The sTeX3 Package *

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

2022-04-10

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 2022-04-10)

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 $\mathsf{sT_EX}$?

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.

$\mathsf{sT_EX}$ 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 $\mathsf{sT_EX}$ workflow combines functionalities provided by several pieces of software:

- The $\mathsf{sT_EX}$ package to use semantic annotations in LaTeX documents,

- $\mathrm{R_{US}T_EX}$ 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 `pdf`s, 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

```
1 \documentclass{article}
2 \usepackage{stex,xcolor,stexthm}
3
4 \begin{document}
5 \begin{smodule}{GeometricSeries}
6     \importmodule[smglom/calculus]{series}
7     \importmodule[smglom/arithmetics]{realarith}
8
9     \symdef{geometricSeries}[name=geometric-series]{\comp{S}}
10
11     \begin{sdefinition}[for=geometricSeries]
12         The \definame{geometricSeries} is the \symname{?series}
13         \[\defeq{\geometricSeries}{\definiens{
14             \infinitesum{\svar{n}}{1}{
15                 \realdivide[frac]{1}{
16                     \realpower{2}{\svar{n}}
17             }}
18         }}.\]
19     \end{sdefinition}
20
21     \begin{sassertion}[name=geometricSeriesConverges,type=theorem]
22     The \symname{geometricSeries} \symname{converges} towards $1$.
23     \end{sassertion}
24 \end{smodule}
25 \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 $\text{S}\!T\!E\!X$ 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 The sdefinition-environment provides two additional macros, \definame and \definiendum
\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 `include`s and other *theory morphisms*, thus giving rise to a *theory graph* in the OMDoc sense.

$$\hookleftarrow M \rightarrow$$
$$\longrightarrow M \rightarrow$$
$$\rightsquigarrow T \rightsquigarrow$$

- 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.

4. Finally, if all else fails, sTEX will look for a file `~/.stex/mathhub.path`. If this file exists, sTEX will assume that it contains the path to the local `MathHub`-directory.

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:

```
1 \documentclass{stex}
2 \libinput{preamble}
3 \begin{document}
4    ...
5    \ifinputref \else \libinput{postamble} \fi
6 \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

$$\texttt{\textbackslash begin\{smodule\}[options]\{ModuleName\}...\textbackslash end\{smodule\}.}$$

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

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

Example 1

Input:

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

Output:

```
Hello World
```

.

<div style="margin-left:1em">\stexpatchmodule</div>

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:

```
1  \stexpatchmodule[display]
2   {\textbf{Module (\smoduletitle)}\par}
3   {\par\noindent\textbf{End of Module (\smoduletitle)}}
4
5  \begin{smodule}[type=display,title={Some New Module}]{SomeModule2}
6     Hello World
7  \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.

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→ `\symdecl` introduces a new OMDoc/Mmt constant in the current mod-
> —M→ ule (=OMDoc/Mmt theory). Correspondingly, they get assigned the URI
> ⤳T↝ `<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:

```
1  \symdecl*{foo}
2  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:

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

Output:

> this is a symbol taking two arguments.

.

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:

```
1 \notation{binarysymbol}{\text{First: }#1\text{; Second: }#2}
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.

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:

```
1 \notation{binarysymbol}[highlight]
2    {\comp{\text{First: }}#1\comp{\text{; Second: }}#2}
3 $\binarysymbol[highlight]{a}{b}$
```

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:
>
> ```
> 1 \symdef{newbinarysymbol}[hl,args=2]
> 2   {\comp{\text{1.: }}#1\comp{\text{; 2.: }}#2}
> 3 $\newbinarysymbol{a}{b}$
> ```
>
> 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.

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:

```
1  \notation{newbinarysymbol}[ab,
2   op={\text{a:}\cdot\text{; b:}\cdot}]
3   {\comp{\text{a:}}#1\comp{\text{; b:}}#2}
4 \symname{newbinarysymbol} is also occasionally written
5 $\newbinarysymbol![ab]$
```

Output:

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

.

> ↪M→
> —M→   `\symbolname!` is translated to OMDoc/Mmt as `<OMS name="...?symbolname"/>`
> ↝T↝   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→ `b`-type arguments behave exactly like `i`-type arguments within TeX, but applica-
> —M→ tions of binding operators, i.e. symbols with `b`-type arguments, are translated to
> ⇝T↝ `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:

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

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:

```
1  \symdef{ascendingchain}[args=iai]
2   {\comp{\forall} #2\comp{.\,}#3}
3   {##1 \comp{<}_{#1} ##2}
4
5 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:

```
1  \symdef{addition}[args=a]{#1}{##1 \comp{+} ##2}
2
3 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 $a = d \land b = d \land c = d$ and $a \in A \land b \in A \land c \in A \land d \in A$, respectively

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

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:

```
1 \symdef{quantforall}[args=Bi]
2  {\comp{\forall}#1\comp{.}#2}
3  {##1\comp,##2}
4
5 $\quantforall{\svar{x},\svar{y},\svar{z}}{P}$
```

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→ The `type` and `def` keys correspond to the `type` and `definiens` components of
> —M→ OMDOC/MMT constants.
> ⤳T⤳ 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 SIEX symbols), e.g. for addition on natural numbers:

Example 13

Input:

```
1  \symdef{Nat}[type=\set]{\comp{\mathbb N}}
2 \symdef{addition}[
3    type=\funtype{\Nat,\Nat}{\Nat},
4    op=+,
5    args=a
6 ]{#1}{##1 \comp+ ##2}
7
8 \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:

```
1  \symdef{successor}[
2     type=\funtype{\Nat}{\Nat},
3     def=\fun{\svar{x}}{\addition{\svar{x},1}},
4     op=\mathtt{succ},
5     args=1
6  ]{\comp{\mathtt{succ(}#1\comp{)}}}}
7
8  The \symname{successor} operation $\funtype{\Nat}{\Nat}$
9  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:

```
1  \symdef{multiplication}[
2     type=\funtype{\Nat,\Nat}{\Nat},
3     op=\cdot,
4     args=a
5  ]{#1}{##1 \comp\cdot ##2}
6
7  \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:

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

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:

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

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:

```
 1  \notation{multiplication}[
 2      op=\cdot,
 3      prec=50
 4  ]{#1}{##1 \comp\cdot ##2}
 5  \notation{addition}[
 6      op=+,
 7      prec=100
 8  ]{#1}{##1 \comp+ ##2}
 9
10  $\addition{a,\multiplication{b,\addition{c,\multiplication{d,e}}}}$
```

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.

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:

```
1     \vardef{varf}[
2         name=f,
3         type=\funtype{\Nat}{\Nat},
4         op=f,
5         args=1,
6         prec=0;\neginfprec
7     ]{\comp{f}#1}
8     \vardef{varn}[name=n,type=\Nat]{\comp{n}}
9     \vardef{varx}[name=x,type=\Nat]{\comp{x}}
10
11    Given a function $\varf!:\funtype{\Nat}{\Nat}$,
12    by $\addition{\varf!,\varn}$ we mean the function
13    $\fun{\varx}{\varf{\addition{\varx,\varn}}}$
```

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.

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:

```
1 \vardef{varn}[name=n,type=\Nat]{\comp{n}}
2 \varseq{seqa}[name=a,type=\Nat]{1}{\varn}{\comp{a}_{#1}}
3
4 The $i$th index of $\seqa!$ is $\seqa{i}$.
```

Output:

The ith 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:

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

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:

```
1  \vardef{varm}[name=m,type=\Nat]{\comp{m}}
2  \varseq{seqa}[
3     name=a,
4     args=2,
5     type=\Nat,
6  ]{1,1}{\varn,\varm}{\comp{a}_{#1}^{#2}}
7
8  $\seqa!$ and $\addition{\seqa}$
```

Output:

$$a_1^1, \ldots, a_n^m \text{ and } a_1^1 + \ldots + a_n^m$$

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

Example 23

Input:

```
1  \varseq{seqa}[
2     name=a,
3     type=\Nat,
4     args=2,
5     mid={\comp{a}_{\varn}^1,\comp{a}_1^2,\ellipses,\comp{a}_{1}^{\varm}}
6  ]{1,1}{\varn,\varm}{\comp{a}_{#1}^{#2}}
7
8  $\seqa!$ and $\addition{\seqa}$
```

Output:

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

.

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}.

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:

```
1 \begin{smodule}[sig=en]{lcm}
2   \notation*{lcm}[de]{\comp{\mathtt{kgV}}(#1,#2)}
3
4   Das \symref{lcm}{kleinste gemeinsame Vielfache}
5   $\lcm{a,b}$ von zwei Zahlen $a,b$ ist...
6 \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[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, SnX 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 SnX 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 $\langle\textit{top-directory}\rangle$`/some/path/Foo[.`$\langle\textit{lang}\rangle$`].tex`, or in $\langle\textit{top-directory}\rangle$`/some/path[.`$\langle\textit{lang}\rangle$`].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:

```
1  \begin{mathstructure}{monoid}
2     \symdef{universe}[type=\set]{\comp{U}}
3     \symdef{op}[
4         args=2,
5         type=\funtype{\universe,\universe}{\universe},
6         op=\circ
7     ]{#1 \comp{\circ} #2}
8     \symdef{unit}[type=\universe]{\comp{e}}
9  \end{mathstructure}
10
11 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:

```
1  \symdef{Int}[type=\set]{\comp{\mathbb Z}}
2  \symdef{addition}[
3     type=\funtype{\Int,\Int}{\Int},
4     args=2,
5     op=+
6  ]{##1 \comp{+} ##2}
7  \symdef{zero}[type=\Int]{\comp{0}}
8
9  $\mathstruct{\Int,\addition!,\zero}$ 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:

```
1  \instantiate{intmonoid}{
2      universe = Int ,
3      op = addition ,
4      unit = zero
5  }{monoid}{\mathbb{Z}_{+,0}}
6
7      $\intmonoid{universe}$, $\intmonoid{unit}$ and $\intmonoid{op}{a}{b}$.
8
9      Also: $\intmonoid!$
```

Output:

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

.

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.

> ↪M→
> —M→
> ⤳T↝
> \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.

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:

```
1  \varinstantiate{varM}{}{monoid}{M}
2
3  A \symname{monoid} is a structure
4  $\varM!:=\mathstruct{\varM{universe},\varM{op}!,\varM{unit}}$
5  such that
6  $\varM{op}!:\funtype{\varM{universe},\varM{universe}}{\varM{universe}}$
7   and...
8
9  \varinstantiate{varMb}{universe = Int}{monoid}{M_2}
10
11  \noindent Let $\varMb!:=\mathstruct{\varMb{universe},\varMb{op}!,\varMb{unit}}$
12  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:

```
1  \begin{smodule}{magma}
2     \symdef{universe}{\comp{\mathcal U}}
3     \symdef{operation}[args=2,op=\circ]{#1 \comp\circ #2}
4  \end{smodule}
5  \begin{smodule}{monoid}
6     \importmodule{magma}
7     \symdef{unit}{\comp e}
8  \end{smodule}
9  \begin{smodule}{group}
10     \importmodule{monoid}
11     \symdef{inverse}[args=1]{{#1}^{\comp{-1}}}
12  \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:

```
 1  \begin{smodule}{ring}
 2     \begin{copymodule}{group}{addition}
 3         \renamedecl[name=universe]{universe}{runiverse}
 4         \renamedecl[name=plus]{operation}{rplus}
 5         \renamedecl[name=zero]{unit}{rzero}
 6         \renamedecl[name=uminus]{inverse}{ruminus}
 7     \end{copymodule}
 8     \notation*{rplus}[plus,op=+,prec=60]{#1 \comp+ #2}
 9         \notation*{rzero}[zero]{\comp0}
10         \notation*{ruminus}[uminus,op=-]{\comp- #1}
11     \begin{copymodule}{monoid}{multiplication}
12     \assign{universe}{\runiverse}
13         \renamedecl[name=times]{operation}{rtimes}
14         \renamedecl[name=one]{unit}{rone}
15     \end{copymodule}
16     \notation*{rtimes}[cdot,op=\cdot,prec=50]{#1 \comp\cdot #2}
17         \notation*{rone}[one]{\comp1}
18         Test: $\rtimes a{\rplus c{\rtimes de}}$
19 \end{smodule}
```

Output:

Test: $a{\cdot}(c{+}d{\cdot}e)$

.

TODO: explain donotclone

3.4.5 The `interpretmodule` Environment

TODO: explain

Example 30

Input:

```
 1  \begin{smodule}{int}
 2     \symdef{Integers}{\comp{\mathbb Z}}
 3     \symdef{plus}[args=2,op=+]{#1 \comp+ #2}
 4     \symdef{zero}{\comp0}
 5     \symdef{uminus}[args=1,op=-]{\comp-#1}
 6
 7     \begin{interpretmodule}{group}{intisgroup}
 8         \assign{universe}{\Integers}
 9         \assign{operation}{\plus!}
10         \assign{unit}{\zero}
11         \assign{inverse}{\uminus!}
12     \end{interpretmodule}
13 \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:

```
1  \symdef{Nat}[
2    name=natural-number,
3    type=\set
4  ]{\comp{\mathbb{N}}}
5
6  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.

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

Example 32

Input:

```
1  \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:

```
1  \addition{\comp{The sum of} \arg{$\svar{n}$} \comp{ and }\arg{$\svar{m}$}}
2 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→ As expected, the above example is translated to OMDoc/Mmt as an
> —M→ `OMA` with `<OMS name="...?addition"/>` as head and `<OMV name="n"/>` and
> ⤳T⤳ `<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:
>
> ```
> 1  \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:
>
> ```
> 1  \addition{\comp{adding}
> 2    \arg[2]{$\svar{k}$}
> 3    \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:

```
1  Given $\addition{\svar{n}}{\svar{m}}$, then
2 $\addition*{
3     \arg*{\addition{\svar{n}}{\svar{m}}}
4     \comp{+}
5     \arg{\svar{k}}
6 }$ 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:

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

Output:

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

.

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 us to resume our earlier example for monoids much more nicely:

Example 38

Input:

```
1  \begin{mathstructure}{monoid}
2     \symdef{universe}[type=\set]{\comp{U}}
3     \symdef{op}[
4         args=2,
5         type=\funtype{\universe,\universe}{\universe},
6         op=\circ
7     ]{#1 \comp{\circ} #2}
8     \symdef{unit}[type=\universe]{\comp{e}}
9
10    \begin{sparagraph}[type=symdoc,for=monoid]
11        A \definame{monoid} is a structure
12        $\mathstruct{\universe,\op!,\unit}$
13        where $\op!:\funtype{\universe}{\universe}$ and
14        $\inset{\unit}{\universe}$ such that
15
16        \begin{sassertion}[name=associative,
17            type=axiom,
18            title=Associativity]
19            $\op!$ is associative
20        \end{sassertion}
21        \begin{sassertion}[name=isunit,
22            type=axiom,
23            title=Unit]
24            $\equal{\op{\svar{x}}{\unit}}{\svar{x}}$
25            for all $\inset{\svar{x}}{\universe}$
26        \end{sassertion}
27    \end{sparagraph}
28 \end{mathstructure}
29
30 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

```
1 \instantiate{intmonoid}{
2     universe = Int ,
3     op = addition ,
4     unit = zero
5 }{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 SpTEX 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 SpTEX 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

```
1 \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:

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

```
4     \else
5        \begin{theorem}[\sassertiontitle]
6   \fi}
7   {\end{theorem}}
```

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

```
1 \stexpatchdefinition
2   {\ifx\sdefinitiontitle\@empty
3        \begin{definition}
4      \else
5        \begin{definition}[\sdefinitiontitle]
6   \fi}
7   {\end{definition}}
```

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

Apart from the environments, we can control how SᴛᴇX highlights variables, notation components, `\symref`s and `\definiendum`s, 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

```
1 \protected\def\symrefemph@uri#1#2{
2   \pdftooltip{
3     \srefsymuri{#2}{\symrefemph{#1}}
4   }{
5     URI:~\detokenize{#2}
6   }
7 }
```

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

$$\texttt{property="stex:}⟨property⟩\texttt{", resource="}⟨resource⟩\texttt{".}$$

`\stex_annotate_invisible:n` adds the attributes

$$\texttt{stex:visible="false", style="display:none".}$$

`\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.

48

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

\stex_filestack_push:n   Push and pop (repsecively) a file path to the file stack, to keep track of the current file.
\stex_filestack_pop: 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 four 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, or

4. a path specified in `~/.stex/mathhub.path`.

In all four 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 $\langle <URI> \rangle$ 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:n`*TF* ⋆

　　　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_module_ns_str`. Additionally, the sub path relative to the current repository is stored in `\l_stex_module_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 is, as and bs),

- `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 !.

`\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]

---

[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

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.

method

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

`step`s, `proofcomment`s, 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 sub-proofs, 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:

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

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

sfragment

id
creators
contributors
short
loadmodules

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.

The structure of the document is given by the `omgroup` environment just like in OM-Doc. 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

```
\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   The `\currentsectionlevel` macro supplies the name of the current sectioning level,
\CurrentSectionLevel   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

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`.

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.

`\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

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.

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.

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

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.

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

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

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).

81

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

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

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

21.2.2 Notes and Slides

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.

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).

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}`.

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.

\inputref*
nparagraph
nfragment
ndefinition
nexample
nsproof
nassertion

21.2.3   Header and Footer Lines of the Slides

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

\setslidelogo

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.

\setsource

\setlicensing

21.2.4   Frame Images

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]

\frameimage

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

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

\activateexcursion   where `\activateexcursion{⟨path⟩}` augments the `\printexcursions` macro by a
\printexcursions   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.

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

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   `\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 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. 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
boxed
test

mh

showmeta

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.

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.

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

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

The main environment provided by the `problem` package is (surprise surprise) the `problem` environment. It is used to mark up problems and exercises. The environ-
ment 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,name=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

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 argu-
ment 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

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.

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.

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

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` 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

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 distri-
bution 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,name=functions1]
  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=Noooooooo,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   This package supplies the `assignment` environment that groups problems into assignment
number   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
title   — the ordinal of the `assignment` environment), `title` (for the assignment title; this is
type   referenced in the title of the assignment sheet), `type` (for the assignment type; e.g. "quiz",
given   or "homework"), `given` (for the date the assignment was given), and `due` (for the date
due   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   `\testspace` takes an argument that expands to a dimension, and leaves vertical
\testnewpage   space accordingly. `\testnewpage` makes a new page in `test` mode, and `\testemptypage`
\testemptypage generates an empty page with the cautionary message that this page was intentionally left empty.

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

23.2.4 Including Assignments

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.

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)

2022-04-10

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.

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

```
1  ⟨*cls⟩
2
3  %%%%%%%%%%%%  basics.dtx   %%%%%%%%%%%%
4
5  \RequirePackage{expl3,l3keys2e}
6  \ProvidesExplClass{stex}{2022/03/03}{3.1.0}{sTeX document class}
7
8  \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
9  \ProcessOptions
10
11  \bool_set_true:N \c_stex_document_class_bool
12
13  \RequirePackage{stex}
14
15  \stex_html_backend:TF {
16    \LoadClass{article}
17  }{
18    \LoadClass[border=1px,varwidth,crop=false]{standalone}
19    \setlength\textwidth{15cm}
20  }
21  \RequirePackage{standalone}
22  ⟨/cls⟩
```

24.2 Preliminaries

```
23  ⟨*package⟩
24
25  %%%%%%%%%%%%  basics.dtx   %%%%%%%%%%%%
26
```

```
27 \RequirePackage{expl3,l3keys2e,ltxcmds}
28 \ProvidesExplPackage{stex}{2022/03/03}{3.1.0}{sTeX package}
29
30 \bool_if_exist:NF \c_stex_document_class_bool {
31   \bool_set_false:N \c_stex_document_class_bool
32   \RequirePackage{standalone}
33 }
34
35 \message{^^J
36   *****************************^^J
37   *~This~is~sTeX~version~3.1.0~*^^J
38   *****************************^^J
39 ^^J}
40
41 %\RequirePackage{morewrites}
42 %\RequirePackage{amsmath}
43
```

Package options:
```
44 \keys_define:nn { stex } {
45   debug      .clist_set:N  = \c_stex_debug_clist ,
46   lang       .clist_set:N  = \c_stex_languages_clist ,
47   mathhub    .tl_set_x:N   = \mathhub ,
48   usesms     .bool_set:N   = \c_stex_persist_mode_bool ,
49   writesms   .bool_set:N   = \c_stex_persist_write_mode_bool ,
50   image      .bool_set:N   = \c_tikzinput_image_bool,
51   unknown    .code:n       = {}
52 }
53 \ProcessKeysOptions { stex }
```

\stex
\sTeX

The sTEXlogo:
```
54 \RequirePackage{xspace}
55 \protected\def\stex{
56   \@ifundefined{texorpdfstring}{\let\texorpdfstring\@firstoftwo}{}
57   \texorpdfstring{\raisebox{-.5ex}S\kern-.5ex\TeX}{sTeX}\xspace
58 }
59 \let\sTeX\stex
```

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

24.3 Messages and logging

```
60 ⟨@@=stex_log⟩
```

Warnings and error messages
```
61 \msg_new:nnn{stex}{error/unknownlanguage}{
62   Unknown~language:~#1
63 }
64 \msg_new:nnn{stex}{warning/nomathhub}{
65   MATHHUB~system~variable~not~found~and~no~
66   \detokenize{\mathhub}-value~set!
67 }
68 \msg_new:nnn{stex}{error/deactivated-macro}{
69   The~\detokenize{#1}~command~is~only~allowed~in~#2!
70 }
```

\stex_debug:nn A simple macro issuing package messages with subpath.

```
71 \cs_new_protected:Nn \stex_debug:nn {
72   \clist_if_in:NnTF \c_stex_debug_clist { all } {
73     \msg_set:nnn{stex}{debug / #1}{
74       \\Debug~#1:~#2\\
75     }
76     \msg_none:nn{stex}{debug / #1}
77   }{
78     \clist_if_in:NnT \c_stex_debug_clist { #1 } {
79       \msg_set:nnn{stex}{debug / #1}{
80         \\Debug~#1:~#2\\
81       }
82       \msg_none:nn{stex}{debug / #1}
83     }
84   }
85 }
```

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

Redirecting messages:

```
86 \clist_if_in:NnTF \c_stex_debug_clist {all} {
87   \msg_redirect_module:nnn{ stex }{ none }{ term }
88 }{
89   \clist_map_inline:Nn \c_stex_debug_clist {
90     \msg_redirect_name:nnn{ stex }{ debug / ##1 }{ term }
91   }
92 }
93
94 \stex_debug:nn{log}{debug~mode~on}
```

24.4   HTML Annotations

```
95 ⟨@@=stex_annotate⟩
```

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

```
96 \tl_new:N \l_stex_html_arg_tl
```

(*End definition for* `\l_stex_html_arg_tl` *and* `\c_stex_html_emptyarg_tl`. *These variables are documented on page* **??**.)

_stex_html_checkempty:n

```
97 \cs_new_protected:Nn \_stex_html_checkempty:n {
98   \tl_set:Nn \l_stex_html_arg_tl { #1 }
99   \tl_if_empty:NT \l_stex_html_arg_tl {
100     \tl_set_eq:NN \l_stex_html_arg_tl \c_stex_html_emptyarg_tl
101   }
102 }
```

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

\stex_if_do_html_p:
\stex_if_do_html:_TF_ Whether to (locally) produce HTML output

```
103 \bool_new:N \_stex_html_do_output_bool
104 \bool_set_true:N \_stex_html_do_output_bool
105
```

```
106 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
107   \bool_if:nTF \_stex_html_do_output_bool
108     \prg_return_true: \prg_return_false:
109 }
```

(*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

```
110 \cs_new_protected:Nn \stex_suppress_html:n {
111   \exp_args:Nne \use:nn {
112     \bool_set_false:N \_stex_html_do_output_bool
113     #1
114   }{
115     \stex_if_do_html:T {
116       \bool_set_true:N \_stex_html_do_output_bool
117     }
118   }
119 }
```

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

\stex_annotate:nnn
\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.

```
120 \tl_if_exist:NF\stex@backend{
121   \ifcsname if@rustex\endcsname
122     \def\stex@backend{rustex}
123   \else
124     \ifcsname if@latexml\endcsname
125       \def\stex@backend{latexml}
126     \else
127       \def\stex@backend{pdflatex}
128     \fi
129   \fi
130 }
131 \input{stex-backend-\stex@backend.cfg}
```

(*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

```
132 ⟨@@=stex_language⟩
```

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

```
133 \prop_const_from_keyval:Nn \c_stex_languages_prop {
134   en = english ,
135   de = ngerman ,
136   ar = arabic ,
137   bg = bulgarian ,
138   ru = russian ,
139   fi = finnish ,
140   ro = romanian ,
```

```
141    tr = turkish ,
142    fr = french
143  }
144
145  \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop {
146    english  = en ,
147    ngerman  = de ,
148    arabic   = ar ,
149    bulgarian = bg ,
150    russian  = ru ,
151    finnish  = fi ,
152    romanian = ro ,
153    turkish  = tr ,
154    french   = fr
155  }
156  % todo: chinese simplified (zhs)
157  %       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:

```
158  \clist_if_empty:NF \c_stex_languages_clist {
159    \clist_clear:N \l_tmpa_clist
160    \clist_map_inline:Nn \c_stex_languages_clist {
161      \prop_get:NnNTF \c_stex_languages_prop { #1 } \l_tmpa_str {
162        \clist_put_right:No \l_tmpa_clist \l_tmpa_str
163      } {
164        \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
165      }
166    }
167    \stex_debug:nn{lang} {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
168    \RequirePackage[\clist_use:Nn \l_tmpa_clist,]{babel}
169  }
170
171  \AtBeginDocument{
172    \stex_html_backend:T {
173      \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
174      \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
175      \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
176      \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
177      \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
178        \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
179        \stex_debug:nn{basics} {Language~\l_tmpa_str~
180          inferred~from~file~name}
181        \stex_annotate_invisible:nnn{language}{ \l_tmpa_str }{}
182      }
183    }
184  }
```

24.6   Persistence

```
185  ⟨@@=stex_persist⟩
186  \bool_if:NTF \c_stex_persist_mode_bool {
```

```
187    \def \stex_persist:n #1 {}
188    \def \stex_persist:x #1 {}
189 }{
190    \bool_if:NTF \c_stex_persist_write_mode_bool {
191    \iow_new:N \c__stex_persist_iow
192    \iow_open:Nn \c__stex_persist_iow{\jobname.sms}
193    \AtEndDocument{
194      \iow_close:N \c__stex_persist_iow
195    }
196    \cs_new_protected:Nn \stex_persist:n {
197      \tl_set:Nn \l_tmpa_tl { #1 }
198      \regex_replace_all:nnN { \cP\# } { \cO\# } \l_tmpa_tl
199      \exp_args:NNo \iow_now:Nn \c__stex_persist_iow \l_tmpa_tl
200    }
201    \cs_generate_variant:Nn \stex_persist:n {x}
202    }{
203      \def \stex_persist:n #1 {}
204      \def \stex_persist:x #1 {}
205    }
206 }
```

24.7   Auxiliary Methods

```
207 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
208    \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
209    \def#1{
210      \msg_error:nnnn{stex}{error/deactivated-macro}{\detokenize{#1}}{#2}
211    }
212 }
```

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

```
213 \cs_new_protected:Nn \stex_reactivate_macro:N {
214    \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname
215 }
```

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

```
216 \protected\def\ignorespacesandpars{
217    \begingroup\catcode13=10\relax
218    \@ifnextchar\par{
219      \endgroup\expandafter\ignorespacesandpars\@gobble
220    }{
221      \endgroup
222    }
223 }
224
225 \cs_new_protected:Nn \stex_copy_control_sequence:NNN {
226    \tl_set:Nx \_tmp_args_tl {\cs_argument_spec:N #2}
227    \exp_args:NNo \tl_remove_all:Nn \_tmp_args_tl \c_hash_str
228    \int_set:Nn \l_tmpa_int {\tl_count:N \_tmp_args_tl}
```

```
229
230   \tl_clear:N \_tmp_args_tl
231   \int_step_inline:nn \l_tmpa_int {
232     \tl_put_right:Nx \_tmp_args_tl {{\exp_not:n{####}\exp_not:n{##1}}}
233   }
234
235   \tl_set:Nn #3 {\cs_generate_from_arg_count:NNnn #1 \cs_set:Npn}
236   \tl_put_right:Nx #3 { {\int_use:N \l_tmpa_int}{
237       \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
238       \exp_after:wN\exp_after:wN\exp_after:wN {
239         \exp_after:wN #2 \_tmp_args_tl
240       }
241   }}
242 }
243 \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {cNN}
244 \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {NcN}
245 \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {ccN}
```

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

\MMTrule

```
246 \NewDocumentCommand \MMTrule {m m}{
247   \seq_set_split:Nnn \l_tmpa_seq , {#2}
248   \int_zero:N \l_tmpa_int
249   \stex_annotate_invisible:nnn{mmtrule}{scala://#1}{
250     $\seq_map_inline:Nn \l_tmpa_seq {
251       \int_incr:N \l_tmpa_int
252       \stex_annotate:nnn{arg}{i\int_use:N \l_tmpa_int}{##1}
253     }$
254   }
255 }
256
257 \NewDocumentCommand \MMTinclude {m}{
258   \stex_annotate_invisible:nnn{import}{#1}{}
259 }
260 ⟨/package⟩
```

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

Chapter 25

STEX
-MathHub Implementation

```
261 ⟨*package⟩
262
263 %%%%%%%%%%%%    mathhub.dtx    %%%%%%%%%%%%
264
265 ⟨@@=stex_path⟩
```

Warnings and error messages

```
266 \msg_new:nnn{stex}{error/norepository}{
267   No~archive~#1~found~in~#2
268 }
269 \msg_new:nnn{stex}{error/notinarchive}{
270   Not~currently~in~an~archive,~but~\detokenize{#1}~
271   needs~one!
272 }
273 \msg_new:nnn{stex}{error/nofile}{
274   \detokenize{#1}~could~not~find~file~#2
275 }
276 \msg_new:nnn{stex}{error/twofiles}{
277   \detokenize{#1}~found~two~candidates~for~#2
278 }
```

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`

```
279 \cs_new_protected:Nn \stex_path_from_string:Nn {
280   \str_set:Nx \l_tmpa_str { #2 }
281   \str_if_empty:NTF \l_tmpa_str {
282     \seq_clear:N #1
283   }{
284     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
285     \sys_if_platform_windows:T{
286       \seq_clear:N \l_tmpa_tl
```

```
287         \seq_map_inline:Nn #1 {
288           \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
289           \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
290         }
291         \seq_set_eq:NN #1 \l_tmpa_tl
292       }
293       \stex_path_canonicalize:N #1
294     }
295 }
296
```

(*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`

```
297 \cs_new_protected:Nn \stex_path_to_string:NN {
298     \exp_args:NNe \str_set:Nn #2 { \seq_use:Nn #1 / }
299 }
300
301 \cs_new:Nn \stex_path_to_string:N {
302     \seq_use:Nn #1 /
303 }
```

(*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.

