfeq}{\seq_use:Nn \l_tmpa_seq {,}}
    \str_if_empty:NF \spftype {
      \stex_annotate_invisible:nnn{type}{\spftype}{}
    }

    \clist_set:No \l_tmpa_clist \spftype
    \tl_clear:N \l_tmpa_tl
    \clist_map_inline:Nn \l_tmpa_clist {
      \tl_if_exist:cT {__stex_sproof_spfeq_##1_start:}{
        \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_start:}}
      }
      \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
        \tl_set:Nn \l_tmpa_tl {\use:n{}}
      }
    }
    \tl_if_empty:NTF \l_tmpa_tl {
      \__stex_sproof_spfeq_start:
    }{
      \l_tmpa_tl
    }{~#2}
```

[9] EdNote: This should really be more like a tabular with an ensuremath in it. or invoke text on the last column
[10] EdNote: document above

```
    \str_if_empty:NF \spfid {
      \stex_ref_new_doc_target:n \spfid
    }
    \begin{displaymath}\begin{array}{rcll}
  }
  \stex_smsmode_do:
}{
  \stex_if_smsmode:F {
    \end{array}\end{displaymath}
    \clist_set:No \l_tmpa_clist \spftype
    \tl_clear:N \l_tmpa_tl
    \clist_map_inline:Nn \l_tmpa_clist {
      \tl_if_exist:cT {__stex_sproof_spfeq_##1_end:}{
        \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_end:}}
      }
    }
    \tl_if_empty:NTF \l_tmpa_tl {
      \__stex_sproof_spfeq_end:
    }{
      \l_tmpa_tl
    }
    \end{stex_annotate_env}
  }
}

\cs_new_protected:Nn \__stex_sproof_spfeq_start: {
  \titleemph{
    \tl_if_empty:NTF \spftitle {
      \spf@proof@kw
    }{
      \spftitle
    }
  }:
}
\cs_new_protected:Nn \__stex_sproof_spfeq_end: {\sproofend}

\newcommand\stexpatchspfeq[3][] {
    \str_set:Nx \l_tmpa_str{ #1 }
    \str_if_empty:NTF \l_tmpa_str {
      \tl_set:Nn \__stex_sproof_spfeq_start: { #2 }
      \tl_set:Nn \__stex_sproof_spfeq_end: { #3 }
    }{
      \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_start:\endcsname{ #2 }
      \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_end:\endcsname{ #3 }
    }
}

```

(*End definition for* `spfeq`. *This function is documented on page* **??**.)

`sproof` In this environment, we initialize the proof depth counter `\count10` to 10, and set up the description environment that will take the proof steps. At the end of the proof, we position the proof end into the last line.

```
\newenvironment{sproof}[2][]{
  \let \premise \stex_proof_premise:
  \intarray_gzero:N \l__stex_sproof_counter_intarray
  \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
  \__stex_sproof_spf_args:n{#1}
  \stex_if_smsmode:TF {
    \str_if_empty:NF \spfid {
      \stex_ref_new_doc_target:n \spfid
    }
  }{
    \seq_clear:N \l_tmpa_seq
    \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
      \tl_if_empty:nF{ ##1 }{
        \stex_get_symbol:n { ##1 }
        \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
          \l_stex_get_symbol_uri_str
        }
      }
    }
    \exp_args:Nnnx
    \begin{stex_annotate_env}{sproof}{\seq_use:Nn \l_tmpa_seq {,}}
    \str_if_empty:NF \spftype {
      \stex_annotate_invisible:nnn{type}{\spftype}{}
    }

    \clist_set:No \l_tmpa_clist \spftype
    \tl_clear:N \l_tmpa_tl
    \clist_map_inline:Nn \l_tmpa_clist {
      \tl_if_exist:cT {__stex_sproof_sproof_##1_start:}{
        \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_start:}}
      }
      \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
        \tl_set:Nn \l_tmpa_tl {\use:n{}}
      }
    }
    \tl_if_empty:NTF \l_tmpa_tl {
      \__stex_sproof_sproof_start:
    }{
      \l_tmpa_tl
    }{~#2}
    \str_if_empty:NF \spfid {
      \stex_ref_new_doc_target:n \spfid
    }
    \begin{description}
  }
  \stex_smsmode_do:
}{
  \stex_if_smsmode:F{
    \end{description}
    \clist_set:No \l_tmpa_clist \spftype
    \tl_clear:N \l_tmpa_tl
    \clist_map_inline:Nn \l_tmpa_clist {
      \tl_if_exist:cT {__stex_sproof_sproof_##1_end:}{
        \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_end:}}
      }
    }
    \tl_if_empty:NTF \l_tmpa_tl {
      \__stex_sproof_sproof_end:
    }{
      \l_tmpa_tl
    }
    \end{stex_annotate_env}
  }
}

\cs_new_protected:Nn \__stex_sproof_sproof_start: {
  \par\noindent\titleemph{
    \tl_if_empty:NTF \spftype {
      \spf@proof@kw
    }{
      \spftype
    }
  }:
}
\cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}

\newcommand\stexpatchproof[3][] {
  \str_set:Nx \l_tmpa_str{ #1 }
  \str_if_empty:NTF \l_tmpa_str {
    \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
    \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
  }{
    \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
    \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
  }
}
```

\spfidea

```
\newcommand\spfidea[2][]{
  \__stex_sproof_spf_args:n{#1}
  \titleemph{
    \tl_if_empty:NTF \spftype {Proof~Idea}{
      \spftype
    }:
  }~#2
  \sproofend
}
```

(*End definition for* `\spfidea`*. This function is documented on page* **??**.)

The next two environments (proof steps) and comments, are mostly semantical, they take `KeyVal` arguments that specify their semantic role. In draft mode, they read these values and show them. If the surrounding proof had `display=flow`, then no new `\item` is generated, otherwise it is. In any case, the proof step number (at the current level) is incremented.

spfstep

```
\newenvironment{spfstep}[1][]{
  \__stex_sproof_spf_args:n{#1}
  \stex_if_smsmode:TF {
    \str_if_empty:NF \spfid {
      \stex_ref_new_doc_target:n \spfid
    }
  }{
    \@in@omtexttrue
    \seq_clear:N \l_tmpa_seq
    \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
      \tl_if_empty:nF{ ##1 }{
        \stex_get_symbol:n { ##1 }
        \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
          \l_stex_get_symbol_uri_str
        }
      }
    }
    \exp_args:Nnnx
    \begin{stex_annotate_env}{spfstep}{\seq_use:Nn \l_tmpa_seq {,}}
    \str_if_empty:NF \spftype {
      \stex_annotate_invisible:nnn{type}{\spftype}{}
    }
    \clist_set:No \l_tmpa_clist \spftype
    \tl_set:Nn \l_tmpa_tl {
      \item[\sproofnumber]
      \bool_set_true:N \l__stex_sproof_inc_counter_bool
    }
    \clist_map_inline:Nn \l_tmpa_clist {
      \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
        \tl_clear:N \l_tmpa_tl
      }
    }
    \l_tmpa_tl
    \tl_if_empty:NF \spftitle {
      {(\titleemph{\spftitle})\enspace}
    }
    \str_if_empty:NF \spfid {
      \stex_ref_new_doc_target:n \spfid
    }
  }
  \stex_smsmode_do:
  \ignorespacesandpars
}{
  \bool_if:NT \l__stex_sproof_inc_counter_bool {
    \__stex_sproof_inc_counter:
  }
  \stex_if_smsmode:F {
    \end{stex_annotate_env}
  }
}
```

sproofcomment

```
\newenvironment{sproofcomment}[1][]{
  \__stex_sproof_spf_args:n{#1}
  \clist_set:No \l_tmpa_clist \spftype
  \tl_set:Nn \l_tmpa_tl {
    \item[\sproofnumber]
    \bool_set_true:N \l__stex_sproof_inc_counter_bool
  }
  \clist_map_inline:Nn \l_tmpa_clist {
    \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
      \tl_clear:N \l_tmpa_tl
    }
  }
  \l_tmpa_tl
}{
  \bool_if:NT \l__stex_sproof_inc_counter_bool {
    \__stex_sproof_inc_counter:
  }
}
```

The next two environments also take a KeyVal argument, but also a regular one, which contains a start text. Both environments start a new numbered proof level.

subproof In the subproof environment, a new (lower-level) proproofof environment is started.

```
\newenvironment{subproof}[2][]{
  \__stex_sproof_spf_args:n{#1}
  \stex_if_smsmode:TF{
    \str_if_empty:NF \spfid {
      \stex_ref_new_doc_target:n \spfid
    }
  }{
    \seq_clear:N \l_tmpa_seq
    \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
      \tl_if_empty:nF{ ##1 }{
        \stex_get_symbol:n { ##1 }
        \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
          \l_stex_get_symbol_uri_str
        }
      }
    }
    \exp_args:Nnnx
    \begin{stex_annotate_env}{subproof}{\seq_use:Nn \l_tmpa_seq {,}}
    \str_if_empty:NF \spftype {
      \stex_annotate_invisible:nnn{type}{\spftype}{}
    }

    \clist_set:No \l_tmpa_clist \spftype
    \tl_set:Nn \l_tmpa_tl {
      \item[\sproofnumber]
      \bool_set_true:N \l__stex_sproof_inc_counter_bool
    }
    \clist_map_inline:Nn \l_tmpa_clist {
      \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
        \tl_clear:N \l_tmpa_tl
      }
    }
    \l_tmpa_tl
    \tl_if_empty:NF \spftitle {
      {(\titleemph{\spftitle})\enspace}
    }
    {~#2}
    \str_if_empty:NF \spfid {
      \stex_ref_new_doc_target:n \spfid
    }
  }
  \__stex_sproof_add_counter:
  \stex_smsmode_do:
}{
  \__stex_sproof_remove_counter:
  \bool_if:NT \l__stex_sproof_inc_counter_bool {
    \__stex_sproof_inc_counter:
  }
  \stex_if_smsmode:F{
    \end{stex_annotate_env}
  }
}
```

spfcases In the pfcases environment, the start text is displayed as the first comment of the proof.

```
\newenvironment{spfcases}[2][]{
  \tl_if_empty:nTF{#1}{
    \begin{subproof}[method=by-cases]{#2}
  }{
    \begin{subproof}[#1,method=by-cases]{#2}
  }
}{
  \end{subproof}
}
```

spfcase In the pfcase environment, the start text is displayed specification of the case after the \item

```
\newenvironment{spfcase}[2][]{
  \__stex_sproof_spf_args:n{#1}
  \stex_if_smsmode:TF {
    \str_if_empty:NF \spfid {
      \stex_ref_new_doc_target:n \spfid
    }
  }{
    \seq_clear:N \l_tmpa_seq
    \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
      \tl_if_empty:nF{ ##1 }{
        \stex_get_symbol:n { ##1 }
        \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
          \l_stex_get_symbol_uri_str
        }
      }
    }
    \exp_args:Nnnx
    \begin{stex_annotate_env}{spfcase}{\seq_use:Nn \l_tmpa_seq {,}}
    \str_if_empty:NF \spftype {
      \stex_annotate_invisible:nnn{type}{\spftype}{}
    }
    \clist_set:No \l_tmpa_clist \spftype
    \tl_set:Nn \l_tmpa_tl {
      \item[\sproofnumber]
      \bool_set_true:N \l__stex_sproof_inc_counter_bool
    }
    \clist_map_inline:Nn \l_tmpa_clist {
      \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
        \tl_clear:N \l_tmpa_tl
      }
    }
    \l_tmpa_tl
    \tl_if_empty:nF{#2}{
      \titleemph{#2}:~
    }
  }
  \__stex_sproof_add_counter:
  \stex_smsmode_do:
}{
  \__stex_sproof_remove_counter:
  \bool_if:NT \l__stex_sproof_inc_counter_bool {
    \__stex_sproof_inc_counter:
  }
  \stex_if_smsmode:F{
    \clist_set:No \l_tmpa_clist \spftype
    \tl_set:Nn \l_tmpa_tl{\sproofend}
    \clist_map_inline:Nn \l_tmpa_clist {
      \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
        \tl_clear:N \l_tmpa_tl
      }
    }
    \l_tmpa_tl
    \end{stex_annotate_env}
  }
}
```

`spfcase` similar to `spfcase`, takes a third argument.