```
304 \str_const:Nn \c__stex_path_dot_str {.}
305 \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.

```
306 \cs_new_protected:Nn \stex_path_canonicalize:N {
307     \seq_if_empty:NF #1 {
308       \seq_clear:N \l_tmpa_seq
309       \seq_get_left:NN #1 \l_tmpa_tl
310       \str_if_empty:NT \l_tmpa_tl {
311         \seq_put_right:Nn \l_tmpa_seq {}
312       }
313       \seq_map_inline:Nn #1 {
314         \str_set:Nn \l_tmpa_tl { ##1 }
315         \str_if_eq:NNF \l_tmpa_tl \c__stex_path_dot_str {
316           \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
317             \seq_if_empty:NTF \l_tmpa_seq {
318               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
319                 \c__stex_path_up_str
320               }
321             }{
322               \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
323               \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
324                 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
325                   \c__stex_path_up_str
326                 }
327               }{
```

```
328                \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
329              }
330            }
331          }{
332            \str_if_empty:NF \l_tmpa_tl {
333              \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
334            }
335          }
336        }
337      }
338      \seq_gset_eq:NN #1 \l_tmpa_seq
339    }
340 }
```

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

```
341 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
342    \seq_if_empty:NTF #1 {
343      \prg_return_false:
344    }{
345      \seq_get_left:NN #1 \l_tmpa_tl
346      \sys_if_platform_windows:TF{
347        \str_if_in:NnTF \l_tmpa_tl {:}{
348          \prg_return_true:
349        }{
350          \prg_return_false:
351        }
352      }{
353        \str_if_empty:NTF \l_tmpa_tl {
354          \prg_return_true:
355        }{
356          \prg_return_false:
357        }
358      }
359    }
360 }
```

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

25.2 PWD and kpsewhich

```
361 \str_new:N\l_stex_kpsewhich_return_str
362 \cs_new_protected:Nn \stex_kpsewhich:n {
363    \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
364    \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
365    \tl_trim_spaces:N \l_stex_kpsewhich_return_str
366 }
```

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

We determine the PWD

```
367 \sys_if_platform_windows:TF{
368   \begingroup\escapechar=-1\catcode`\\=12
369   \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
370   \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
371   \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_
372 }{
373   \stex_kpsewhich:n{-var-value~PWD}
374 }
375
376 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
377 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
378 \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

```
379 ⟨@@=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 ST_EX-purposes.

keeps track of file changes

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

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

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

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

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

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

```
385 \cs_new_protected:Nn \stex_filestack_push:n {
386   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
387   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
388     \stex_path_from_string:Nn\g_stex_currentfile_seq{
389       \c_stex_pwd_str/#1
390     }
391   }
392   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
393   \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
394 }
```

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

\stex_filestack_pop:

```
395 \cs_new_protected:Nn \stex_filestack_pop: {
396   \seq_if_empty:NF\g__stex_files_stack{
397     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
398   }
399   \seq_if_empty:NTF\g__stex_files_stack{
400     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
401   }{
402     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
403     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
404   }
405 }
```

Hooks for the current file:

```
406 \AddToHook{file/before}{
407   \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
408 }
409 \AddToHook{file/after}{
410   \stex_filestack_pop:
411 }
```

25.4 MathHub Repositories

```
412 ⟨@@=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.

```
413 \str_if_empty:NTF\mathhub{
414   \sys_if_platform_windows:TF{
415     \begingroup\escapechar=-1\catcode`\\=12
416     \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str MATHHUB\c_percent_str}
417     \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
418     \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_ste
419   }{
420     \stex_kpsewhich:n{-var-value~MATHHUB}
421   }
422   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
423
424   \str_if_empty:NT \c_stex_mathhub_str {
425     \sys_if_platform_windows:TF{
426       \begingroup\escapechar=-1\catcode`\\=12
427       \exp_args:Nx\stex_kpsewhich:n{-var-value~HOME}
428       \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
429       \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_s
430     }{
431       \stex_kpsewhich:n{-var-value~HOME}
432     }
433     \ior_open:NnT \l_tmpa_ior{\l_stex_kpsewhich_return_str / .stex / mathhub.path}{
434       \begingroup\escapechar=-1\catcode`\\=12
435       \ior_str_get:NN \l_tmpa_ior \l_tmpa_str
```

```
436         \sys_if_platform_windows:T{
437           \exp_args:NNx\str_replace_all:Nnn\l_tmpa_str{\c_backslash_str}/
438         }
439         \str_gset_eq:NN \c_stex_mathhub_str\l_tmpa_str
440         \endgroup
441         \ior_close:N \l_tmpa_ior
442       }
443     }
444     \str_if_empty:NTF\c_stex_mathhub_str{
445       \msg_warning:nn{stex}{warning/nomathhub}
446     }{
447       \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
448       \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
449     }
450   }{
451     \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
452     \stex_path_if_absolute:NF \c_stex_mathhub_seq {
453       \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
454         \c_stex_pwd_str/\mathhub
455       }
456     }
457     \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
458     \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
459   }
```

(*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

```
460 \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
461   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
462     \str_set:Nx \l_tmpa_str { #1 }
463     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
464     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
465     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
466     \__stex_mathhub_find_manifest:N \l_tmpa_seq
467     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
468       \msg_error:nnxx{stex}{error/norepository}{#1}{
469         \stex_path_to_string:N \c_stex_mathhub_str
470       }
471     } {
472       \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
473     }
474   }
475 }
```

(*End definition for* __stex_mathhub_do_manifest:n.)

\l__stex_mathhub_manifest_file_seq

```
476 \seq_new:N\l__stex_mathhub_manifest_file_seq
```

(*End definition for* \l__stex_mathhub_manifest_file_seq.)

Attempts to find the `MANIFEST.MF` in some file path and stores its path in `\l__stex_-mathhub_manifest_file_seq`:

```
477 \cs_new_protected:Nn \__stex_mathhub_find_manifest:N {
478   \seq_set_eq:NN\l_tmpa_seq #1
479   \bool_set_true:N\l_tmpa_bool
480   \bool_while_do:Nn \l_tmpa_bool {
481     \seq_if_empty:NTF \l_tmpa_seq {
482       \bool_set_false:N\l_tmpa_bool
483     }{
484       \file_if_exist:nTF{
485         \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
486       }{
487         \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
488         \bool_set_false:N\l_tmpa_bool
489       }{
490         \file_if_exist:nTF{
491           \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
492         }{
493           \seq_put_right:Nn\l_tmpa_seq{META-INF}
494           \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
495           \bool_set_false:N\l_tmpa_bool
496         }{
497           \file_if_exist:nTF{
498             \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
499           }{
500             \seq_put_right:Nn\l_tmpa_seq{meta-inf}
501             \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
502             \bool_set_false:N\l_tmpa_bool
503           }{
504             \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
505           }
506         }
507       }
508     }
509   }
510   \seq_set_eq:NN\l__stex_mathhub_manifest_file_seq\l_tmpa_seq
511 }
```

(*End definition for* `\__stex_mathhub_find_manifest:N`.)

File variable used for `MANIFEST`-files

```
512 \ior_new:N \c__stex_mathhub_manifest_ior
```

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

Stores the entries in manifest file in the corresponding property list:

```
513 \cs_new_protected:Nn \__stex_mathhub_parse_manifest:n {
514   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
515   \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
516   \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
517     \str_set:Nn \l_tmpa_str {##1}
518     \exp_args:NNoo \seq_set_split:Nnn
519       \l_tmpb_seq \c_colon_str \l_tmpa_str
520     \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
```

```
521        \exp_args:NNe \str_set:Nn \l_tmpb_tl {
522          \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
523        }
524        \exp_args:No \str_case:nnTF \l_tmpa_tl {
525          {id} {
526            \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
527              { id } \l_tmpb_tl
528          }
529          {narration-base} {
530            \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
531              { narr } \l_tmpb_tl
532          }
533          {url-base} {
534            \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
535              { docurl } \l_tmpb_tl
536          }
537          {source-base} {
538            \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
539              { ns } \l_tmpb_tl
540          }
541          {ns} {
542            \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
543              { ns } \l_tmpb_tl
544          }
545          {dependencies} {
546            \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
547              { deps } \l_tmpb_tl
548          }
549        }{}{}
550      }{}
551    }
552    \ior_close:N \c__stex_mathhub_manifest_ior
553    \stex_persist:x {
554      \prop_set_from_keyval:cn{ c_stex_mathhub_#1_manifest_prop }{
555        \exp_after:wN \prop_to_keyval:N \csname c_stex_mathhub_#1_manifest_prop\endcsname
556      }
557    }
558  }
```

(End definition for __stex_mathhub_parse_manifest:n.*)*

```
559  \cs_new_protected:Nn \stex_set_current_repository:n {
560    \stex_require_repository:n { #1 }
561    \prop_set_eq:Nc \l_stex_current_repository_prop {
562      c_stex_mathhub_#1_manifest_prop
563    }
564  }
```

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

```
565  \cs_new_protected:Nn \stex_require_repository:n {
566    \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
567      \stex_debug:nn{mathhub}{Opening~archive:~#1}
```

```
568     \__stex_mathhub_do_manifest:n { #1 }
569   }
570 }
```

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

\l_stex_current_repository_prop Current MathHub repository

```
571 %\prop_new:N \l_stex_current_repository_prop
572 \bool_if:NF \c_stex_persist_mode_bool {
573   \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
574   \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
575     \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
576   } {
577     \__stex_mathhub_parse_manifest:n { main }
578     \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
579       \l_tmpa_str
580     \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str _manifest_prop }
581       \c_stex_mathhub_main_manifest_prop
582     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
583     \stex_debug:nn{mathhub}{Current~repository:~
584       \prop_item:Nn \l_stex_current_repository_prop {id}
585     }
586   }
587 }
```

(*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.

```
588 \cs_new_protected:Nn \stex_in_repository:nn {
589   \str_set:Nx \l_tmpa_str { #1 }
590   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
591   \str_if_empty:NTF \l_tmpa_str {
592     \prop_if_exist:NTF \l_stex_current_repository_prop {
593       \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
594       \exp_args:Ne \l_tmpa_cs{
595         \prop_item:Nn \l_stex_current_repository_prop { id }
596       }
597     }{
598       \l_tmpa_cs{}
599     }
600   }{
601     \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}
602     \stex_require_repository:n \l_tmpa_str
603     \str_set:Nx \l_tmpa_str { #1 }
604     \exp_args:Nne \use:nn {
605       \stex_set_current_repository:n \l_tmpa_str
606       \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
607     }{
608       \stex_debug:nn{mathhub}{switching~back~to:~
609         \prop_if_exist:NTF \l_stex_current_repository_prop {
610           \prop_item:Nn \l_stex_current_repository_prop { id }:~
611           \meaning\l_stex_current_repository_prop
612         }{
```

```
613            no~repository
614          }
615        }
616        \prop_if_exist:NTF \l_stex_current_repository_prop {
617          \stex_set_current_repository:n {
618            \prop_item:Nn \l_stex_current_repository_prop { id }
619          }
620        }{
621          \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
622        }
623      }
624    }
625  }
```

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

25.5 Using Content in Archives

\mhpath

```
626  \def \mhpath #1 #2 {
627    \exp_args:Ne \tl_if_empty:nTF{#1}{
628      \c_stex_mathhub_str /
629        \prop_item:Nn \l_stex_current_repository_prop { id }
630        / source / #2
631    }{
632      \c_stex_mathhub_str / #1 / source / #2
633    }
634  }
```

(*End definition for* `\mhpath`. *This function is documented on page 50.*)

\inputref
\mhinput

```
635  \newif \ifinputref \inputreffalse
636
637  \cs_new_protected:Nn \__stex_mathhub_mhinput:nn {
638    \stex_in_repository:nn {#1} {
639      \ifinputref
640        \input{ \c_stex_mathhub_str / ##1 / source / #2 }
641      \else
642        \inputreftrue
643        \input{ \c_stex_mathhub_str / ##1 / source / #2 }
644        \inputreffalse
645      \fi
646    }
647  }
648  \NewDocumentCommand \mhinput { O{} m}{
649    \__stex_mathhub_mhinput:nn{ #1 }{ #2 }
650  }
651
652  \cs_new_protected:Nn \__stex_mathhub_inputref:nn {
653    \stex_in_repository:nn {#1} {
654      \stex_html_backend:TF {
655        \str_clear:N \l_tmpa_str
```

```
656        \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
657          \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
658        }
659        \stex_annotate_invisible:nnn{inputref}{
660          \l_tmpa_str / #2
661        }{}
662      }{
663        \begingroup
664          \inputreftrue
665          \tl_if_empty:nTF{ ##1 }{
666            \input{#2}
667          }{
668            \input{ \c_stex_mathhub_str / ##1 / source / #2 }
669          }
670        \endgroup
671      }
672    }
673  }
674  \NewDocumentCommand \inputref { O{} m }{
675    \__stex_mathhub_inputref:nn{ #1 }{ #2 }
676  }
```

(*End definition for* `\inputref` *and* `\mhinput`*. These functions are documented on page 50.*)

```
677  \cs_new_protected:Nn \__stex_mathhub_mhbibresource:nn {
678    \stex_in_repository:nn {#1} {
679      \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
680    }
681  }
682  \newcommand\addmhbibresource[2][]{
683    \__stex_mathhub_mhbibresource:nn{ #1 }{ #2 }
684  }
```

(*End definition for* `\addmhbibresource`*. This function is documented on page 50.*)

```
685  \cs_new_protected:Npn \libinput #1 {
686    \prop_if_exist:NF \l_stex_current_repository_prop {
687      \msg_error:nnn{stex}{error/notinarchive}\libinput
688    }
689    \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
690      \msg_error:nnn{stex}{error/notinarchive}\libinput
691    }
692    \seq_clear:N \l__stex_mathhub_libinput_files_seq
693    \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
694    \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
695
696    \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
697      \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #1.tex}
698      \IfFileExists{ \l_tmpa_str }{
699        \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
700      }{}
701      \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
702      \seq_put_right:No \l_tmpa_seq \l_tmpa_str
```

```
703    }

705    \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}
706    \IfFileExists{ \l_tmpa_str }{
707      \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
708    }{}

710    \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
711      \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
712    }{
713      \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
714        \input{ ##1 }
715      }
716    }
717  }
```

(End definition for `\libinput`*. This function is documented on page* *50.)*

\libusepackage

```
718  \NewDocumentCommand \libusepackage {O{} m} {
719    \prop_if_exist:NF \l_stex_current_repository_prop {
720      \msg_error:nnn{stex}{error/notinarchive}\libusepackage
721    }
722    \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
723      \msg_error:nnn{stex}{error/notinarchive}\libusepackage
724    }
725    \seq_clear:N \l__stex_mathhub_libinput_files_seq
726    \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
727    \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str

729    \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
730      \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #2}
731      \IfFileExists{ \l_tmpa_str.sty }{
732        \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
733      }{}
734      \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
735      \seq_put_right:No \l_tmpa_seq \l_tmpa_str
736    }

738    \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2}
739    \IfFileExists{ \l_tmpa_str.sty }{
740      \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
741    }{}

743    \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
744      \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
745    }{
746      \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
747        \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
748          \usepackage[#1]{ ##1 }
749        }
750      }{
751        \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
752      }
```

```
753        }
754  }
```

(*End definition for* `\libusepackage`*. This function is documented on page 50.*)

`\mhgraphics`
`\cmhgraphics`

```
755
756  \AddToHook{begindocument}{
757  \ltx@ifpackageloaded{graphicx}{
758      \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
759      \newcommand\mhgraphics[2][]{%
760        \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
761        \includegraphics[#1]{\mhpath\Gin@mhrepos{#2}}}
762      \newcommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
763    }{}
```

(*End definition for* `\mhgraphics` *and* `\cmhgraphics`*. These functions are documented on page 50.*)

`\lstinputmhlisting`
`\clstinputmhlisting`

```
764  \ltx@ifpackageloaded{listings}{
765      \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
766      \newcommand\lstinputmhlisting[2][]{%
767        \def\lst@mhrepos{}\setkeys{lst}{#1}%
768        \lstinputlisting[#1]{\mhpath\lst@mhrepos{#2}}}
769      \newcommand\clstinputmhlisting[2][]{\begin{center}\lstinputmhlisting[#1]{#2}\end{center}
770    }{}
771  }
772
773  ⟨/package⟩
```

(*End definition for* `\lstinputmhlisting` *and* `\clstinputmhlisting`*. These functions are documented on page 50.*)

114

Chapter 26

sTEX
-References Implementation

774 ⟨∗package⟩

775

776 %%%%%%%%%%%%   references.dtx   %%%%%%%%%%%%

777

778 ⟨@@=stex_refs⟩

Warnings and error messages

779

References are stored in the file `\jobname.sref`, to enable cross-referencing external documents.

780 %\iow_new:N \c__stex_refs_refs_iow
781 \AtBeginDocument{
782 % \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
783 }
784 \AtEndDocument{
785 % \iow_close:N \c__stex_refs_refs_iow
786 }

\STEXreftitle

787 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}

788

789 \NewDocumentCommand \STEXreftitle { m } {
790   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
791 }

(*End definition for* \STEXreftitle. *This function is documented on page 51.*)

26.1   Document URIs and URLs

\l_stex_current_docns_str

792 \str_new:N \l_stex_current_docns_str

(*End definition for* \l_stex_current_docns_str. *This variable is documented on page 51.*)

```
793 \cs_new_protected:Nn \stex_get_document_uri: {
794   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
795   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
796   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
797   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
798   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
799
800   \str_clear:N \l_tmpa_str
801   \prop_if_exist:NT \l_stex_current_repository_prop {
802     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
803       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
804     }
805   }
806
807   \str_if_empty:NTF \l_tmpa_str {
808     \str_set:Nx \l_stex_current_docns_str {
809       file:/\stex_path_to_string:N \l_tmpa_seq
810     }
811   }{
812     \bool_set_true:N \l_tmpa_bool
813     \bool_while_do:Nn \l_tmpa_bool {
814       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
815       \exp_args:No \str_case:nnTF { \l_tmpb_str } {
816         {source} { \bool_set_false:N \l_tmpa_bool }
817       }{}{
818         \seq_if_empty:NT \l_tmpa_seq {
819           \bool_set_false:N \l_tmpa_bool
820         }
821       }
822     }
823
824     \seq_if_empty:NTF \l_tmpa_seq {
825       \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
826     }{
827       \str_set:Nx \l_stex_current_docns_str {
828         \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
829       }
830     }
831   }
832 }
```

(End definition for \stex_get_document_uri:*. This function is documented on page 51.)*

```
833 \str_new:N \l_stex_current_docurl_str
```

(End definition for \l_stex_current_docurl_str*. This variable is documented on page 51.)*

```
834 \cs_new_protected:Nn \stex_get_document_url: {
835   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
836   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
837   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
```

116

```
838    \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
839    \seq_put_right:No \l_tmpa_seq \l_tmpb_str
840
841    \str_clear:N \l_tmpa_str
842    \prop_if_exist:NT \l_stex_current_repository_prop {
843      \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
844        \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
845          \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
846        }
847      }
848    }
849
850    \str_if_empty:NTF \l_tmpa_str {
851      \str_set:Nx \l_stex_current_docurl_str {
852        file:/\stex_path_to_string:N \l_tmpa_seq
853      }
854    }{
855      \bool_set_true:N \l_tmpa_bool
856      \bool_while_do:Nn \l_tmpa_bool {
857        \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
858        \exp_args:No \str_case:nnTF { \l_tmpb_str } {
859          {source} { \bool_set_false:N \l_tmpa_bool }
860        }{}{
861          \seq_if_empty:NT \l_tmpa_seq {
862            \bool_set_false:N \l_tmpa_bool
863          }
864        }
865      }
866
867      \seq_if_empty:NTF \l_tmpa_seq {
868        \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
869      }{
870        \str_set:Nx \l_stex_current_docurl_str {
871          \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
872        }
873      }
874    }
875  }
```

(*End definition for* `\stex_get_document_url:`. *This function is documented on page* *51.*)

26.2   Setting Reference Targets

```
876  \str_const:Nn \c__stex_refs_url_str{URL}
877  \str_const:Nn \c__stex_refs_ref_str{REF}
878  \str_new:N \l__stex_refs_curr_label_str
879  % @currentlabel -> number
880  % @currentlabelname -> title
881  % @currentHref -> name.number <- id of some kind
882  % \theH# -> \arabic{section}
883  % \the#  -> number
884  % \hyper@makecurrent{#}
885  \int_new:N \l__stex_refs_unnamed_counter_int
```

```
886 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
887    \stex_get_document_uri:
888    \str_clear:N \l__stex_refs_curr_label_str
889    \str_set:Nx \l_tmpa_str { #1 }
890    \str_if_empty:NT \l_tmpa_str {
891       \int_incr:N \l__stex_refs_unnamed_counter_int
892       \str_set:Nx \l_tmpa_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
893    }
894    \str_set:Nx \l__stex_refs_curr_label_str {
895       \l_stex_current_docns_str?\l_tmpa_str
896    }
897    \seq_if_exist:cF{g__stex_refs_labels_\l_tmpa_str _seq}{
898       \seq_new:c {g__stex_refs_labels_\l_tmpa_str _seq}
899    }
900    \seq_if_in:coF{g__stex_refs_labels_\l_tmpa_str _seq}\l__stex_refs_curr_label_str {
901       \seq_gput_right:co{g__stex_refs_labels_\l_tmpa_str _seq}\l__stex_refs_curr_label_str
902    }
903    \stex_if_smsmode:TF {
904       \stex_get_document_url:
905       \str_gset_eq:cN {sref_url_\l__stex_refs_curr_label_str _str}\l_stex_current_docurl_str
906       \str_gset_eq:cN {sref_\l__stex_refs_curr_label_str _type}\c__stex_refs_url_str
907    }{
908       %\iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=~\expandafter\unexpanded\expandafter{
909       \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
910       \immediate\write\@auxout{\stexauxadddocref{\l_stex_current_docns_str}{\l_tmpa_str}}
911       \str_gset:cx {sref_\l__stex_refs_curr_label_str _type}\c__stex_refs_ref_str
912    }
913 }
```

(*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.

```
914 \cs_new_protected:Npn \stexauxadddocref #1 #2 {
915    \str_set:Nn \l_tmpa_str {#1?#2}
916    \str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
917    \seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
918       \seq_new:c {g__stex_refs_labels_#2_seq}
919    }
920    \seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
921       \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
922    }
923 }
```

To avoid resetting the same macros when the .aux-file is read at the end of the document:

```
924 \AtEndDocument{
925    \def\stexauxadddocref#1 #2 {}{}
926 }
```

```
927 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
928    \stex_if_smsmode:TF {
929       \str_if_exist:cF{sref_sym_#1_type}{
930          \stex_get_document_url:
931          \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str
```

```
932         \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
933       }
934   }{
935     \str_if_empty:NF \l__stex_refs_curr_label_str {
936       \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
937       \immediate\write\@auxout{
938         \exp_not:N\expandafter\def\exp_not:N\csname \exp_not:N\detokenize{sref_sym_#1_label_
939           \l__stex_refs_curr_label_str
940         }
941       }
942     }
943   }
944 }
```

(*End definition for* `\stex_ref_new_sym_target:n`*. This function is documented on page* *51.*)

26.3   Using References

```
945 \str_new:N \l__stex_refs_indocument_str
```

`\sref`   Optional arguments:

```
946
947 \keys_define:nn { stex / sref } {
948   linktext     .tl_set:N  = \l__stex_refs_linktext_tl ,
949   fallback     .tl_set:N  = \l__stex_refs_fallback_tl ,
950   pre          .tl_set:N  = \l__stex_refs_pre_tl ,
951   post         .tl_set:N  = \l__stex_refs_post_tl ,
952 }
953 \cs_new_protected:Nn \__stex_refs_args:n {
954   \tl_clear:N \l__stex_refs_linktext_tl
955   \tl_clear:N \l__stex_refs_fallback_tl
956   \tl_clear:N \l__stex_refs_pre_tl
957   \tl_clear:N \l__stex_refs_post_tl
958   \str_clear:N \l__stex_refs_repo_str
959   \keys_set:nn { stex / sref } { #1 }
960 }
```

The actual macro:

```
961 \NewDocumentCommand \sref { O{} m}{
962   \__stex_refs_args:n { #1 }
963   \str_if_empty:NTF \l__stex_refs_indocument_str {
964     \str_set:Nx \l_tmpa_str { #2 }
965     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
966     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
967       \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
968         \seq_get_left:cNF {g__stex_refs_labels_\l_tmpa_str _seq} \l_tmpa_str {
969           \str_clear:N \l_tmpa_str
970         }
971       }{
972         \str_clear:N \l_tmpa_str
973       }
974     }{
975       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
976       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
```

```
977        \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
978        \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
979          \str_set_eq:NN \l_tmpc_str \l_tmpa_str
980          \str_clear:N \l_tmpa_str
981          \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str _seq} {
982            \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
983              \str_range:nnn { ##1 }{ -\l_tmpa_int}{ -1 }
984            }{
985              \seq_map_break:n {
986                \str_set:Nn \l_tmpa_str { ##1 }
987              }
988            }
989          }
990        }{
991          \str_clear:N \l_tmpa_str
992        }
993      }
994      \str_if_empty:NTF \l_tmpa_str {
995        \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_lin
996      }{
997        \str_if_eq:cNTF {sref_\l_tmpa_str _type} \c__stex_refs_ref_str {
998          \tl_if_empty:NTF \l__stex_refs_linktext_tl {
999            \cs_if_exist:cTF{autoref}{
1000               \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1001            }{
1002               \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1003            }
1004          }{
1005            \ltx@ifpackageloaded{hyperref}{
1006               \hyperref[sref_\l_tmpa_str]\l__stex_refs_linktext_tl
1007            }{
1008               \l__stex_refs_linktext_tl
1009            }
1010          }
1011        }{
1012          \ltx@ifpackageloaded{hyperref}{
1013             \href{\use:c{sref_url_\l_tmpa_str _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_t
1014          }{
1015             \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs
1016          }
1017        }
1018      }
1019    }{
1020      % TODO
1021    }
1022 }
```

(*End definition for* \sref. *This function is documented on page 52.*)

```
1023 \NewDocumentCommand \srefsym { O{} m}{
1024   \stex_get_symbol:n { #2 }
1025   \__stex_refs_sym_aux:nn{#1}{\l_stex_get_symbol_uri_str}
1026 }
```

120

```
1027
1028  \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1029    \str_if_exist:cTF {sref_sym_#2 _label_str }{
1030      \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1031    }{
1032      \__stex_refs_args:n { #1 }
1033      \str_if_empty:NTF \l__stex_refs_indocument_str {
1034        \tl_if_exist:cTF{sref_sym_#2 _type}{
1035          % doc uri in \l_tmpb_str
1036          \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1037          \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1038            % reference
1039            \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1040              \cs_if_exist:cTF{autoref}{
1041                \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1042              }{
1043                \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1044              }
1045            }{
1046              \ltx@ifpackageloaded{hyperref}{
1047                \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1048              }{
1049                \l__stex_refs_linktext_tl
1050              }
1051            }
1052          }{
1053            % URL
1054            \ltx@ifpackageloaded{hyperref}{
1055              \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl \
1056            }{
1057              \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_re
1058            }
1059          }
1060        }{
1061          \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_l
1062        }
1063      }{
1064        % TODO
1065      }
1066    }
1067  }
```

(*End definition for* \srefsym. *This function is documented on page* *52.*)

```
1068  \cs_new_protected:Npn \srefsymuri #1 #2 {
1069    \__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1070  }
```

(*End definition for* \srefsymuri. *This function is documented on page* *52.*)

```
1071  ⟨/package⟩
```

Chapter 27

STEX
-Modules Implementation

```
1072 ⟨*package⟩
1073
1074 %%%%%%%%%%%%%    modules.dtx    %%%%%%%%%%%%%
1075
1076 ⟨@@=stex_modules⟩
```

Warnings and error messages

```
1077 \msg_new:nnn{stex}{error/unknownmodule}{
1078   No~module~#1~found
1079 }
1080 \msg_new:nnn{stex}{error/syntax}{
1081   Syntax~error:~#1
1082 }
1083 \msg_new:nnn{stex}{error/siglanguage}{
1084   Module~#1~declares~signature~#2,~but~does~not~
1085   declare~its~language
1086 }
1087 \msg_new:nnn{stex}{warning/deprecated}{
1088   #1~is~deprecated;~please~use~#2~instead!
1089 }
1090
1091 \msg_new:nnn{stex}{error/conflictingmodules}{
1092   Conflicting~imports~for~module~#1
1093 }
```

\l_stex_current_module_str   The current module:

```
1094 \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

```
1095 \seq_new:N \l_stex_all_modules_seq
```

(*End definition for* \l_stex_all_modules_seq. *This variable is documented on page 54.*)

```
1096 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1097   \str_if_empty:NTF \l_stex_current_module_str
1098     \prg_return_false: \prg_return_true:
1099 }
```

(*End definition for* \stex_if_in_module:TF. *This function is documented on page 54.*)

```
1100 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1101   \prop_if_exist:cTF { c_stex_module_#1_prop }
1102     \prg_return_true: \prg_return_false:
1103 }
```

(*End definition for* \stex_if_module_exists:nTF. *This function is documented on page 54.*)

Only allowed within modules:

```
1104 \cs_new_protected:Nn \stex_execute_in_module:n { \stex_if_in_module:T {
1105   \stex_add_to_current_module:n { #1 }
1106   \stex_do_up_to_module:n { #1 }
1107 }}
1108 \cs_generate_variant:Nn \stex_execute_in_module:n {x}
1109
1110 \cs_new_protected:Nn \stex_add_to_current_module:n {
1111   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1112 }
1113 \cs_generate_variant:Nn \stex_add_to_current_module:n {x}
1114 \cs_new_protected:Npn \STEXexport {
1115   \begingroup
1116   \newlinechar=-1\relax
1117   \endlinechar=-1\relax
1118   %\catcode'\ = 9\relax
1119   \expandafter\endgroup\__stex_modules_export:n
1120 }
1121 \cs_new_protected:Nn \__stex_modules_export:n {
1122   \ignorespaces #1
1123   \stex_add_to_current_module:n { \ignorespaces #1 }
1124   \stex_smsmode_do:
1125 }
1126 \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.*)

```
1127 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1128   \str_set:Nx \l_tmpa_str { #1 }
1129   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
1130 }
```

(*End definition for* \stex_add_constant_to_current_module:n. *This function is documented on page 54.*)

```
1131 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1132   \str_set:Nx \l_tmpa_str { #1 }
1133   \exp_args:Nno
1134   \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1135     \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1136   }
1137 }
```

(*End definition for* \stex_add_import_to_current_module:n. *This function is documented on page* *54.*)

```
1138 \cs_new_protected:Nn \stex_collect_imports:n {
1139   \seq_clear:N \l_stex_collect_imports_seq
1140   \__stex_modules_collect_imports:n {#1}
1141 }
1142 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1143   \seq_map_inline:cn {c_stex_module_#1_imports} {
1144     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1145       \__stex_modules_collect_imports:n { ##1 }
1146     }
1147   }
1148   \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1149     \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1150   }
1151 }
```

(*End definition for* \stex_collect_imports:n. *This function is documented on page* *54.*)

```
1152 \int_new:N \l__stex_modules_group_depth_int
1153 \cs_new_protected:Nn \stex_do_up_to_module:n {
1154   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1155     #1
1156   }{
1157     #1
1158     \expandafter \tl_gset:Nn
1159     \csname l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1160     \expandafter\expandafter\expandafter\endcsname
1161     \expandafter\expandafter\expandafter { \csname
1162       l__stex_modules_aftergroup_\l_stex_current_module_str _tl\endcsname #1 }
1163     \aftergroup\__stex_modules_aftergroup_do:
1164   }
1165 }
1166 \cs_generate_variant:Nn \stex_do_up_to_module:n {x}
1167 \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1168   \stex_debug:nn{aftergroup}{\cs_meaning:c{
1169     l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1170   }}
1171   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1172     \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1173     \tl_gclear:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1174   }{
1175     \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
```

```
1176      \aftergroup\__stex_modules_aftergroup_do:
1177    }
1178  }
1179  \cs_new_protected:Nn \_stex_reset_up_to_module:n {
1180    \expandafter\let\csname l__stex_modules_aftergroup_#1_tl\endcsname\undefined
1181  }
```

(*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`).