```
\newcommand\spfcasesketch[3][]{
  \begin{spfcase}[#1]{#2}#3\end{spfcase}
}
```

33.3 Justifications

We define the actions that are undertaken, when the keys for justifications are encountered. Here this is very simple, we just define an internal macro with the value, so that we can use it later.

```
\keys_define:nn { stex / just }{
  id        .str_set_x:N  = \l__stex_sproof_just_id_str,
  method    .tl_set:N     = \l__stex_sproof_just_method_tl,
  premises  .tl_set:N     = \l__stex_sproof_just_premises_tl,
  args      .tl_set:N     = \l__stex_sproof_just_args_tl
}
```

The next three environments and macros are purely semantic, so we ignore the keyval arguments for now and only display the content.[11] EdN:11

[11] EdNote: need to do something about the premise in draft mode.

`justification`

```
\newenvironment{justification}[1][]{}{}
```

`\premise`

```
\newcommand\stex_proof_premise:[2][]{#2}
```

(*End definition for* `\premise`*. This function is documented on page* **??**.)

`\justarg` the `\justarg` macro is purely semantic, so we ignore the keyval arguments for now and only display the content.

```
\newcommand\justarg[2][]{#2}
⟨/package⟩
```

(*End definition for* `\justarg`*. This function is documented on page* **??**.)

Some auxiliary code, and clean up to be executed at the end of the package.

Chapter 34

sTeX -Others Implementation

```
⟨*package⟩

%%%%%%%%%%%%%   others.dtx   %%%%%%%%%%%%%

⟨@@=stex_others⟩
```

Warnings and error messages

```
  % None
```

\MSC Math subject classifier

```
\NewDocumentCommand \MSC {m} {
  % TODO
}
```

(*End definition for \MSC. This function is documented on page* ??.)

Patching tikzinput, if loaded

```
\@ifpackageloaded{tikzinput}{
  \RequirePackage{stex-tikzinput}
}{}
⟨/package⟩
```

Chapter 35

sTeX -Metatheory Implementation

```
⟨*package⟩
⟨@@=stex_modules⟩

%%%%%%%%%%%%%   metatheory.dtx   %%%%%%%%%%%%%

\str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
\begingroup
\stex_module_setup:nn{
  ns=\c_stex_metatheory_ns_str,
  meta=NONE
}{Metatheory}
\stex_reactivate_macro:N \symdecl
\stex_reactivate_macro:N \notation
\stex_reactivate_macro:N \symdef
\ExplSyntaxOff
\csname stex_suppress_html:n\endcsname{
  % is-a (a:A, a \in A, a is an A, etc.)
  \symdecl{isa}[args=ai]
  \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
  \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
  \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}

  % bind (\forall, \Pi, \lambda etc.)
  \symdecl{bind}[args=Bi]
  \notation{bind}[forall]{\comp\forall #1.\;#2}{##1 \comp, ##2}
  \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
  \notation{bind}[depfun]{\comp( #1 \comp{)\;\to\;} #2}{##1 \comp, ##2}

  % implicit bind
  \symdef{implicitbind}[args=Bi]{\comp\prod_{#1}#2}{##1\comp,##2}

  % dummy variable
  \symdecl{dummyvar}
  \notation{dummyvar}[underscore]{\comp\_}
  \notation{dummyvar}[dot]{\comp\cdot}
```

```
\notation{dummyvar}[dash]{\comp{{\rm --}}}

%fromto (function space, Hom-set, implication etc.)
\symdecl{fromto}[args=ai]
\notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
\notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}

% mapto (lambda etc.)
%\symdecl{mapto}[args=Bi]
%\notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
%\notation{mapto}[lambda]{\comp\lambda #1 \comp.\; #2}{#1 \comp, #2}
%\notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.\; #2}{#1 \comp, #2}

% function/operator application
\symdecl{apply}[args=ia]
\notation{apply}[prec=0;0x\infprec,parens]{#1 \comp( #2 \comp)}{##1 \comp, ##2}
\notation{apply}[prec=0;0x\infprec,lambda]{#1 \; #2 }{##1 \; ##2}

% collection of propositions/booleans/truth values
\symdecl{prop}[name=proposition]
\notation{prop}[prop]{\comp{{\rm prop}}}
\notation{prop}[BOOL]{\comp{{\rm BOOL}}}

\symdecl{judgmentholds}[args=1]
\notation{judgmentholds}[vdash,op=\vdash]{\comp\vdash\; #1}

% sequences
\symdecl{seqtype}[args=1]
\notation{seqtype}[kleene]{#1^{\comp\ast}}

\symdecl{seqexpr}[args=a]
\notation{seqexpr}[angle,prec=nobrackets]{\comp\langle #1\comp\rangle}{##1\comp,##2}

\symdef{sequence-index}[args=2,li,prec=nobrackets]{{#1}_{#2}}
\notation{sequence-index}[ui,prec=nobrackets]{{#1}^{#2}}

\symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{,\ellipses}}{##1\comp,##2}
\symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{,\ellipses,}#2}{##1\comp,##2}
\symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{,\ellipses,}#2\comp{,\ellipses,}#

% letin (‘‘let’’, local definitions, variable substitution)
\symdecl{letin}[args=bii]
\notation{letin}[let]{\comp{{\rm let}}\;#1\comp{=}#2\;\comp{{\rm in}}\;#3}
\notation{letin}[subst]{#3 \comp[ #1 \comp/ #2 \comp]}
\notation{letin}[frac]{#3 \comp[ \frac{#2}{#1} \comp]}

% structures
\symdecl*{module-type}[args=1]
\notation{module-type}{\comp{\mathtt{MOD}} #1}
\symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
\notation{mathstruct}[angle,prec=nobrackets]{\comp\langle #1 \comp\rangle}{##1 \comp, ##2}

% objects
\symdecl{object}
```

```
  \notation{object}{\comp{\mathtt{OBJECT}}}

}
  \ExplSyntaxOn
  \stex_add_to_current_module:n{
    \let\nappa\apply
    \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
    \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
    \def\livar{\csname sequence-index\endcsname[li]}
    \def\uivar{\csname sequence-index\endcsname[ui]}
    \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
    \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
    \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
  }
\__stex_modules_end_module:
\endgroup
```
⟨/package⟩

Chapter 36

Tikzinput Implementation

```
⟨*package⟩

%%%%%%%%%%%%%   tikzinput.dtx   %%%%%%%%%%%%%

\ProvidesExplPackage{tikzinput}{2022/02/26}{3.0.1}{tikzinput package}
\RequirePackage{l3keys2e}

\keys_define:nn { tikzinput } {
  image   .bool_set:N   = \c_tikzinput_image_bool,
  image   .default:n    = false ,
  unknown   .code:n       = {}
}

\ProcessKeysOptions { tikzinput }

\bool_if:NTF \c_tikzinput_image_bool {
  \RequirePackage{graphicx}

  \providecommand\usetikzlibrary[]{}
  \newcommand\tikzinput[2][]{\includegraphics[#1]{#2}}
}{
  \RequirePackage{tikz}
  \RequirePackage{standalone}

  \newcommand \tikzinput [2] [] {
    \setkeys{Gin}{#1}
    \ifx \Gin@ewidth \Gin@exclamation
      \ifx \Gin@eheight \Gin@exclamation
        \input { #2 }
      \else
        \resizebox{!}{ \Gin@eheight }{
          \input { #2 }
        }
      \fi
    \else
      \ifx \Gin@eheight \Gin@exclamation
        \resizebox{ \Gin@ewidth }{!}{
          \input { #2 }
      }
    \else
      \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
        \input { #2 }
      }
    \fi
  \fi
 }
}

\newcommand \ctikzinput [2] [] {
  \begin{center}
    \tikzinput [#1] {#2}
  \end{center}
}

\@ifpackageloaded{stex}{
  \RequirePackage{stex-tikzinput}
}{}

⟨/package⟩
⟨*stex⟩
\ProvidesExplPackage{stex-tikzinput}{2022/02/26}{3.0.1}{stex-tikzinput}
\RequirePackage{stex}
\RequirePackage{tikzinput}

\newcommand\mhtikzinput[2][]{%
  \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
  \stex_in_repository:nn\Gin@mhrepos{
    \tikzinput[#1]{\mhpath{##1}{#2}}
  }
}
\newcommand\cmhtikzinput[2][]{\begin{center}\mhtikzinput[#1]{#2}\end{center}}
⟨/stex⟩
```

LocalWords: bibfolder jobname.dtx tikzinput.dtx usetikzlibrary Gin@ewidth Gin@eheight
LocalWords: resizebox ctikzinput mhtikzinput Gin@mhrepos mhpath

Chapter 37

document-structure.sty Implementation

37.1 The document-structure Class

The functionality is spread over the `document-structure` class and package. The class provides the `document` environment and the `document-structure` element corresponds to it, whereas the package provides the concrete functionality.

```
⟨*cls⟩
⟨@@=document_structure⟩
\ProvidesExplClass{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure Class}
\RequirePackage{l3keys2e}
```

37.2 Class Options

`\omdoc@cls@class` To initialize the `document-structure` class, we declare and process the necessary options using the `kvoptions` package for key/value options handling. For `omdoc.cls` this is quite simple. We have options `report` and `book`, which set the `\omdoc@cls@class` macro and pass on the macro to `omdoc.sty` for further processing.

```
\keys_define:nn{ document-structure / pkg }{
  class       .str_set_x:N  = \c_document_structure_class_str,
  minimal     .bool_set:N   = \c_document_structure_minimal_bool,
  report      .code:n       = {
    \ClassWarning{document-structure}{the option 'report' is deprecated, use 'class=report',
    \str_set:Nn \c_document_structure_class_str {report}
  },
  book        .code:n       = {
    \ClassWarning{document-structure}{the option 'book' is deprecated, use 'class=book', ins
    \str_set:Nn \c_document_structure_class_str {book}
  },
  bookpart    .code:n       = {
    \ClassWarning{document-structure}{the option 'bookpart' is deprecated, use 'class=book,t
    \str_set:Nn \c_document_structure_class_str {book}
    \str_set:Nn \c_document_structure_topsect_str {chapter}
  },
  docopt      .str_set_x:N  = \c_document_structure_docopt_str,
  unknown     .code:n       = {
    \PassOptionsToPackage{ \CurrentOption }{ document-structure }
  }
}
\ProcessKeysOptions{ document-structure / pkg }
\str_if_empty:NT \c_document_structure_class_str {
  \str_set:Nn \c_document_structure_class_str {article}
}
\exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
  {\c_document_structure_class_str}

```

37.3 Beefing up the `document` environment

Now, – unless the option `minimal` is defined – we include the `stex` package

```
\RequirePackage{document-structure}
\bool_if:NF \c_document_structure_minimal_bool {
```

And define the environments we need. The top-level one is the `document` environment, which we redefined so that we can provide keyval arguments.

`document` For the moment we do not use them on the LaTeX level, but the document identifier is picked up by LaTeXML.[12] EdN:12

```
\keys_define:nn { document-structure / document }{
  id .str_set_x:N = \c_document_structure_document_id_str
}
\let\__document_structure_orig_document=\document
\renewcommand{\document}[1][]{
  \keys_set:nn{ document-structure / document }{ #1 }
  \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
  \__document_structure_orig_document
}
```

Finally, we end the test for the `minimal` option.

```
}
⟨/cls⟩
```

37.4 Implementation: document-structure Package

```
⟨*package⟩
\ProvidesExplPackage{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure}
\RequirePackage{l3keys2e}
```

37.5 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).

[12] EdNote: faking documentkeys for now. @HANG, please implement

```

\keys_define:nn{ document-structure / pkg }{
  class       .str_set_x:N  = \c_document_structure_class_str,
  topsect     .str_set_x:N  = \c_document_structure_topsect_str,
%  showignores .bool_set:N   = \c_document_structure_showignores_bool,
}
\ProcessKeysOptions{ document-structure / pkg }
\str_if_empty:NT \c_document_structure_class_str {
  \str_set:Nn \c_document_structure_class_str {article}
}
\str_if_empty:NT \c_document_structure_topsect_str {
  \str_set:Nn \c_document_structure_topsect_str {section}
}
```

Then we need to set up the packages by requiring the sref package to be loaded, and set up triggers for other languages

```
\RequirePackage{xspace}
\RequirePackage{comment}
\AddToHook{begindocument}{
\ltx@ifpackageloaded{babel}{
    \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
    \clist_if_in:NnT \l_tmpa_clist {ngerman}{
      \makeatletter\input{document-structure-ngerman.ldf}\makeatother
    }
  }{}
}
```

\section@level

Finally, we set the \section@level macro that governs sectioning. The default is two (corresponding to the article class), then we set the defaults for the standard classes book and report and then we take care of the levels passed in via the topsect option.

```
\int_new:N \l_document_structure_section_level_int
\str_case:VnF \c_document_structure_topsect_str {
  {part}{
    \int_set:Nn \l_document_structure_section_level_int {0}
  }
  {chapter}{
    \int_set:Nn \l_document_structure_section_level_int {1}
  }
}{
  \str_case:VnF \c_document_structure_class_str {
    {book}{
      \int_set:Nn \l_document_structure_section_level_int {0}
    }
    {report}{
      \int_set:Nn \l_document_structure_section_level_int {0}
    }
  }{
    \int_set:Nn \l_document_structure_section_level_int {2}
  }
}
```

37.6 Document Structure

The structure of the document is given by the `omgroup` environment just like in OMDoc. The hierarchy is adjusted automatically according to the LaTeX class in effect.

`\currentsectionlevel` For the `\currentsectionlevel` and `\Currentsectionlevel` macros we use an internal macro `\current@section@level` that only contains the keyword (no markup). We initialize it with "document" as a default. In the generated OMDoc, we only generate a text element of class `omdoc_currentsectionlevel`, wich will be instantiated by CSS later.[13] EdN:13

```
\def\current@section@level{document}%
\newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
\newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

(*End definition for* `\currentsectionlevel`. *This function is documented on page* **??**.)

`\skipomgroup`

```
\cs_new_protected:Npn \skipomgroup {
  \ifcase\l_document_structure_section_level_int
  \or\stepcounter{part}
  \or\stepcounter{chapter}
  \or\stepcounter{section}
  \or\stepcounter{subsection}
  \or\stepcounter{subsubsection}
  \or\stepcounter{paragraph}
  \or\stepcounter{subparagraph}
  \fi
}
```

(*End definition for* `\skipomgroup`. *This function is documented on page* **??**.)

`blindfragment`

```
\newcommand\at@begin@blindomgroup[1]{}
\newenvironment{blindfragment}
{
  \int_incr:N\l_document_structure_section_level_int
  \at@begin@blindomgroup\l_document_structure_section_level_int
}{}
```

`\omgroup@nonum` convenience macro: `\omgroup@nonum{`⟨*level*⟩`}{`⟨*title*⟩`}` makes an unnumbered sectioning with title ⟨*title*⟩ at level ⟨*level*⟩.

```
\newcommand\omgroup@nonum[2]{
  \ifx\hyper@anchor\@undefined\else\phantomsection\fi
  \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
}
```

(*End definition for* `\omgroup@nonum`. *This function is documented on page* **??**.)

`\omgroup@num` convenience macro: `\omgroup@nonum{`⟨*level*⟩`}{`⟨*title*⟩`}` makes numbered sectioning with title ⟨*title*⟩ at level ⟨*level*⟩. We have to check the `short` key was given in the `omgroup` environment and – if it is use it. But how to do that depends on whether the `rdfmeta` package has been loaded. In the end we call `\sref@label@id` to enable crossreferencing.

```
\newcommand\omgroup@num[2]{
```

[13] EdNote: MK: we may have to experiment with the more powerful uppercasing macro from `mfirstuc.sty` once we internationalize.

```
  \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
    \@nameuse{#1}{#2}
  }{
    \cs_if_exist:NTF\rdfmeta@sectioning{
      \@nameuse{rdfmeta@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
    }{
      \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
    }
  }
%\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\omgroup@id
}
```

(*End definition for* `\omgroup@num`*. This function is documented on page* **??**.)

`sfragment`

```
\keys_define:nn { document-structure / omgroup }{
  id            .str_set_x:N = \l__document_structure_omgroup_id_str,
  date          .str_set_x:N = \l__document_structure_omgroup_date_str,
  creators      .clist_set:N = \l__document_structure_omgroup_creators_clist,
  contributors  .clist_set:N = \l__document_structure_omgroup_contributors_clist,
  srccite       .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
  type          .tl_set:N    = \l__document_structure_omgroup_type_tl,
  short         .tl_set:N    = \l__document_structure_omgroup_short_tl,
  display       .tl_set:N    = \l__document_structure_omgroup_display_tl,
  intro         .tl_set:N    = \l__document_structure_omgroup_intro_tl,
  loadmodules   .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
}
\cs_new_protected:Nn \__document_structure_omgroup_args:n {
  \str_clear:N \l__document_structure_omgroup_id_str
  \str_clear:N \l__document_structure_omgroup_date_str
  \clist_clear:N \l__document_structure_omgroup_creators_clist
  \clist_clear:N \l__document_structure_omgroup_contributors_clist
  \tl_clear:N \l__document_structure_omgroup_srccite_tl
  \tl_clear:N \l__document_structure_omgroup_type_tl
  \tl_clear:N \l__document_structure_omgroup_short_tl
  \tl_clear:N \l__document_structure_omgroup_display_tl
  \tl_clear:N \l__document_structure_omgroup_intro_tl
  \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
  \keys_set:nn { document-structure / omgroup } { #1 }
}
```

`\at@begin@omgroup`

we define a switch for numbering lines and a hook for the beginning of groups: The `\at@begin@omgroup` macro allows customization. It is run at the beginning of the `omgroup`, i.e. after the section heading.

```
\newif\if@mainmatter\@mainmattertrue
\newcommand\at@begin@omgroup[3][]{}
```

Then we define a helper macro that takes care of the sectioning magic. It comes with its own key/value interface for customization.

```
\keys_define:nn { document-structure / sectioning }{
  name    .str_set_x:N  = \l__document_structure_sect_name_str   ,
  ref     .str_set_x:N  = \l__document_structure_sect_ref_str    ,
  clear   .bool_set:N   = \l__document_structure_sect_clear_bool ,
  clear   .default:n    = {true}                    ,
  num     .bool_set:N   = \l__document_structure_sect_num_bool   ,
  num      .default:n    = {true}
}
\cs_new_protected:Nn \__document_structure_sect_args:n {
  \str_clear:N \l__document_structure_sect_name_str
  \str_clear:N \l__document_structure_sect_ref_str
  \bool_set_false:N \l__document_structure_sect_clear_bool
  \bool_set_false:N \l__document_structure_sect_num_bool
  \keys_set:nn { document-structure / sectioning } { #1 }
}
\newcommand\omdoc@sectioning[3][]{
  \__document_structure_sect_args:n {#1 }
  \let\omdoc@sect@name\l__document_structure_sect_name_str
  \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
  \if@mainmatter% numbering not overridden by frontmatter, etc.
    \bool_if:NTF \l__document_structure_sect_num_bool {
      \omgroup@num{#2}{#3}
    }{
      \omgroup@nonum{#2}{#3}
    }
    \def\current@section@level{\omdoc@sect@name}
  \else
    \omgroup@nonum{#2}{#3}
  \fi
}% if@mainmatter
```

and another one, if redefines the `\addtocontentsline` macro of LaTeX to import the respective macros. It takes as an argument a list of module names.

```
\newcommand\omgroup@redefine@addtocontents[1]{%
%\edef\__document_structureimport{#1}%
%\@for\@I:=\__document_structureimport\do{%
%\edef\@path{\csname module@\@I  @path\endcsname}%
%\@ifundefined{tf@toc}\relax%
%      {\protected@write\tf@toc{}{\string\@requiremodules{\@path}}}}
%\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
%\def\addcontentsline##1##2##3{%
%\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}
%\else% hyperref.sty not loaded
%\def\addcontentsline##1##2##3{%
%\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}{
%\fi
}% hypreref.sty loaded?
```

now the `omgroup` environment itself. This takes care of the table of contents via the helper macro above and then selects the appropriate sectioning command from `article.cls`. It also registeres the current level of omgroups in the `\omgroup@level` counter.

```
\newenvironment{sfragment}[2][]% keys, title
{
  \__document_structure_omgroup_args:n { #1 }%\sref@target%
```

If the `loadmodules` key is set on `\begin{sfragment}`, we redefine the `\addcontetsline` macro that determines how the sectioning commands below construct the entries for the table of contents.

```
  \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
    \omgroup@redefine@addtocontents{
      %\@ifundefined{module@id}\used@modules%
      %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}
    }
  }
```

now we only need to construct the right sectioning depending on the value of `\section@level`.

```
  \int_incr:N\l_document_structure_section_level_int
  \ifcase\l_document_structure_section_level_int
    \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
    \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
    \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
    \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
    \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
    \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#
    \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{paragr
  \fi
  \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
  \str_if_empty:NF \l__document_structure_omgroup_id_str {
    \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
  }
}% for customization
{}
```

and finally, we localize the sections

```
\newcommand\omdoc@part@kw{Part}
\newcommand\omdoc@chapter@kw{Chapter}
\newcommand\omdoc@section@kw{Section}
\newcommand\omdoc@subsection@kw{Subsection}
\newcommand\omdoc@subsubsection@kw{Subsubsection}
\newcommand\omdoc@paragraph@kw{paragraph}
\newcommand\omdoc@subparagraph@kw{subparagraph}
```

37.7 Front and Backmatter

Index markup is provided by the `omtext` package [Koh20c], so in the `document-structure` package we only need to supply the corresponding `\printindex` command, if it is not already defined

`\printindex`

```
\providecommand\printindex{\IfFileExists{\jobname.ind}{\input{\jobname.ind}}{}}
```

(*End definition for* `\printindex`. *This function is documented on page* ??.)

some classes (e.g. `book.cls`) already have `\frontmatter`, `\mainmatter`, and `\backmatter` macros. As we want to define `frontmatter` and `backmatter` environments, we save their behavior (possibly defining it) in `orig@*matter` macros and make them undefined (so that we can define the environments).

```
\cs_if_exist:NTF\frontmatter{
  \let\__document_structure_orig_frontmatter\frontmatter
  \let\frontmatter\relax
}{
  \tl_set:Nn\__document_structure_orig_frontmatter{
    \clearpage
    \@mainmatterfalse
    \pagenumbering{roman}
  }
}
\cs_if_exist:NTF\backmatter{
  \let\__document_structure_orig_backmatter\backmatter
  \let\backmatter\relax
}{
  \tl_set:Nn\__document_structure_orig_backmatter{
    \clearpage
    \@mainmatterfalse
    \pagenumbering{roman}
  }
}
```

Using these, we can now define the `frontmatter` and `backmatter` environments

`frontmatter` we use the `\orig@frontmatter` macro defined above and `\mainmatter` if it exists, otherwise we define it.

```
\newenvironment{frontmatter}{
  \__document_structure_orig_frontmatter
}{
  \cs_if_exist:NTF\mainmatter{
    \mainmatter
  }{
    \clearpage
    \@mainmattertrue
    \pagenumbering{arabic}
  }
}
```

`backmatter` As backmatter is at the end of the document, we do nothing for `\endbackmatter`.

```
\newenvironment{backmatter}{
  \__document_structure_orig_backmatter
}{
  \cs_if_exist:NTF\mainmatter{
    \mainmatter
  }{
    \clearpage
    \@mainmattertrue
    \pagenumbering{arabic}
  }
}
```

finally, we make sure that page numbering is arabic and we have main matter as the default

```
\@mainmattertrue\pagenumbering{arabic}
```

`\prematurestop` We initialize `\afterprematurestop`, and provide `\prematurestop@endomgroup` which looks up `\omgroup@level` and recursively ends enough `{sfragment}`s.

```
\def \c__document_structure_document_str{document}
\newcommand\afterprematurestop{}
\def\prematurestop@endomgroup{
  \unless\ifx\@currenvir\c__document_structure_document_str
    \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafte
    \expandafter\prematurestop@endomgroup
  \fi
}
\providecommand\prematurestop{
  \message{Stopping~sTeX~processing~prematurely}
  \prematurestop@endomgroup
  \afterprematurestop
  \end{document}
}
```

(*End definition for* `\prematurestop`. *This function is documented on page* **??**.)

37.8 Global Variables

`\setSGvar` set a global variable

```
\RequirePackage{etoolbox}
\newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}
```

(*End definition for* `\setSGvar`. *This function is documented on page* **??**.)