```
1182
```

(*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.

```
1183  \str_new:N \l_stex_module_ns_str
1184  \str_new:N \l_stex_module_subpath_str
1185  \cs_new_protected:Nn \__stex_modules_compute_namespace:nN {
1186    \seq_set_eq:NN \l_tmpa_seq #2
1187    % split off file extension
1188    \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str % <- filename
1189    \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1190    \seq_get_left:NN \l_tmpb_seq \l_tmpb_str % <- filename without suffixes
1191    \seq_put_right:No \l_tmpa_seq \l_tmpb_str % <- file path including name without suffixes
1192
1193    \bool_set_true:N \l_tmpa_bool
1194    \bool_while_do:Nn \l_tmpa_bool {
1195      \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1196      \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1197        {source} { \bool_set_false:N \l_tmpa_bool }
1198      }{}{
1199        \seq_if_empty:NT \l_tmpa_seq {
1200          \bool_set_false:N \l_tmpa_bool
1201        }
1202      }
1203    }
1204
1205    \stex_path_to_string:NN \l_tmpa_seq \l_stex_module_subpath_str
1206    % \l_tmpa_seq <- sub-path relative to archive
1207    \str_if_empty:NTF \l_stex_module_subpath_str {
1208      \str_set:Nx \l_stex_module_ns_str {#1}
1209    }{
1210      \str_set:Nx \l_stex_module_ns_str {
1211        #1/\l_stex_module_subpath_str
1212      }
1213    }
1214  }
1215
1216  \cs_new_protected:Nn \stex_modules_current_namespace: {
1217    \str_clear:N \l_stex_module_subpath_str
1218    \prop_if_exist:NTF \l_stex_current_repository_prop {
1219      \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
```

```
1220        \__stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1221    }{
1222        % split off file extension
1223        \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1224        \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1225        \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1226        \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1227        \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1228        \str_set:Nx \l_stex_module_ns_str {
1229            file:/\stex_path_to_string:N \l_tmpa_seq
1230        }
1231    }
1232 }
```

(*End definition for* `\stex_modules_current_namespace:`. *This function is documented on page 55.*)

27.1   The `smodule` environment

`smodule` arguments:

```
1233 \keys_define:nn { stex / module } {
1234    title          .tl_set:N     = \smoduletitle ,
1235    type           .str_set_x:N  = \smoduletype ,
1236    id             .str_set_x:N  = \smoduleid ,
1237    deprecate      .str_set_x:N  = \l_stex_module_deprecate_str ,
1238    ns             .str_set_x:N  = \l_stex_module_ns_str ,
1239    lang           .str_set_x:N  = \l_stex_module_lang_str ,
1240    sig            .str_set_x:N  = \l_stex_module_sig_str ,
1241    creators       .str_set_x:N  = \l_stex_module_creators_str ,
1242    contributors   .str_set_x:N  = \l_stex_module_contributors_str ,
1243    meta           .str_set_x:N  = \l_stex_module_meta_str ,
1244    srccite        .str_set_x:N  = \l_stex_module_srccite_str
1245 }
1246
1247 \cs_new_protected:Nn \__stex_modules_args:n {
1248    \str_clear:N \smoduletitle
1249    \str_clear:N \smoduletype
1250    \str_clear:N \smoduleid
1251    \str_clear:N \l_stex_module_ns_str
1252    \str_clear:N \l_stex_module_deprecate_str
1253    \str_clear:N \l_stex_module_lang_str
1254    \str_clear:N \l_stex_module_sig_str
1255    \str_clear:N \l_stex_module_creators_str
1256    \str_clear:N \l_stex_module_contributors_str
1257    \str_clear:N \l_stex_module_meta_str
1258    \str_clear:N \l_stex_module_srccite_str
1259    \keys_set:nn { stex / module } { #1 }
1260 }
1261
1262 % module parameters here? In the body?
1263
```

`\stex_module_setup:nn`   Sets up a new module property list:

```
1264 \cs_new_protected:Nn \stex_module_setup:nn {
```

126

```
1265    \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1266    \str_set:Nx \l_stex_module_name_str { #2 }
1267    \__stex_modules_args:n { #1 }
```

First, we set up the name and namespace of the module.
Are we in a nested module?

```
1268    \stex_if_in_module:TF {
1269      % Nested module
1270      \prop_get:cnN {c_stex_module_\l_stex_current_module_str _prop}
1271        { ns } \l_stex_module_ns_str
1272      \str_set:Nx \l_stex_module_name_str {
1273        \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
1274          { name } / \l_stex_module_name_str
1275      }
1276    }{
1277      % not nested:
1278      \str_if_empty:NT \l_stex_module_ns_str {
1279        \stex_modules_current_namespace:
1280        \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1281          / {\l_stex_module_ns_str}
1282        \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1283        \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1284          \str_set:Nx \l_stex_module_ns_str {
1285            \stex_path_to_string:N \l_tmpa_seq
1286          }
1287        }
1288      }
1289    }
```

Next, we determine the language of the module:

```
1290    \str_if_empty:NT \l_stex_module_lang_str {
1291      \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1292      \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1293      \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1294      \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
1295        \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
1296          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
1297        }
1298      }
1299      \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1300      \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
1301        \seq_pop_right:NN \l_tmpa_seq \l_stex_module_lang_str
1302        \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1303          inferred~from~file~name}
1304      }
1305    }
1306
1307    \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
1308      \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1309        \l_tmpa_str {
1310          \ltx@ifpackageloaded{babel}{
1311            \exp_args:Nx \selectlanguage { \l_tmpa_str }
1312          }{}
1313        } {
```

```
1314        \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
1315      }
1316    }}
```

We check if we need to extend a signature module, and set `\l_stex_current_-module_prop` accordingly:

```
1317    \str_if_empty:NTF \l_stex_module_sig_str {
1318      \exp_args:Nnx \prop_gset_from_keyval:cn {
1319        c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1320      } {
1321        name      = \l_stex_module_name_str ,
1322        ns        = \l_stex_module_ns_str ,
1323        file      = \exp_not:o { \g_stex_currentfile_seq } ,
1324        lang      = \l_stex_module_lang_str ,
1325        sig       = \l_stex_module_sig_str ,
1326        deprecate = \l_stex_module_deprecate_str ,
1327        meta      = \l_stex_module_meta_str
1328      }
1329      \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1330      \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1331      \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _copymodules}
1332      \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1333      \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
```

We load the metatheory:

```
1334      \str_if_empty:NT \l_stex_module_meta_str {
1335        \str_set:Nx \l_stex_module_meta_str {
1336          \c_stex_metatheory_ns_str ? Metatheory
1337        }
1338      }
1339      \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1340        \bool_set_true:N \l_stex_in_meta_bool
1341        \exp_args:Nx \stex_add_to_current_module:n {
1342          \bool_set_true:N \l_stex_in_meta_bool
1343          \stex_activate_module:n {\l_stex_module_meta_str}
1344          \bool_set_false:N \l_stex_in_meta_bool
1345        }
1346        \stex_activate_module:n {\l_stex_module_meta_str}
1347        \bool_set_false:N \l_stex_in_meta_bool
1348      }
1349    }{
1350      \str_if_empty:NT \l_stex_module_lang_str {
1351        \msg_error:nnxx{stex}{error/siglanguage}{
1352          \l_stex_module_ns_str?\l_stex_module_name_str
1353        }{\l_stex_module_sig_str}
1354      }
1355      \stex_debug:nn{modules}{Signature~\l_stex_module_sig_str~for~\l_stex_module_ns_str?\l_st
1356      \stex_if_module_exists:nTF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1357        \stex_debug:nn{modules}{(already exists)}
1358      }{
1359        \stex_debug:nn{modules}{(needs loading)}
1360        \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1361        \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1362        \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
```

```
1363         \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1364         \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1365         \str_set:Nx \l_tmpa_str {
1366           \stex_path_to_string:N \l_tmpa_seq /
1367           \l_tmpa_str . \l_stex_module_sig_str .tex
1368         }
1369         \IfFileExists \l_tmpa_str {
1370           \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1371             \str_clear:N \l_stex_current_module_str
1372             \seq_clear:N \l_stex_all_modules_seq
1373             \stex_debug:nn{modules}{Loading~signature}
1374           }
1375         }{
1376           \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1377         }
1378       }
1379       \stex_if_smsmode:F {
1380         \stex_activate_module:n {
1381           \l_stex_module_ns_str ? \l_stex_module_name_str
1382         }
1383       }
1384       \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1385     }
1386     \str_if_empty:NF \l_stex_module_deprecate_str {
1387       \msg_warning:nnxx{stex}{warning/deprecated}{
1388         Module~\l_stex_current_module_str
1389       }{
1390         \l_stex_module_deprecate_str
1391       }
1392     }
1393     \seq_put_right:Nx \l_stex_all_modules_seq {
1394       \l_stex_module_ns_str ? \l_stex_module_name_str
1395     }
1396     \tl_clear:c{l__stex_modules_aftergroup_\l_stex_module_ns_str ? \l_stex_module_name_str _tl
1397 }
```

(*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}`

```
1398 \cs_new_protected:Nn \__stex_modules_begin_module: {
1399   \stex_reactivate_macro:N \STEXexport
1400   \stex_reactivate_macro:N \importmodule
1401   \stex_reactivate_macro:N \symdecl
1402   \stex_reactivate_macro:N \notation
1403   \stex_reactivate_macro:N \symdef
1404
1405   \stex_debug:nn{modules}{
1406     New~module:\\
1407     Namespace:~\l_stex_module_ns_str\\
1408     Name:~\l_stex_module_name_str\\
1409     Language:~\l_stex_module_lang_str\\
1410     Signature:~\l_stex_module_sig_str\\
```

129

```
1411        Metatheory:~\l_stex_module_meta_str\\
1412        File:~\stex_path_to_string:N \g_stex_currentfile_seq
1413    }
1414
1415    \stex_if_do_html:T{
1416        \begin{stex_annotate_env} {theory} {
1417            \l_stex_module_ns_str ? \l_stex_module_name_str
1418        }
1419
1420        \stex_annotate_invisible:nnn{header}{} {
1421            \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1422            \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1423            \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1424                \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1425            }
1426            \str_if_empty:NF \smoduletype {
1427                \stex_annotate:nnn{type}{\smoduletype}{}
1428            }
1429        }
1430    }
1431    % TODO: Inherit metatheory for nested modules?
1432 }
1433 \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}

```
1434 \cs_new_protected:Nn \__stex_modules_end_module: {
1435    \stex_debug:nn{modules}{Closing~module~\prop_item:cn {c_stex_module_\l_stex_current_module
1436    \_stex_reset_up_to_module:n \l_stex_current_module_str
1437    \stex_if_smsmode:T {
1438        \stex_persist:x {
1439            \prop_set_from_keyval:cn{c_stex_module_\l_stex_current_module_str _prop}{
1440                \exp_after:wN \prop_to_keyval:N \csname c_stex_module_\l_stex_current_module_str _pr
1441            }
1442            \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _constants}{
1443                \seq_use:cn{c_stex_module_\l_stex_current_module_str _constants},
1444            }
1445            \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _imports}{
1446                \seq_use:cn{c_stex_module_\l_stex_current_module_str _imports},
1447            }
1448            \tl_set:cn {c_stex_module_\l_stex_current_module_str _code}
1449        }
1450        \exp_after:wN \let \exp_after:wN \l_tmpa_tl \csname c_stex_module_\l_stex_current_module
1451        \exp_after:wN \stex_persist:n \exp_after:wN { \exp_after:wN { \l_tmpa_tl } }
1452    }
1453 }
```

(*End definition for* __stex_modules_end_module:.)

The core environment

```
1454 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1455 \NewDocumentEnvironment { smodule } { O{} m } {
1456    \stex_module_setup:nn{#1}{#2}
1457    \par
```

130

```
1458    \stex_if_smsmode:F{
1459      \tl_clear:N \l_tmpa_tl
1460      \clist_map_inline:Nn \smoduletype {
1461        \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1462          \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1463        }
1464      }
1465      \tl_if_empty:NTF \l_tmpa_tl {
1466        \__stex_modules_smodule_start:
1467      }{
1468        \l_tmpa_tl
1469      }
1470    }
1471    \__stex_modules_begin_module:
1472    \str_if_empty:NF \smoduleid {
1473      \stex_ref_new_doc_target:n \smoduleid
1474    }
1475    \stex_smsmode_do:
1476  } {
1477    \__stex_modules_end_module:
1478    \stex_if_smsmode:F {
1479      \end{stex_annotate_env}
1480      \clist_set:No \l_tmpa_clist \smoduletype
1481      \tl_clear:N \l_tmpa_tl
1482      \clist_map_inline:Nn \l_tmpa_clist {
1483        \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1484          \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1485        }
1486      }
1487      \tl_if_empty:NTF \l_tmpa_tl {
1488        \__stex_modules_smodule_end:
1489      }{
1490        \l_tmpa_tl
1491      }
1492    }
1493  }
```

\stexpatchmodule

```
1494  \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1495  \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1496
1497  \newcommand\stexpatchmodule[3][] {
1498      \str_set:Nx \l_tmpa_str{ #1 }
1499      \str_if_empty:NTF \l_tmpa_str {
1500        \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1501        \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1502      }{
1503        \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1504        \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1505      }
1506  }
```

(*End definition for* \stexpatchmodule. *This function is documented on page* *55.*)

27.2 Invoking modules

```
1507 \NewDocumentCommand \STEXModule { m } {
1508   \exp_args:NNx \str_set:Nn \l_tmpa_str { # 1 }
1509   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1510   \tl_set:Nn \l_tmpa_tl {
1511     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1512   }
1513   \seq_map_inline:Nn \l_stex_all_modules_seq {
1514     \str_set:Nn \l_tmpb_str { ##1 }
1515     \str_if_eq:eeT { \l_tmpa_str } {
1516       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1517     } {
1518       \seq_map_break:n {
1519         \tl_set:Nn \l_tmpa_tl {
1520           \stex_invoke_module:n { ##1 }
1521         }
1522       }
1523     }
1524   }
1525   \l_tmpa_tl
1526 }
1527
1528 \cs_new_protected:Nn \stex_invoke_module:n {
1529   \stex_debug:nn{modules}{Invoking~module~#1}
1530   \peek_charcode_remove:NTF ! {
1531     \__stex_modules_invoke_uri:nN { #1 }
1532   } {
1533     \peek_charcode_remove:NTF ? {
1534       \__stex_modules_invoke_symbol:nn { #1 }
1535     } {
1536       \msg_error:nnx{stex}{error/syntax}{
1537         ?~or~!~expected~after~
1538         \c_backslash_str STEXModule{#1}
1539       }
1540     }
1541   }
1542 }
1543
1544 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1545   \str_set:Nn #2 { #1 }
1546 }
1547
1548 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1549   \stex_invoke_symbol:n{#1?#2}
1550 }
```

(*End definition for* \STEXModule *and* \stex_invoke_module:n. *These functions are documented on page 55.*)

```
1551 \bool_new:N \l_stex_in_meta_bool
1552 \bool_set_false:N \l_stex_in_meta_bool
```

```
1553 \cs_new_protected:Nn \stex_activate_module:n {
1554   \stex_debug:nn{modules}{Activating~module~#1}
1555   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1556     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1557     \use:c{ c_stex_module_#1_code }
1558   }
1559 }
```

(*End definition for* `\stex_activate_module:n`*. This function is documented on page* *56**.*)

```
1560 ⟨/package⟩
```

Chapter 28

SТEX
-Module Inheritance
Implementation

```
1561 ⟨∗package⟩
1562
1563 %%%%%%%%%%%%  inheritance.dtx  %%%%%%%%%%%%
1564
```

28.1   SMS Mode

```
1565 ⟨@@=stex_smsmode⟩
```

```
1566 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1567 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1568 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1569
1570 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1571   \makeatletter
1572   \makeatother
1573   \ExplSyntaxOn
1574   \ExplSyntaxOff
1575   \rustexBREAK
1576 }
1577
1578 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1579   \symdef
1580   \importmodule
1581   \notation
1582   \symdecl
1583   \STEXexport
1584   \inlineass
1585   \inlinedef
1586   \inlineex
1587   \endinput
1588   \setnotation
```

```
1589     \copynotation
1590     \assign
1591     \renamedecl
1592     \donotcopy
1593     \instantiate
1594 }
1595
1596 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1597     \tl_to_str:n {
1598         smodule,
1599         copymodule,
1600         interpretmodule,
1601         sdefinition,
1602         sexample,
1603         sassertion,
1604         sparagraph,
1605         mathstructure
1606     }
1607 }
```

(*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*

```
1608 \bool_new:N \g__stex_smsmode_bool
1609 \bool_set_false:N \g__stex_smsmode_bool
1610 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1611     \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1612 }
```

(*End definition for* `\stex_if_smsmode:TF`*. This function is documented on page 57.*)

`\__stex_smsmode_in_smsmode:nn`

```
1613 \cs_new_protected:Nn \__stex_smsmode_in_smsmode:nn { \stex_suppress_html:n {
1614     \vbox_set:Nn \l_tmpa_box {
1615         \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1616         \bool_gset_true:N \g__stex_smsmode_bool
1617         #2
1618         \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1619     }
1620     \box_clear:N \l_tmpa_box
1621 } }
```

(*End definition for* `\__stex_smsmode_in_smsmode:nn`*.*)

`\stex_file_in_smsmode:nn`

```
1622 \quark_new:N \q__stex_smsmode_break
1623
1624 \NewDocumentCommand \__stex_smsmode_importmodule: { O{} m } {
1625     \seq_gput_right:Nn \l__stex_smsmode_importmodules_seq {{#1}{#2}}
1626     \stex_smsmode_do:
1627 }
1628
1629 \cs_new_protected:Nn \__stex_smsmode_module:nn {
1630     \__stex_modules_args:n{#1}
```

135

```
1631    \stex_if_in_module:F {
1632      \str_if_empty:NF \l_stex_module_sig_str {
1633        \stex_modules_current_namespace:
1634        \str_set:Nx \l_stex_module_name_str { #2 }
1635        \stex_if_module_exists:nF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1636          \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1637          \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1638          \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1639          \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1640          \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1641          \str_set:Nx \l_tmpa_str {
1642            \stex_path_to_string:N \l_tmpa_seq /
1643            \l_tmpa_str . \l_stex_module_sig_str .tex
1644          }
1645          \IfFileExists \l_tmpa_str {
1646            \exp_args:NNx \seq_gput_right:Nn \l__stex_smsmode_sigmodules_seq \l_tmpa_str
1647          }{
1648            \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1649          }
1650        }
1651      }
1652    }
1653  }
1654
1655  \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1656    \stex_filestack_push:n{#1}
1657    \seq_gclear:N \l__stex_smsmode_importmodules_seq
1658    \seq_gclear:N \l__stex_smsmode_sigmodules_seq
1659    % ----- new ---------------------------
1660    \__stex_smsmode_in_smsmode:nn{#1}{
1661      \let\importmodule\__stex_smsmode_importmodule:
1662      \let\stex_module_setup:nn\__stex_smsmode_module:nn
1663      \let\__stex_modules_begin_module:\relax
1664      \let\__stex_modules_end_module:\relax
1665      \seq_clear:N \g_stex_smsmode_allowedenvs_seq
1666      \exp_args:NNx \seq_put_right:Nn \g_stex_smsmode_allowedenvs_seq {\tl_to_str:n{smodule}}
1667      \tl_clear:N \g_stex_smsmode_allowedmacros_tl
1668      \tl_clear:N \g_stex_smsmode_allowedmacros_escape_tl
1669      \tl_put_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\importmodule}
1670      \everyeof{\q__stex_smsmode_break\noexpand}
1671      \expandafter\expandafter\expandafter
1672      \stex_smsmode_do:
1673      \csname @ @ input\endcsname "#1"\relax
1674
1675      \seq_map_inline:Nn \l__stex_smsmode_sigmodules_seq {
1676        \stex_filestack_push:n{##1}
1677        \expandafter\expandafter\expandafter
1678        \stex_smsmode_do:
1679        \csname @ @ input\endcsname "##1"\relax
1680        \stex_filestack_pop:
1681      }
1682    }
1683    % ----- new ---------------------------
1684    \__stex_smsmode_in_smsmode:nn{#1} {
```

```
1685        #2
1686        % ----- new ----------------------------
1687        \begingroup
1688        %\stex_debug:nn{smsmode}{Here:~\seq_use:Nn\l__stex_smsmode_importmodules_seq, }
1689        \seq_map_inline:Nn \l__stex_smsmode_importmodules_seq {
1690          \stex_import_module_uri:nn ##1
1691          \stex_import_require_module:nnnn
1692            \l_stex_import_ns_str
1693            \l_stex_import_archive_str
1694            \l_stex_import_path_str
1695            \l_stex_import_name_str
1696        }
1697        \endgroup
1698        \stex_debug:nn{smsmode}{Actually~loading~file~#1}
1699        % ----- new ----------------------------
1700        \everyeof{\q__stex_smsmode_break\noexpand}
1701        \expandafter\expandafter\expandafter
1702        \stex_smsmode_do:
1703        \csname @ @ input\endcsname "#1"\relax
1704      }
1705      \stex_filestack_pop:
1706 }
```

(*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:

```
1707 \cs_new_protected:Npn \stex_smsmode_do: {
1708   \stex_if_smsmode:T {
1709     \__stex_smsmode_do:w
1710   }
1711 }
1712 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1713   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 }}{
1714     \expandafter\if\expandafter\relax\noexpand#1
1715       \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1716     \else\expandafter\__stex_smsmode_do:w\fi
1717   }{
1718     \__stex_smsmode_do:w %#1
1719   }
1720 }
1721 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1722   \cs_if_eq:NNF #1 \q__stex_smsmode_break {
1723     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1724       #1\__stex_smsmode_do:w
1725     }{
1726       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1727         #1
1728       }{
1729         \cs_if_eq:NNTF \begin #1 {
1730           \__stex_smsmode_check_begin:n
1731         }{
1732           \cs_if_eq:NNTF \end #1 {
1733             \__stex_smsmode_check_end:n
```

137

```
1734              }{
1735                \__stex_smsmode_do:w
1736              }
1737            }
1738          }
1739        }
1740      }
1741    }
1742
1743    \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1744      \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1745        \begin{#1}
1746      }{
1747        \__stex_smsmode_do:w
1748      }
1749    }
1750    \cs_new_protected:Nn \__stex_smsmode_check_end:n {
1751      \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1752        \end{#1}\__stex_smsmode_do:w
1753      }{
1754        \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1755      }
1756    }
```

(*End definition for* `\stex_smsmode_do:`. *This function is documented on page 58.*)

28.2   Inheritance

1757 ⟨@@=stex_importmodule⟩

\stex_import_module_uri:nn

```
1758    \cs_new_protected:Nn \stex_import_module_uri:nn {
1759      \str_set:Nx \l_stex_import_archive_str { #1 }
1760      \str_set:Nn \l_stex_import_path_str { #2 }
1761
1762      \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1763      \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1764      \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1765
1766      \stex_modules_current_namespace:
1767      \bool_lazy_all:nTF {
1768        {\str_if_empty_p:N \l_stex_import_archive_str}
1769        {\str_if_empty_p:N \l_stex_import_path_str}
1770        {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1771      }{
1772        \str_set_eq:NN \l_stex_import_path_str \l_stex_module_subpath_str
1773        \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1774      }{
1775        \str_if_empty:NT \l_stex_import_archive_str {
1776          \prop_if_exist:NT \l_stex_current_repository_prop {
1777            \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1778          }
1779        }
1780        \str_if_empty:NTF \l_stex_import_archive_str {
```

138

```
1781         \str_if_empty:NF \l_stex_import_path_str {
1782           \str_set:Nx \l_stex_import_ns_str {
1783             \l_stex_module_ns_str / \l_stex_import_path_str
1784           }
1785         }
1786     }{
1787       \stex_require_repository:n \l_stex_import_archive_str
1788       \prop_get:cnN { c_stex_mathhub_\l_stex_import_archive_str _manifest_prop } { ns }
1789         \l_stex_import_ns_str
1790       \str_if_empty:NF \l_stex_import_path_str {
1791         \str_set:Nx \l_stex_import_ns_str {
1792           \l_stex_import_ns_str / \l_stex_import_path_str
1793         }
1794       }
1795     }
1796   }
1797 }
```

(*End definition for* `\stex_import_module_uri:nn`. *This function is documented on page 59.*)

<div style="color:red">\l_stex_import_name_str
\l_stex_import_archive_str
\l_stex_import_path_str
\l_stex_import_ns_str</div> Store the return values of `\stex_import_module_uri:nn`.

```
1798 \str_new:N \l_stex_import_name_str
1799 \str_new:N \l_stex_import_archive_str
1800 \str_new:N \l_stex_import_path_str
1801 \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*⟩}

```
1802 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1803   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1804
1805     %\stex_debug:nn{requiremodule}{Here:\\~~1:~#1\\~~2:~#2\\~~3:~#3\\~~4:~#4}
1806
1807     \exp_args:NNxx \seq_set_split:Nnn \l_tmpa_seq {\tl_to_str:n{/}} {#4}
1808     \seq_get_left:NN \l_tmpa_seq \l_tmpc_str
1809
1810     %\stex_debug:nn{requiremodule}{Top~module:\l_tmpc_str}
1811
1812     % archive
1813     \str_set:Nx \l_tmpa_str { #2 }
1814     \str_if_empty:NTF \l_tmpa_str {
1815       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1816     } {
1817       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1818       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1819       \seq_put_right:Nn \l_tmpa_seq { source }
1820     }
1821
1822     % path
1823     \str_set:Nx \l_tmpb_str { #3 }
1824     \str_if_empty:NTF \l_tmpb_str {
1825       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / \l_tmpc_str }
1826
```

```
1827          \ltx@ifpackageloaded{babel} {
1828            \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1829                { \languagename } \l_tmpb_str {
1830                  \msg_error:nnx{stex}{error/unknownlanguage}{\languagename}
1831                }
1832          } {
1833            \str_clear:N \l_tmpb_str
1834          }

1835
1836          %\stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1837          \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1838            \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1839          }{
1840            %\stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1841            \IfFileExists{ \l_tmpa_str.tex }{
1842              \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1843            }{
1844              % try english as default
1845              %\stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1846              \IfFileExists{ \l_tmpa_str.en.tex }{
1847                \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1848              }{
1849                \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1850              }
1851            }
1852          }

1853
1854        } {
1855          \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1856          \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq

1857
1858          \ltx@ifpackageloaded{babel} {
1859            \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1860                { \languagename } \l_tmpb_str {
1861                  \msg_error:nnx{stex}{error/unknownlanguage}{\languagename}
1862                }
1863          } {
1864            \str_clear:N \l_tmpb_str
1865          }

1866
1867          \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str

1868
1869          %\stex_debug:nn{modules}{Checking~\l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex}
1870          \IfFileExists{ \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }{
1871            \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.te
1872          }{
1873            %\stex_debug:nn{modules}{Checking~\l_tmpa_str/\l_tmpc_str.tex}
1874            \IfFileExists{ \l_tmpa_str/\l_tmpc_str.tex }{
1875              \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.tex }
1876            }{
1877              % try english as default
1878              %\stex_debug:nn{modules}{Checking~\l_tmpa_str/\l_tmpc_str.en.tex}
1879              \IfFileExists{ \l_tmpa_str/\l_tmpc_str.en.tex }{
1880                \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.en.tex }
```

```
1881          }{
1882            %\stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1883            \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1884              \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1885            }{
1886              %\stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1887              \IfFileExists{ \l_tmpa_str.tex }{
1888                \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1889              }{
1890                % try english as default
1891                %\stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1892                \IfFileExists{ \l_tmpa_str.en.tex }{
1893                  \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1894                }{
1895                  \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1896                }
1897              }
1898            }
1899          }
1900        }
1901      }
1902    }
1903
1904    \str_if_eq:eeF{\g__stex_importmodule_file_str}{\seq_use:Nn \g_stex_currentfile_seq /}{
1905      \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
1906        \seq_clear:N \l_stex_all_modules_seq
1907        \str_clear:N \l_stex_current_module_str
1908        \str_set:Nx \l_tmpb_str { #2 }
1909        \str_if_empty:NF \l_tmpb_str {
1910          \stex_set_current_repository:n { #2 }
1911        }
1912        \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1913      }
1914
1915      \stex_if_module_exists:nF { #1 ? #4 } {
1916        \msg_error:nnx{stex}{error/unknownmodule}{
1917          #1?#4~(in~file~\g__stex_importmodule_file_str)
1918        }
1919      }
1920    }
1921
1922  }
1923  \stex_activate_module:n { #1 ? #4 }
1924 }
```

(*End definition for* \stex_import_require_module:nnnn. *This function is documented on page 59.*)

```
1925 \NewDocumentCommand \importmodule { O{} m } {
1926   \stex_import_module_uri:nn { #1 } { #2 }
1927   \stex_debug:nn{modules}{Importing~module:~
1928     \l_stex_import_ns_str ? \l_stex_import_name_str
1929   }
1930   \stex_import_require_module:nnnn
```

141

```
1931    { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1932    { \l_stex_import_path_str } { \l_stex_import_name_str }
1933    \stex_if_smsmode:F {
1934      \stex_annotate_invisible:nnn
1935        {import} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1936    }
1937    \exp_args:Nx \stex_add_to_current_module:n {
1938      \stex_import_require_module:nnnn
1939      { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1940      { \l_stex_import_path_str } { \l_stex_import_name_str }
1941    }
1942    \exp_args:Nx \stex_add_import_to_current_module:n {
1943      \l_stex_import_ns_str ? \l_stex_import_name_str
1944    }
1945    \stex_smsmode_do:
1946    \ignorespacesandpars
1947 }
1948 \stex_deactivate_macro:Nn \importmodule {module~environments}
```

(*End definition for* \importmodule. *This function is documented on page* *58.*)

\usemodule

```
1949 \NewDocumentCommand \usemodule { O{} m } {
1950   \stex_if_smsmode:F {
1951     \stex_import_module_uri:nn { #1 } { #2 }
1952     \stex_import_require_module:nnnn
1953     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1954     { \l_stex_import_path_str } { \l_stex_import_name_str }
1955     \stex_annotate_invisible:nnn
1956       {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1957   }
1958   \stex_smsmode_do:
1959   \ignorespacesandpars
1960 }
```

(*End definition for* \usemodule. *This function is documented on page* *58.*)