`\useSGvar` use a global variable

```
\newrobustcmd\useSGvar[1]{%
  \@ifundefined{sTeX@Gvar@#1}
  {\PackageError{document-structure}
    {The sTeX Global variable #1 is undefined}
    {set it with \protect\setSGvar}}
\@nameuse{sTeX@Gvar@#1}}
```

(*End definition for* `\useSGvar`. *This function is documented on page* **??**.)

`\ifSGvar` execute something conditionally based on the state of the global variable.

```
\newrobustcmd\ifSGvar[3]{\def\@test{#2}%
  \@ifundefined{sTeX@Gvar@#1}
  {\PackageError{document-structure}
    {The sTeX Global variable #1 is undefined}
    {set it with \protect\setSGvar}}
  {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}
```

(*End definition for* `\ifSGvar`. *This function is documented on page* **??**.)

Chapter 38

NotesSlides – Implementation

38.1 Class and Package Options

We define some Package Options and switches for the notesslides class and activate them by passing them on to beamer.cls and omdoc.cls and the notesslides package. We pass the nontheorem option to the statements package when we are not in notes mode, since the beamer package has its own (overlay-aware) theorem environments.

```
⟨*cls⟩
⟨@@=notesslides⟩
\ProvidesExplClass{notesslides}{2022/02/28}{3.1.0}{notesslides Class}
\RequirePackage{l3keys2e}

\keys_define:nn{notesslides / cls}{
  class   .code:n   = {
    \PassOptionsToClass{\CurrentOption}{document-structure}
    \str_if_eq:nnT{#1}{book}{
      \PassOptionsToPackage{defaulttopsec=part}{notesslides}
    }
    \str_if_eq:nnT{#1}{report}{
      \PassOptionsToPackage{defaulttopsec=part}{notesslides}
    }
  },
  notes   .bool_set:N  = \c__notesslides_notes_bool ,
  slides  .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
  unknown .code:n      = {
    \PassOptionsToClass{\CurrentOption}{document-structure}
    \PassOptionsToClass{\CurrentOption}{beamer}
    \PassOptionsToPackage{\CurrentOption}{notesslides}
  }
}
\ProcessKeysOptions{ notesslides / cls }
\bool_if:NTF \c__notesslides_notes_bool {
  \PassOptionsToPackage{notes=true}{notesslides}
}{
  \PassOptionsToPackage{notes=false}{notesslides}
}
⟨/cls⟩
```

now we do the same for the `notesslides` package.

```
⟨*package⟩
\ProvidesExplPackage{notesslides}{2022/02/28}{3.1.0}{notesslides Package}
\RequirePackage{l3keys2e}

\keys_define:nn{notesslides / pkg}{
  topsect         .str_set_x:N  = \c__notesslides_topsect_str,
  defaulttopsect  .str_set_x:N  = \c__notesslides_defaulttopsec_str,
  notes           .bool_set:N   = \c__notesslides_notes_bool ,
  slides          .code:n       = { \bool_set_false:N \c__notesslides_notes_bool },
  sectocframes    .bool_set:N   = \c__notesslides_sectocframes_bool ,
  frameimages     .bool_set:N   = \c__notesslides_frameimages_bool ,
  fiboxed         .bool_set:N   = \c__notesslides_fiboxed_bool ,
  noproblems      .bool_set:N   = \c__notesslides_noproblems_bool,
  unknown         .code:n       = {
    \PassOptionsToClass{\CurrentOption}{stex}
    \PassOptionsToClass{\CurrentOption}{tikzinput}
  }
}
\ProcessKeysOptions{ notesslides / pkg }
\newif\ifnotes
\bool_if:NTF \c__notesslides_notes_bool {
  \notestrue
}{
  \notesfalse
}

```

we give ourselves a macro `\@@topsect` that needs only be evaluated once, so that the `\ifdefstring` conditionals work below.

```
\str_if_empty:NTF \c__notesslides_topsect_str {
  \str_set_eq:NN \__notesslidestopsect \c__notesslides_defaulttopsec_str
}{
  \str_set_eq:NN \__notesslidestopsect \c__notesslides_topsect_str
}
⟨/package⟩
```

Depending on the options, we either load the `article`-based `document-structure` or the `beamer` class (and set some counters).

```
⟨*cls⟩
\bool_if:NTF \c__notesslides_notes_bool {
  \LoadClass{document-structure}
}{
  \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
  \newcounter{Item}
  \newcounter{paragraph}
  \newcounter{subparagraph}
  \newcounter{Hfootnote}
  \RequirePackage{document-structure}
}
```

now it only remains to load the `notesslides` package that does all the rest.

```
\RequirePackage{notesslides}
⟨/cls⟩
```

In `notes` mode, we also have to make the `beamer`-specific things available to `article` via the `beamerarticle` package. We use options to avoid loading theorem-like environments, since we want to use our own from the sTeX packages. The first batch of packages we want are loaded on `notesslides.sty`. These are the general ones, we will load the sTeX-specific ones after we have done some work (e.g. defined the counters `m*`). Only the `stex-logo` package is already needed now for the default theme.

```
⟨*package⟩
\bool_if:NT \c__notesslides_notes_bool {
  \RequirePackage{a4wide}
  \RequirePackage{marginnote}
  \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
  \RequirePackage{mdframed}
  \RequirePackage[noxcolor,noamsthm]{beamerarticle}
  \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
}
\RequirePackage{stex-tikzinput}
\RequirePackage{etoolbox}
\RequirePackage{amssymb}
\RequirePackage{amsmath}
\RequirePackage{comment}
\RequirePackage{textcomp}
\RequirePackage{url}
\RequirePackage{graphicx}
\RequirePackage{pgf}
```

38.2 Notes and Slides

For the lecture notes cases, we also provide the `\usetheme` macro that would otherwise come from the the `beamer` class. While the latter loads `beamertheme`⟨*theme*⟩.sty, the notes version loads `beamernotestheme`⟨*theme*⟩`.sty`.[14]

EdN:14

```
\bool_if:NT \c__notesslides_notes_bool {
  \renewcommand\usetheme[2][]{\usepackage[#1]{beamernotestheme#2}}
}


\NewDocumentCommand \libusetheme {O{} m} {
  \bool_if:NTF \c__notesslides_notes_bool {
    \libusepackage[#1]{beamernotestheme#2}
  }{
  \libusepackage[#1]{beamertheme#2}
  }
}
```

We define the sizes of slides in the notes. Somehow, we cannot get by with the same here.

```
\newcounter{slide}
\newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
\newlength{\slideheight}\setlength{\slideheight}{9cm}
```

[14]EdNote: MK: This is not ideal, but I am not sure that I want to be able to provide the full theme functionality there.

note The `note` environment is used to leave out text in the `slides` mode. It does not have a counterpart in OMDoc. So for course notes, we define the `note` environment to be a no-operation otherwise we declare the `note` environment as a comment via the `comment` package.

```
\bool_if:NTF \c__notesslides_notes_bool {
  \renewenvironment{note}{\ignorespaces}{}
}{
  \excludecomment{note}
}
```

We first set up the slide boxes in `article` mode. We set up sizes and provide a box register for the frames and a counter for the slides.

```
\bool_if:NT \c__notesslides_notes_bool {
  \newlength{\slideframewidth}
  \setlength{\slideframewidth}{1.5pt}
```

frame We first define the keys.

```
  \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
    \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
      \bool_set_true:N #1
    }{
      \bool_set_false:N #1
    }
  }
  \keys_define:nn{notesslides / frame}{
    label              .str_set_x:N  = \l__notesslides_frame_label_str,
    allowframebreaks   .code:n       = {
      \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
    },
    allowdisplaybreaks  .code:n       = {
      \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
    },
    fragile            .code:n       = {
      \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
    },
    shrink             .code:n       = {
      \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
    },
    squeeze            .code:n       = {
      \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
    },
    t                  .code:n       = {
      \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
    },
  }
  \cs_new_protected:Nn \__notesslides_frame_args:n {
    \str_clear:N \l__notesslides_frame_label_str
    \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
    \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
    \bool_set_true:N \l__notesslides_frame_fragile_bool
    \bool_set_true:N \l__notesslides_frame_shrink_bool
    \bool_set_true:N \l__notesslides_frame_squeeze_bool
    \bool_set_true:N \l__notesslides_frame_t_bool
    \keys_set:nn { notesslides / frame }{ #1 }
  }
```

We define the environment, read them, and construct the slide number and label.

```
  \renewenvironment{frame}[1][]{
    \__notesslides_frame_args:n{#1}
    \sffamily
    \stepcounter{slide}
    \def\@currentlabel{\theslide}
    \str_if_empty:NF \l__notesslides_frame_label_str {
      \label{\l__notesslides_frame_label_str}
    }
```

We redefine the itemize environment so that it looks more like the one in beamer.

```
    \def\itemize@level{outer}
    \def\itemize@outer{outer}
    \def\itemize@inner{inner}
    \renewcommand\newpage{\addtocounter{framenumber}{1}}
    \newcommand\metakeys@show@keys[2]{\marginnote{{\scriptsize ##2}}}
    \renewenvironment{itemize}{
      \ifx\itemize@level\itemize@outer
        \def\itemize@label{$\rhd$}
      \fi
      \ifx\itemize@level\itemize@inner
        \def\itemize@label{$\scriptstyle\rhd$}
      \fi
      \begin{list}
      {\itemize@label}
      {\setlength{\labelsep}{.3em}
       \setlength{\labelwidth}{.5em}
       \setlength{\leftmargin}{1.5em}
      }
      \edef\itemize@level{\itemize@inner}
    }{
      \end{list}
    }
```

We create the box with the mdframed environment from the equinymous package.

```
    \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth
  }{
    \medskip\miko@slidelabel\end{mdframed}
  }
```

Now, we need to redefine the frametitle (we are still in course notes mode).

\frametitle

```
  \renewcommand{\frametitle}[1]{{\Large\bf\sf\color{blue}{#1}}\medskip}
}
```

(*End definition for* \frametitle. *This function is documented on page* **??**.)

EdN:15 \pause [15]

```
\bool_if:NT \c__notesslides_notes_bool {
  \newcommand\pause{}
}
```

[15]EdNote: MK: fake it in notes mode for now

(*End definition for* `\pause`*. This function is documented on page* **??**.)

nparagraph

```
\bool_if:NTF \c__notesslides_notes_bool {
  \newenvironment{nparagraph}[1][]{\begin{sparagraph}[#1]}{\end{sparagraph}}
}{
  \excludecomment{nparagraph}
}
```

nfragment

```
\bool_if:NTF \c__notesslides_notes_bool {
  \newenvironment{nfragment}[2][]{\begin{sfragment}[#1]{#2}}{\end{sfragment}}
}{
  \excludecomment{nfragment}
}
```

ndefinition

```
\bool_if:NTF \c__notesslides_notes_bool {
  \newenvironment{ndefinition}[1][]{\begin{sdefinition}[#1]}{\end{sdefinition}}
}{
  \excludecomment{ndefinition}
}
```

nassertion

```
\bool_if:NTF \c__notesslides_notes_bool {
  \newenvironment{nassertion}[1][]{\begin{sassertion}[#1]}{\end{sassertion}}
}{
  \excludecomment{nassertion}
}
```

nsproof

```
\bool_if:NTF \c__notesslides_notes_bool {
  \newenvironment{nproof}[2][]{\begin{sproof}[#1]{#2}}{\end{sproof}}
}{
  \excludecomment{nproof}
}
```

nexample

```
\bool_if:NTF \c__notesslides_notes_bool {
  \newenvironment{nexample}[1][]{\begin{sexample}[#1]}{\end{sexample}}
}{
  \excludecomment{nexample}
}
```

`\inputref@*skip` We customize the hooks for in `\inputref`.

```
\def\inputref@preskip{\smallskip}
\def\inputref@postskip{\medskip}
```

(*End definition for* `\inputref@*skip`*. This function is documented on page* **??**.)

`\inputref*`

```
\let\orig@inputref\inputref
\def\inputref{\@ifstar\ninputref\orig@inputref}
\newcommand\ninputref[2][]{
  \bool_if:NT \c__notesslides_notes_bool {
    \orig@inputref[#1]{#2}
  }
}
```

(*End definition for* `\inputref*`. *This function is documented on page* **??**.)

38.3 Header and Footer Lines

Now, we set up the infrastructure for the footer line of the slides, we use boxes for the logos, so that they are only loaded once, that considerably speeds up processing.