```
1961 \cs_new_protected:Nn \stex_csl_to_imports:Nn {
1962   \tl_if_empty:nF{#2}{
1963     \clist_set:Nn \l_tmpa_clist {#2}
1964     \clist_map_inline:Nn \l_tmpa_clist {
1965       \tl_if_head_eq_charcode:nNTF {##1}[{
1966         #1 ##1
1967       }{
1968         #1{##1}
1969       }
1970     }
1971   }
1972 }
1973 \cs_generate_variant:Nn \stex_csl_to_imports:Nn {No}
1974
1975
1976 ⟨/package⟩
```

142

Chapter 29

sTeX -Symbols Implementation

```
1977 ⟨∗package⟩
1978
1979 %%%%%%%%%%%%   symbols.dtx   %%%%%%%%%%%%
1980
```

Warnings and error messages
```
1981 \msg_new:nnn{stex}{error/wrongargs}{
1982   args~value~in~symbol~declaration~for~#1~
1983   needs~to~be~i,~a,~b~or~B,~but~#2~given
1984 }
1985 \msg_new:nnn{stex}{error/unknownsymbol}{
1986   No~symbol~#1~found!
1987 }
1988 \msg_new:nnn{stex}{error/seqlength}{
1989   Expected~#1~arguments;~got~#2!
1990 }
1991 \msg_new:nnn{stex}{error/unknownnotation}{
1992   Unknown~notation~#1~for~#2!
1993 }
```

29.1   Symbol Declarations

```
1994 ⟨@@=stex_symdecl⟩
```

\stex_all_symbols:n   Map over all available symbols
```
1995 \cs_new_protected:Nn \stex_all_symbols:n {
1996   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
1997   \seq_map_inline:Nn \l_stex_all_modules_seq {
1998     \seq_map_inline:cn{c_stex_module_##1_constants}{
1999       \__stex_symdecl_all_symbols_cs{##1?####1}
2000     }
2001   }
2002 }
```

(*End definition for* \stex_all_symbols:n. *This function is documented on page 61.*)

```
2003 \NewDocumentCommand \STEXsymbol { m } {
2004   \stex_get_symbol:n { #1 }
2005   \exp_args:No
2006   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
2007 }
```

(*End definition for* \STEXsymbol. *This function is documented on page 62.*)

symdecl arguments:

```
2008 \keys_define:nn { stex / symdecl } {
2009   name        .str_set_x:N  = \l_stex_symdecl_name_str ,
2010   local       .bool_set:N   = \l_stex_symdecl_local_bool ,
2011   args        .str_set_x:N  = \l_stex_symdecl_args_str ,
2012   type        .tl_set:N     = \l_stex_symdecl_type_tl ,
2013   deprecate   .str_set_x:N  = \l_stex_symdecl_deprecate_str ,
2014   align       .str_set:N    = \l_stex_symdecl_align_str , % TODO(?)
2015   gfc         .str_set:N    = \l_stex_symdecl_gfc_str , % TODO(?)
2016   specializes .str_set:N    = \l_stex_symdecl_specializes_str , % TODO(?)
2017   def         .tl_set:N     = \l_stex_symdecl_definiens_tl ,
2018   assoc       .choices:nn   =
2019     {bin,binl,binr,pre,conj,pwconj}
2020     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}
2021 }
2022
2023 \bool_new:N \l_stex_symdecl_make_macro_bool
2024
2025 \cs_new_protected:Nn \__stex_symdecl_args:n {
2026   \str_clear:N \l_stex_symdecl_name_str
2027   \str_clear:N \l_stex_symdecl_args_str
2028   \str_clear:N \l_stex_symdecl_deprecate_str
2029   \str_clear:N \l_stex_symdecl_assoctype_str
2030   \bool_set_false:N \l_stex_symdecl_local_bool
2031   \tl_clear:N \l_stex_symdecl_type_tl
2032   \tl_clear:N \l_stex_symdecl_definiens_tl
2033
2034   \keys_set:nn { stex / symdecl } { #1 }
2035 }
```

\symdecl Parses the optional arguments and passes them on to \stex_symdecl_do: (so that \symdef can do the same)

```
2036
2037 \NewDocumentCommand \symdecl { s m O{} } {
2038   \__stex_symdecl_args:n { #3 }
2039   \IfBooleanTF #1 {
2040     \bool_set_false:N \l_stex_symdecl_make_macro_bool
2041   } {
2042     \bool_set_true:N \l_stex_symdecl_make_macro_bool
2043   }
2044   \stex_symdecl_do:n { #2 }
2045   \stex_smsmode_do:
2046 }
2047
2048 \cs_new_protected:Nn \stex_symdecl_do:nn {
```

144

```
2049      \__stex_symdecl_args:n{#1}
2050      \bool_set_false:N \l_stex_symdecl_make_macro_bool
2051      \stex_symdecl_do:n{#2}
2052    }
2053
2054    \stex_deactivate_macro:Nn \symdecl {module~environments}
```

(*End definition for* \symdecl. *This function is documented on page* 60.)

\stex_symdecl_do:n

```
2055    \cs_new_protected:Nn \stex_symdecl_do:n {
2056      \stex_if_in_module:F {
2057        % TODO throw error? some default namespace?
2058      }
2059
2060      \str_if_empty:NT \l_stex_symdecl_name_str {
2061        \str_set:Nx \l_stex_symdecl_name_str { #1 }
2062      }
2063
2064      \prop_if_exist:cT { l_stex_symdecl_
2065          \l_stex_current_module_str ?
2066          \l_stex_symdecl_name_str
2067        _prop
2068      }{
2069        % TODO throw error (beware of circular dependencies)
2070      }
2071
2072      \prop_clear:N \l_tmpa_prop
2073      \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
2074      \seq_clear:N \l_tmpa_seq
2075      \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
2076      \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
2077
2078      \str_if_empty:NT \l_stex_symdecl_deprecate_str {
2079        \str_if_empty:NF \l_stex_module_deprecate_str {
2080          \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
2081        }
2082      }
2083      \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
2084
2085      \exp_args:No \stex_add_constant_to_current_module:n {
2086        \l_stex_symdecl_name_str
2087      }
2088
2089      % arity/args
2090      \int_zero:N \l_tmpb_int
2091
2092      \bool_set_true:N \l_tmpa_bool
2093      \str_map_inline:Nn \l_stex_symdecl_args_str {
2094        \token_case_meaning:NnF ##1 {
2095          0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2096          {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2097          {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2098          {\tl_to_str:n a} {
```

145

```
2099        \bool_set_false:N \l_tmpa_bool
2100        \int_incr:N \l_tmpb_int
2101      }
2102      {\tl_to_str:n B} {
2103        \bool_set_false:N \l_tmpa_bool
2104        \int_incr:N \l_tmpb_int
2105      }
2106    }{
2107      \msg_error:nnxx{stex}{error/wrongargs}{
2108        \l_stex_current_module_str ?
2109        \l_stex_symdecl_name_str
2110      }{##1}
2111    }
2112  }
2113  \bool_if:NTF \l_tmpa_bool {
2114    % possibly numeric
2115    \str_if_empty:NTF \l_stex_symdecl_args_str {
2116      \prop_put:Nnn \l_tmpa_prop { args } {}
2117      \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2118    }{
2119      \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
2120      \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2121      \str_clear:N \l_tmpa_str
2122      \int_step_inline:nn \l_tmpa_int {
2123        \str_put_right:Nn \l_tmpa_str i
2124      }
2125      \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
2126    }
2127  } {
2128    \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
2129    \prop_put:Nnx \l_tmpa_prop { arity }
2130      { \str_count:N \l_stex_symdecl_args_str }
2131  }
2132  \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2133
2134  \tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
2135    \prop_put:Nnx \l_tmpa_prop { defined }{ false }
2136  }{
2137    \prop_put:Nnx \l_tmpa_prop { defined }{ true }
2138  }
2139
2140  % semantic macro
2141
2142  \bool_if:NT \l_stex_symdecl_make_macro_bool {
2143    \exp_args:Nx \stex_do_up_to_module:n {
2144      \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2145        \l_stex_current_module_str ? \l_stex_symdecl_name_str
2146      }}
2147    }
2148  }
2149
2150  \stex_debug:nn{symbols}{New~symbol:~
2151    \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2152    Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
```

146

```
2153    Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
2154    Definiens:~\exp_not:o {\l_stex_symdecl_definiens_tl}
2155  }
2156
2157  % circular dependencies require this:
2158  \stex_if_do_html:T {
2159    \stex_annotate_invisible:nnn {symdecl} {
2160      \l_stex_current_module_str ? \l_stex_symdecl_name_str
2161    } {
2162      \tl_if_empty:NF \l_stex_symdecl_type_tl {
2163        \stex_annotate_invisible:nnn{type}{}{$\l_stex_symdecl_type_tl$}
2164      }
2165      \stex_annotate_invisible:nnn{args}{}{
2166        \prop_item:Nn \l_tmpa_prop { args }
2167      }
2168      \stex_annotate_invisible:nnn{macroname}{#1}{}
2169      \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2170        \stex_annotate_invisible:nnn{definiens}{}
2171          {$\l_stex_symdecl_definiens_tl$}
2172      }
2173      \str_if_empty:NF \l_stex_symdecl_assoctype_str {
2174        \stex_annotate_invisible:nnn{assoctype}{\l_stex_symdecl_assoctype_str}{}
2175      }
2176    }
2177  }
2178  \prop_if_exist:cF {
2179    l_stex_symdecl_
2180    \l_stex_current_module_str ? \l_stex_symdecl_name_str
2181    _prop
2182  } {
2183    \bool_if:NTF \l_stex_symdecl_local_bool \stex_do_up_to_module:x \stex_execute_in_module:
2184      \__stex_symdecl_restore_symbol:nnnnnnn
2185        {\l_stex_symdecl_name_str}
2186        { \prop_item:Nn \l_tmpa_prop {args} }
2187        { \prop_item:Nn \l_tmpa_prop {arity} }
2188        { \prop_item:Nn \l_tmpa_prop {assocs} }
2189        { \prop_item:Nn \l_tmpa_prop {defined} }
2190        {\bool_if:NT \l_stex_symdecl_make_macro_bool {#1} }
2191        {\l_stex_current_module_str}
2192    }
2193  }
2194 }
2195 \cs_new_protected:Nn \__stex_symdecl_restore_symbol:nnnnnnn {
2196   \prop_clear:N \l_tmpa_prop
2197   \prop_put:Nnn \l_tmpa_prop { module } { #7 }
2198   \prop_put:Nnn \l_tmpa_prop { name } { #1}
2199   \prop_put:Nnn \l_tmpa_prop { args } {#2}
2200   \prop_put:Nnn \l_tmpa_prop { arity } { #3 }
2201   \prop_put:Nnn \l_tmpa_prop { assocs } { #4 }
2202   \prop_put:Nnn \l_tmpa_prop { defined } { #5 }
2203   \tl_if_empty:nF{#6}{
2204     \tl_set:cx{#6}{\stex_invoke_symbol:n{\detokenize{#7 ? #1}}}
2205   }
2206   \prop_set_eq:cN{l_stex_symdecl_ \detokenize{#7 ? #1} _prop}\l_tmpa_prop
```

```
2207     \seq_clear:c{l_stex_symdecl_ \detokenize{#7 ? #1} _notations}
2208 }
```

(End definition for `\stex_symdecl_do:n`*. This function is documented on page 61.)*

`\stex_get_symbol:n`

```
2209 \str_new:N \l_stex_get_symbol_uri_str
2210
2211 \cs_new_protected:Nn \stex_get_symbol:n {
2212   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2213     \tl_set:Nn \l_tmpa_tl { #1 }
2214     \__stex_symdecl_get_symbol_from_cs:
2215   }{
2216     % argument is a string
2217     % is it a command name?
2218     \cs_if_exist:cTF { #1 }{
2219       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2220       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2221       \str_if_empty:NTF \l_tmpa_str {
2222         \exp_args:Nx \cs_if_eq:NNTF {
2223           \tl_head:N \l_tmpa_tl
2224         } \stex_invoke_symbol:n {
2225           \__stex_symdecl_get_symbol_from_cs:
2226         }{
2227           \__stex_symdecl_get_symbol_from_string:n { #1 }
2228         }
2229       } {
2230         \__stex_symdecl_get_symbol_from_string:n { #1 }
2231       }
2232     }{
2233       % argument is not a command name
2234       \__stex_symdecl_get_symbol_from_string:n { #1 }
2235       % \l_stex_all_symbols_seq
2236     }
2237   }
2238   \str_if_eq:eeF {
2239     \prop_item:cn {
2240       l_stex_symdecl_\l_stex_get_symbol_uri_str _prop
2241     }{ deprecate }
2242   }{}{
2243     \msg_warning:nnxx{stex}{warning/deprecated}{
2244       Symbol~\l_stex_get_symbol_uri_str
2245     }{
2246       \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{ deprecate }
2247     }
2248   }
2249 }
2250
2251 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2252   \tl_set:Nn \l_tmpa_tl {
2253     \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2254   }
2255   \str_set:Nn \l_tmpa_str { #1 }
2256   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
```

```
2257
2258    \stex_all_symbols:n {
2259      \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2260        \seq_map_break:n{\seq_map_break:n{
2261          \tl_set:Nn \l_tmpa_tl {
2262            \str_set:Nn \l_stex_get_symbol_uri_str { ##1 }
2263          }
2264        }}
2265      }
2266    }
2267
2268    \l_tmpa_tl
2269 }
2270
2271 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
2272    \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2273      { \tl_tail:N \l_tmpa_tl }
2274    \tl_if_single:NTF \l_tmpa_tl {
2275      \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2276        \exp_after:wN \str_set:Nn \exp_after:wN
2277          \l_stex_get_symbol_uri_str \l_tmpa_tl
2278      }{
2279        % TODO
2280        % tail is not a single group
2281      }
2282    }{
2283      % TODO
2284      % tail is not a single group
2285    }
2286 }
```

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

29.2   Notations

notation arguments:
```
2288 \keys_define:nn { stex / notation } {
2289 %  lang    .tl_set_x:N  = \l__stex_notation_lang_str ,
2290    variant .tl_set_x:N  = \l__stex_notation_variant_str ,
2291    prec    .str_set_x:N = \l__stex_notation_prec_str ,
2292    op      .tl_set:N    = \l__stex_notation_op_tl ,
2293    primary .bool_set:N  = \l__stex_notation_primary_bool ,
2294    primary .default:n   = {true} ,
2295    unknown .code:n      = \str_set:Nx
2296        \l__stex_notation_variant_str \l_keys_key_str
2297 }
2298
2299 \cs_new_protected:Nn \_stex_notation_args:n {
2300 %  \str_clear:N \l__stex_notation_lang_str
2301    \str_clear:N \l__stex_notation_variant_str
2302    \str_clear:N \l__stex_notation_prec_str
2303    \tl_clear:N \l__stex_notation_op_tl
```

```
2304    \bool_set_false:N \l__stex_notation_primary_bool
2305
2306    \keys_set:nn { stex / notation } { #1 }
2307  }
```

\notation

```
2308  \NewDocumentCommand \notation { s m O{}} {
2309    \__stex_notation_args:n { #3 }
2310    \tl_clear:N \l_stex_symdecl_definiens_tl
2311    \stex_get_symbol:n { #2 }
2312    \tl_set:Nn \l_stex_notation_after_do_tl {
2313      \__stex_notation_final:
2314      \IfBooleanTF#1{
2315        \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2316      }{}
2317      \stex_smsmode_do:\ignorespacesandpars
2318    }
2319    \stex_notation_do:nnnnn
2320      { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } } }
2321      { \prop_item:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } } }
2322      { \l__stex_notation_variant_str }
2323      { \l__stex_notation_prec_str}
2324  }
2325  \stex_deactivate_macro:Nn \notation {module~environments}
```

(*End definition for* \notation. *This function is documented on page 61.*)

\stex_notation_do:nnnnn

```
2326  \seq_new:N \l__stex_notation_precedences_seq
2327  \tl_new:N \l__stex_notation_opprec_tl
2328  \int_new:N \l__stex_notation_currarg_int
2329  \tl_new:N \stex_symbol_after_invokation_tl
2330
2331  \cs_new_protected:Nn \stex_notation_do:nnnnn {
2332    \let\l_stex_current_symbol_str\relax
2333    \seq_clear:N \l__stex_notation_precedences_seq
2334    \tl_clear:N \l__stex_notation_opprec_tl
2335    \str_set:Nx \l__stex_notation_args_str { #1 }
2336    \str_set:Nx \l__stex_notation_arity_str { #2 }
2337    \str_set:Nx \l__stex_notation_suffix_str { #3 }
2338    \str_set:Nx \l__stex_notation_prec_str { #4 }
2339
2340    % precedences
2341    \str_if_empty:NTF \l__stex_notation_prec_str {
2342      \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2343        \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2344      }{
2345        \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2346      }
2347    } {
2348      \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2349        \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2350        \int_step_inline:nn { \l__stex_notation_arity_str } {
2351          \exp_args:NNo
2352          \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
```

```
2353        }
2354      }{
2355        \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2356        \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2357          \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2358          \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2359            \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2360              \l_tmpa_seq {\tl_to_str:n{x} } { \l_tmpa_str }
2361            \seq_map_inline:Nn \l_tmpa_seq {
2362              \seq_put_right:Nn \l_tmpb_seq { ##1 }
2363            }
2364          }
2365        }{
2366          \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2367            \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2368          }{
2369            \tl_set:No \l__stex_notation_opprec_tl { 0 }
2370          }
2371        }
2372      }
2373    }
2374
2375    \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2376    \int_step_inline:nn { \l__stex_notation_arity_str } {
2377      \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
2378        \exp_args:NNo
2379        \seq_put_right:No \l__stex_notation_precedences_seq {
2380          \l__stex_notation_opprec_tl
2381        }
2382      }
2383    }
2384    \tl_clear:N \l_stex_notation_dummyargs_tl
2385
2386    \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2387      \exp_args:NNe
2388      \cs_set:Npn \l_stex_notation_macrocode_cs {
2389        \_stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2390          { \l__stex_notation_suffix_str }
2391          { \l__stex_notation_opprec_tl }
2392          { \exp_not:n { #5 } }
2393      }
2394      \l_stex_notation_after_do_tl
2395    }{
2396      \str_if_in:NnTF \l__stex_notation_args_str b {
2397        \exp_args:Nne \use:nn
2398        {
2399        \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2400        \cs_set:Npn \l__stex_notation_arity_str } { {
2401          \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2402            { \l__stex_notation_suffix_str }
2403            { \l__stex_notation_opprec_tl }
2404            { \exp_not:n { #5 } }
2405        }}
2406      }{
```

```
2407        \str_if_in:NnTF \l__stex_notation_args_str B {
2408          \exp_args:Nne \use:nn
2409          {
2410          \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2411          \cs_set:Npn \l__stex_notation_arity_str } { { 
2412            \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2413              { \l__stex_notation_suffix_str }
2414              { \l__stex_notation_opprec_tl }
2415              { \exp_not:n { #5 } }
2416          } }
2417        }{
2418          \exp_args:Nne \use:nn
2419          {
2420          \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2421          \cs_set:Npn \l__stex_notation_arity_str } { { 
2422            \_stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2423              { \l__stex_notation_suffix_str }
2424              { \l__stex_notation_opprec_tl }
2425              { \exp_not:n { #5 } }
2426          } }
2427        }
2428      }
2429
2430      \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2431      \int_zero:N \l__stex_notation_currarg_int
2432      \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2433      \__stex_notation_arguments:
2434    }
2435 }
```

(*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

```
2436 \cs_new_protected:Nn \__stex_notation_arguments: {
2437    \int_incr:N \l__stex_notation_currarg_int
2438    \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2439    \l_stex_notation_after_do_tl
2440  }{
2441    \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2442    \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2443    \str_if_eq:VnTF \l_tmpa_str a {
2444      \__stex_notation_argument_assoc:nn{a}
2445    }{
2446      \str_if_eq:VnTF \l_tmpa_str B {
2447        \__stex_notation_argument_assoc:nn{B}
2448      }{
2449        \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpb_str
2450        \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2451          { \_stex_term_math_arg:nnn
2452            { \l_tmpa_str\int_use:N \l__stex_notation_currarg_int }
2453            { \l_tmpb_str }
2454            { ####\int_use:N \l__stex_notation_currarg_int }
2455          }
2456        }
```

```
2457            \__stex_notation_arguments:
2458          }
2459        }
2460      }
2461    }
```

(*End definition for* `\__stex_notation_arguments:`.)

```
2462  \cs_new_protected:Nn \__stex_notation_argument_assoc:nn {
2463
2464    \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2465      {\l__stex_notation_arity_str}{
2466      #2
2467    }
2468    \int_zero:N \l_tmpa_int
2469    \tl_clear:N \l_tmpa_tl
2470    \str_map_inline:Nn \l__stex_notation_args_str {
2471      \int_incr:N \l_tmpa_int
2472      \tl_put_right:Nx \l_tmpa_tl {
2473        \str_if_eq:nnTF {##1}{a}{ {} }{
2474          \str_if_eq:nnTF {##1}{B}{ {} }{
2475            {\_stex_term_arg:nn{##1\int_use:N \l_tmpa_int}{############### \int_use:N \l_tmpa
2476          }
2477        }
2478      }
2479    }
2480    \exp_after:wN\exp_after:wN\exp_after:wN \def
2481    \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2482    \exp_after:wN\exp_after:wN\exp_after:wN ##
2483    \exp_after:wN\exp_after:wN\exp_after:wN 1
2484    \exp_after:wN\exp_after:wN\exp_after:wN ##
2485    \exp_after:wN\exp_after:wN\exp_after:wN 2
2486    \exp_after:wN\exp_after:wN\exp_after:wN {
2487      \exp_after:wN \exp_after:wN \exp_after:wN
2488      \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2489        \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2490      }
2491    }
2492
2493    \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2494    \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
2495      \_stex_term_math_assoc_arg:nnnn
2496        { #1\int_use:N \l__stex_notation_currarg_int }
2497        { \l_tmpa_str }
2498        { ####\int_use:N \l__stex_notation_currarg_int }
2499        { \l_tmpa_cs {####1} {####2} }
2500    } }
2501    \__stex_notation_arguments:
2502  }
```

(*End definition for* `\__stex_notation_argument_assoc:nn`.)

`\__stex_notation_final:`   Called after processing all notation arguments

```
2503 \cs_new_protected:Nn \__stex_notation_restore_notation:nnnnn {
2504   \cs_generate_from_arg_count:cNnn{stex_notation_\detokenize{#1} \c_hash_str \detokenize{#2}
2505   \cs_set_nopar:Npn {#3}{#4}
2506   \tl_if_empty:nF {#5}{
2507     \tl_set:cn{stex_op_notation_\detokenize{#1} \c_hash_str \detokenize{#2}_cs}{ \comp{ #5 }
2508   }
2509   \seq_if_exist:cT { l_stex_symdecl_\detokenize{#1} _notations }{
2510     \seq_put_right:cx { l_stex_symdecl_\detokenize{#1} _notations } { \detokenize{#2} }
2511   }
2512 }
2513
2514 \cs_new_protected:Nn \__stex_notation_final: {
2515
2516   \stex_execute_in_module:x {
2517     \__stex_notation_restore_notation:nnnnn
2518       {\l_stex_get_symbol_uri_str}
2519       {\l__stex_notation_suffix_str}
2520       {\l__stex_notation_arity_str}
2521       {
2522         \exp_after:wN \exp_after:wN \exp_after:wN
2523         \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2524         { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2525       }
2526       {\exp_args:No \exp_not:n \l__stex_notation_op_tl }
2527   }
2528
2529   \stex_debug:nn{symbols}{
2530     Notation~\l__stex_notation_suffix_str
2531     ~for~\l_stex_get_symbol_uri_str^^J
2532     Operator~precedence:~\l__stex_notation_opprec_tl^^J
2533     Argument~precedences:~
2534       \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2535     Notation: \cs_meaning:c {
2536       stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2537       \l__stex_notation_suffix_str
2538       _cs
2539     }
2540   }
2541     % HTML annotations
2542   \stex_if_do_html:T {
2543     \stex_annotate_invisible:nnn { notation }
2544     { \l_stex_get_symbol_uri_str } {
2545       \stex_annotate_invisible:nnn { notationfragment }
2546         { \l__stex_notation_suffix_str }{}
2547       \stex_annotate_invisible:nnn { precedence }
2548         { \l__stex_notation_prec_str }{}
2549
2550       \int_zero:N \l_tmpa_int
2551       \str_set_eq:NN \l_stex_notation_remaining_args_str \l__stex_notation_args_str
2552       \tl_clear:N \l_tmpa_tl
2553       \int_step_inline:nn { \l__stex_notation_arity_str }{
2554         \int_incr:N \l_tmpa_int
2555         \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2556         \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_rem
```

154

```
2557          \str_if_eq:VnTF \l_tmpb_str a {
2558            \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2559              \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2560              \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2561            } }
2562          }{
2563            \str_if_eq:VnTF \l_tmpb_str B {
2564              \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2565                \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2566                \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2567              } }
2568            }{
2569              \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2570                \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
2571              } }
2572            }
2573          }
2574        }
2575        \stex_annotate_invisible:nnn { notationcomp }{}{
2576          \str_set:Nx \l_stex_current_symbol_str {\l_stex_get_symbol_uri_str }
2577          $ \exp_args:Nno \use:nn { \use:c {
2578            stex_notation_ \l_stex_current_symbol_str
2579            \c_hash_str \l__stex_notation_suffix_str _cs
2580          } } { \l_tmpa_tl } $
2581        }
2582      }
2583    }
2584 }
```

(End definition for `\__stex_notation_final:`*.)*

\setnotation

```
2585 \keys_define:nn { stex / setnotation } {
2586 %  lang     .tl_set_x:N  = \l__stex_notation_lang_str ,
2587   variant .tl_set_x:N  = \l__stex_notation_variant_str ,
2588   unknown .code:n      = \str_set:Nx
2589       \l__stex_notation_variant_str \l_keys_key_str
2590 }
2591
2592 \cs_new_protected:Nn \_stex_setnotation_args:n {
2593 % \str_clear:N \l__stex_notation_lang_str
2594   \str_clear:N \l__stex_notation_variant_str
2595   \keys_set:nn { stex / setnotation } { #1 }
2596 }
2597
2598 \cs_new_protected:Nn \__stex_notation_setnotation:nn {
2599   \seq_if_exist:cT{l_stex_symdecl_#1_notations}{
2600     \seq_remove_all:cn { l_stex_symdecl_#1 _notations }{ #2 }
2601     \seq_put_left:cn { l_stex_symdecl_#1 _notations }{ #2 }
2602   }
2603 }
2604
2605 \cs_new_protected:Nn \stex_setnotation:n {
2606   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1 _notations }
```

```
2607      { \l__stex_notation_variant_str }{
2608        \stex_execute_in_module:x{ \__stex_notation_setnotation:nn {#1}{\l__stex_notation_vari
2609        \stex_debug:nn {notations}{
2610          Setting~default~notation~
2611          {\l__stex_notation_variant_str }~for~
2612          #1 \\
2613          \expandafter\meaning\csname
2614          l_stex_symdecl_#1 _notations\endcsname
2615        }
2616      }{
2617        \msg_error:nnxx{stex}{unknownnotation}{\l__stex_notation_variant_str}{#1}
2618      }
2619 }
2620
2621 \NewDocumentCommand \setnotation {m m} {
2622   \stex_get_symbol:n { #1 }
2623   \_stex_setnotation_args:n { #2 }
2624   \stex_setnotation:n{\l_stex_get_symbol_uri_str}
2625   \stex_smsmode_do:\ignorespacesandpars
2626 }
2627
2628 \cs_new_protected:Nn \stex_copy_notations:nn {
2629   \stex_debug:nn {notations}{
2630     Copying~notations~from~#2~to~#1\\
2631     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
2632   }
2633   \tl_clear:N \l_tmpa_tl
2634   \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } } {
2635     \tl_put_right:Nn \l_tmpa_tl { {## ##1} }
2636   }
2637   \seq_map_inline:cn {l_stex_symdecl_#2_notations}{
2638     \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2639     \edef \l_tmpa_tl {
2640       \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2641       \exp_after:wN\exp_after:wN\exp_after:wN {
2642         \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2643       }
2644     }
2645
2646     \stex_execute_in_module:x {
2647       \__stex_notation_restore_notation:nnnnn
2648         {#1}{##1}
2649         { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }
2650         { \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl} }
2651         {
2652           \cs_if_exist:cT{stex_op_notation_ #2\c_hash_str ##1 _cs}{
2653             \exp_args:NNo\exp_args:No\exp_not:n{\csname stex_op_notation_ #2\c_hash_str ##1
2654           }
2655         }
2656     }
2657   }
2658 }
2659
2660 \NewDocumentCommand \copynotation {m m} {
```

```
2661      \stex_get_symbol:n { #1 }
2662      \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2663      \stex_get_symbol:n { #2 }
2664      \exp_args:Noo
2665      \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2666      \stex_smsmode_do:\ignorespacesandpars
2667  }
2668
```

(*End definition for* \setnotation. *This function is documented on page* *18.*)

\symdef

```
2669  \keys_define:nn { stex / symdef } {
2670    name     .str_set_x:N = \l_stex_symdecl_name_str ,
2671    local    .bool_set:N  = \l_stex_symdecl_local_bool ,
2672    args     .str_set_x:N = \l_stex_symdecl_args_str ,
2673    type     .tl_set:N    = \l_stex_symdecl_type_tl ,
2674    def      .tl_set:N    = \l_stex_symdecl_definiens_tl ,
2675    op       .tl_set:N    = \l__stex_notation_op_tl ,
2676  % lang     .str_set_x:N = \l__stex_notation_lang_str ,
2677    variant  .str_set_x:N = \l__stex_notation_variant_str ,
2678    prec     .str_set_x:N = \l__stex_notation_prec_str ,
2679    assoc    .choices:nn  =
2680        {bin,binl,binr,pre,conj,pwconj}
2681        {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
2682    unknown .code:n       = \str_set:Nx
2683        \l__stex_notation_variant_str \l_keys_key_str
2684  }
2685
2686  \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2687    \str_clear:N \l_stex_symdecl_name_str
2688    \str_clear:N \l_stex_symdecl_args_str
2689    \str_clear:N \l_stex_symdecl_assoctype_str
2690    \bool_set_false:N \l_stex_symdecl_local_bool
2691    \tl_clear:N \l_stex_symdecl_type_tl
2692    \tl_clear:N \l_stex_symdecl_definiens_tl
2693  % \str_clear:N \l__stex_notation_lang_str
2694    \str_clear:N \l__stex_notation_variant_str
2695    \str_clear:N \l__stex_notation_prec_str
2696    \tl_clear:N \l__stex_notation_op_tl
2697
2698    \keys_set:nn { stex / symdef } { #1 }
2699  }
2700
2701  \NewDocumentCommand \symdef { m O{} } {
2702    \__stex_notation_symdef_args:n { #2 }
2703    \bool_set_true:N \l_stex_symdecl_make_macro_bool
2704    \stex_symdecl_do:n { #1 }
2705    \tl_set:Nn \l_stex_notation_after_do_tl {
2706      \__stex_notation_final:
2707      \stex_smsmode_do:\ignorespacesandpars
2708    }
2709    \str_set:Nx \l_stex_get_symbol_uri_str {
2710      \l_stex_current_module_str ? \l_stex_symdecl_name_str
```