`\setslidelogo` The default logo is the sTEX logo. Customization can be done by `\setslidelogo{`⟨*logo name*⟩`}`.

```
\newlength{\slidelogoheight}

\bool_if:NTF \c__notesslides_notes_bool {
  \setlength{\slidelogoheight}{.4cm}
}{
  \setlength{\slidelogoheight}{1cm}
}
\newsavebox{\slidelogo}
\sbox{\slidelogo}{\sTeX}
\newrobustcmd{\setslidelogo}[1]{
  \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
}
```

(*End definition for* `\setslidelogo`. *This function is documented on page* **??**.)

`\setsource` `\source` stores the writer's name. By default it is *Michael Kohlhase* since he is the main user and designer of this package. `\setsource{`⟨*name*⟩`}` can change the writer's name.

```
\def\source{Michael Kohlhase}% customize locally
\newrobustcmd{\setsource}[1]{\def\source{#1}}
```

(*End definition for* `\setsource`. *This function is documented on page* **??**.)

`\setlicensing` Now, we set up the copyright and licensing. By default we use the Creative Commons Attribuition-ShareAlike license to strengthen the public domain. If package `hyperref` is loaded, then we can attach a hyperlink to the license logo. `\setlicensing[`⟨*url*⟩`]{`⟨*logo name*⟩`}` is used for customization, where ⟨*url*⟩ is optional.

```
\def\copyrightnotice{\footnotesize\copyright :\hspace{.3ex}{\source}}
\newsavebox{\cclogo}
\sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{stex-cc_somerights}}
\newif\ifcchref\cchreffalse
\AtBeginDocument{
  \@ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
}
\def\licensing{
  \ifcchref
```

```
    \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
  \else
    {\usebox{\cclogo}}
  \fi
}
\newrobustcmd{\setlicensing}[2][]{
  \def\@url{#1}
  \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
  \ifx\@url\@empty
    \def\licensing{{\usebox{\cclogo}}}
  \else
    \def\licensing{
      \ifcchref
      \href{#1}{\usebox{\cclogo}}
      \else
      {\usebox{\cclogo}}
      \fi
    }
  \fi
}
```

(*End definition for* `\setlicensing`. *This function is documented on page* **??**.)

EdN:16 `\slidelabel` Now, we set up the slide label for the `article` mode.[16]

```
\newrobustcmd\miko@slidelabel{
  \vbox to \slidelogoheight{
    \vss\hbox to \slidewidth
    {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
  }
}
```

(*End definition for* `\slidelabel`. *This function is documented on page* **??**.)

38.4 Frame Images

`\frameimage` We have to make sure that the width is overwritten, for that we check the `\Gin@ewidth` macro from the `graphicx` package. We also add the `label` key.

```
\def\Gin@mhrepos{}
\define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
\define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
\newrobustcmd\frameimage[2][]{
  \stepcounter{slide}
  \bool_if:NT \c__notesslides_frameimages_bool {
    \def\Gin@ewidth{}\setkeys{Gin}{#1}
    \bool_if:NF \c__notesslides_notes_bool { \vfill }
    \begin{center}
      \bool_if:NTF \c__notesslides_fiboxed_bool {
        \fbox{
          \ifx\Gin@ewidth\@empty
            \ifx\Gin@mhrepos\@empty
              \mhgraphics[width=\slidewidth,#1]{#2}
            \else
```

[16] EdNote: see that we can use the themes for the slides some day. This is all fake.

```
          \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
        \fi
      \else% Gin@ewidth empty
        \ifx\Gin@mhrepos\@empty
          \mhgraphics[#1]{#2}
        \else
          \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
        \fi
      \fi% Gin@ewidth empty
    }
  }{
    \ifx\Gin@ewidth\@empty
      \ifx\Gin@mhrepos\@empty
        \mhgraphics[width=\slidewidth,#1]{#2}
      \else
        \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
      \fi
      \ifx\Gin@mhrepos\@empty
        \mhgraphics[#1]{#2}
      \else
        \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
      \fi
    \fi% Gin@ewidth empty
  }
  \end{center}
  \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
  \bool_if:NF \c__notesslides_notes_bool { \vfill }
}
} % ifmks@sty@frameimages
```

(*End definition for* `\frameimage`. *This function is documented on page* **??**.)

38.5 Colors and Highlighting

We first specify sans serif fonts as the default.

```
\sffamily
```

Now, we set up an infrastructure for highlighting phrases in slides. Note that we use content-oriented macros for highlighting rather than directly using color markup. The first thing to to is to adapt the green so that it is dark enough for most beamers

```
\AddToHook{begindocument}{
  \definecolor{green}{rgb}{0,.5,0}
  \definecolor{purple}{cmyk}{.3,1,0,.17}
}
```

We customize the `\defemph`, `\symrefemph`, `\compemph`, and `\titleemph` macros with colors. Furthermore we customize the `\__omtextlec` macro for the appearance of line end comments in `\lec`.

```
% \def\STpresent#1{\textcolor{blue}{#1}}
\def\defemph#1{{\textcolor{magenta}{#1}}}
\def\symrefemph#1{{\textcolor{cyan}{#1}}}
\def\compemph#1{{\textcolor{blue}{#1}}}
\def\titleemph#1{{\textcolor{blue}{#1}}}
\def\__omtext_lec#1{(\textcolor{green}{#1})}
```

I like to use the dangerous bend symbol for warnings, so we provide it here.

`\textwarning` as the macro can be used quite often we put it into a box register, so that it is only loaded once.

```
\pgfdeclareimage[width=.8em]{miko@small@dbend}{stex-dangerous-bend}
\def\smalltextwarning{
  \pgfuseimage{miko@small@dbend}
  \xspace
}
\pgfdeclareimage[width=1.2em]{miko@dbend}{stex-dangerous-bend}
\newrobustcmd\textwarning{
  \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
  \xspace
}
\pgfdeclareimage[width=2.5em]{miko@big@dbend}{stex-dangerous-bend}
\newrobustcmd\bigtextwarning{
  \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
  \xspace
}
```

(*End definition for* `\textwarning`. *This function is documented on page* **??**.)

```
\newrobustcmd\putgraphicsat[3]{
  \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
}
\newrobustcmd\putat[2]{
  \begin{picture}(0,0)\put(#1){#2}\end{picture}
}
```

38.6 Sectioning

If the `sectocframes` option is set, then we make section frames. We first define counters for `part` and `chapter`, which `beamer.cls` does not have and we make the `section` counter which it does dependent on `chapter`.

```
\bool_if:NT \c__notesslides_sectocframes_bool {
  \str_if_eq:VnTF \__notesslidestopsect{part}{
    \newcounter{chapter}\counterwithin*{section}{chapter}
  }{
    \str_if_eq:VnT\__notesslidestopsect{chapter}{
      \newcounter{chapter}\counterwithin*{section}{chapter}
    }
  }
}
```

`\section@level` We set the `\section@level` counter that governs sectioning according to the class options. We also introduce the sectioning counters accordingly.

`\section@level`

```
\def\part@prefix{}
\@ifpackageloaded{document-structure}{}{
  \str_case:VnF \__notesslidestopsect {
    {part}{
      \int_set:Nn \l_document_structure_section_level_int {0}
      \def\thesection{\arabic{chapter}.\arabic{section}}
      \def\part@prefix{\arabic{chapter}.}
    }
    {chapter}{
      \int_set:Nn \l_document_structure_section_level_int {1}
      \def\thesection{\arabic{chapter}.\arabic{section}}
      \def\part@prefix{\arabic{chapter}.}
    }
  }{
    \int_set:Nn \l_document_structure_section_level_int {2}
    \def\part@prefix{}
  }
}

\bool_if:NF \c__notesslides_notes_bool { % only in slides
```

(*End definition for* `\section@level`*. This function is documented on page* **??**.)

The new counters are used in the `omgroup` environment that choses the LaTeX sectioning macros according to `\section@level`.

`sfragment`

```
  \renewenvironment{sfragment}[2][]{
    \__document_structure_omgroup_args:n { #1 }
    \int_incr:N \l_document_structure_section_level_int
    \bool_if:NT \c__notesslides_sectocframes_bool {
      \stepcounter{slide}
      \begin{frame}[noframenumbering]
      \vfill\Large\centering
      \red{
        \ifcase\l_document_structure_section_level_int\or
          \stepcounter{part}
          \def\__notesslideslabel{\omdoc@part@kw~\Roman{part}}
          \def\currentsectionlevel{\omdoc@part@kw}
        \or
          \stepcounter{chapter}
          \def\__notesslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
          \def\currentsectionlevel{\omdoc@chapter@kw}
        \or
          \stepcounter{section}
          \def\__notesslideslabel{\part@prefix\arabic{section}}
          \def\currentsectionlevel{\omdoc@section@kw}
        \or
          \stepcounter{subsection}
          \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
          \def\currentsectionlevel{\omdoc@subsection@kw}
        \or
          \stepcounter{subsubsection}
          \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
          \def\currentsectionlevel{\omdoc@subsubsection@kw}
        \or
          \stepcounter{paragraph}
          \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
          \def\currentsectionlevel{\omdoc@paragraph@kw}
        \else
          \def\__notesslideslabel{}
          \def\currentsectionlevel{\omdoc@paragraph@kw}
        \fi% end ifcase
        \__notesslideslabel%\sref@label@id\__notesslideslabel
        \quad #2%
      }%
      \vfill%
      \end{frame}%
    }
    \str_if_empty:NF \l__document_structure_omgroup_id_str {
      \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
    }
  }{}
}
```

We set up a beamer template for theorems like ams style, but without a block environment.

```
\def\inserttheorembodyfont{\normalfont}
%\bool_if:NF \c__notesslides_notes_bool {
%  \defbeamertemplate{theorem begin}{miko}
%  {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
%    \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
%    \inserttheorempunctuation\inserttheorembodyfont\xspace}
%  \defbeamertemplate{theorem end}{miko}{}
```

and we set it as the default one.

```
%  \setbeamertemplate{theorems}[miko]
```

The following fixes an error I do not understand, this has something to do with beamer compatibility, which has similar definitions but only up to 1.

```
%  \expandafter\def\csname Parent2\endcsname{}
%}

\AddToHook{begindocument}{ % this does not work for some reasone
  \setbeamertemplate{theorems}[ams style]
}
\bool_if:NT \c__notesslides_notes_bool {
  \renewenvironment{columns}[1][]{%
    \par\noindent%
    \begin{minipage}%
    \slidewidth\centering\leavevmode%
  }{%
    \end{minipage}\par\noindent%
  }%
  \newsavebox\columnbox%
  \renewenvironment<>{column}[2][]{%
    \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
  }{%
    \end{minipage}\end{lrbox}\usebox\columnbox%
  }%
}
\bool_if:NTF \c__notesslides_noproblems_bool {
  \newenvironment{problems}{}{}
}{
  \excludecomment{problems}
}
```

38.7 Excursions

`\excursion` The excursion macros are very simple, we define a new internal macro `\excursionref` and use it in `\excursion`, which is just an `\inputref` that checks if the new macro is defined before formatting the file in the argument.

```
\gdef\printexcursions{}
\newcommand\excursionref[2]{% label, text
  \bool_if:NT \c__notesslides_notes_bool {
    \begin{sparagraph}[title=Excursion]
      #2 \sref[fallback=the appendix]{#1}.
    \end{sparagraph}
  }
}
\newcommand\activate@excursion[2][]{
  \gappto\printexcursions{\inputref[#1]{#2}}
}
\newcommand\excursion[4][]{% repos, label, path, text
  \bool_if:NT \c__notesslides_notes_bool {
    \activate@excursion[#1]{#3}\excursionref{#2}{#4}
  }
}
```

(*End definition for* `\excursion`. *This function is documented on page* **??**.)

`\excursiongroup`

```
\keys_define:nn{notesslides / excursiongroup }{
  id        .str_set_x:N  = \l__notesslides_excursion_id_str,
  intro     .tl_set:N     = \l__notesslides_excursion_intro_tl,
  mhrepos   .str_set_x:N  = \l__notesslides_excursion_mhrepos_str
}
\cs_new_protected:Nn \__notesslides_excursion_args:n {
  \tl_clear:N \l__notesslides_excursion_intro_tl
  \str_clear:N \l__notesslides_excursion_id_str
  \str_clear:N \l__notesslides_excursion_mhrepos_str
  \keys_set:nn {notesslides / excursiongroup }{ #1 }
}
\newcommand\excursiongroup[1][]{
  \__notesslides_excursion_args:n{ #1 }
  \ifdefempty\printexcursions{}% only if there are excursions
  {\begin{note}
    \begin{sfragment}[#1]{Excursions}%
      \ifdefempty\l__notesslides_excursion_intro_tl{}{
        \inputref[\l__notesslides_excursion_mhrepos_str]{
          \l__notesslides_excursion_intro_tl
        }
      }
      \printexcursions%
    \end{sfragment}
  \end{note}}
}
\ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
⟨/package⟩
```

(*End definition for* `\excursiongroup`. *This function is documented on page* **??**.)

Chapter 39

The Implementation

39.1 Package Options

The first step is to declare (a few) package options that handle whether certain information is printed or not. They all come with their own conditionals that are set by the options.

```
⟨*package⟩
⟨@@=problems⟩
\ProvidesExplPackage{problem}{2022/02/26}{3.0.1}{Semantic Markup for Problems}
\RequirePackage{l3keys2e,stex}

\keys_define:nn { problem / pkg }{
  notes     .default:n    = { true },
  notes     .bool_set:N   = \c__problems_notes_bool,
  gnotes    .default:n    = { true },
  gnotes    .bool_set:N   = \c__problems_gnotes_bool,
  hints     .default:n    = { true },
  hints     .bool_set:N   = \c__problems_hints_bool,
  solutions .default:n    = { true },
  solutions .bool_set:N   = \c__problems_solutions_bool,
  pts       .default:n    = { true },
  pts       .bool_set:N   = \c__problems_pts_bool,
  min       .default:n    = { true },
  min       .bool_set:N   = \c__problems_min_bool,
  boxed     .default:n    = { true },
  boxed     .bool_set:N   = \c__problems_boxed_bool,
  unknown   .code:n       = {}
}
\newif\ifsolutions

\ProcessKeysOptions{ problem / pkg }
\bool_if:NTF \c__problems_solutions_bool {
  \solutionstrue
}{
  \solutionsfalse
}
```

Then we make sure that the necessary packages are loaded (in the right versions).

```
\RequirePackage{comment}
```

The next package relies on the LaTeX3 kernel, which LaTeXMLonly partially supports. As it is purely presentational, we only load it when the boxed option is given and we run LaTeXML.

```
\bool_if:NT \c__problems_boxed_bool { \RequirePackage{mdframed} }
```

`\prob@*@kw` For multilinguality, we define internal macros for keywords that can be specialized in `*.ldf` files.

```
\def\prob@problem@kw{Problem}
\def\prob@solution@kw{Solution}
\def\prob@hint@kw{Hint}
\def\prob@note@kw{Note}
\def\prob@gnote@kw{Grading}
\def\prob@pt@kw{pt}
\def\prob@min@kw{min}
```

(*End definition for* `\prob@*@kw`. *This function is documented on page* **??**.)

For the other languages, we set up triggers

```
\AddToHook{begindocument}{
  \ltx@ifpackageloaded{babel}{
      \makeatletter
      \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
      \clist_if_in:NnT \l_tmpa_clist {ngerman}{
        \input{problem-ngerman.ldf}
      }
      \clist_if_in:NnT \l_tmpa_clist {finnish}{
        \input{problem-finnish.ldf}
      }
      \clist_if_in:NnT \l_tmpa_clist {french}{
        \input{problem-french.ldf}
      }
      \clist_if_in:NnT \l_tmpa_clist {russian}{
        \input{problem-russian.ldf}
      }
      \makeatother
  }{}
}
```

39.2 Problems and Solutions

We now prepare the KeyVal support for problems. The key macros just set appropriate internal macros.

```
\keys_define:nn{ problem / problem }{
  id      .str_set_x:N  = \l__problems_prob_id_str,
  pts     .tl_set:N     = \l__problems_prob_pts_tl,
  min     .tl_set:N     = \l__problems_prob_min_tl,
  title   .tl_set:N     = \l__problems_prob_title_tl,
  type    .tl_set:N     = \l__problems_prob_type_tl,
  refnum  .int_set:N    = \l__problems_prob_refnum_int
}
\cs_new_protected:Nn \__problems_prob_args:n {
  \str_clear:N \l__problems_prob_id_str
  \tl_clear:N \l__problems_prob_pts_tl
  \tl_clear:N \l__problems_prob_min_tl
  \tl_clear:N \l__problems_prob_title_tl
  \tl_clear:N \l__problems_prob_type_tl
  \int_zero_new:N \l__problems_prob_refnum_int
  \keys_set:nn { problem / problem }{ #1 }
  \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
    \let\l__problems_prob_refnum_int\undefined
  }
}
```

Then we set up a counter for problems.

`\numberproblemsin`

```
\newcounter{problem}
\newcommand\numberproblemsin[1]{\@addtoreset{problem}{#1}}
```

(*End definition for* `\numberproblemsin`. *This function is documented on page* ??.)

`\prob@label` We provide the macro `\prob@label` to redefine later to get context involved.

```
\newcommand\prob@label[1]{#1}
```

(*End definition for* `\prob@label`. *This function is documented on page* ??.)

`\prob@number` We consolidate the problem number into a reusable internal macro

```
\newcommand\prob@number{
  \int_if_exist:NTF \l__problems_inclprob_refnum_int {
    \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
  }{
    \int_if_exist:NTF \l__problems_prob_refnum_int {
      \prob@label{\int_use:N \l__problems_prob_refnum_int }
    }{
        \prob@label\theproblem
    }
  }
}
```

(*End definition for* `\prob@number`. *This function is documented on page* ??.)

`\prob@title` We consolidate the problem title into a reusable internal macro as well. `\prob@title` takes three arguments the first is the fallback when no title is given at all, the second and third go around the title, if one is given.

```
\newcommand\prob@title[3]{%
  \tl_if_exist:NTF \l__problems_inclprob_title_tl {
    #2 \l__problems_inclprob_title_tl #3
  }{
    \tl_if_exist:NTF \l__problems_prob_title_tl {
      #2 \l__problems_prob_title_tl #3
    }{
      #1
    }
  }
}
```

(*End definition for* `\prob@title`. *This function is documented on page* ??.)

With these the problem header is a one-liner

`\prob@heading` We consolidate the problem header line into a separate internal macro that can be reused in various settings.

```
\def\prob@heading{
  {\prob@problem@kw}\ \prob@number\prob@title{~}{~(}{)\strut}
  %\sref@label@id{\prob@problem@kw~\prob@number}{}
}
```

(*End definition for* `\prob@heading`. *This function is documented on page* ??.)

With this in place, we can now define the `problem` environment. It comes in two shapes, depending on whether we are in boxed mode or not. In both cases we increment the problem number and output the points and minutes (depending) on whether the respective options are set.

`sproblem`

```
\newenvironment{sproblem}[1][]{
  \__problems_prob_args:n{#1}%\sref@target%
  \@in@omtexttrue% we are in a statement (for inline definitions)
  \stepcounter{problem}\record@problem
  \def\current@section@level{\prob@problem@kw}
  \tl_if_exist:NTF \l__problems_inclprob_type_tl {
    \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
  }{
    \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
  }
  \str_if_exist:NTF \l__problems_inclprob_id_str {
    \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
  }{
    \str_set_eq:NN \sproblemid \l__problems_prob_id_str
  }


  \clist_set:No \l_tmpa_clist \sproblemtype
  \tl_clear:N \l_tmpa_tl
  \clist_map_inline:Nn \l_tmpa_clist {
    \tl_if_exist:cT {__problems_sproblem_##1_start:}{
      \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_start:}}
    }
  }
  \tl_if_empty:NTF \l_tmpa_tl {
    \__problems_sproblem_start:
  }{
    \l_tmpa_tl
  }
  \stex_ref_new_doc_target:n \sproblemid
}{
  \clist_set:No \l_tmpa_clist \sproblemtype
  \tl_clear:N \l_tmpa_tl
  \clist_map_inline:Nn \l_tmpa_clist {
    \tl_if_exist:cT {__problems_sproblem_##1_end:}{
      \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_end:}}
    }
  }
  \tl_if_empty:NTF \l_tmpa_tl {
    \__problems_sproblem_end:
  }{
    \l_tmpa_tl
  }


  \smallskip
}


\cs_new_protected:Nn \__problems_sproblem_start: {
  \par\noindent\textbf\prob@heading\show@pts\show@min\\\ignorespacesandpars
}
\cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}

\newcommand\stexpatchproblem[3][] {
    \str_set:Nx \l_tmpa_str{ #1 }
    \str_if_empty:NTF \l_tmpa_str {
      \tl_set:Nn \__problems_sproblem_start: { #2 }
      \tl_set:Nn \__problems_sproblem_end: { #3 }
    }{
      \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
      \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
    }
}


\bool_if:NT \c__problems_boxed_bool {
  \surroundwithmdframed{problem}
}
```

`\record@problem` This macro records information about the problems in the `*.aux` file.

```
\def\record@problem{
  \protected@write\@auxout{}
  {
    \string\@problem{\prob@number}
    {
      \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
        \l__problems_inclprob_pts_tl
      }{
        \l__problems_prob_pts_tl
      }
    }%
    {
      \tl_if_exist:NTF \l__problems_inclprob_min_tl {
        \l__problems_inclprob_min_tl
      }{
        \l__problems_prob_min_tl
      }
    }
  }
}
```

(*End definition for* `\record@problem`. *This function is documented on page* **??**.)

`\@problem` This macro acts on a problem's record in the `*.aux` file. It does not have any functionality here, but can be redefined elsewhere (e.g. in the `assignment` package).

```
\def\@problem#1#2#3{}
```

(*End definition for* `\@problem`. *This function is documented on page* **??**.)

`solution` The `solution` environment is similar to the `problem` environment, only that it is independent of the boxed mode. It also has it's own keys that we need to define first.

```
\keys_define:nn { problem / solution }{
  id           .str_set_x:N  = \l__problems_solution_id_str ,
  for          .tl_set:N     = \l__problems_solution_for_tl ,
  height       .dim_set:N    = \l__problems_solution_height_dim ,
  creators     .clist_set:N  = \l__problems_solution_creators_clist ,
  contributors .clist_set:N  = \l__problems_solution_contributors_clist ,
  srccite      .tl_set:N     = \l__problems_solution_srccite_tl
}
\cs_new_protected:Nn \__problems_solution_args:n {
  \str_clear:N \l__problems_solution_id_str
  \tl_clear:N \l__problems_solution_for_tl
  \tl_clear:N \l__problems_solution_srccite_tl
  \clist_clear:N \l__problems_solution_creators_clist
  \clist_clear:N \l__problems_solution_contributors_clist
  \dim_zero:N \l__problems_solution_height_dim
  \keys_set:nn { problem / solution }{ #1 }
}
```

the next step is to define a helper macro that does what is needed to start a solution.

```
\newcommand\@startsolution[1][]{
  \__problems_solution_args:n { #1 }
  \@in@omtexttrue% we are in a statement.
  \bool_if:NF \c__problems_boxed_bool { \hrule }
  \smallskip\noindent
  {\textbf\prob@solution@kw :\enspace}
  \begin{small}
  \def\current@section@level{\prob@solution@kw}
  \ignorespacesandpars
}
```

`\startsolutions` for the `\startsolutions` macro we use the `\specialcomment` macro from the `comment` package. Note that we use the `\@startsolution` macro in the start codes, that parses the optional argument.

```
\newcommand\startsolutions{
  \specialcomment{solution}{\@startsolution}{
    \bool_if:NF \c__problems_boxed_bool {
      \hrule\medskip
    }
    \end{small}%
  }
  \bool_if:NT \c__problems_boxed_bool {
    \surroundwithmdframed{solution}
  }
}
```

(*End definition for* `\startsolutions`. *This function is documented on page* **??**.)