```
2711      }
2712    \exp_args:Nx \stex_notation_do:nnnnn
2713      { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } }
2714      { \prop_item:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } }
2715      { \l__stex_notation_variant_str }
2716      { \l__stex_notation_prec_str}
2717    }
2718    \stex_deactivate_macro:Nn \symdef {module~environments}
```

(*End definition for* \symdef. *This function is documented on page* *61*.)

29.3   Variables

```
2719    ⟨@@=stex_variables⟩
2720
2721    \keys_define:nn { stex / vardef } {
2722      name     .str_set_x:N  = \l__stex_variables_name_str ,
2723      args     .str_set_x:N  = \l__stex_variables_args_str ,
2724      type     .tl_set:N     = \l__stex_variables_type_tl ,
2725      def      .tl_set:N     = \l__stex_variables_def_tl ,
2726      op       .tl_set:N     = \l__stex_variables_op_tl ,
2727      prec     .str_set_x:N  = \l__stex_variables_prec_str ,
2728      assoc    .choices:nn   =
2729         {bin,binl,binr,pre,conj,pwconj}
2730         {\str_set:Nx \l_stex_variables_assoctype_str {\l_keys_choice_tl}},
2731      bind     .choices:nn   =
2732         {forall,exists}
2733         {\str_set:Nx \l_stex_variables_bind_str {\l_keys_choice_tl}}
2734    }
2735
2736    \cs_new_protected:Nn \__stex_variables_args:n {
2737      \str_clear:N \l__stex_variables_name_str
2738      \str_clear:N \l__stex_variables_args_str
2739      \str_clear:N \l__stex_variables_prec_str
2740      \str_clear:N \l__stex_variables_assoctype_str
2741      \str_clear:N \l__stex_variables_bind_str
2742      \tl_clear:N \l__stex_variables_type_tl
2743      \tl_clear:N \l__stex_variables_def_tl
2744      \tl_clear:N \l__stex_variables_op_tl
2745
2746      \keys_set:nn { stex / vardef } { #1 }
2747    }
2748
2749    \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{}} {
2750      \__stex_variables_args:n {#2}
2751      \str_if_empty:NT \l__stex_variables_name_str {
2752         \str_set:Nx \l__stex_variables_name_str { #1 }
2753      }
2754      \prop_clear:N \l_tmpa_prop
2755      \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
2756
2757      \int_zero:N \l_tmpb_int
2758      \bool_set_true:N \l_tmpa_bool
2759      \str_map_inline:Nn \l__stex_variables_args_str {
```

```
2760     \token_case_meaning:NnF ##1 {
2761       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2762       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2763       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2764       {\tl_to_str:n a} {
2765         \bool_set_false:N \l_tmpa_bool
2766         \int_incr:N \l_tmpb_int
2767       }
2768       {\tl_to_str:n B} {
2769         \bool_set_false:N \l_tmpa_bool
2770         \int_incr:N \l_tmpb_int
2771       }
2772     }{
2773       \msg_error:nnxx{stex}{error/wrongargs}{
2774         variable~\l__stex_variables_name_str
2775       }{##1}
2776     }
2777   }
2778   \bool_if:NTF \l_tmpa_bool {
2779     % possibly numeric
2780     \str_if_empty:NTF \l__stex_variables_args_str {
2781       \prop_put:Nnn \l_tmpa_prop { args } {}
2782       \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2783     }{
2784       \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
2785       \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2786       \str_clear:N \l_tmpa_str
2787       \int_step_inline:nn \l_tmpa_int {
2788         \str_put_right:Nn \l_tmpa_str i
2789       }
2790       \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
2791       \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2792     }
2793   } {
2794     \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2795     \prop_put:Nnx \l_tmpa_prop { arity }
2796       { \str_count:N \l__stex_variables_args_str }
2797   }
2798   \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2799   \tl_set:cx { #1 }{ \stex_invoke_variable:n { \l__stex_variables_name_str } }
2800
2801   \prop_set_eq:cN { l_stex_variable_\l__stex_variables_name_str _prop} \l_tmpa_prop
2802
2803   \tl_if_empty:NF \l__stex_variables_op_tl {
2804     \cs_set:cpx {
2805       stex_var_op_notation_ \l__stex_variables_name_str _cs
2806     } { \exp_not:N\comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } } }
2807   }
2808
2809   \tl_set:Nn \l_stex_notation_after_do_tl {
2810     \exp_args:Nne \use:nn {
2811       \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str _cs }
2812         \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } } }
2813     } {{
```

159

```
2814        \exp_after:wN \exp_after:wN \exp_after:wN
2815        \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2816        { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_symb
2817      }}
2818      \stex_if_do_html:T {
2819        \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
2820          \stex_annotate_invisible:nnn { precedence }
2821            { \l_stex_variables_prec_str }{}
2822          \tl_if_empty:NF \l__stex_variables_type_tl {\stex_annotate_invisible:nnn{type}{}{$\l
2823          \stex_annotate_invisible:nnn{args}{}{ \l__stex_variables_args_str }
2824          \stex_annotate_invisible:nnn{macroname}{#1}{}
2825          \tl_if_empty:NF \l__stex_variables_def_tl {
2826            \stex_annotate_invisible:nnn{definiens}{}
2827              {$\l__stex_variables_def_tl$}
2828          }
2829          \str_if_empty:NF \l__stex_variables_assoctype_str {
2830            \stex_annotate_invisible:nnn{assoctype}{\l__stex_variables_assoctype_str}{}
2831          }
2832          \str_if_empty:NF \l__stex_variables_bind_str {
2833            \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{}
2834          }
2835          \int_zero:N \l_tmpa_int
2836          \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
2837          \tl_clear:N \l_tmpa_tl
2838          \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
2839            \int_incr:N \l_tmpa_int
2840            \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
2841            \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables
2842            \str_if_eq:VnTF \l_tmpb_str a {
2843              \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2844                \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2845                \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2846              } }
2847            }{
2848              \str_if_eq:VnTF \l_tmpb_str B {
2849                \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2850                  \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2851                  \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2852                } }
2853              }{
2854                \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2855                  \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
2856                } }
2857              }
2858            }
2859          }
2860          \stex_annotate_invisible:nnn { notationcomp }{}{
2861            \str_set:Nx \l_stex_current_symbol_str {var://\l__stex_variables_name_str }
2862            $ \exp_args:Nno \use:nn { \use:c {
2863              stex_var_notation_\l__stex_variables_name_str _cs
2864            } } { \l_tmpa_tl } $
2865          }
2866        }
2867      }\ignorespacesandpars
```

160

```
2868    }
2869
2870    \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
2871 }
2872
2873 \cs_new:Nn \_stex_reset:N {
2874    \tl_if_exist:NTF #1 {
2875      \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
2876    }{
2877      \let \exp_not:N #1 \exp_not:N \undefined
2878    }
2879 }
2880
2881 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
2882    \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
2883    \exp_args:Nnx \use:nn {
2884      % TODO
2885      \stex_annotate_invisible:nnn {vardecl}{\clist_use:Nn\l__stex_variables_names,}{
2886        #2
2887      }
2888    }{
2889      \_stex_reset:N \varnot
2890      \_stex_reset:N \vartype
2891      \_stex_reset:N \vardefi
2892    }
2893 }
2894
2895 \NewDocumentCommand \vardef { s } {
2896    \IfBooleanTF#1 {
2897      \__stex_variables_do_complex:nn
2898    }{
2899      \__stex_variables_do_simple:nnn
2900    }
2901 }
2902
2903 \NewDocumentCommand \svar { O{} m }{
2904    \tl_if_empty:nTF {#1}{
2905      \str_set:Nn \l_tmpa_str { #2 }
2906    }{
2907      \str_set:Nn \l_tmpa_str { #1 }
2908    }
2909    \_stex_term_omv:nn {
2910      var://\l_tmpa_str
2911    }{
2912      \exp_args:Nnx \use:nn {
2913        \def\comp{\_varcomp}
2914        \str_set:Nx \l_stex_current_symbol_str { var://\l_tmpa_str }
2915        \comp{ #2 }
2916      }{
2917        \_stex_reset:N \comp
2918        \_stex_reset:N \l_stex_current_symbol_str
2919      }
2920    }
2921 }
```

```
2922

2923

2924

2925 \keys_define:nn { stex / varseq } {
2926   name    .str_set_x:N  = \l__stex_variables_name_str ,
2927   args    .int_set:N    = \l__stex_variables_args_int ,
2928   type    .tl_set:N     = \l__stex_variables_type_tl  ,
2929   mid     .tl_set:N     = \l__stex_variables_mid_tl   ,
2930   bind    .choices:nn   =
2931       {forall,exists}
2932       {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2933 }

2934

2935 \cs_new_protected:Nn \__stex_variables_seq_args:n {
2936   \str_clear:N \l__stex_variables_name_str
2937   \int_set:Nn \l__stex_variables_args_int 1
2938   \tl_clear:N \l__stex_variables_type_tl
2939   \str_clear:N \l__stex_variables_bind_str

2940

2941   \keys_set:nn { stex / varseq } { #1 }
2942 }

2943

2944 \NewDocumentCommand \varseq {m O{} m m m}{
2945   \__stex_variables_seq_args:n { #2 }
2946   \str_if_empty:NT \l__stex_variables_name_str {
2947     \str_set:Nx \l__stex_variables_name_str { #1 }
2948   }
2949   \prop_clear:N \l_tmpa_prop
2950   \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}

2951

2952   \seq_set_from_clist:Nn \l_tmpa_seq {#3}
2953   \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
2954     \msg_error:nnxx{stex}{error/seqlength}
2955       {\int_use:N \l__stex_variables_args_int}
2956       {\seq_count:N \l_tmpa_seq}
2957   }
2958   \seq_set_from_clist:Nn \l_tmpb_seq {#4}
2959   \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
2960     \msg_error:nnxx{stex}{error/seqlength}
2961       {\int_use:N \l__stex_variables_args_int}
2962       {\seq_count:N \l_tmpb_seq}
2963   }
2964   \prop_put:Nnn \l_tmpa_prop {starts} {#3}
2965   \prop_put:Nnn \l_tmpa_prop {ends} {#4}

2966

2967   \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
2968     \cs_set:Npn  {\int_use:N \l__stex_variables_args_int} { #5 }

2969

2970   \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
2971   \int_step_inline:nn \l__stex_variables_args_int {
2972     \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
2973   }
2974   \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
2975   \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
```

```
2976    \tl_if_empty:NF \l__stex_variables_mid_tl {
2977      \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
2978      \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
2979    }
2980    \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
2981    \int_step_inline:nn \l__stex_variables_args_int {
2982      \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
2983    }
2984    \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
2985    \tl_put_right:No \l_tmpa_tl \l_tmpb_tl


2988    \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl

2990    \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}

2992    \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}

2994    \int_step_inline:nn \l__stex_variables_args_int {
2995      \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
2996        \_stex_term_math_arg:nnn{i##1}{0}{\exp_not:n{####}##1}
2997      }}
2998    }

3000    \tl_set:Nx \l_tmpa_tl {
3001      \_stex_term_math_oma:nnnn { varseq://\l__stex_variables_name_str}{}{0}{
3002        \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
3003      }
3004    }

3006    \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \stex_symbol_after_invokation_tl} }

3008    \exp_args:Nno \use:nn {
3009    \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
3010      \cs_set:Npn {\int_use:N \l__stex_variables_args_int}}{\l_tmpa_tl}

3012    \stex_debug:nn{sequences}{New~Sequence:~
3013      \expandafter\meaning\csname stex_varseq_\l__stex_variables_name_str _cs\endcsname\\~\\
3014      \prop_to_keyval:N \l_tmpa_prop
3015    }
3016    \stex_if_do_html:T{\stex_annotate_invisible:nnn{varseq}{\l__stex_variables_name_str}{
3017      \tl_if_empty:NF \l__stex_variables_type_tl {
3018        \stex_annotate:nnn {type}{}{$\seqtype\l__stex_variables_type_tl$}
3019      }
3020      \stex_annotate:nnn {args}{\int_use:N \l__stex_variables_args_int}{}
3021      \str_if_empty:NF \l__stex_variables_bind_str {
3022        \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{}
3023      }
3024    }}

3026    \prop_set_eq:cN {stex_varseq_\l__stex_variables_name_str _prop}\l_tmpa_prop
3027    \ignorespacesandpars
3028 }

3029
```

163

3030 ⟨/package⟩

Chapter 30

$\mathrm{S\kern-0.1emT\kern-0.15emE\kern-0.1emX}$ -Terms Implementation

```
3031 ⟨∗package⟩
3032
3033 %%%%%%%%%%%%    terms.dtx    %%%%%%%%%%%%
3034
3035 ⟨@@=stex_terms⟩
```

Warnings and error messages

```
3036 \msg_new:nnn{stex}{error/nonotation}{
3037   Symbol~#1~invoked,~but~has~no~notation#2!
3038 }
3039 \msg_new:nnn{stex}{error/notationarg}{
3040   Error~in~parsing~notation~#1
3041 }
3042 \msg_new:nnn{stex}{error/noop}{
3043   Symbol~#1~has~no~operator~notation~for~notation~#2
3044 }
3045 \msg_new:nnn{stex}{error/notallowed}{
3046   Symbol~invokation~#1~not~allowed~in~notation~component~of~#2
3047 }
3048 \msg_new:nnn{stex}{error/doubleargument}{
3049   Argument~#1~of~symbol~#2~already~assigned
3050 }
3051 \msg_new:nnn{stex}{error/overarity}{
3052   Argument~#1~invalid~for~symbol~#2~with~arity~#3
3053 }
3054
```

30.1   Symbol Invocations

`\stex_invoke_symbol:n`   Invokes a semantic macro

```
3055
3056
3057 \bool_new:N \l_stex_allow_semantic_bool
3058 \bool_set_true:N \l_stex_allow_semantic_bool
3059
```

```
3060 \cs_new_protected:Nn \stex_invoke_symbol:n {
3061   \bool_if:NTF \l_stex_allow_semantic_bool {
3062     \str_if_eq:eeF {
3063       \prop_item:cn {
3064         l_stex_symdecl_#1_prop
3065       }{ deprecate }
3066     }{}{
3067       \msg_warning:nnxx{stex}{warning/deprecated}{
3068         Symbol~#1
3069       }{
3070         \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
3071       }
3072     }
3073     \if_mode_math:
3074       \exp_after:wN \__stex_terms_invoke_math:n
3075     \else:
3076       \exp_after:wN \__stex_terms_invoke_text:n
3077     \fi: { #1 }
3078   }{
3079     \msg_error:nnxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
3080   }
3081 }
3082
3083 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
3084   \peek_charcode_remove:NTF ! {
3085     \__stex_terms_invoke_op_custom:nn {#1}
3086   }{
3087     \__stex_terms_invoke_custom:nn {#1}
3088   }
3089 }
3090
3091 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
3092   \peek_charcode_remove:NTF ! {
3093     % operator
3094     \peek_charcode_remove:NTF * {
3095       % custom op
3096       \__stex_terms_invoke_op_custom:nn {#1}
3097     }{
3098       % op notation
3099       \peek_charcode:NTF [ {
3100         \__stex_terms_invoke_op_notation:nw {#1}
3101       }{
3102         \__stex_terms_invoke_op_notation:nw {#1}[]
3103       }
3104     }
3105   }{
3106     \peek_charcode_remove:NTF * {
3107       \__stex_terms_invoke_custom:nn {#1}
3108       % custom
3109     }{
3110       % normal
3111       \peek_charcode:NTF [ {
3112         \__stex_terms_invoke_notation:nw {#1}
3113       }{
```

```
3114            \__stex_terms_invoke_notation:nw {#1}[]
3115          }
3116        }
3117    }
3118 }
3119
3120
3121 \cs_new_protected:Nn \__stex_terms_invoke_op_custom:nn {
3122    \exp_args:Nnx \use:nn {
3123      \def\comp{\_comp}
3124      \str_set:Nn \l_stex_current_symbol_str { #1 }
3125      \bool_set_false:N \l_stex_allow_semantic_bool
3126      \_stex_term_oms:nnn {#1}{#1 \c_hash_str CUSTOM-}{
3127        \comp{ #2 }
3128      }
3129    }{
3130      \_stex_reset:N \comp
3131      \_stex_reset:N \l_stex_current_symbol_str
3132      \bool_set_true:N \l_stex_allow_semantic_bool
3133    }
3134 }
3135
3136 \keys_define:nn { stex / terms } {
3137 %  lang     .tl_set_x:N = \l_stex_notation_lang_str ,
3138    variant .tl_set_x:N = \l_stex_notation_variant_str ,
3139    unknown .code:n      = \str_set:Nx
3140        \l_stex_notation_variant_str \l_keys_key_str
3141 }
3142
3143 \cs_new_protected:Nn \__stex_terms_args:n {
3144  % \str_clear:N \l_stex_notation_lang_str
3145    \str_clear:N \l_stex_notation_variant_str
3146
3147    \keys_set:nn { stex / terms } { #1 }
3148 }
3149
3150 \cs_new_protected:Nn \stex_find_notation:nn {
3151    \__stex_terms_args:n { #2 }
3152    \seq_if_empty:cTF {
3153      l_stex_symdecl_ #1 _notations
3154    } {
3155      \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
3156    } {
3157      \str_if_empty:NTF \l_stex_notation_variant_str {
3158        \seq_get_left:cN {l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
3159      }{
3160        \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
3161          \l_stex_notation_variant_str
3162        }{
3163        %  \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3164        }{
3165          \msg_error:nnxx{stex}{error/nonotation}{#1}{
3166            ~\l_stex_notation_variant_str
3167          }
```

```
3168            }
3169          }
3170        }
3171    }
3172
3173    \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3174        \exp_args:Nnx \use:nn {
3175          \def\comp{\_comp}
3176          \str_set:Nn \l_stex_current_symbol_str { #1 }
3177          \stex_find_notation:nn { #1 }{ #2 }
3178          \bool_set_false:N \l_stex_allow_semantic_bool
3179          \cs_if_exist:cTF {
3180            stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3181          }{
3182            \_stex_term_oms:nnn { #1 }{
3183              #1 \c_hash_str \l_stex_notation_variant_str
3184            }{
3185              \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3186            }
3187          }{
3188            \int_compare:nNnTF {\prop_item:cn {l_stex_symdecl_#1_prop}{arity}} = 0{
3189              \cs_if_exist:cTF {
3190                stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3191              }{
3192                \tl_set:Nx \stex_symbol_after_invokation_tl {
3193                  \_stex_reset:N \comp
3194                  \_stex_reset:N \stex_symbol_after_invokation_tl
3195                  \_stex_reset:N \l_stex_current_symbol_str
3196                  \bool_set_true:N \l_stex_allow_semantic_bool
3197                }
3198                \def\comp{\_comp}
3199                \str_set:Nn \l_stex_current_symbol_str { #1 }
3200                \bool_set_false:N \l_stex_allow_semantic_bool
3201                \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3202              }{
3203                \msg_error:nnxx{stex}{error/nonotation}{#1}{
3204                  ~\l_stex_notation_variant_str
3205                }
3206              }
3207            }{
3208              \msg_error:nnxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3209            }
3210          }
3211        }{
3212          \_stex_reset:N \comp
3213          \_stex_reset:N \l_stex_current_symbol_str
3214          \bool_set_true:N \l_stex_allow_semantic_bool
3215        }
3216    }
3217
3218    \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {
3219        \stex_find_notation:nn { #1 }{ #2 }
3220        \cs_if_exist:cTF {
3221          stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
```

168

```
3222   }{
3223     \tl_set:Nx \stex_symbol_after_invokation_tl {
3224       \_stex_reset:N \comp
3225       \_stex_reset:N \stex_symbol_after_invokation_tl
3226       \_stex_reset:N \l_stex_current_symbol_str
3227       \bool_set_true:N \l_stex_allow_semantic_bool
3228     }
3229     \def\comp{\_comp}
3230     \str_set:Nn \l_stex_current_symbol_str { #1 }
3231     \bool_set_false:N \l_stex_allow_semantic_bool
3232     \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3233   }{
3234     \msg_error:nnxx{stex}{error/nonotation}{#1}{
3235       ~\l_stex_notation_variant_str
3236     }
3237   }
3238 }
3239
3240 \prop_new:N \l__stex_terms_custom_args_prop
3241
3242 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3243   \exp_args:Nnx \use:nn {
3244     \bool_set_false:N \l_stex_allow_semantic_bool
3245     \def\comp{\_comp}
3246     \str_set:Nn \l_stex_current_symbol_str { # 1 }
3247     \prop_clear:N \l__stex_terms_custom_args_prop
3248     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3249     \prop_get:cnN {
3250       l_stex_symdecl_#1 _prop
3251     }{ args } \l_tmpa_str
3252     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3253     \tl_set:Nn \arg { \__stex_terms_arg: }
3254     \str_if_empty:NTF \l_tmpa_str {
3255       \_stex_term_oms:nnn {#1}{#1\c_hash_str CUSTOM-}{#2}
3256     }{
3257       \str_if_in:NnTF \l_tmpa_str b {
3258         \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3259       }{
3260         \str_if_in:NnTF \l_tmpa_str B {
3261           \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3262         }{
3263           \_stex_term_oma:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3264         }
3265       }
3266     }
3267     % TODO check that all arguments exist
3268   }{
3269     \_stex_reset:N \l_stex_current_symbol_str
3270     \_stex_reset:N \arg
3271     \_stex_reset:N \comp
3272     \_stex_reset:N \l__stex_terms_custom_args_prop
3273     \bool_set_true:N \l_stex_allow_semantic_bool
3274   }
3275 }
```

```
3276
3277 \NewDocumentCommand \__stex_terms_arg: { s O{} m }{
3278   \tl_if_empty:nTF {#2}{
3279     \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
3280     \bool_set_true:N \l_tmpa_bool
3281     \bool_do_while:Nn \l_tmpa_bool {
3282       \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int
3283         \int_incr:N \l_tmpa_int
3284       }{
3285         \bool_set_false:N \l_tmpa_bool
3286       }
3287     }
3288   }{
3289     \int_set:Nn \l_tmpa_int { #2 }
3290   }
3291   \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
3292   \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
3293     \msg_error:nnxxx{stex}{error/overarity}
3294       {\int_use:N \l_tmpa_int}
3295       {\l_stex_current_symbol_str}
3296       {\str_count:N \l_tmpa_str}
3297   }
3298   \str_set:Nx \l_tmpa_str {\str_item:Nn \l_tmpa_str \l_tmpa_int}
3299   \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
3300     \bool_lazy_any:nF {
3301       {\str_if_eq_p:Vn \l_tmpa_str {a}}
3302       {\str_if_eq_p:Vn \l_tmpa_str {B}}
3303     }{
3304       \msg_error:nnxx{stex}{error/doubleargument}
3305         {\int_use:N \l_tmpa_int}
3306         {\l_stex_current_symbol_str}
3307     }
3308   }
3309   \exp_args:NNx \prop_put:Nnn \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {#3}
3310   \bool_set_true:N \l_stex_allow_semantic_bool
3311   \IfBooleanTF#1{
3312     \stex_annotate_invisible:n { %TODO
3313       \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{#3}
3314     }
3315   }{ %TODO
3316     \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{#3}
3317   }
3318   \bool_set_false:N \l_stex_allow_semantic_bool
3319 }
3320
3321
3322 \cs_new_protected:Nn \_stex_term_arg:nn {
3323   \bool_set_true:N \l_stex_allow_semantic_bool
3324   \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3325   \bool_set_false:N \l_stex_allow_semantic_bool
3326 }
3327
3328 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
3329   \exp_args:Nnx \use:nn
```

170

```
3330      { \int_set:Nn \l__stex_terms_downprec { #2 }
3331          \_stex_term_arg:nn { #1 }{ #3 }
3332      }
3333      { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3334 }
```

(End definition for \stex_invoke_symbol:n. This function is documented on page 62.)

```
3335 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
3336    \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3337    \tl_set:Nn \l_tmpb_tl {\_stex_term_math_arg:nnn{#1}{#2}}
3338    \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 }}{
3339       \expandafter\if\expandafter\relax\noexpand#3
3340          \expandafter\__stex_terms_math_assoc_arg_maybe_sequence:N\expandafter#3
3341       \else\expandafter\__stex_terms_math_assoc_arg_simple:nn
3342       \expandafter{\expandafter}\expandafter#3\fi
3343    }{
3344       \__stex_terms_math_assoc_arg_simple:nn{#1}{#3}
3345    }
3346 }
3347
3348 \cs_new_protected:Nn \__stex_terms_math_assoc_arg_maybe_sequence:N {
3349    \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
3350    \str_if_empty:NTF \l_tmpa_str {
3351       \exp_args:Nx \cs_if_eq:NNTF {
3352          \tl_head:N #1
3353       } \stex_invoke_sequence:n {
3354          \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
3355          \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
3356          \tl_set:Nx \l_tmpa_tl {\prop_item:cn {stex_varseq_\l_tmpa_str _prop}{notation}}
3357          \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
3358          \tl_set:Nx \l_tmpa_tl {{\exp_not:N \exp_not:n{
3359             \exp_not:n{\exp_args:Nnx \use:nn} {
3360                \exp_not:n {
3361                   \def\comp{\_varcomp}
3362                   \str_set:Nn \l_stex_current_symbol_str
3363                } {varseq://\l_tmpa_str}
3364                \exp_not:n{ ##1 }
3365             }{
3366                \exp_not:n {
3367                   \_stex_reset:N \comp
3368                   \_stex_reset:N \l_stex_current_symbol_str
3369                }
3370             }
3371          }}}
3372          \exp_args:Nno \use:nn {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
3373          \seq_reverse:N \l_tmpa_seq
3374          \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
3375          \seq_map_inline:Nn \l_tmpa_seq {
3376             \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3377                \exp_args:Nno
3378                \l_tmpa_cs { ##1 } \l_tmpa_tl
3379             }
```

171

```
3380            }
3381            \tl_set:Nx \l_tmpa_tl {
3382                \_stex_term_omv:nn {varseq://\l_tmpa_str}{
3383                    \exp_args:No \exp_not:n \l_tmpa_tl
3384                }
3385            }
3386            \exp_args:No\l_tmpb_tl\l_tmpa_tl
3387        }{
3388            \__stex_terms_math_assoc_arg_simple:nn{} { #1 }
3389        }
3390    } {
3391        \__stex_terms_math_assoc_arg_simple:nn{} { #1 }
3392    }
3393
3394 }
3395
3396 \cs_new_protected:Nn \__stex_terms_math_assoc_arg_simple:nn {
3397    \clist_set:Nn \l_tmpa_clist{ #2 }
3398    \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3399        \tl_set:Nn \l_tmpa_tl { #2 }
3400    }{
3401        \clist_reverse:N \l_tmpa_clist
3402        \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3403        \tl_set:Nx \l_tmpa_tl { \_stex_term_arg:nn{A#1}{
3404            \exp_args:No \exp_not:n \l_tmpa_tl
3405        }}
3406        \clist_map_inline:Nn \l_tmpa_clist {
3407            \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3408                \exp_args:Nno
3409                \l_tmpa_cs { \_stex_term_arg:nn{A#1}{##1} } \l_tmpa_tl
3410            }
3411        }
3412    }
3413    \exp_args:No\l_tmpb_tl\l_tmpa_tl
3414 }
```

(*End definition for* `\_stex_term_math_assoc_arg:nnnn`*. This function is documented on page* *62.*)

30.2   Terms

Precedences:

```
3415 \tl_const:Nx \infprec {\int_use:N \c_max_int}
3416 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
3417 \int_new:N \l__stex_terms_downprec
3418 \int_set_eq:NN \l__stex_terms_downprec \infprec
```

(*End definition for* `\infprec`*,* `\neginfprec`*, and* `\l__stex_terms_downprec`*. These variables are docu-mented on page* *63.*)

Bracketing:

\l__stex_terms_left_bracket_str
\l__stex_terms_right_bracket_str

```
3419 \tl_set:Nn \l__stex_terms_left_bracket_str (
3420 \tl_set:Nn \l__stex_terms_right_bracket_str )
```

`\__stex_terms_maybe_brackets:nn` Compares precedences and insert brackets accordingly

```
3421 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
3422   \bool_if:NTF \l__stex_terms_brackets_done_bool {
3423     \bool_set_false:N \l__stex_terms_brackets_done_bool
3424     #2
3425   } {
3426     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
3427       \bool_if:NTF \l_stex_inparray_bool { #2 }{
3428         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
3429         \dobrackets { #2 }
3430       }
3431     }{ #2 }
3432   }
3433 }
```

(*End definition for* `\__stex_terms_maybe_brackets:nn`*.*)

`\dobrackets`

```
3434 \bool_new:N \l__stex_terms_brackets_done_bool
3435 %\RequirePackage{scalerel}
3436 \cs_new_protected:Npn \dobrackets #1 {
3437   %\ThisStyle{\if D\m@switch
3438   %   \exp_args:Nnx \use:nn
3439   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
3440   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
3441   %  \else
3442     \exp_args:Nnx \use:nn
3443     {
3444       \bool_set_true:N \l__stex_terms_brackets_done_bool
3445       \int_set:Nn \l__stex_terms_downprec \infprec
3446       \l__stex_terms_left_bracket_str
3447       #1
3448     }
3449     {
3450       \bool_set_false:N \l__stex_terms_brackets_done_bool
3451       \l__stex_terms_right_bracket_str
3452       \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3453     }
3454   %\fi}
3455 }
```

(*End definition for* `\dobrackets`*. This function is documented on page 63.*)

`\withbrackets`

```
3456 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
3457   \exp_args:Nnx \use:nn
3458   {
3459     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
3460     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
3461     #3
3462   }
3463   {
```

173

```
3464      \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
3465         {\l__stex_terms_left_bracket_str}
3466      \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
3467         {\l__stex_terms_right_bracket_str}
3468    }
3469 }
```

(*End definition for* `\withbrackets`*. This function is documented on page 63.*)

\STEXinvisible

```
3470 \cs_new_protected:Npn \STEXinvisible #1 {
3471    \stex_annotate_invisible:n { #1 }
3472 }
```

(*End definition for* `\STEXinvisible`*. This function is documented on page 63.*)

OMDoc terms:

_stex_term_math_oms:nnnn

```
3473 \cs_new_protected:Nn \_stex_term_oms:nnn {
3474    \stex_annotate:nnn{ OMID }{ #2 }{
3475       #3
3476    }
3477 }
3478
3479 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
3480    \__stex_terms_maybe_brackets:nn { #3 }{
3481       \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3482    }
3483 }
```

(*End definition for* `\_stex_term_math_oms:nnnn`*. This function is documented on page 62.*)

_stex_term_math_omv:nn

```
3484 \cs_new_protected:Nn \_stex_term_omv:nn {
3485    \stex_annotate:nnn{ OMV }{ #1 }{
3486       #2
3487    }
3488 }
```

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

_stex_term_math_oma:nnnn

```
3489 \cs_new_protected:Nn \_stex_term_oma:nnn {
3490    \stex_annotate:nnn{ OMA }{ #2 }{
3491       #3
3492    }
3493 }
3494
3495 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
3496    \__stex_terms_maybe_brackets:nn { #3 }{
3497       \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3498    }
3499 }
```

(*End definition for* `\_stex_term_math_oma:nnnn`*. This function is documented on page 62.*)

174

```
3500 \cs_new_protected:Nn \_stex_term_ombind:nnn {
3501   \stex_annotate:nnn{ OMBIND }{ #2 }{
3502     #3
3503   }
3504 }
3505
3506 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
3507   \__stex_terms_maybe_brackets:nn { #3 }{
3508     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3509   }
3510 }
```

(End definition for _stex_term_math_omb:nnnn*. This function is documented on page 62.)*

```
3511 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3512
3513 \keys_define:nn { stex / symname } {
3514   pre     .tl_set_x:N    = \l__stex_terms_pre_tl ,
3515   post    .tl_set_x:N    = \l__stex_terms_post_tl ,
3516   root    .tl_set_x:N    = \l__stex_terms_root_tl
3517 }
3518
3519 \cs_new_protected:Nn \stex_symname_args:n {
3520   \tl_clear:N \l__stex_terms_post_tl
3521   \tl_clear:N \l__stex_terms_pre_tl
3522   \tl_clear:N \l__stex_terms_root_str
3523   \keys_set:nn { stex / symname } { #1 }
3524 }
3525
3526 \NewDocumentCommand \symref { m m }{
3527   \let\compemph_uri_prev:\compemph@uri
3528   \let\compemph@uri\symrefemph@uri
3529   \STEXsymbol{#1}!{ #2 }
3530   \let\compemph@uri\compemph_uri_prev:
3531 }
3532
3533 \NewDocumentCommand \synonym { O{} m m}{
3534   \stex_symname_args:n { #1 }
3535   \let\compemph_uri_prev:\compemph@uri
3536   \let\compemph@uri\symrefemph@uri
3537   % TODO
3538   \STEXsymbol{#2}!{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
3539   \let\compemph@uri\compemph_uri_prev:
3540 }
3541
3542 \NewDocumentCommand \symname { O{} m }{
3543   \stex_symname_args:n { #1 }
3544   \stex_get_symbol:n { #2 }
3545   \str_set:Nx \l_tmpa_str {
3546     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3547   }
3548   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
```

```
3549
3550    \let\compemph_uri_prev:\compemph@uri
3551    \let\compemph@uri\symrefemph@uri
3552    \exp_args:NNx \use:nn
3553    \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!{
3554      \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
3555    } }
3556    \let\compemph@uri\compemph_uri_prev:
3557  }
3558
3559  \NewDocumentCommand \Symname { O{} m }{
3560    \stex_symname_args:n { #1 }
3561    \stex_get_symbol:n { #2 }
3562    \str_set:Nx \l_tmpa_str {
3563      \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3564    }
3565    \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3566    \let\compemph_uri_prev:\compemph@uri
3567    \let\compemph@uri\symrefemph@uri
3568    \exp_args:NNx \use:nn
3569    \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!{
3570      \exp_after:wN \stex_capitalize:n \l_tmpa_str
3571        \l__stex_terms_post_tl
3572    } }
3573    \let\compemph@uri\compemph_uri_prev:
3574  }
```

(*End definition for* \symref *and* \symname*. These functions are documented on page 62.*)

30.3   Notation Components

```
3575  ⟨@@=stex_notationcomps⟩
```

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

```
3576  \cs_new_protected:Npn \_comp #1 {
3577    \str_if_empty:NF \l_stex_current_symbol_str {
3578      \stex_html_backend:TF {
3579        \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
3580      }{
3581        \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3582      }
3583    }
3584  }
3585
3586  \cs_new_protected:Npn \_varcomp #1 {
3587    \str_if_empty:NF \l_stex_current_symbol_str {
3588      \stex_html_backend:TF {
3589        \stex_annotate:nnn { varcomp }{ \l_stex_current_symbol_str }{ #1 }
3590      }{
3591        \exp_args:Nnx \varemph@uri { #1 } { \l_stex_current_symbol_str }
3592      }
3593    }
3594  }
3595
```

176

```
3596  \def\comp{\_comp}
3597
3598  \cs_new_protected:Npn \compemph@uri #1 #2 {
3599      \compemph{ #1 }
3600  }
3601
3602
3603  \cs_new_protected:Npn \compemph #1 {
3604      #1
3605  }
3606
3607  \cs_new_protected:Npn \defemph@uri #1 #2 {
3608      \defemph{#1}
3609  }
3610
3611  \cs_new_protected:Npn \defemph #1 {
3612      \textbf{#1}
3613  }
3614
3615  \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3616      \symrefemph{#1}
3617  }
3618
3619  \cs_new_protected:Npn \symrefemph #1 {
3620      \textbf{#1}
3621  }
3622
3623  \cs_new_protected:Npn \varemph@uri #1 #2 {
3624      \varemph{#1}
3625  }
3626
3627  \cs_new_protected:Npn \varemph #1 {
3628      #1
3629  }
```

(*End definition for* \comp *and others. These functions are documented on page 63.*)

\ellipses

```
3630  \NewDocumentCommand \ellipses {} { \ldots }
```

(*End definition for* \ellipses. *This function is documented on page 63.*)

\parray
\prmatrix
\parrayline
\parraylineh
\parraycell

```
3631  \bool_new:N \l_stex_inparray_bool
3632  \bool_set_false:N \l_stex_inparray_bool
3633  \NewDocumentCommand \parray { m m } {
3634    \begingroup
3635    \bool_set_true:N \l_stex_inparray_bool
3636    \begin{array}{#1}
3637      #2
3638    \end{array}
3639    \endgroup
3640  }
3641
3642  \NewDocumentCommand \prmatrix { m } {
```

177

```
3643     \begingroup
3644     \bool_set_true:N \l_stex_inparray_bool
3645     \begin{matrix}
3646       #1
3647     \end{matrix}
3648     \endgroup
3649 }
3650
3651 \def \maybephline {
3652     \bool_if:NT \l_stex_inparray_bool {\hline}
3653 }
3654
3655 \def \parrayline #1 #2 {
3656     #1 #2 \bool_if:NT \l_stex_inparray_bool {\\}
3657 }
3658
3659 \def \pmrow #1 { \parrayline{}{ #1 } }
3660
3661 \def \parraylineh #1 #2 {
3662     #1 #2 \bool_if:NT \l_stex_inparray_bool {\\\hline}
3663 }
3664
3665 \def \parraycell #1 {
3666     #1 \bool_if:NT \l_stex_inparray_bool {&}
3667 }
```

(*End definition for* \parray *and others. These functions are documented on page* **??**.)

30.4   Variables

```
3668 ⟨@@=stex_variables⟩
```

\stex_invoke_variable:n   Invokes a variable

```
3669 \cs_new_protected:Nn \stex_invoke_variable:n {
3670     \if_mode_math:
3671       \exp_after:wN \__stex_variables_invoke_math:n
3672     \else:
3673       \exp_after:wN \__stex_variables_invoke_text:n
3674     \fi: {#1}
3675 }
3676
3677 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
3678     %TODO
3679 }
3680
3681
3682 \cs_new_protected:Nn \__stex_variables_invoke_math:n {
3683     \peek_charcode_remove:NTF ! {
3684       \peek_charcode_remove:NTF ! {
3685         \peek_charcode:NTF [ {
3686           \__stex_variables_invoke_op_custom:nw
3687         }{
3688           % TODO throw error
3689         }
```

```
3690        }{
3691          \__stex_variables_invoke_op:n { #1 }
3692        }
3693      }{
3694        \peek_charcode_remove:NTF * {
3695          \__stex_variables_invoke_text:n { #1 }
3696        }{
3697          \__stex_variables_invoke_math_ii:n { #1 }
3698        }
3699      }
3700    }
3701
3702    \cs_new_protected:Nn \__stex_variables_invoke_op:n {
3703      \cs_if_exist:cTF {
3704        stex_var_op_notation_ #1 _cs
3705      }{
3706        \exp_args:Nnx \use:nn {
3707          \def\comp{\_varcomp}
3708          \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3709          \_stex_term_omv:nn { var://#1 }{
3710            \use:c{stex_var_op_notation_ #1 _cs }
3711          }
3712        }{
3713          \_stex_reset:N \comp
3714          \_stex_reset:N \l_stex_current_symbol_str
3715        }
3716      }{
3717        \int_compare:nNnTF {\prop_item:cn {l_stex_variable_#1_prop}{arity}}} = 0{
3718          \__stex_variables_invoke_math_ii:n {#1}
3719        }{
3720          \msg_error:nnxx{stex}{error/noop}{variable~#1}{}
3721        }
3722      }
3723    }
3724
3725    \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n  #1 {
3726      \cs_if_exist:cTF {
3727        stex_var_notation_#1_cs
3728      }{
3729        \tl_set:Nx \stex_symbol_after_invokation_tl {
3730          \_stex_reset:N \comp
3731          \_stex_reset:N \stex_symbol_after_invokation_tl
3732          \_stex_reset:N \l_stex_current_symbol_str
3733          \bool_set_true:N \l_stex_allow_semantic_bool
3734        }
3735        \def\comp{\_varcomp}
3736        \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3737        \bool_set_false:N \l_stex_allow_semantic_bool
3738        \use:c{stex_var_notation_#1_cs}
3739      }{
3740        \msg_error:nnxx{stex}{error/nonotation}{variable~#1}{s}
3741      }
3742    }
```

(*End definition for* \stex_invoke_variable:n. *This function is documented on page* **??**.)

30.5 Sequences

```
3743  ⟨@@=stex_sequences⟩
3744
3745  \cs_new_protected:Nn \stex_invoke_sequence:n {
3746    \peek_charcode_remove:NTF ! {
3747      \_stex_term_omv:nn {varseq://#1}{
3748        \exp_args:Nnx \use:nn {
3749          \def\comp{\_varcomp}
3750          \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3751          \prop_item:cn{stex_varseq_#1_prop}{notation}
3752        }{
3753          \_stex_reset:N \comp
3754          \_stex_reset:N \l_stex_current_symbol_str
3755        }
3756      }
3757    }{
3758      \bool_set_false:N \l_stex_allow_semantic_bool
3759      \def\comp{\_varcomp}
3760      \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3761      \tl_set:Nx \stex_symbol_after_invokation_tl {
3762        \_stex_reset:N \comp
3763        \_stex_reset:N \stex_symbol_after_invokation_tl
3764        \_stex_reset:N \l_stex_current_symbol_str
3765        \bool_set_true:N \l_stex_allow_semantic_bool
3766      }
3767      \use:c { stex_varseq_#1_cs }
3768    }
3769  }
3770  ⟨/package⟩
```

Chapter 31

sTeX
-Structural Features
Implementation

```
3771 ⟨∗package⟩
3772
3773 %%%%%%%%%%%%   features.dtx   %%%%%%%%%%%%
3774
```

Warnings and error messages
```
3775 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3776   Symbol~#1~can~not~be~assigned~in~copymodule~#2
3777 }
3778 \msg_new:nnn{stex}{error/interpretmodule/nodefiniens}{
3779   Symbol~#1~not~assigned~in~interpretmodule~#2
3780 }
3781
3782 \msg_new:nnn{stex}{error/unknownstructure}{
3783   No~structure~#1~found!
3784 }
3785
3786 \msg_new:nnn{stex}{error/unknownfield}{
3787   No~field~#1~in~instance~#2~found!\\#3
3788 }
3789
3790 \msg_new:nnn{stex}{error/keyval}{
3791   Invalid~key=value~pair:#1
3792 }
3793 \msg_new:nnn{stex}{error/instantiate/missing}{
3794   Assignments~missing~in~instantiate:~#1
3795 }
3796 \msg_new:nnn{stex}{error/incompatible}{
3797   Incompatible~signature:~#1~(#2)~and~#3~(#4)
3798 }
3799
```

31.1 Imports with modification

```
3800  ⟨@@=stex_copymodule⟩
3801  \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
3802    \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3803      \tl_set:Nn \l_tmpa_tl { #1 }
3804      \__stex_copymodule_get_symbol_from_cs:
3805    }{
3806      % argument is a string
3807      % is it a command name?
3808      \cs_if_exist:cTF { #1 }{
3809        \cs_set_eq:Nc \l_tmpa_tl { #1 }
3810        \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3811        \str_if_empty:NTF \l_tmpa_str {
3812          \exp_args:Nx \cs_if_eq:NNTF {
3813            \tl_head:N \l_tmpa_tl
3814          } \stex_invoke_symbol:n {
3815            \__stex_copymodule_get_symbol_from_cs:n{ #2 }
3816          }{
3817            \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3818          }
3819        } {
3820          \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3821        }
3822      }{
3823        % argument is not a command name
3824        \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3825        % \l_stex_all_symbols_seq
3826      }
3827    }
3828  }
3829
3830  \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
3831    \str_set:Nn \l_tmpa_str { #1 }
3832    \bool_set_false:N \l_tmpa_bool
3833    \bool_if:NF \l_tmpa_bool {
3834      \tl_set:Nn \l_tmpa_tl {
3835        \msg_error:nnn{stex}{error/unknownsymbol}{#1}
3836      }
3837      \str_set:Nn \l_tmpa_str { #1 }
3838      \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3839      \seq_map_inline:Nn #2 {
3840        \str_set:Nn \l_tmpb_str { ##1 }
3841        \str_if_eq:eeT { \l_tmpa_str } {
3842          \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3843        } {
3844          \seq_map_break:n {
3845            \tl_set:Nn \l_tmpa_tl {
3846              \str_set:Nn \l_stex_get_symbol_uri_str {
3847                ##1
3848              }
3849            }
3850          }
3851        }
```

```
3852        }
3853      \l_tmpa_tl
3854    }
3855  }
3856
3857  \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
3858    \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3859      { \tl_tail:N \l_tmpa_tl }
3860    \tl_if_single:NTF \l_tmpa_tl {
3861      \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3862        \exp_after:wN \str_set:Nn \exp_after:wN
3863          \l_stex_get_symbol_uri_str \l_tmpa_tl
3864        \__stex_copymodule_get_symbol_check:n { #1 }
3865      }{
3866        % TODO
3867        % tail is not a single group
3868      }
3869    }{
3870      % TODO
3871      % tail is not a single group
3872    }
3873  }
3874
3875  \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
3876    \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
3877      \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3878        :~\seq_use:Nn #1 {,~}
3879      }
3880    }
3881  }
3882
3883  \cs_new_protected:Nn \stex_copymodule_start:nnnn {
3884    % import module
3885    \stex_import_module_uri:nn { #1 } { #2 }
3886    \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3887    \stex_import_require_module:nnnn
3888      { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3889      { \l_stex_import_path_str } { \l_stex_import_name_str }
3890
3891    \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3892    \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
3893
3894    % fields
3895    \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
3896    \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
3897      \seq_map_inline:cn {c_stex_module_##1_constants}{
3898        \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
3899          ##1 ? ####1
3900        }
3901      }
3902    }
3903
3904    % setup prop
3905    \seq_clear:N \l_tmpa_seq
```

```
3906    \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3907      name    = \l_stex_current_copymodule_name_str ,
3908      module  = \l_stex_current_module_str ,
3909      from    = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3910      includes = \l_tmpa_seq %,
3911  %   fields   = \l_tmpa_seq
3912    }
3913    \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3914      as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3915      \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_se
3916    \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,
3917
3918    \stex_if_do_html:T {
3919      \begin{stex_annotate_env} {#4} {
3920        \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3921      }
3922      \stex_annotate_invisible:nnn{domain}{\l_stex_import_ns_str ?\l_stex_import_name_str}{}
3923    }
3924  }
3925
3926  \cs_new_protected:Nn \stex_copymodule_end:n {
3927    % apply to every field
3928    \def \l_tmpa_cs ##1 ##2 {#1}
3929
3930    \tl_clear:N \__stex_copymodule_module_tl
3931    \tl_clear:N \__stex_copymodule_exec_tl
3932
3933    %\prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3934    \seq_clear:N \__stex_copymodule_fields_seq
3935
3936    \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
3937      \seq_map_inline:cn {c_stex_module_##1_constants}{
3938
3939        \tl_clear:N \__stex_copymodule_curr_symbol_tl % <- wrap in current symbol html
3940        \l_tmpa_cs{##1}{####1}
3941
3942        \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
3943          \str_set_eq:Nc \__stex_copymodule_curr_name_str {l__stex_copymodule_copymodule_##1?#
3944          \stex_if_do_html:T {
3945            \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
3946              \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?###
3947            }
3948          }
3949        }{
3950          \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str /
3951        }
3952
3953        \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
3954        \prop_put:Nnx \l_tmpa_prop { name } \__stex_copymodule_curr_name_str
3955        \prop_put:Nnx \l_tmpa_prop { module } \l_stex_current_module_str
3956
3957        \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
3958          \stex_if_do_html:T {
3959            \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
```

184

```
3960          $\stex_annotate_invisible:nnn{definiens}{}{\exp_after:wN \exp_not:N\csname l__st
3961        }
3962      }
3963      \prop_put:Nnn \l_tmpa_prop { defined } { true }
3964    }
3965
3966    \stex_add_constant_to_current_module:n \__stex_copymodule_curr_name_str
3967    \tl_put_right:Nx \__stex_copymodule_module_tl {
3968      \seq_clear:c {l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_n
3969      \prop_set_from_keyval:cn {
3970        l_stex_symdecl_\l_stex_current_module_str ? \__stex_copymodule_curr_name_str _prop
3971      }{
3972        \prop_to_keyval:N \l_tmpa_prop
3973      }
3974    }
3975
3976    \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
3977      \stex_if_do_html:T {
3978        \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
3979          \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1
3980        }
3981      }
3982      \tl_put_right:Nx \__stex_copymodule_module_tl {
3983        \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
3984          \stex_invoke_symbol:n {
3985            \l_stex_current_module_str ? \__stex_copymodule_curr_name_str
3986          }
3987        }
3988      }
3989    }
3990
3991    \seq_put_right:Nx \__stex_copymodule_fields_seq {\l_stex_current_module_str ? \__stex_
3992
3993    \tl_put_right:Nx \__stex_copymodule_exec_tl {
3994      \stex_copy_notations:nn {\l_stex_current_module_str ? \__stex_copymodule_curr_name_s
3995    }
3996
3997    \tl_put_right:Nx \__stex_copymodule_exec_tl {
3998      \stex_if_do_html:TF{
3999        \stex_annotate_invisible:nnn{assignment} {##1?####1} { \exp_after:wN \exp_not:n \e
4000      }{
4001        \exp_after:wN \exp_not:n \exp_after:wN {\__stex_copymodule_curr_symbol_tl}
4002      }
4003    }
4004  }
4005 }
4006
4007
4008 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \__stex_copymodule_fields_seq
4009 \tl_put_left:Nx \__stex_copymodule_module_tl {
4010   \prop_set_from_keyval:cn {
4011     l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _pro
4012   }{
4013     \prop_to_keyval:N \l_stex_current_copymodule_prop
```

185

```
4014        }
4015      }
4016
4017      \seq_gput_right:cx{c_stex_module_\l_stex_current_module_str _copymodules}{
4018        \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4019      }
4020
4021      \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4022      \stex_debug:nn{copymodule}{result:\meaning \__stex_copymodule_module_tl}
4023      \stex_debug:nn{copymodule}{output:\meaning \__stex_copymodule_exec_tl}
4024
4025      \__stex_copymodule_exec_tl
4026      \stex_if_do_html:T {
4027        \end{stex_annotate_env}
4028      }
4029    }
4030
4031    \NewDocumentEnvironment {copymodule} { O{} m m}{
4032      \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ copymodule }
4033      \stex_deactivate_macro:Nn \symdecl {module~environments}
4034      \stex_deactivate_macro:Nn \symdef {module~environments}
4035      \stex_deactivate_macro:Nn \notation {module~environments}
4036      \stex_reactivate_macro:N \assign
4037      \stex_reactivate_macro:N \renamedecl
4038      \stex_reactivate_macro:N \donotcopy
4039      \stex_smsmode_do:
4040    }{
4041      \stex_copymodule_end:n {}
4042    }
4043
4044    \NewDocumentEnvironment {interpretmodule} { O{} m m}{
4045      \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ interpretmodule }
4046      \stex_deactivate_macro:Nn \symdecl {module~environments}
4047      \stex_deactivate_macro:Nn \symdef {module~environments}
4048      \stex_deactivate_macro:Nn \notation {module~environments}
4049      \stex_reactivate_macro:N \assign
4050      \stex_reactivate_macro:N \renamedecl
4051      \stex_reactivate_macro:N \donotcopy
4052      \stex_smsmode_do:
4053    }{
4054      \stex_copymodule_end:n {
4055        \tl_if_exist:cF {
4056          l__stex_copymodule_copymodule_##1?##2_def_tl
4057        }{
4058          \str_if_eq:eeF {
4059            \prop_item:cn{
4060              l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
4061          }{ true }{
4062            \msg_error:nnxx{stex}{error/interpretmodule/nodefiniens}{
4063              ##1?##2
4064            }{\l_stex_current_copymodule_name_str}
4065          }
4066        }
4067      }
```

186

```
4068 }
4069
4070 \iffalse \begin{stex_annotate_env} \fi
4071 \NewDocumentEnvironment {realization} { O{} m}{
4072   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #2 }{ realize }
4073   \stex_deactivate_macro:Nn \symdecl {module~environments}
4074   \stex_deactivate_macro:Nn \symdef {module~environments}
4075   \stex_deactivate_macro:Nn \notation {module~environments}
4076   \stex_reactivate_macro:N \donotcopy
4077   \stex_reactivate_macro:N \assign
4078   \stex_smsmode_do:
4079 }{
4080   \stex_import_module_uri:nn { #1 } { #2 }
4081   \tl_clear:N \__stex_copymodule_exec_tl
4082   \tl_set:Nx \__stex_copymodule_module_tl {
4083     \stex_import_require_module:nnnn
4084       { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4085       { \l_stex_import_path_str } { \l_stex_import_name_str }
4086   }
4087
4088   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4089     \seq_map_inline:cn {c_stex_module_##1_constants}{
4090       \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str / #
4091       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
4092         \stex_if_do_html:T {
4093           \tl_put_right:Nx \__stex_copymodule_exec_tl {
4094             \stex_annotate_invisible:nnn{assignment} {##1?####1} {
4095               $\stex_annotate_invisible:nnn{definiens}{}{\exp_after:wN \exp_not:N\csname l__
4096             }
4097           }
4098         }
4099         \tl_put_right:Nx \__stex_copymodule_module_tl {
4100           \prop_put:cnn {l_stex_symdecl_##1?####1_prop}{ defined }{ true }
4101         }
4102       }
4103   }}
4104
4105   \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4106
4107   \__stex_copymodule_exec_tl
4108   \stex_if_do_html:T {\end{stex_annotate_env}}
4109 }
4110
4111 \NewDocumentCommand \donotcopy { m }{
4112   \str_clear:N \l_stex_import_name_str
4113   \str_set:Nn \l_tmpa_str { #1 }
4114   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4115   \seq_map_inline:Nn \l_stex_all_modules_seq {
4116     \str_set:Nn \l_tmpb_str { ##1 }
4117     \str_if_eq:eeT { \l_tmpa_str } {
4118       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4119     } {
4120       \seq_map_break:n {
4121         \stex_if_do_html:T {
```

```
4122            \stex_if_smsmode:F {
4123              \stex_annotate_invisible:nnn{donotcopy}{##1}{
4124                \stex_annotate:nnn{domain}{##1}{}
4125              }
4126            }
4127          }
4128          \str_set_eq:NN \l_stex_import_name_str \l_tmpb_str
4129        }
4130      }
4131      \seq_map_inline:cn {c_stex_module_##1_copymodules}{
4132        \str_set:Nn \l_tmpb_str { ####1 }
4133        \str_if_eq:eeT { \l_tmpa_str } {
4134          \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4135        } {
4136          \seq_map_break:n {\seq_map_break:n {
4137            \stex_if_do_html:T {
4138              \stex_if_smsmode:F {
4139                \stex_annotate_invisible:nnn{donotcopy}{####1}{
4140                  \stex_annotate:nnn{domain}{
4141                    \prop_item:cn {l_stex_copymodule_ ####1 _prop}{module}
4142                  }{}
4143                }
4144              }
4145            }
4146            \str_set:Nx \l_stex_import_name_str {
4147              \prop_item:cn {l_stex_copymodule_ ####1 _prop}{module}
4148            }
4149          }}
4150        }
4151      }
4152    }
4153    \str_if_empty:NTF \l_stex_import_name_str {
4154      % TODO throw error
4155    }{
4156      \stex_collect_imports:n {\l_stex_import_name_str }
4157      \seq_map_inline:Nn \l_stex_collect_imports_seq {
4158        \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
4159        \seq_map_inline:cn {c_stex_module_##1_constants}{
4160          \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? ####1 }
4161          \bool_lazy_any:nT {
4162            { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_name_str}}
4163            { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_macroname_str}}
4164            { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_def_tl}}
4165          }{
4166            % TODO throw error
4167          }
4168        }
4169      }
4170      \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
4171      \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_name_str }
4172      \prop_put:Nno \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
4173    }
4174    \stex_smsmode_do:
4175 }
```

```
4176
4177 \NewDocumentCommand \assign { m m }{
4178   \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
4179   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
4180   \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _def_tl}{#2}
4181   \stex_smsmode_do:
4182 }
4183
4184 \keys_define:nn { stex / renamedecl } {
4185   name          .str_set_x:N  = \l_stex_renamedecl_name_str
4186 }
4187 \cs_new_protected:Nn \__stex_copymodule_renamedecl_args:n {
4188   \str_clear:N \l_stex_renamedecl_name_str
4189   \keys_set:nn { stex / renamedecl } { #1 }
4190 }
4191
4192 \NewDocumentCommand \renamedecl { O{} m m}{
4193   \__stex_copymodule_renamedecl_args:n { #1 }
4194   \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
4195   \stex_debug:nn{renamedecl}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
4196   \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
4197   \str_if_empty:NTF \l_stex_renamedecl_name_str {
4198     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4199       \l_stex_get_symbol_uri_str
4200     } }
4201   } {
4202     \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex
4203     \stex_debug:nn{renamedecl}{@~\l_stex_current_module_str ? \l_stex_renamedecl_name_str}
4204     \prop_set_eq:cc {l_stex_symdecl_
4205       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4206       _prop
4207     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
4208     \seq_set_eq:cc {l_stex_symdecl_
4209       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4210       _notations
4211     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
4212     \prop_put:cnx {l_stex_symdecl_
4213       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4214       _prop
4215     }{ name }{ \l_stex_renamedecl_name_str }
4216     \prop_put:cnx {l_stex_symdecl_
4217       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4218       _prop
4219     }{ module }{ \l_stex_current_module_str }
4220     \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
4221       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4222     }
4223     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4224       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4225     } }
4226   }
4227   \stex_smsmode_do:
4228 }
4229
```

```
4230  \stex_deactivate_macro:Nn \assign {copymodules}
4231  \stex_deactivate_macro:Nn \renamedecl {copymodules}
4232  \stex_deactivate_macro:Nn \donotcopy {copymodules}

4233

4234
```

31.2   The feature environment

```
4235  ⟨@@=stex_features⟩

4236
4237  \NewDocumentEnvironment{structural_feature_module}{ m m m }{
4238    \stex_if_in_module:F {
4239      \msg_set:nnn{stex}{error/nomodule}{
4240        Structural~Feature~has~to~occur~in~a~module:\\
4241        Feature~#2~of~type~#1\\
4242        In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
4243      }
4244      \msg_error:nn{stex}{error/nomodule}
4245    }

4246
4247    \str_set_eq:NN \l_tmpa_str \l_stex_current_module_str

4248
4249    \stex_module_setup:nn{meta=NONE}{#2 - #1}

4250
4251    \stex_if_do_html:T {
4252      \begin{stex_annotate_env}{ feature:#1 }{\l_tmpa_str ? #2 - #1}
4253        \stex_annotate_invisible:nnn{header}{}{ #3 }
4254    }
4255  }{
4256    \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
4257    \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
4258    \stex_debug:nn{features}{
4259      Feature: \l_stex_last_feature_str
4260    }
4261    \stex_if_do_html:T {
4262      \end{stex_annotate_env}
4263    }
4264  }
```

31.3   Structure

```
4265  ⟨@@=stex_structures⟩
4266  \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
4267    \prop_if_exist:cF {c_stex_module_\l_stex_current_module_str _structures}{
4268      \prop_new:c {c_stex_module_\l_stex_current_module_str _structures}
4269    }
4270    \prop_gput:cxx{c_stex_module_\l_stex_current_module_str _structures}
4271      {#1}{#2}
4272  }

4273
```

```
4274 \keys_define:nn { stex / features / structure } {
4275   name          .str_set_x:N  = \l__stex_structures_name_str ,
4276 }
4277
4278 \cs_new_protected:Nn \__stex_structures_structure_args:n {
4279   \str_clear:N \l__stex_structures_name_str
4280   \keys_set:nn { stex / features / structure } { #1 }
4281 }
4282
4283 \NewDocumentEnvironment{mathstructure}{m O{}}{
4284   \__stex_structures_structure_args:n { #2 }
4285   \str_if_empty:NT \l__stex_structures_name_str {
4286     \str_set:Nx \l__stex_structures_name_str { #1 }
4287   }
4288   \stex_suppress_html:n {
4289     \exp_args:Nx \stex_symdecl_do:nn {
4290       name = \l__stex_structures_name_str ,
4291       def  = {\STEXsymbol{module-type}{
4292         \_stex_term_math_oms:nnnn {
4293           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4294             { ns } ?
4295             \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4296               { name } / \l__stex_structures_name_str - structure
4297         }{}{0}{}
4298       }}
4299     }{ #1 }
4300   }
4301   \exp_args:Nnnx
4302   \begin{structural_feature_module}{ structure }
4303     { \l__stex_structures_name_str }{}
4304   \stex_smsmode_do:
4305 }{
4306   \end{structural_feature_module}
4307   \_stex_reset_up_to_module:n \l_stex_last_feature_str
4308   \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
4309   \seq_clear:N \l_tmpa_seq
4310   \seq_map_inline:Nn \l_stex_collect_imports_seq {
4311     \seq_map_inline:cn{c_stex_module_##1_constants}{
4312       \seq_put_right:Nn \l_tmpa_seq { ##1 ? ####1 }
4313     }
4314   }
4315   \exp_args:Nnno
4316   \prop_gput:cnn {c_stex_module_ \l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
4317   \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
4318   \stex_add_structure_to_current_module:nn
4319     \l__stex_structures_name_str
4320     \l_stex_last_feature_str
4321
4322   \stex_execute_in_module:x {
4323     \tl_set:cn { #1 }{
4324       \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structure
4325     }
4326   }
4327 }
```

191