`\stopsolutions`

```
\newcommand\stopsolutions{\excludecomment{solution}}
```

(*End definition for* `\stopsolutions`. *This function is documented on page* **??**.)

so it only remains to start/stop solutions depending on what option was specified.

```
\ifsolutions
  \startsolutions
\else
  \stopsolutions
\fi
```

`exnote`

```
\bool_if:NTF \c__problems_notes_bool {
  \newenvironment{exnote}[1][]{
    \par\smallskip\hrule\smallskip
    \noindent\textbf{\prob@note@kw : }\small
  }{
    \smallskip\hrule
  }
}{
  \excludecomment{exnote}
}
```

`hint`

```
\bool_if:NTF \c__problems_notes_bool {
  \newenvironment{hint}[1][]{
    \par\smallskip\hrule\smallskip
    \noindent\textbf{\prob@hint@kw :~ }\small
  }{
    \smallskip\hrule
  }
  \newenvironment{exhint}[1][]{
    \par\smallskip\hrule\smallskip
    \noindent\textbf{\prob@hint@kw :~ }\small
  }{
    \smallskip\hrule
  }
}{
  \excludecomment{hint}
  \excludecomment{exhint}
}
```

`gnote`

```
\bool_if:NTF \c__problems_notes_bool {
  \newenvironment{gnote}[1][]{
    \par\smallskip\hrule\smallskip
    \noindent\textbf{\prob@gnote@kw : }\small
  }{
    \smallskip\hrule
  }
}{
  \excludecomment{gnote}
}
```

39.3 Multiple Choice Blocks

EdN:17 mcb [17]

```
\newenvironment{mcb}{
  \begin{enumerate}
}{
  \end{enumerate}
}
```

we define the keys for the mcc macro

```
\cs_new_protected:Nn \__problems_do_yes_param:Nn {
  \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
    \bool_set_true:N #1
  }{
    \bool_set_false:N #1
  }
}
\keys_define:nn { problem / mcc }{
  id        .str_set_x:N  = \l__problems_mcc_id_str ,
  feedback  .tl_set:N     = \l__problems_mcc_feedback_tl ,
  T         .default:n    = { true } ,
  T         .bool_set:N   = \l__problems_mcc_t_bool ,
  F         .default:n    = { true } ,
  F         .bool_set:N   = \l__problems_mcc_f_bool ,
  Ttext     .code:n       = {
    \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
  } ,
  Ftext     .code:n       = {
    \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
  }
}
\cs_new_protected:Nn \l__problems_mcc_args:n {
  \str_clear:N \l__problems_mcc_id_str
  \tl_clear:N \l__problems_mcc_feedback_tl
  \bool_set_true:N \l__problems_mcc_t_bool
  \bool_set_true:N \l__problems_mcc_f_bool
  \bool_set_true:N \l__problems_mcc_Ttext_bool
  \bool_set_false:N \l__problems_mcc_Ftext_bool
  \keys_set:nn { problem / mcc }{ #1 }
}
```

\mcc

```
\newcommand\mcc[2][]{
  \l__problems_mcc_args:n{ #1 }
  \item #2
  \ifsolutions
    \\
    \bool_if:NT \l__problems_mcc_t_bool {
      % TODO!
      % \ifcsstring{mcc@T}{T}{}{\mcc@Ttext}%
    }
    \bool_if:NT \l__problems_mcc_f_bool {
```

[17] EdNote: MK: maybe import something better here from a dedicated MC package

```
      % TODO!
      % \ifcsstring{mcc@F}{F}{}{\mcc@Ftext}%
    }
    \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
      !
    }{
      \l__problems_mcc_feedback_tl
    }
  \fi
} %solutions
```

(*End definition for* `\mcc`. *This function is documented on page* **??**.)

39.4 Including Problems

`\includeproblem` The `\includeproblem` command is essentially a glorified `\input` statement, it sets some internal macros first that overwrite the local points. Importantly, it resets the `inclprob` keys after the input.

```

\keys_define:nn{ problem / inclproblem }{
  id      .str_set_x:N  = \l__problems_inclprob_id_str,
  pts     .tl_set:N     = \l__problems_inclprob_pts_tl,
  min     .tl_set:N     = \l__problems_inclprob_min_tl,
  title   .tl_set:N     = \l__problems_inclprob_title_tl,
  refnum  .int_set:N    = \l__problems_inclprob_refnum_int,
  type    .tl_set:N     = \l__problems_inclprob_type_tl,
  mhrepos .str_set_x:N  = \l__problems_inclprob_mhrepos_str
}
\cs_new_protected:Nn \__problems_inclprob_args:n {
  \str_clear:N \l__problems_prob_id_str
  \tl_clear:N \l__problems_inclprob_pts_tl
  \tl_clear:N \l__problems_inclprob_min_tl
  \tl_clear:N \l__problems_inclprob_title_tl
  \tl_clear:N \l__problems_inclprob_type_tl
  \int_zero_new:N \l__problems_inclprob_refnum_int
  \str_clear:N \l__problems_inclprob_mhrepos_str
  \keys_set:nn { problem / inclproblem }{ #1 }
  \tl_if_empty:NT \l__problems_inclprob_pts_tl {
    \let\l__problems_inclprob_pts_tl\undefined
  }
  \tl_if_empty:NT \l__problems_inclprob_min_tl {
    \let\l__problems_inclprob_min_tl\undefined
  }
  \tl_if_empty:NT \l__problems_inclprob_title_tl {
    \let\l__problems_inclprob_title_tl\undefined
  }
  \tl_if_empty:NT \l__problems_inclprob_type_tl {
    \let\l__problems_inclprob_type_tl\undefined
  }
  \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
    \let\l__problems_inclprob_refnum_int\undefined
  }
}
```

```

\cs_new_protected:Nn \__problems_inclprob_clear: {
  \let\l__problems_inclprob_id_str\undefined
  \let\l__problems_inclprob_pts_tl\undefined
  \let\l__problems_inclprob_min_tl\undefined
  \let\l__problems_inclprob_title_tl\undefined
  \let\l__problems_inclprob_type_tl\undefined
  \let\l__problems_inclprob_refnum_int\undefined
  \let\l__problems_inclprob_mhrepos_str\undefined
}
\__problems_inclprob_clear:

\newcommand\includeproblem[2][]{
  \__problems_inclprob_args:n{ #1 }
  \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
    \input{#2}
  }{
    \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
      \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
    }
  }
  \__problems_inclprob_clear:
}
```

(*End definition for* `\includeproblem`. *This function is documented on page* **??**.)

39.5 Reporting Metadata

For messages it is OK to have them in English as the whole documentation is, and we can therefore assume authors can deal with it.

```
\AddToHook{enddocument}{
  \bool_if:NT \c__problems_pts_bool {
    \message{Total:~\arabic{pts}~points}
  }
  \bool_if:NT \c__problems_min_bool {
    \message{Total:~\arabic{min}~minutes}
  }
}
```

The margin pars are reader-visible, so we need to translate

```
\def\pts#1{
  \bool_if:NT \c__problems_pts_bool {
    \marginpar{#1~\prob@pt@kw}
  }
}
\def\min#1{
  \bool_if:NT \c__problems_min_bool {
    \marginpar{#1~\prob@min@kw}
  }
}
```

`\show@pts` The `\show@pts` shows the points: if no points are given from the outside and also no points are given locally do nothing, else show and add. If there are outside points then we show them in the margin.

```
\newcounter{pts}
\def\show@pts{
  \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
    \bool_if:NT \c__problems_pts_bool {
      \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
      \addtocounter{pts}{\l__problems_inclprob_pts_tl}
    }
  }{
    \tl_if_exist:NT \l__problems_prob_pts_tl {
      \bool_if:NT \c__problems_pts_bool {
        \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
        \addtocounter{pts}{\l__problems_prob_pts_tl}
      }
    }
  }
}
```

(*End definition for* `\show@pts`. *This function is documented on page* **??**.)

and now the same for the minutes

`\show@min`

```
\newcounter{min}
\def\show@min{
  \tl_if_exist:NTF \l__problems_inclprob_min_tl {
    \bool_if:NT \c__problems_min_bool {
      \marginpar{\l__problems_inclprob_pts_tl\ min}
      \addtocounter{min}{\l__problems_inclprob_min_tl}
    }
  }{
    \tl_if_exist:NT \l__problems_prob_min_tl {
      \bool_if:NT \c__problems_min_bool {
        \marginpar{\l__problems_prob_min_tl\ min}
        \addtocounter{min}{\l__problems_prob_min_tl}
      }
    }
  }
}
⟨/package⟩
```

(*End definition for* `\show@min`. *This function is documented on page* **??**.)

Chapter 40

Implementation: The hwexam Class

The functionality is spread over the `hwexam` class and package. The class provides the `document` environment and pre-loads some convenience packages, whereas the package provides the concrete functionality.

40.1 Class Options

To initialize the `hwexam` class, we declare and process the necessary options by passing them to the respective packages and classes they come from.

```
⟨@@=hwexam⟩
⟨*cls⟩
\ProvidesExplClass{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
\RequirePackage{l3keys2e}
\DeclareOption*{
  \PassOptionsToClass{\CurrentOption}{document-structure}
  \PassOptionsToPackage{\CurrentOption}{stex}
  \PassOptionsToPackage{\CurrentOption}{hwexam}
  \PassOptionsToPackage{\CurrentOption}{tikzinput}
}
\ProcessOptions
```

We load `omdoc.cls`, and the desired packages. For the LaTeXML bindings, we make sure the right packages are loaded.

```
\LoadClass{document-structure}
\RequirePackage{stex}
\RequirePackage{hwexam}
\RequirePackage{tikzinput}
\RequirePackage{graphicx}
\RequirePackage{a4wide}
\RequirePackage{amssymb}
\RequirePackage{amstext}
\RequirePackage{amsmath}
```

Finally, we register another keyword for the `document` environment. We give a default assignment type to prevent errors

```
\newcommand\assig@default@type{\hwexam@assignment@kw}
\def\document@hwexamtype{\assig@default@type}
⟨@@=document_structure⟩
\keys_define:nn { document-structure / document }{
id .str_set_x:N = \c_document_structure_document_id_str,
hwexamtype .tl_set:N = \document@hwexamtype
}
⟨@@=hwexam⟩
⟨/cls⟩
```

Chapter 41

Implementation: The hwexam Package

41.1 Package Options

The first step is to declare (a few) package options that handle whether certain information is printed or not. Some come with their own conditionals that are set by the options, the rest is just passed on to the `problems` package.

```
⟨*package⟩
\ProvidesExplPackage{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
\RequirePackage{l3keys2e}

\newif\iftest\testfalse
\DeclareOption{test}{\testtrue}
\newif\ifmultiple\multiplefalse
\DeclareOption{multiple}{\multipletrue}
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
\ProcessOptions
```

Then we make sure that the necessary packages are loaded (in the right versions).

```
\RequirePackage{keyval}[1997/11/10]
\RequirePackage{problem}
```

`\hwexam@*@kw` For multilinguality, we define internal macros for keywords that can be specialized in `*.ldf` files.

```
\newcommand\hwexam@assignment@kw{Assignment}
\newcommand\hwexam@given@kw{Given}
\newcommand\hwexam@due@kw{Due}
\newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
blank~for~extra~space}
\def\hwexam@minutes@kw{minutes}
\newcommand\correction@probs@kw{prob.}
\newcommand\correction@pts@kw{total}
\newcommand\correction@reached@kw{reached}
\newcommand\correction@sum@kw{Sum}
\newcommand\correction@grade@kw{grade}
\newcommand\correction@forgrading@kw{To~be~used~for~grading,~do~not~write~here}
```

(*End definition for* \hwexam@*@kw. *This function is documented on page* **??**.)

For the other languages, we set up triggers

```
\AddToHook{begindocument}{
\ltx@ifpackageloaded{babel}{
\makeatletter
\clist_set:Nx \l_tmpa_clist {\bbl@loaded}
\clist_if_in:NnT \l_tmpa_clist {ngerman}{
  \input{hwexam-ngerman.ldf}
}
\clist_if_in:NnT \l_tmpa_clist {finnish}{
  \input{hwexam-finnish.ldf}
}
\clist_if_in:NnT \l_tmpa_clist {french}{
  \input{hwexam-french.ldf}
}
\clist_if_in:NnT \l_tmpa_clist {russian}{
  \input{hwexam-russian.ldf}
}
\makeatother
}{}
}

```

41.2 Assignments

Then we set up a counter for problems and make the problem counter inherited from `problem.sty` depend on it. Furthermore, we specialize the `\prob@label` macro to take the assignment counter into account.

```
\newcounter{assignment}
\numberproblemsin{assignment}
\renewcommand\prob@label[1]{\assignment@number.#1}
```

We will prepare the keyval support for the `assignment` environment.

```
\keys_define:nn { hwexam / assignment } {
id  .str_set_x:N = \l__hwexam_assign_id_str,
number  .int_set:N  = \l__hwexam_assign_number_int,
title  .tl_set:N  = \l__hwexam_assign_title_tl,
type  .tl_set:N  = \l__hwexam_assign_type_tl,
given .tl_set:N  = \l__hwexam_assign_given_tl,
due .tl_set:N  = \l__hwexam_assign_due_tl,
loadmodules .code:n  = {
\bool_set_true:N \l__hwexam_assign_loadmodules_bool
}
}
\cs_new_protected:Nn \__hwexam_assignment_args:n {
\str_clear:N \l__hwexam_assign_id_str
\int_set:Nn \l__hwexam_assign_number_int {-1}
\tl_clear:N \l__hwexam_assign_title_tl
\tl_clear:N \l__hwexam_assign_type_tl
\tl_clear:N \l__hwexam_assign_given_tl
\tl_clear:N \l__hwexam_assign_due_tl
\bool_set_false:N \l__hwexam_assign_loadmodules_bool
\keys_set:nn { hwexam / assignment }{ #1 }
}
```

The next three macros are intermediate functions that handle the case gracefully, where the respective token registers are undefined.

The `\given@due` macro prints information about the given and due status of the assignment. Its arguments specify the brackets.

```
\newcommand\given@due[2]{
\bool_lazy_all:nF {
{\tl_if_empty_p:V \l__hwexam_inclassign_given_tl}
{\tl_if_empty_p:V \l__hwexam_assign_given_tl}
{\tl_if_empty_p:V \l__hwexam_inclassign_due_tl}
{\tl_if_empty_p:V \l__hwexam_assign_due_tl}
}{ #1 }

\tl_if_empty:NTF \l__hwexam_inclassign_given_tl {
\tl_if_empty:NF \l__hwexam_assign_given_tl {
\hwexam@given@kw\xspace\l__hwexam_assign_given_tl
}
}{
\hwexam@given@kw\xspace\l__hwexam_inclassign_given_tl
}

\bool_lazy_or:nnF {
\bool_lazy_and_p:nn {
\tl_if_empty_p:V \l__hwexam_inclassign_due_tl
}{
\tl_if_empty_p:V \l__hwexam_assign_due_tl
}
}{
\bool_lazy_and_p:nn {
\tl_if_empty_p:V \l__hwexam_inclassign_due_tl
}{
\tl_if_empty_p:V \l__hwexam_assign_due_tl
}
}{ ,~ }

\tl_if_empty:NTF \l__hwexam_inclassign_due_tl {
\tl_if_empty:NF \l__hwexam_assign_due_tl {
\hwexam@due@kw\xspace \l__hwexam_assign_due_tl
}
}{
\hwexam@due@kw\xspace \l__hwexam_inclassign_due_tl
}

\bool_lazy_all:nF {
{ \tl_if_empty_p:V \l__hwexam_inclassign_given_tl }
{ \tl_if_empty_p:V \l__hwexam_assign_given_tl }
{ \tl_if_empty_p:V \l__hwexam_inclassign_due_tl }
{ \tl_if_empty_p:V \l__hwexam_assign_due_tl }
}{ #2 }
}
```

`\assignment@title` This macro prints the title of an assignment, the local title is overwritten, if there is one from the `\inputassignment`. `\assignment@title` takes three arguments the first is the fallback when no title is given at all, the second and third go around the title, if one is given.

```
\newcommand\assignment@title[3]{
\tl_if_empty:NTF \l__hwexam_inclassign_title_tl {
\tl_if_empty:NTF \l__hwexam_assign_title_tl {
#1
}{
#2\l__hwexam_assign_title_tl#3
}
}{
#2\l__hwexam_inclassign_title_tl#3
}
}
```

(*End definition for* `\assignment@title`. *This function is documented on page* **??**.)

`\assignment@number` Like `\assignment@title` only for the number, and no around part.

```
\newcommand\assignment@number{
\int_compare:nNnTF \l__hwexam_inclassign_number_int = {-1} {
\int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
\arabic{assignment}
} {
\int_use:N \l__hwexam_assign_number_int
}
}{
\int_use:N \l__hwexam_inclassign_number_int
}
}
```

(*End definition for* `\assignment@number`. *This function is documented on page* **??**.)

With them, we can define the central `assignment` environment. This has two forms (separated by `\ifmultiple`) in one we make a title block for an assignment sheet, and in the other we make a section heading and add it to the table of contents. We first define an assignment counter

`assignment` For the `assignment` environment we delegate the work to the `@assignment` environment that depends on whether `multiple` option is given.

```
\newenvironment{assignment}[1][]{
\__hwexam_assignment_args:n { #1 }
%\sref@target
\int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
\global\stepcounter{assignment}
}{
\global\setcounter{assignment}{\int_use:N\l__hwexam_assign_number_int}
}
\setcounter{problem}{0}
\def\current@section@level{\document@hwexamtype}
%\sref@label@id{\document@hwexamtype \thesection}
\begin{@assignment}
}{
\end{@assignment}
}
```

In the multi-assignment case we just use the omdoc environment for suitable sectioning.

```
\def\ass@title{
\protect\document@hwexamtype~\arabic{assignment}
\assignment@title{}{\;(}{)\;} -- \given@due{}{}
}
\ifmultiple
\newenvironment{@assignment}{
\bool_if:NTF \l__hwexam_assign_loadmodules_bool {
\begin{sfragment}[loadmodules]{\ass@title}
}{
\begin{sfragment}{\ass@title}
}
}{
\end{sfragment}
}
```

for the single-page case we make a title block from the same components.

```
\else
\newenvironment{@assignment}{
\begin{center}\bf
\Large\@title\strut\\
\document@hwexamtype~\arabic{assignment}\assignment@title{\;}{:\;}{\\}
\large\given@due{--\;}{\;--}
\end{center}
}{}
\fi% multiple
```

41.3 Including Assignments

\in*assignment This macro is essentially a glorified \include statement, it just sets some internal macros first that overwrite the local points Importantly, it resets the inclassig keys after the input.

```
\keys_define:nn { hwexam / inclassignment } {
%id  .str_set_x:N = \l__hwexam_assign_id_str,
number  .int_set:N  = \l__hwexam_inclassign_number_int,
title  .tl_set:N  = \l__hwexam_inclassign_title_tl,
type  .tl_set:N  = \l__hwexam_inclassign_type_tl,
given .tl_set:N  = \l__hwexam_inclassign_given_tl,
due .tl_set:N  = \l__hwexam_inclassign_due_tl,
mhrepos  .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
}
\cs_new_protected:Nn \__hwexam_inclassignment_args:n {
\int_set:Nn \l__hwexam_inclassign_number_int {-1}
\tl_clear:N \l__hwexam_inclassign_title_tl
\tl_clear:N \l__hwexam_inclassign_type_tl
\tl_clear:N \l__hwexam_inclassign_given_tl
\tl_clear:N \l__hwexam_inclassign_due_tl
\str_clear:N \l__hwexam_inclassign_mhrepos_str
\keys_set:nn { hwexam / inclassignment }{ #1 }
}
\__hwexam_inclassignment_args:n {}

\newcommand\inputassignment[2][]{
```

```
\__hwexam_inclassignment_args:n { #1 }
\str_if_empty:NTF \l__hwexam_inclassign_mhrepos_str {
\input{#2}
}{
\stex_in_repository:nn{\l__hwexam_inclassign_mhrepos_str}{
\input{\mhpath{\l__hwexam_inclassign_mhrepos_str}{#2}}
}
}
\__hwexam_inclassignment_args:n {}
}
\newcommand\includeassignment[2][]{
\newpage
\inputassignment[#1]{#2}
}
```

(*End definition for* `\in*assignment`*. This function is documented on page* **??**.)

41.4 Typesetting Exams

`\quizheading`

```
\ExplSyntaxOff
\newcommand\quizheading[1]{%
\def\@tas{#1}%
\large\noindent NAME: \hspace{8cm}  MAILBOX:\\[2ex]%
\ifx\@tas\@empty\else%
\noindent TA:~\@for\@I:=\@tas\do{{\Large$\Box$}\@I\hspace*{1em}}\\[2ex]%
\fi%
}
\ExplSyntaxOn
```

(*End definition for* `\quizheading`*. This function is documented on page* **??**.)

`\testheading`

```

\def\hwexamheader{\input{hwexam-default.header}}

\def\hwexamminutes{
\tl_if_empty:NTF \testheading@duration {
{\testheading@min}~\hwexam@minutes@kw
}{
\testheading@duration
}
}

\keys_define:nn { hwexam / testheading } {
min  .tl_set:N  = \testheading@min,
duration .tl_set:N  = \testheading@duration,
reqpts .tl_set:N  = \testheading@reqpts,
tools .tl_set:N  = \testheading@tools
}
\cs_new_protected:Nn \__hwexam_testheading_args:n {
\tl_clear:N \testheading@min
\tl_clear:N \testheading@duration
\tl_clear:N \testheading@reqpts
\tl_clear:N \testheading@tools
\keys_set:nn { hwexam / testheading }{ #1 }
}
\newenvironment{testheading}[1][]{
\__hwexam_testheading_args:n{ #1 }
\newcount\check@time\check@time=\testheading@min
\advance\check@time by -\theassignment@totalmin
\newif\if@bonuspoints
\tl_if_empty:NTF \testheading@reqpts {
\@bonuspointsfalse
}{
\newcount\bonus@pts
\bonus@pts=\theassignment@totalpts
\advance\bonus@pts by -\testheading@reqpts
\edef\bonus@pts{\the\bonus@pts}
\@bonuspointstrue
}
\edef\check@time{\the\check@time}

\makeatletter\hwexamheader\makeatother
}{
\newpage
}
```

(*End definition for* `\testheading`. *This function is documented on page* **??**.)

`\testspace`

```
\newcommand\testspace[1]{\iftest\vspace*{#1}\fi}
```

(*End definition for* `\testspace`. *This function is documented on page* **??**.)

`\testnewpage`

```
\newcommand\testnewpage{\iftest\newpage\fi}
```

(*End definition for* `\testnewpage`. *This function is documented on page* **??**.)

`\testemptypage`

```
\newcommand\testemptypage[1][]{\iftest\begin{center}\hwexam@testemptypage@kw\end{center}\vfi
```

(*End definition for* `\testemptypage`. *This function is documented on page* **??**.)

`\@problem` This macro acts on a problem's record in the `*.aux` file. Here we redefine it (it was defined to do nothing in `problem.sty`) to generate the correction table.

```
⟨@@=problems⟩
\renewcommand\@problem[3]{
\stepcounter{assignment@probs}
\def\__problemspts{#2}
\ifx\__problemspts\@empty\else
\addtocounter{assignment@totalpts}{#2}
\fi
\def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\f
\xdef\correction@probs{\correction@probs & #1}%
\xdef\correction@pts{\correction@pts & #2}
\xdef\correction@reached{\correction@reached &}
}
⟨@@=hwexam⟩
```

(*End definition for* `\@problem`. *This function is documented on page* **??**.)

`\correction@table` This macro generates the correction table

```
\newcounter{assignment@probs}
\newcounter{assignment@totalpts}
\newcounter{assignment@totalmin}
\def\correction@probs{\correction@probs@kw}
\def\correction@pts{\correction@pts@kw}
\def\correction@reached{\correction@reached@kw}
\stepcounter{assignment@probs}
\newcommand\correction@table{
\resizebox{\textwidth}{!}{%
\begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
&\multicolumn{\theassignment@probs}{c||}%|
{\footnotesize\correction@forgrading@kw} &\\\hline
\correction@probs & \correction@sum@kw & \correction@grade@kw\\\hline
\correction@pts &\theassignment@totalpts & \\\hline
\correction@reached & & \\[.7cm]\hline
\end{tabular}}}
⟨/package⟩
```

(*End definition for* `\correction@table`. *This function is documented on page* **??**.)

41.5 Leftovers

at some point, we may want to reactivate the logos font, then we use

```
here we define the logos that characterize the assignment
\font\bierfont=../assignments/bierglas
\font\denkerfont=../assignments/denker
\font\uhrfont=../assignments/uhr
\font\warnschildfont=../assignments/achtung

\newcommand\bierglas{{\bierfont\char65}}
\newcommand\denker{{\denkerfont\char65}}
\newcommand\uhr{{\uhrfont\char65}}
\newcommand\warnschild{{\warnschildfont\char 65}}
\newcommand\hardA{\warnschild}
\newcommand\longA{\uhr}
\newcommand\thinkA{\denker}
\newcommand\discussA{\bierglas}
```