```
4328
4329 \cs_new:Nn \stex_invoke_structure:nn {
4330   \stex_invoke_symbol:n { #1?#2 }
4331 }
4332
4333 \cs_new_protected:Nn \stex_get_structure:n {
4334   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4335     \tl_set:Nn \l_tmpa_tl { #1 }
4336     \__stex_structures_get_from_cs:
4337   }{
4338     \cs_if_exist:cTF { #1 }{
4339       \cs_set_eq:Nc \l_tmpa_cs { #1 }
4340       \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
4341       \str_if_empty:NTF \l_tmpa_str {
4342         \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs} \stex_invoke_structure:nn {
4343           \__stex_structures_get_from_cs:
4344         }{
4345           \__stex_structures_get_from_string:n { #1 }
4346         }
4347       }{
4348         \__stex_structures_get_from_string:n { #1 }
4349       }
4350     }{
4351       \__stex_structures_get_from_string:n { #1 }
4352     }
4353   }
4354 }
4355
4356 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
4357   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4358     { \tl_tail:N \l_tmpa_tl }
4359   \str_set:Nx \l_tmpa_str {
4360     \exp_after:wN \use_i:nn \l_tmpa_tl
4361   }
4362   \str_set:Nx \l_tmpb_str {
4363     \exp_after:wN \use_ii:nn \l_tmpa_tl
4364   }
4365   \str_set:Nx \l_stex_get_structure_str {
4366     \l_tmpa_str ? \l_tmpb_str
4367   }
4368   \str_set:Nx \l_stex_get_structure_module_str {
4369     \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
4370   }
4371 }
4372
4373 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
4374   \tl_set:Nn \l_tmpa_tl {
4375     \msg_error:nnn{stex}{error/unknownstructure}{#1}
4376   }
4377   \str_set:Nn \l_tmpa_str { #1 }
4378   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4379
4380   \seq_map_inline:Nn \l_stex_all_modules_seq {
4381     \prop_if_exist:cT {c_stex_module_##1_structures} {
```

```
4382        \prop_map_inline:cn {c_stex_module_##1_structures} {
4383          \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
4384            \prop_map_break:n{\seq_map_break:n{
4385              \tl_set:Nn \l_tmpa_tl {
4386                \str_set:Nn \l_stex_get_structure_str {##1?####1}
4387                \str_set:Nn \l_stex_get_structure_module_str {####2}
4388              }
4389            }}
4390          }
4391        }
4392      }
4393    }
4394    \l_tmpa_tl
4395 }
```

```
4396
4397 \keys_define:nn { stex / instantiate } {
4398    name        .str_set_x:N  = \l__stex_structures_name_str
4399 }
4400 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
4401    \str_clear:N \l__stex_structures_name_str
4402    \keys_set:nn { stex / instantiate } { #1 }
4403 }
4404
4405 \NewDocumentCommand \instantiate {m O{} m m m}{
4406    \begingroup
4407      \stex_get_structure:n {#4}
4408      \__stex_structures_instantiate_args:n { #2 }
4409      \str_if_empty:NT \l__stex_structures_name_str {
4410        \str_set:Nn \l__stex_structures_name_str { #1 }
4411      }
4412      \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4413      \seq_clear:N \l__stex_structures_fields_seq
4414      \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4415      \seq_map_inline:Nn \l_stex_collect_imports_seq {
4416        \seq_map_inline:cn {c_stex_module_##1_constants}{
4417          \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4418        }
4419      }
4420
4421      \tl_if_empty:nF{#3}{
4422        \seq_set_split:Nnn \l_tmpa_seq , {#3}
4423        \prop_clear:N \l_tmpa_prop
4424        \seq_map_inline:Nn \l_tmpa_seq {
4425          \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4426          \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4427            \msg_error:nnn{stex}{error/keyval}{##1}
4428          }
4429          \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
4430          \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4431          \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
4432          \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}
4433          \exp_args:Nxx \str_if_eq:nnF
```

193

```
4434              {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4435              {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
4436              \msg_error:nnxxxx{stex}{error/incompatible}
4437                {\l__stex_structures_dom_str}
4438                {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4439                {\l_stex_get_symbol_uri_str}
4440                {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
4441          }
4442          \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str
4443        }
4444      }
4445
4446      \seq_map_inline:Nn \l__stex_structures_fields_seq {
4447        \str_set:Nx \l_tmpa_str {field:\l__stex_structures_name_str . \prop_item:cn {l_stex_sy
4448        \stex_debug:nn{instantiate}{Field~\l_tmpa_str :~##1}
4449
4450        \stex_add_constant_to_current_module:n {\l_tmpa_str}
4451        \stex_execute_in_module:x {
4452          \prop_set_from_keyval:cn { l_stex_symdecl_ \l_stex_current_module_str?\l_tmpa_str _p
4453            name  = \l_tmpa_str ,
4454            args  = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4455            arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4456            assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4457          }
4458          \seq_clear:c {l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _notations}
4459        }
4460
4461        \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4462          \stex_find_notation:nn{##1}{}
4463          \stex_execute_in_module:x {
4464            \seq_put_right:cn {l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _notation
4465          }
4466
4467          \stex_copy_control_sequence:ccN
4468            {stex_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_notation_
4469            {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4470            \l_tmpa_tl
4471          \exp_args:No \stex_execute_in_module:n \l_tmpa_tl
4472
4473
4474          \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
4475            \tl_set_eq:Nc \l_tmpa_cs {stex_op_notation_##1\c_hash_str \l_stex_notation_variant
4476            \stex_execute_in_module:x {
4477              \tl_set:cn
4478              {stex_op_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_nota
4479              { \exp_args:No \exp_not:n \l_tmpa_cs}
4480            }
4481          }
4482
4483        }
4484
4485        \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\l_stex_cur
4486      }
4487
```

194

```
4488      \stex_execute_in_module:x {
4489        \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
4490          domain = \l_stex_get_structure_module_str ,
4491          \prop_to_keyval:N \l_tmpa_prop
4492        }
4493        \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
4494      }
4495      \stex_debug:nn{instantiate}{
4496        Instance~\l_stex_current_module_str?\l__stex_structures_name_str \\
4497        \prop_to_keyval:N \l_tmpa_prop
4498      }
4499      \exp_args:Nxx \stex_symdecl_do:nn {
4500        type={\STEXsymbol{module-type}{
4501          \_stex_term_math_oms:nnnn {
4502            \l_stex_get_structure_module_str
4503          }{}{0}{}
4504        }}
4505      }{\l__stex_structures_name_str}
4506 %    {
4507        \str_set:Nx \l_stex_get_symbol_uri_str {\l_stex_current_module_str?\l__stex_structures
4508        \tl_set:Nn \l_stex_notation_after_do_tl {\__stex_notation_final:}
4509        \stex_notation_do:nnnnn{}{0}{}{}{\comp{#5}}
4510 %    }
4511      %\exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}
4512    \endgroup
4513    \stex_smsmode_do:\ignorespacesandpars
4514 }
4515
4516 \cs_new_protected:Nn \stex_symbol_or_var:n {
4517    \cs_if_exist:cTF{#1}{
4518      \cs_set_eq:Nc \l_tmpa_tl { #1 }
4519      \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4520      \str_if_empty:NTF \l_tmpa_str {
4521        \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
4522          \stex_invoke_variable:n {
4523            \bool_set_true:N \l_stex_symbol_or_var_bool
4524            \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
4525            \str_set:Nx \l_stex_get_symbol_uri_str {
4526              \exp_after:wN \use:n \l_tmpa_tl
4527            }
4528          }{
4529            \bool_set_false:N \l_stex_symbol_or_var_bool
4530            \stex_get_symbol:n{#1}
4531          }
4532      }{
4533        \__stex_structures_symbolorvar_from_string:n{ #1 }
4534      }
4535    }{
4536      \__stex_structures_symbolorvar_from_string:n{ #1 }
4537    }
4538 }
4539
4540 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
4541    \prop_if_exist:cTF {l_stex_variable_#1 _prop}{
```

195

```
4542     \bool_set_true:N \l_stex_symbol_or_var_bool
4543     \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
4544   }{
4545     \bool_set_false:N \l_stex_symbol_or_var_bool
4546     \stex_get_symbol:n{#1}
4547   }
4548 }
4549
4550 \keys_define:nn { stex / varinstantiate } {
4551   name        .str_set_x:N  = \l__stex_structures_name_str,
4552   bind        .choices:nn   =
4553      {forall,exists}
4554      {\str_set:Nx \l__stex_structures_bind_str {\l_keys_choice_tl}}
4555
4556 }
4557 \cs_new_protected:Nn \__stex_structures_varinstantiate_args:n {
4558   \str_clear:N \l__stex_structures_name_str
4559   \str_clear:N \l__stex_structures_bind_str
4560   \keys_set:nn { stex / varinstantiate } { #1 }
4561 }
4562
4563 \NewDocumentCommand \varinstantiate {m O{} m m m}{
4564   \begingroup
4565     \stex_get_structure:n {#4}
4566     \__stex_structures_varinstantiate_args:n { #2 }
4567     \str_if_empty:NT \l__stex_structures_name_str {
4568       \str_set:Nn \l__stex_structures_name_str { #1 }
4569     }
4570     \stex_if_do_html:TF{
4571       \stex_annotate:nnn{varinstance}{\l__stex_structures_name_str}
4572     }{\use:n}
4573     {
4574       \stex_if_do_html:T{
4575         \stex_annotate:nnn{domain}{\l_stex_get_structure_module_str}{}
4576       }
4577       \seq_clear:N \l__stex_structures_fields_seq
4578       \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4579       \seq_map_inline:Nn \l_stex_collect_imports_seq {
4580         \seq_map_inline:cn {c_stex_module_##1_constants}{
4581           \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4582         }
4583       }
4584       \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4585       \prop_clear:N \l_tmpa_prop
4586       \tl_if_empty:nF {#3} {
4587         \seq_set_split:Nnn \l_tmpa_seq , {#3}
4588         \seq_map_inline:Nn \l_tmpa_seq {
4589           \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4590           \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4591             \msg_error:nnn{stex}{error/keyval}{##1}
4592           }
4593           \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_stru
4594           \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4595           \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol
```

```
4596            \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
4597            \stex_if_do_html:T{
4598              \stex_annotate:nnn{assign}{\l__stex_structures_dom_str,\l_stex_get_symbol_uri_st
4599            }
4600            \bool_if:NTF \l_stex_symbol_or_var_bool {
4601              \exp_args:Nxx \str_if_eq:nnF
4602                {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4603                {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}{
4604              \msg_error:nnxxxx{stex}{error/incompatible}
4605                {\l__stex_structures_dom_str}
4606                {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4607                {\l_stex_get_symbol_uri_str}
4608                {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}
4609            }
4610            \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n
4611          }{
4612              \exp_args:Nxx \str_if_eq:nnF
4613                {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4614                {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
4615              \msg_error:nnxxxx{stex}{error/incompatible}
4616                {\l__stex_structures_dom_str}
4617                {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4618                {\l_stex_get_symbol_uri_str}
4619                {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
4620            }
4621            \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n {
4622          }
4623        }
4624      }
4625      \tl_gclear:N \g__stex_structures_aftergroup_tl
4626      \seq_map_inline:Nn \l__stex_structures_fields_seq {
4627        \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdec
4628        \stex_debug:nn{varinstantiate}{Field~\l_tmpa_str :~##1}
4629        \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4630          \stex_find_notation:nn{##1}{}
4631          \cs_gset_eq:cc{g__stex_structures_tmpa_\l_tmpa_str _cs}
4632            {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4633          \stex_debug:nn{varinstantiate}{Notation:~\cs_meaning:c{g__stex_structures_tmpa_\l_
4634          \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
4635            \cs_gset_eq:cc {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
4636              {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4637            \stex_debug:nn{varinstantiate}{Operator~Notation:~\cs_meaning:c{g__stex_struct
4638        }
4639      }
4640
4641      \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4642        \prop_set_from_keyval:cn { l_stex_variable_ \l_tmpa_str _prop}{
4643          name  = \l_tmpa_str ,
4644          args  = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4645          arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4646          assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4647        }
4648        \cs_set_eq:cc {stex_var_notation_\l_tmpa_str _cs}
4649          {g__stex_structures_tmpa_\l_tmpa_str _cs}
```

```
4650             \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str _cs}
4651               {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
4652           }
4653           \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\stex_inv
4654         }
4655       \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4656         \prop_set_from_keyval:cn {l_stex_varinstance_\l__stex_structures_name_str _prop }{
4657           domain = \l_stex_get_structure_module_str ,
4658           \prop_to_keyval:N \l_tmpa_prop
4659         }
4660         \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
4661         \tl_set:cn {l_stex_varinstance_\l__stex_structures_name_str _op_tl}{
4662           \exp_args:Nnx \exp_not:N \use:nn {
4663             \str_set:Nn \exp_not:N \l_stex_current_symbol_str {var://\l__stex_structures_nam
4664             \_stex_term_omv:nn {var://\l__stex_structures_name_str}{
4665               \exp_not:n{
4666                 \_varcomp{#5}
4667               }
4668             }
4669           }{
4670             \exp_not:n{\_stex_reset:N \l_stex_current_symbol_str}
4671           }
4672         }
4673       }
4674     }
4675     \stex_debug:nn{varinstantiate}{\expandafter\detokenize\expandafter{\g__stex_structures_a
4676     \aftergroup\g__stex_structures_aftergroup_tl
4677   \endgroup
4678   \stex_smsmode_do:\ignorespacesandpars
4679 }
4680
4681 \cs_new_protected:Nn \stex_invoke_instance:n {
4682   \peek_charcode_remove:NTF ! {
4683     \stex_invoke_symbol:n{#1}
4684   }{
4685     \_stex_invoke_instance:nn {#1}
4686   }
4687 }
4688
4689
4690 \cs_new_protected:Nn \stex_invoke_varinstance:n {
4691   \peek_charcode_remove:NTF ! {
4692     \exp_args:Nnx \use:nn {
4693       \def\comp{\_varcomp}
4694       \use:c{l_stex_varinstance_#1_op_tl}
4695     }{
4696       \_stex_reset:N \comp
4697     }
4698   }{
4699     \_stex_invoke_varinstance:nn {#1}
4700   }
4701 }
4702
4703 \cs_new_protected:Nn \_stex_invoke_instance:nn {
```

198

```
4704      \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
4705        \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
4706      }{
4707        \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
4708        \msg_error:nnxxx{stex}{error/unknownfield}{#2}{#1}{
4709          \prop_to_keyval:N \l_tmpa_prop
4710        }
4711      }
4712    }
4713
4714    \cs_new_protected:Nn \_stex_invoke_varinstance:nn {
4715      \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
4716        \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
4717        \l_tmpa_tl
4718      }{
4719        \msg_error:nnnnn{stex}{error/unknownfield}{#2}{#1}{}
4720      }
4721    }
```

(End definition for \instantiate*. This function is documented on page 31.)*

\stex_invoke_structure:nnn

```
4722    % #1: URI of the instance
4723    % #2: URI of the instantiated module
4724    \cs_new_protected:Nn \stex_invoke_structure:nnn {
4725      \tl_if_empty:nTF{ #3 }{
4726        \prop_set_eq:Nc \l__stex_structures_structure_prop {
4727          c_stex_feature_ #2 _prop
4728        }
4729        \tl_clear:N \l_tmpa_tl
4730        \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
4731        \seq_map_inline:Nn \l_tmpa_seq {
4732          \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
4733          \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
4734          \cs_if_exist:cT {
4735            stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
4736          }{
4737            \tl_if_empty:NF \l_tmpa_tl {
4738              \tl_put_right:Nn \l_tmpa_tl {,}
4739            }
4740            \tl_put_right:Nx \l_tmpa_tl {
4741              \stex_invoke_symbol:n {#1/\l_tmpa_str}!
4742            }
4743          }
4744        }
4745        \exp_args:No \mathstruct \l_tmpa_tl
4746      }{
4747        \stex_invoke_symbol:n{#1/#3}
4748      }
4749    }
```

(End definition for \stex_invoke_structure:nnn*. This function is documented on page* **??**.*)*

```
4750  ⟨/package⟩
```

199

Chapter 32

STEX
-Statements Implementation

4752

4753 %%%%%%%%%%%%   features.dtx   %%%%%%%%%%%%

4754

4755 ⟨@@=stex_statements⟩

Warnings and error messages

4756

\titleemph

4757 \def\titleemph#1{\textbf{#1}}

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

32.1   Definitions

definiendum

4758 \keys_define:nn {stex / definiendum }{
4759   pre   .tl_set:N   = \l__stex_statements_definiendum_pre_tl,
4760   post   .tl_set:N   = \l__stex_statements_definiendum_post_tl,
4761   root   .str_set_x:N = \l__stex_statements_definiendum_root_str,
4762   gfa   .str_set_x:N = \l__stex_statements_definiendum_gfa_str
4763 }
4764 \cs_new_protected:Nn __stex_statements_definiendum_args:n {
4765   \str_clear:N \l__stex_statements_definiendum_root_str
4766   \tl_clear:N \l__stex_statements_definiendum_post_tl
4767   \str_clear:N \l__stex_statements_definiendum_gfa_str
4768   \keys_set:nn { stex / definiendum }{ #1 }
4769 }
4770 \NewDocumentCommand \definiendum { O{} m m} {
4771   __stex_statements_definiendum_args:n { #1 }
4772   \stex_get_symbol:n { #2 }
4773   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4774   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
4775   \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {

```
4776        \tl_set:Nn \l_tmpa_tl { #3 }
4777      } {
4778        \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
4779        \tl_set:Nn \l_tmpa_tl {
4780          \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
4781        }
4782      }
4783    } {
4784      \tl_set:Nn \l_tmpa_tl { #3 }
4785    }
4786
4787    % TODO root
4788    \stex_html_backend:TF {
4789      \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
4790    } {
4791      \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
4792    }
4793 }
4794 \stex_deactivate_macro:Nn \definiendum {definition~environments}
```

(*End definition for* `definiendum`. *This function is documented on page 40.*)

definame

```
4795
4796 \NewDocumentCommand \definame { O{} m } {
4797    \__stex_statements_definiendum_args:n { #1 }
4798    % TODO: root
4799    \stex_get_symbol:n { #2 }
4800    \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4801    \str_set:Nx \l_tmpa_str {
4802      \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4803    }
4804    \str_replace_all:Nnn \l_tmpa_str {-} {~}
4805    \stex_html_backend:TF {
4806      \stex_if_do_html:T {
4807        \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4808          \l_tmpa_str\l__stex_statements_definiendum_post_tl
4809        }
4810      }
4811    } {
4812      \exp_args:Nnx \defemph@uri {
4813        \l_tmpa_str\l__stex_statements_definiendum_post_tl
4814      } { \l_stex_get_symbol_uri_str }
4815    }
4816 }
4817 \stex_deactivate_macro:Nn \definame {definition~environments}
4818
4819 \NewDocumentCommand \Definame { O{} m } {
4820    \__stex_statements_definiendum_args:n { #1 }
4821    \stex_get_symbol:n { #2 }
4822    \str_set:Nx \l_tmpa_str {
4823      \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4824    }
4825    \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
```

```
4826     \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4827     \stex_html_backend:TF {
4828       \stex_if_do_html:T {
4829         \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4830           \l_tmpa_str\l__stex_statements_definiendum_post_tl
4831         }
4832       }
4833     } {
4834       \exp_args:Nnx \defemph@uri {
4835         \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
4836       } { \l_stex_get_symbol_uri_str }
4837     }
4838   }
4839   \stex_deactivate_macro:Nn \Definame {definition~environments}
4840
4841   \NewDocumentCommand \premise { m }{
4842     \stex_annotate:nnn{ premise }{}{ #1 }
4843   }
4844   \NewDocumentCommand \conclusion { m }{
4845     \stex_annotate:nnn{ conclusion }{}{ #1 }
4846   }
4847   \NewDocumentCommand \definiens { O{} m }{
4848     \str_clear:N \l_stex_get_symbol_uri_str
4849     \tl_if_empty:nF {#1} {
4850       \stex_get_symbol:n { #1 }
4851     }
4852     \str_if_empty:NT \l_stex_get_symbol_uri_str {
4853       \int_compare:nNnTF {\clist_count:N \l__stex_statements_sdefinition_for_clist} = 1 {
4854         \str_set:Nx \l_stex_get_symbol_uri_str {\clist_item:Nn \l__stex_statements_sdefinition
4855       }{
4856         % TODO throw error
4857       }
4858     }
4859     \str_if_eq:eeT {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{module}}
4860       {\l_stex_current_module_str}{
4861         \str_if_eq:eeF {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defin
4862         {true}{
4863           \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
4864           \exp_args:Nx \stex_add_to_current_module:n {
4865             \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
4866           }
4867         }
4868     }
4869     \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
4870   }
4871
4872   \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
4873   \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
4874   \stex_deactivate_macro:Nn \definiens {definition~environments}
4875
```

(*End definition for* `definame`. *This function is documented on page [40](#).*)

sdefinition

```
4876
4877  \keys_define:nn {stex / sdefinition }{
4878    type     .str_set_x:N  = \sdefinitiontype,
4879    id       .str_set_x:N  = \sdefinitionid,
4880    name     .str_set_x:N  = \sdefinitionname,
4881    for      .clist_set:N   = \l__stex_statements_sdefinition_for_clist ,
4882    title    .tl_set:N      = \sdefinitiontitle
4883  }
4884  \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
4885    \str_clear:N \sdefinitiontype
4886    \str_clear:N \sdefinitionid
4887    \str_clear:N \sdefinitionname
4888    \clist_clear:N \l__stex_statements_sdefinition_for_clist
4889    \tl_clear:N \sdefinitiontitle
4890    \keys_set:nn { stex / sdefinition }{ #1 }
4891  }
4892
4893  \NewDocumentEnvironment{sdefinition}{O{}}{
4894    \__stex_statements_sdefinition_args:n{ #1 }
4895    \stex_reactivate_macro:N \definiendum
4896    \stex_reactivate_macro:N \definame
4897    \stex_reactivate_macro:N \Definame
4898    \stex_reactivate_macro:N \premise
4899    \stex_reactivate_macro:N \definiens
4900    \stex_if_smsmode:F{
4901      \seq_clear:N \l_tmpa_seq
4902      \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4903        \tl_if_empty:nF{ ##1 }{
4904          \stex_get_symbol:n { ##1 }
4905          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4906            \l_stex_get_symbol_uri_str
4907          }
4908        }
4909      }
4910      \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpa_seq
4911      \exp_args:Nnnx
4912      \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {,}}
4913      \str_if_empty:NF \sdefinitiontype {
4914        \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
4915      }
4916      \str_if_empty:NF \sdefinitionname {
4917        \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
4918      }
4919      \clist_set:No \l_tmpa_clist \sdefinitiontype
4920      \tl_clear:N \l_tmpa_tl
4921      \clist_map_inline:Nn \l_tmpa_clist {
4922        \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
4923          \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
4924        }
4925      }
4926      \tl_if_empty:NTF \l_tmpa_tl {
4927        \__stex_statements_sdefinition_start:
4928      }{
4929        \l_tmpa_tl
```

```
4930          }
4931        }
4932      \stex_ref_new_doc_target:n \sdefinitionid
4933      \stex_smsmode_do:
4934   }{
4935      \stex_suppress_html:n {
4936        \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
4937      }
4938      \stex_if_smsmode:F {
4939        \clist_set:No \l_tmpa_clist \sdefinitiontype
4940        \tl_clear:N \l_tmpa_tl
4941        \clist_map_inline:Nn \l_tmpa_clist {
4942          \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
4943            \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
4944          }
4945        }
4946        \tl_if_empty:NTF \l_tmpa_tl {
4947          \__stex_statements_sdefinition_end:
4948        }{
4949          \l_tmpa_tl
4950        }
4951        \end{stex_annotate_env}
4952      }
4953   }
```

```
4954   \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
4955      \par\noindent\titleemph{Definition\tl_if_empty:NF \sdefinitiontitle {
4956        ~(\sdefinitiontitle)
4957      }~}
4958   }
4959   \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\par\medskip}
4960
4961   \newcommand\stexpatchdefinition[3][] {
4962      \str_set:Nx \l_tmpa_str{ #1 }
4963      \str_if_empty:NTF \l_tmpa_str {
4964        \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
4965        \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
4966      }{
4967        \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2
4968        \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
4969      }
4970   }
```

(*End definition for* \stexpatchdefinition. *This function is documented on page 42.*)

\inlinedef inline:

```
4971   \keys_define:nn {stex / inlinedef }{
4972      type    .str_set_x:N  = \sdefinitiontype,
4973      id      .str_set_x:N  = \sdefinitionid,
4974      for     .clist_set:N   = \l__stex_statements_sdefinition_for_clist ,
4975      name    .str_set_x:N  = \sdefinitionname
4976   }
4977   \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
```

```
4978    \str_clear:N \sdefinitiontype
4979    \str_clear:N \sdefinitionid
4980    \str_clear:N \sdefinitionname
4981    \clist_clear:N \l__stex_statements_sdefinition_for_clist
4982    \keys_set:nn { stex / inlinedef }{ #1 }
4983 }
4984 \NewDocumentCommand \inlinedef { O{} m } {
4985    \begingroup
4986    \__stex_statements_inlinedef_args:n{ #1 }
4987    \stex_reactivate_macro:N \definiendum
4988    \stex_reactivate_macro:N \definame
4989    \stex_reactivate_macro:N \Definame
4990    \stex_reactivate_macro:N \premise
4991    \stex_reactivate_macro:N \definiens
4992    \stex_ref_new_doc_target:n \sdefinitionid
4993    \stex_if_smsmode:TF{\stex_suppress_html:n {
4994      \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
4995    }}{
4996      \seq_clear:N \l_tmpa_seq
4997      \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4998        \tl_if_empty:nF{ ##1 }{
4999          \stex_get_symbol:n { ##1 }
5000          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5001            \l_stex_get_symbol_uri_str
5002          }
5003        }
5004      }
5005      \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpa_seq
5006      \exp_args:Nnx
5007      \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
5008        \str_if_empty:NF \sdefinitiontype {
5009          \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5010        }
5011        #2
5012        \str_if_empty:NF \sdefinitionname {
5013          \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sdefinitionname}}
5014          \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5015        }
5016      }
5017    }
5018    \endgroup
5019    \stex_smsmode_do:
5020 }
```

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

32.2 Assertions

sassertion

```
5021
5022 \keys_define:nn {stex / sassertion }{
5023    type    .str_set_x:N  = \sassertiontype,
5024    id      .str_set_x:N  = \sassertionid,
```

```
5025    title    .tl_set:N    = \sassertiontitle ,
5026    for      .clist_set:N = \l__stex_statements_sassertion_for_clist ,
5027    name     .str_set_x:N = \sassertionname
5028 }
5029 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
5030    \str_clear:N \sassertiontype
5031    \str_clear:N \sassertionid
5032    \str_clear:N \sassertionname
5033    \clist_clear:N \l__stex_statements_sassertion_for_clist
5034    \tl_clear:N \sassertiontitle
5035    \keys_set:nn { stex / sassertion }{ #1 }
5036 }
5037
5038 %\tl_new:N \g__stex_statements_aftergroup_tl
5039
5040 \NewDocumentEnvironment{sassertion}{O{}}{
5041    \__stex_statements_sassertion_args:n{ #1 }
5042    \stex_reactivate_macro:N \premise
5043    \stex_reactivate_macro:N \conclusion
5044    \stex_if_smsmode:F {
5045      \seq_clear:N \l_tmpa_seq
5046      \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5047        \tl_if_empty:nF{ ##1 }{
5048          \stex_get_symbol:n { ##1 }
5049          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5050            \l_stex_get_symbol_uri_str
5051          }
5052        }
5053      }
5054      \exp_args:Nnnx
5055      \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}
5056      \str_if_empty:NF \sassertiontype {
5057        \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
5058      }
5059      \str_if_empty:NF \sassertionname {
5060        \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
5061      }
5062      \clist_set:No \l_tmpa_clist \sassertiontype
5063      \tl_clear:N \l_tmpa_tl
5064      \clist_map_inline:Nn \l_tmpa_clist {
5065        \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
5066          \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
5067        }
5068      }
5069      \tl_if_empty:NTF \l_tmpa_tl {
5070        \__stex_statements_sassertion_start:
5071      }{
5072        \l_tmpa_tl
5073      }
5074    }
5075    \str_if_empty:NTF \sassertionid {
5076      \str_if_empty:NF \sassertionname {
5077        \stex_ref_new_doc_target:n {}
5078      }
```

206

```
5079      } {
5080        \stex_ref_new_doc_target:n \sassertionid
5081      }
5082      \stex_smsmode_do:
5083    }{
5084      \str_if_empty:NF \sassertionname {
5085        \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5086        \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5087      }
5088      \stex_if_smsmode:F {
5089        \clist_set:No \l_tmpa_clist \sassertiontype
5090        \tl_clear:N \l_tmpa_tl
5091        \clist_map_inline:Nn \l_tmpa_clist {
5092          \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
5093            \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
5094          }
5095        }
5096        \tl_if_empty:NTF \l_tmpa_tl {
5097          \__stex_statements_sassertion_end:
5098        }{
5099          \l_tmpa_tl
5100        }
5101        \end{stex_annotate_env}
5102      }
5103    }
```

```
5104
5105    \cs_new_protected:Nn \__stex_statements_sassertion_start: {
5106      \par\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {
5107        (\sassertiontitle)
5108      }~}
5109    }
5110    \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
5111
5112    \newcommand\stexpatchassertion[3][] {
5113        \str_set:Nx \l_tmpa_str{ #1 }
5114        \str_if_empty:NTF \l_tmpa_str {
5115          \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
5116          \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
5117        }{
5118          \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
5119          \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3
5120        }
5121    }
```

(*End definition for* \stexpatchassertion. *This function is documented on page 42.*)

\inlineass   inline:

```
5122    \keys_define:nn {stex / inlineass }{
5123      type    .str_set_x:N  = \sassertiontype,
5124      id      .str_set_x:N  = \sassertionid,
5125      for     .clist_set:N  = \l__stex_statements_sassertion_for_clist ,
5126      name    .str_set_x:N  = \sassertionname
```

```
5127 }
5128 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
5129   \str_clear:N \sassertiontype
5130   \str_clear:N \sassertionid
5131   \str_clear:N \sassertionname
5132   \clist_clear:N \l__stex_statements_sassertion_for_clist
5133   \keys_set:nn { stex / inlineass }{ #1 }
5134 }
5135 \NewDocumentCommand \inlineass { O{} m } {
5136   \begingroup
5137   \stex_reactivate_macro:N \premise
5138   \stex_reactivate_macro:N \conclusion
5139   \__stex_statements_inlineass_args:n{ #1 }
5140   \str_if_empty:NTF \sassertionid {
5141     \str_if_empty:NF \sassertionname {
5142       \stex_ref_new_doc_target:n {}
5143     }
5144   } {
5145     \stex_ref_new_doc_target:n \sassertionid
5146   }
5147
5148   \stex_if_smsmode:TF{
5149     \str_if_empty:NF \sassertionname {
5150       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5151       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5152     }
5153   }{
5154     \seq_clear:N \l_tmpa_seq
5155     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5156       \tl_if_empty:nF{ ##1 }{
5157         \stex_get_symbol:n { ##1 }
5158         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5159           \l_stex_get_symbol_uri_str
5160         }
5161       }
5162     }
5163     \exp_args:Nnx
5164     \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{
5165       \str_if_empty:NF \sassertiontype {
5166         \stex_annotate_invisible:nnn{typestrings}{\sassertiontype}{}
5167       }
5168       #2
5169       \str_if_empty:NF \sassertionname {
5170         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5171         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5172         \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
5173       }
5174     }
5175   }
5176   \endgroup
5177   \stex_smsmode_do:
5178 }
```

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

208

32.3 Examples

```
5179
5180 \keys_define:nn {stex / sexample }{
5181   type    .str_set_x:N  = \exampletype,
5182   id      .str_set_x:N  = \sexampleid,
5183   title   .tl_set:N     = \sexampletitle,
5184   name    .str_set_x:N  = \sexamplename ,
5185   for     .clist_set:N  = \l__stex_statements_sexample_for_clist,
5186 }
5187 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
5188   \str_clear:N \sexampletype
5189   \str_clear:N \sexampleid
5190   \str_clear:N \sexamplename
5191   \tl_clear:N \sexampletitle
5192   \clist_clear:N \l__stex_statements_sexample_for_clist
5193   \keys_set:nn { stex / sexample }{ #1 }
5194 }
5195
5196 \NewDocumentEnvironment{sexample}{O{}}{
5197   \__stex_statements_sexample_args:n{ #1 }
5198   \stex_reactivate_macro:N \premise
5199   \stex_reactivate_macro:N \conclusion
5200   \stex_if_smsmode:F {
5201     \seq_clear:N \l_tmpa_seq
5202     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5203       \tl_if_empty:nF{ ##1 }{
5204         \stex_get_symbol:n { ##1 }
5205         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5206           \l_stex_get_symbol_uri_str
5207         }
5208       }
5209     }
5210     \exp_args:Nnnx
5211     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
5212     \str_if_empty:NF \sexampletype {
5213       \stex_annotate_invisible:nnn{typestrings}{\sexampletype}{}
5214     }
5215     \str_if_empty:NF \sexamplename {
5216       \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}
5217     }
5218     \clist_set:No \l_tmpa_clist \sexampletype
5219     \tl_clear:N \l_tmpa_tl
5220     \clist_map_inline:Nn \l_tmpa_clist {
5221       \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
5222         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
5223       }
5224     }
5225     \tl_if_empty:NTF \l_tmpa_tl {
5226       \__stex_statements_sexample_start:
5227     }{
5228       \l_tmpa_tl
5229     }
```

209

```
5230        }
5231        \str_if_empty:NF \sexampleid {
5232          \stex_ref_new_doc_target:n \sexampleid
5233        }
5234        \stex_smsmode_do:
5235    }{
5236        \str_if_empty:NF \sexamplename {
5237          \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5238        }
5239        \stex_if_smsmode:F {
5240          \clist_set:No \l_tmpa_clist \sexampletype
5241          \tl_clear:N \l_tmpa_tl
5242          \clist_map_inline:Nn \l_tmpa_clist {
5243            \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
5244              \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
5245            }
5246          }
5247          \tl_if_empty:NTF \l_tmpa_tl {
5248            \__stex_statements_sexample_end:
5249          }{
5250            \l_tmpa_tl
5251          }
5252          \end{stex_annotate_env}
5253        }
5254    }
```

\stexpatchexample

```
5255
5256    \cs_new_protected:Nn \__stex_statements_sexample_start: {
5257      \par\noindent\titleemph{Example~\tl_if_empty:NF \sexampletitle {
5258        (\sexampletitle)
5259      }~}
5260    }
5261    \cs_new_protected:Nn \__stex_statements_sexample_end: {\par\medskip}
5262
5263    \newcommand\stexpatchexample[3][] {
5264        \str_set:Nx \l_tmpa_str{ #1 }
5265        \str_if_empty:NTF \l_tmpa_str {
5266          \tl_set:Nn \__stex_statements_sexample_start: { #2 }
5267          \tl_set:Nn \__stex_statements_sexample_end: { #3 }
5268        }{
5269          \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
5270          \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
5271        }
5272    }
```

(End definition for \stexpatchexample*. This function is documented on page 42.)*

\inlineex   inline:

```
5273    \keys_define:nn {stex / inlineex }{
5274      type      .str_set_x:N  = \sexampletype,
5275      id        .str_set_x:N  = \sexampleid,
5276      for       .clist_set:N  = \l__stex_statements_sexample_for_clist ,
5277      name      .str_set_x:N  = \sexamplename
```

```
5278 }
5279 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
5280   \str_clear:N \sexampletype
5281   \str_clear:N \sexampleid
5282   \str_clear:N \sexamplename
5283   \clist_clear:N \l__stex_statements_sexample_for_clist
5284   \keys_set:nn { stex / inlineex }{ #1 }
5285 }
5286 \NewDocumentCommand \inlineex { O{} m } {
5287   \begingroup
5288   \stex_reactivate_macro:N \premise
5289   \stex_reactivate_macro:N \conclusion
5290   \__stex_statements_inlineex_args:n{ #1 }
5291   \str_if_empty:NF \sexampleid {
5292     \stex_ref_new_doc_target:n \sexampleid
5293   }
5294   \stex_if_smsmode:TF{
5295     \str_if_empty:NF \sexamplename {
5296       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\examplename}}
5297     }
5298   }{
5299     \seq_clear:N \l_tmpa_seq
5300     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5301       \tl_if_empty:nF{ ##1 }{
5302         \stex_get_symbol:n { ##1 }
5303         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5304           \l_stex_get_symbol_uri_str
5305         }
5306       }
5307     }
5308     \exp_args:Nnx
5309     \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpa_seq {,}}{
5310       \str_if_empty:NF \sexampletype {
5311         \stex_annotate_invisible:nnn{typestrings}{\sexampletype}{}
5312       }
5313       #2
5314       \str_if_empty:NF \sexamplename {
5315         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5316         \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}
5317       }
5318     }
5319   }
5320   \endgroup
5321   \stex_smsmode_do:
5322 }
```

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

32.4 Logical Paragraphs

sparagraph

```
5323 \keys_define:nn { stex / sparagraph} {
5324   id       .str_set_x:N  = \sparagraphid ,
```

```
5325   title    .tl_set:N      = \l_stex_sparagraph_title_tl ,
5326   type     .str_set_x:N   = \sparagraphtype ,
5327   for      .clist_set:N   = \l__stex_statements_sparagraph_for_clist ,
5328   from     .tl_set:N      = \sparagraphfrom ,
5329   to       .tl_set:N      = \sparagraphto ,
5330   start    .tl_set:N      = \l_stex_sparagraph_start_tl ,
5331   name     .str_set:N     = \sparagraphname ,
5332   imports .tl_set:N       = \l__stex_statements_sparagraph_imports_tl
5333 }
5334
5335 \cs_new_protected:Nn \stex_sparagraph_args:n {
5336   \tl_clear:N \l_stex_sparagraph_title_tl
5337   \tl_clear:N \sparagraphfrom
5338   \tl_clear:N \sparagraphto
5339   \tl_clear:N \l_stex_sparagraph_start_tl
5340   \tl_clear:N \l__stex_statements_sparagraph_imports_tl
5341   \str_clear:N \sparagraphid
5342   \str_clear:N \sparagraphtype
5343   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5344   \str_clear:N \sparagraphname
5345   \keys_set:nn { stex / sparagraph }{ #1 }
5346 }
5347 \newif\if@in@omtext\@in@omtextfalse
5348
5349 \NewDocumentEnvironment {sparagraph} { O{} } {
5350   \stex_sparagraph_args:n { #1 }
5351   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5352     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
5353   }{
5354     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
5355   }
5356   \@in@omtexttrue
5357   \stex_if_smsmode:F {
5358     \seq_clear:N \l_tmpa_seq
5359     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5360       \tl_if_empty:nF{ ##1 }{
5361         \stex_get_symbol:n { ##1 }
5362         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5363           \l_stex_get_symbol_uri_str
5364         }
5365       }
5366     }
5367     \exp_args:Nnnx
5368     \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
5369     \str_if_empty:NF \sparagraphtype {
5370       \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
5371     }
5372     \str_if_empty:NF \sparagraphfrom {
5373       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5374     }
5375     \str_if_empty:NF \sparagraphto {
5376       \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5377     }
5378     \str_if_empty:NF \sparagraphname {
```

```
5379            \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
5380          }
5381        \clist_set:No \l_tmpa_clist \sparagraphtype
5382        \tl_clear:N \l_tmpa_tl
5383        \clist_map_inline:Nn \sparagraphtype {
5384          \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
5385            \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
5386          }
5387        }
5388        \stex_csl_to_imports:No \usemodule \l__stex_statements_sparagraph_imports_tl
5389        \tl_if_empty:NTF \l_tmpa_tl {
5390          \__stex_statements_sparagraph_start:
5391        }{
5392          \l_tmpa_tl
5393        }
5394      }
5395      \clist_set:No \l_tmpa_clist \sparagraphtype
5396      \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}
5397      {
5398        \stex_reactivate_macro:N \definiendum
5399        \stex_reactivate_macro:N \definame
5400        \stex_reactivate_macro:N \Definame
5401        \stex_reactivate_macro:N \premise
5402        \stex_reactivate_macro:N \definiens
5403      }
5404      \str_if_empty:NTF \sparagraphid {
5405        \str_if_empty:NTF \sparagraphname {
5406          \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5407            \stex_ref_new_doc_target:n {}
5408          }
5409        } {
5410          \stex_ref_new_doc_target:n {}
5411        }
5412      } {
5413        \stex_ref_new_doc_target:n \sparagraphid
5414      }
5415      \exp_args:NNx
5416      \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5417        \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5418          \tl_if_empty:nF{ ##1 }{
5419            \stex_get_symbol:n { ##1 }
5420            \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5421          }
5422        }
5423      }
5424      \stex_smsmode_do:
5425      \ignorespacesandpars
5426    }{
5427      \str_if_empty:NF \sparagraphname {
5428        \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5429        \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5430      }
5431      \stex_if_smsmode:F {
5432        \clist_set:No \l_tmpa_clist \sparagraphtype
```

```
5433        \tl_clear:N \l_tmpa_tl
5434        \clist_map_inline:Nn \l_tmpa_clist {
5435          \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
5436            \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}}
5437        }
5438      }
5439      \tl_if_empty:NTF \l_tmpa_tl {
5440        \__stex_statements_sparagraph_end:
5441      }{
5442        \l_tmpa_tl
5443      }
5444      \end{stex_annotate_env}
5445    }
5446 }
```

```
5447
5448 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
5449   \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5450     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
5451       \titleemph{\l_stex_sparagraph_title_tl}:~
5452     }
5453   }{
5454     \titleemph{\l_stex_sparagraph_start_tl}~
5455   }
5456 }
5457 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
5458
5459 \newcommand\stexpatchparagraph[3][] {
5460     \str_set:Nx \l_tmpa_str{ #1 }
5461     \str_if_empty:NTF \l_tmpa_str {
5462       \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
5463       \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
5464     }{
5465       \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
5466       \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
5467     }
5468 }
5469
5470 \keys_define:nn { stex / inlinepara} {
5471   id      .str_set_x:N  = \sparagraphid ,
5472   type    .str_set_x:N  = \sparagraphtype ,
5473   for     .clist_set:N   = \l__stex_statements_sparagraph_for_clist ,
5474   from    .tl_set:N      = \sparagraphfrom ,
5475   to      .tl_set:N      = \sparagraphto ,
5476   name    .str_set:N     = \sparagraphname
5477 }
5478 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
5479   \tl_clear:N \sparagraphfrom
5480   \tl_clear:N \sparagraphto
5481   \str_clear:N \sparagraphid
5482   \str_clear:N \sparagraphtype
5483   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5484   \str_clear:N \sparagraphname
```

```
5485    \keys_set:nn { stex / inlinepara }{ #1 }
5486 }
5487 \NewDocumentCommand \inlinepara { O{} m } {
5488    \begingroup
5489    \__stex_statements_inlinepara_args:n{ #1 }
5490    \clist_set:No \l_tmpa_clist \sparagraphtype
5491    \str_if_empty:NTF \sparagraphid {
5492       \str_if_empty:NTF \sparagraphname {
5493          \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5494             \stex_ref_new_doc_target:n {}
5495          }
5496       } {
5497          \stex_ref_new_doc_target:n {}
5498       }
5499    } {
5500       \stex_ref_new_doc_target:n \sparagraphid
5501    }
5502    \stex_if_smsmode:TF{
5503       \str_if_empty:NF \sparagraphname {
5504          \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5505          \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5506       }
5507    }{
5508       \seq_clear:N \l_tmpa_seq
5509       \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5510          \tl_if_empty:nF{ ##1 }{
5511             \stex_get_symbol:n { ##1 }
5512             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5513                \l_stex_get_symbol_uri_str
5514             }
5515          }
5516       }
5517       \exp_args:Nnx
5518       \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}{
5519          \str_if_empty:NF \sparagraphtype {
5520             \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
5521          }
5522          \str_if_empty:NF \sparagraphfrom {
5523             \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5524          }
5525          \str_if_empty:NF \sparagraphto {
5526             \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5527          }
5528          \str_if_empty:NF \sparagraphname {
5529             \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5530             \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
5531             \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5532          }
5533          \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5534             \clist_map_inline:Nn \l_tmpa_seq {
5535                \stex_ref_new_sym_target:n {##1}
5536             }
5537          }
5538          #2
```

```
5539        }
5540      }
5541    \endgroup
5542    \stex_smsmode_do:
5543 }
5544
```

(*End definition for* `\stexpatchparagraph`*. This function is documented on page* *42**.*)

```
5545 ⟨/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]

```
5546 ⟨*package⟩
5547 ⟨@@=stex_sproof⟩
5548
5549 %%%%%%%%%%%%    sproof.dtx    %%%%%%%%%%%%
5550
```

33.2 Proofs

We first define some keys for the `proof` environment.

```
5551 \keys_define:nn { stex / spf } {
5552   id          .str_set_x:N  = \spfid,
5553   for         .clist_set:N  = \l__stex_sproof_spf_for_clist ,
5554   from        .tl_set:N     = \l__stex_sproof_spf_from_tl ,
5555   proofend    .tl_set:N     = \l__stex_sproof_spf_proofend_tl,
5556   type        .str_set_x:N  = \spftype,
5557   title       .tl_set:N     = \spftitle,
5558   continues   .tl_set:N     = \l__stex_sproof_spf_continues_tl,
5559   functions   .tl_set:N     = \l__stex_sproof_spf_functions_tl,
5560   method      .tl_set:N     = \l__stex_sproof_spf_method_tl
5561 }
5562 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
5563 \str_clear:N \spfid
5564 \tl_clear:N \l__stex_sproof_spf_for_tl
5565 \tl_clear:N \l__stex_sproof_spf_from_tl
5566 \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
5567 \str_clear:N \spftype
5568 \tl_clear:N \spftitle
5569 \tl_clear:N \l__stex_sproof_spf_continues_tl
5570 \tl_clear:N \l__stex_sproof_spf_functions_tl
```

---

[8]EdNote: need an implementation for LaTeXML

```
5571  \tl_clear:N \l__stex_sproof_spf_method_tl
5572    \bool_set_false:N \l__stex_sproof_inc_counter_bool
5573  \keys_set:nn { stex / spf }{ #1 }
5574  }
```

\c__stex_sproof_flow_str We define this macro, so that we can test whether the `display` key has the value `flow`

```
5575  \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.

```
5576  \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
5577  \cs_new_protected:Npn \sproofnumber {
5578    \int_set:Nn \l_tmpa_int {1}
5579    \bool_while_do:nn {
5580      \int_compare_p:nNn {
5581        \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5582      } > 0
5583    }{
5584      \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
5585      \int_incr:N \l_tmpa_int
5586    }
5587  }
5588  \cs_new_protected:Npn \__stex_sproof_inc_counter: {
5589    \int_set:Nn \l_tmpa_int {1}
5590    \bool_while_do:nn {
5591      \int_compare_p:nNn {
5592        \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5593      } > 0
5594    }{
5595      \int_incr:N \l_tmpa_int
5596    }
5597    \int_compare:nNnF \l_tmpa_int = 1 {
5598      \int_decr:N \l_tmpa_int
5599    }
5600    \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
5601      \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1
```

---

[7]This gets the labeling right but only works 8 levels deep

```
5602        }
5603    }
5604
5605    \cs_new_protected:Npn \__stex_sproof_add_counter: {
5606      \int_set:Nn \l_tmpa_int {1}
5607      \bool_while_do:nn {
5608        \int_compare_p:nNn {
5609          \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5610        } > 0
5611      }{
5612        \int_incr:N \l_tmpa_int
5613      }
5614      \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 1 }
5615    }
5616
5617    \cs_new_protected:Npn \__stex_sproof_remove_counter: {
5618      \int_set:Nn \l_tmpa_int {1}
5619      \bool_while_do:nn {
5620        \int_compare_p:nNn {
5621          \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5622        } > 0
5623      }{
5624        \int_incr:N \l_tmpa_int
5625      }
5626      \int_decr:N \l_tmpa_int
5627      \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 0 }
5628    }
```

\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

```
5629    \def\sproof@box{
5630      \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
5631    }
5632    \def\sproofend{
5633      \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
5634        \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
5635      }
5636    }
```

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

spf@*@kw

```
5637    \def\spf@proofsketch@kw{Proof~Sketch}
5638    \def\spf@proof@kw{Proof}
5639    \def\spf@step@kw{Step}
```

(*End definition for* spf@*@kw. *This function is documented on page* **??**.)

For the other languages, we set up triggers

```
5640    \AddToHook{begindocument}{
5641      \ltx@ifpackageloaded{babel}{
5642        \makeatletter
5643        \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5644        \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5645          \input{sproof-ngerman.ldf}
```

```
5646        }
5647        \clist_if_in:NnT \l_tmpa_clist {finnish}{
5648          \input{sproof-finnish.ldf}
5649        }
5650        \clist_if_in:NnT \l_tmpa_clist {french}{
5651          \input{sproof-french.ldf}
5652        }
5653        \clist_if_in:NnT \l_tmpa_clist {russian}{
5654          \input{sproof-russian.ldf}
5655        }
5656        \makeatother
5657      }{}
5658    }
```

spfsketch

```
5659 \newcommand\spfsketch[2][]{
5660    \begingroup
5661    \let \premise \stex_proof_premise:
5662    \__stex_sproof_spf_args:n{#1}
5663    \stex_if_smsmode:TF {
5664      \str_if_empty:NF \spfid {
5665        \stex_ref_new_doc_target:n \spfid
5666      }
5667    }{
5668      \seq_clear:N \l_tmpa_seq
5669      \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5670        \tl_if_empty:nF{ ##1 }{
5671          \stex_get_symbol:n { ##1 }
5672          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5673            \l_stex_get_symbol_uri_str
5674          }
5675        }
5676      }
5677      \exp_args:Nnx
5678      \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
5679        \str_if_empty:NF \spftype {
5680          \stex_annotate_invisible:nnn{type}{\spftype}{}
5681        }
5682        \clist_set:No \l_tmpa_clist \spftype
5683        \tl_set:Nn \l_tmpa_tl {
5684          \titleemph{
5685            \tl_if_empty:NTF \spftitle {
5686              \spf@proofsketch@kw
5687            }{
5688              \spftitle
5689            }
5690          }:~
5691        }
5692        \clist_map_inline:Nn \l_tmpa_clist {
5693          \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5694            \tl_clear:N \l_tmpa_tl
5695          }
5696        }
5697        \str_if_empty:NF \spfid {
```

```
5698        \stex_ref_new_doc_target:n \spfid
5699      }
5700      \l_tmpa_tl #2 \sproofend
5701    }
5702  }
5703  \endgroup
5704  \stex_smsmode_do:
5705 }
5706
```

(*End definition for* spfsketch. *This function is documented on page* **??**.)

spfeq   This is very similar to \spfsketch, but uses a computation array[9][10]

```
5707 \newenvironment{spfeq}[2][]{
5708   \__stex_sproof_spf_args:n{#1}
5709   \let \premise \stex_proof_premise:
5710   \stex_if_smsmode:TF {
5711     \str_if_empty:NF \spfid {
5712       \stex_ref_new_doc_target:n \spfid
5713     }
5714   }{
5715     \seq_clear:N \l_tmpa_seq
5716     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5717       \tl_if_empty:nF{ ##1 }{
5718         \stex_get_symbol:n { ##1 }
5719         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5720           \l_stex_get_symbol_uri_str
5721         }
5722       }
5723     }
5724     \exp_args:Nnnx
5725     \begin{stex_annotate_env}{spfeq}{\seq_use:Nn \l_tmpa_seq {,}}
5726     \str_if_empty:NF \spftype {
5727       \stex_annotate_invisible:nnn{type}{\spftype}{}
5728     }
5729
5730     \clist_set:No \l_tmpa_clist \spftype
5731     \tl_clear:N \l_tmpa_tl
5732     \clist_map_inline:Nn \l_tmpa_clist {
5733       \tl_if_exist:cT {__stex_sproof_spfeq_##1_start:}{
5734         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_start:}}
5735       }
5736       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5737         \tl_set:Nn \l_tmpa_tl {\use:n{}}
5738       }
5739     }
5740     \tl_if_empty:NTF \l_tmpa_tl {
5741       \__stex_sproof_spfeq_start:
5742     }{
5743       \l_tmpa_tl
5744     }{~#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

```
5745      \str_if_empty:NF \spfid {
5746        \stex_ref_new_doc_target:n \spfid
5747      }
5748      \begin{displaymath}\begin{array}{rcll}
5749    }
5750    \stex_smsmode_do:
5751  }{
5752    \stex_if_smsmode:F {
5753      \end{array}\end{displaymath}
5754      \clist_set:No \l_tmpa_clist \spftype
5755      \tl_clear:N \l_tmpa_tl
5756      \clist_map_inline:Nn \l_tmpa_clist {
5757        \tl_if_exist:cT {__stex_sproof_spfeq_##1_end:}{
5758          \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_end:}}
5759        }
5760      }
5761      \tl_if_empty:NTF \l_tmpa_tl {
5762        \__stex_sproof_spfeq_end:
5763      }{
5764        \l_tmpa_tl
5765      }
5766      \end{stex_annotate_env}
5767    }
5768  }
5769
5770  \cs_new_protected:Nn \__stex_sproof_spfeq_start: {
5771    \titleemph{
5772      \tl_if_empty:NTF \spftitle {
5773        \spf@proof@kw
5774      }{
5775        \spftitle
5776      }
5777    }:
5778  }
5779  \cs_new_protected:Nn \__stex_sproof_spfeq_end: {\sproofend}
5780
5781  \newcommand\stexpatchspfeq[3][] {
5782      \str_set:Nx \l_tmpa_str{ #1 }
5783      \str_if_empty:NTF \l_tmpa_str {
5784        \tl_set:Nn \__stex_sproof_spfeq_start: { #2 }
5785        \tl_set:Nn \__stex_sproof_spfeq_end: { #3 }
5786      }{
5787        \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_start:\endcsname{ #2 }
5788        \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_end:\endcsname{ #3 }
5789      }
5790  }
5791
```

(*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.

```
5792  \newenvironment{sproof}[2][]{
```

```
5793    \let \premise \stex_proof_premise:
5794    \intarray_gzero:N \l__stex_sproof_counter_intarray
5795    \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
5796    \__stex_sproof_spf_args:n{#1}
5797    \stex_if_smsmode:TF {
5798      \str_if_empty:NF \spfid {
5799        \stex_ref_new_doc_target:n \spfid
5800      }
5801    }{
5802      \seq_clear:N \l_tmpa_seq
5803      \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5804        \tl_if_empty:nF{ ##1 }{
5805          \stex_get_symbol:n { ##1 }
5806          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5807            \l_stex_get_symbol_uri_str
5808          }
5809        }
5810      }
5811      \exp_args:Nnnx
5812      \begin{stex_annotate_env}{sproof}{\seq_use:Nn \l_tmpa_seq {,}}
5813      \str_if_empty:NF \spftype {
5814        \stex_annotate_invisible:nnn{type}{\spftype}{}
5815      }
5816
5817      \clist_set:No \l_tmpa_clist \spftype
5818      \tl_clear:N \l_tmpa_tl
5819      \clist_map_inline:Nn \l_tmpa_clist {
5820        \tl_if_exist:cT {__stex_sproof_sproof_##1_start:}{
5821          \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_start:}}
5822        }
5823        \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5824          \tl_set:Nn \l_tmpa_tl {\use:n{}}
5825        }
5826      }
5827      \tl_if_empty:NTF \l_tmpa_tl {
5828        \__stex_sproof_sproof_start:
5829      }{
5830        \l_tmpa_tl
5831      }{~#2}
5832      \str_if_empty:NF \spfid {
5833        \stex_ref_new_doc_target:n \spfid
5834      }
5835      \begin{description}
5836    }
5837    \stex_smsmode_do:
5838  }{
5839    \stex_if_smsmode:F{
5840      \end{description}
5841      \clist_set:No \l_tmpa_clist \spftype
5842      \tl_clear:N \l_tmpa_tl
5843      \clist_map_inline:Nn \l_tmpa_clist {
5844        \tl_if_exist:cT {__stex_sproof_sproof_##1_end:}{
5845          \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_end:}}
5846        }
```

```
5847        }
5848      \tl_if_empty:NTF \l_tmpa_tl {
5849        \__stex_sproof_sproof_end:
5850      }{
5851        \l_tmpa_tl
5852      }
5853      \end{stex_annotate_env}
5854    }
5855  }
5856
5857  \cs_new_protected:Nn \__stex_sproof_sproof_start: {
5858    \par\noindent\titleemph{
5859      \tl_if_empty:NTF \spftype {
5860        \spf@proof@kw
5861      }{
5862        \spftype
5863      }
5864    }:
5865  }
5866  \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
5867
5868  \newcommand\stexpatchproof[3][] {
5869    \str_set:Nx \l_tmpa_str{ #1 }
5870    \str_if_empty:NTF \l_tmpa_str {
5871      \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
5872      \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
5873    }{
5874      \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
5875      \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
5876    }
5877  }
```

\spfidea

```
5878  \newcommand\spfidea[2][]{
5879    \__stex_sproof_spf_args:n{#1}
5880    \titleemph{
5881      \tl_if_empty:NTF \spftype {Proof~Idea}{
5882        \spftype
5883      }:
5884    }~#2
5885    \sproofend
5886  }
```

(*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

```
5887  \newenvironment{spfstep}[1][]{
5888    \__stex_sproof_spf_args:n{#1}
5889    \stex_if_smsmode:TF {
```

```
5890        \str_if_empty:NF \spfid {
5891          \stex_ref_new_doc_target:n \spfid
5892        }
5893      }{
5894        \@in@omtexttrue
5895        \seq_clear:N \l_tmpa_seq
5896        \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5897          \tl_if_empty:nF{ ##1 }{
5898            \stex_get_symbol:n { ##1 }
5899            \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5900              \l_stex_get_symbol_uri_str
5901            }
5902          }
5903        }
5904        \exp_args:Nnnx
5905        \begin{stex_annotate_env}{spfstep}{\seq_use:Nn \l_tmpa_seq {,}}
5906        \str_if_empty:NF \spftype {
5907          \stex_annotate_invisible:nnn{type}{\spftype}{}
5908        }
5909        \clist_set:No \l_tmpa_clist \spftype
5910        \tl_set:Nn \l_tmpa_tl {
5911          \item[\sproofnumber]
5912          \bool_set_true:N \l__stex_sproof_inc_counter_bool
5913        }
5914        \clist_map_inline:Nn \l_tmpa_clist {
5915          \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5916            \tl_clear:N \l_tmpa_tl
5917          }
5918        }
5919        \l_tmpa_tl
5920        \tl_if_empty:NF \spftitle {
5921          {(\titleemph{\spftitle})\enspace}
5922        }
5923        \str_if_empty:NF \spfid {
5924          \stex_ref_new_doc_target:n \spfid
5925        }
5926      }
5927      \stex_smsmode_do:
5928      \ignorespacesandpars
5929    }{
5930      \bool_if:NT \l__stex_sproof_inc_counter_bool {
5931        \__stex_sproof_inc_counter:
5932      }
5933      \stex_if_smsmode:F {
5934        \end{stex_annotate_env}
5935      }
5936    }
```

sproofcomment

```
5937    \newenvironment{sproofcomment}[1][]{
5938      \__stex_sproof_spf_args:n{#1}
5939      \clist_set:No \l_tmpa_clist \spftype
5940      \tl_set:Nn \l_tmpa_tl {
5941        \item[\sproofnumber]
```

```
5942        \bool_set_true:N \l__stex_sproof_inc_counter_bool
5943      }
5944      \clist_map_inline:Nn \l_tmpa_clist {
5945        \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5946          \tl_clear:N \l_tmpa_tl
5947        }
5948      }
5949      \l_tmpa_tl
5950    }{
5951      \bool_if:NT \l__stex_sproof_inc_counter_bool {
5952        \__stex_sproof_inc_counter:
5953      }
5954    }
```

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.

```
5955 \newenvironment{subproof}[2][]{
5956    \__stex_sproof_spf_args:n{#1}
5957    \stex_if_smsmode:TF{
5958      \str_if_empty:NF \spfid {
5959        \stex_ref_new_doc_target:n \spfid
5960      }
5961    }{
5962      \seq_clear:N \l_tmpa_seq
5963      \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5964        \tl_if_empty:nF{ ##1 }{
5965          \stex_get_symbol:n { ##1 }
5966          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5967            \l_stex_get_symbol_uri_str
5968          }
5969        }
5970      }
5971      \exp_args:Nnnx
5972      \begin{stex_annotate_env}{subproof}{\seq_use:Nn \l_tmpa_seq {,}}
5973      \str_if_empty:NF \spftype {
5974        \stex_annotate_invisible:nnn{type}{\spftype}{}
5975      }
5976
5977      \clist_set:No \l_tmpa_clist \spftype
5978      \tl_set:Nn \l_tmpa_tl {
5979        \item[\sproofnumber]
5980        \bool_set_true:N \l__stex_sproof_inc_counter_bool
5981      }
5982      \clist_map_inline:Nn \l_tmpa_clist {
5983        \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5984          \tl_clear:N \l_tmpa_tl
5985        }
5986      }
5987      \l_tmpa_tl
5988      \tl_if_empty:NF \spftitle {
5989        {(\titleemph{\spftitle})\enspace}
5990      }
```

```
5991        {~#2}
5992        \str_if_empty:NF \spfid {
5993          \stex_ref_new_doc_target:n \spfid
5994        }
5995      }
5996      \__stex_sproof_add_counter:
5997      \stex_smsmode_do:
5998    }{
5999      \__stex_sproof_remove_counter:
6000      \bool_if:NT \l__stex_sproof_inc_counter_bool {
6001        \__stex_sproof_inc_counter:
6002      }
6003      \stex_if_smsmode:F{
6004        \end{stex_annotate_env}
6005      }
6006    }
```

spfcases   In the `pfcases` environment, the start text is displayed as the first comment of the proof.

```
6007    \newenvironment{spfcases}[2][]{
6008      \tl_if_empty:nTF{#1}{
6009        \begin{subproof}[method=by-cases]{#2}
6010      }{
6011        \begin{subproof}[#1,method=by-cases]{#2}
6012      }
6013    }{
6014      \end{subproof}
6015    }
```

spfcase   In the `pfcase` environment, the start text is displayed specification of the case after the
           `\item`

```
6016    \newenvironment{spfcase}[2][]{
6017      \__stex_sproof_spf_args:n{#1}
6018      \stex_if_smsmode:TF {
6019        \str_if_empty:NF \spfid {
6020          \stex_ref_new_doc_target:n \spfid
6021        }
6022      }{
6023        \seq_clear:N \l_tmpa_seq
6024        \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6025          \tl_if_empty:nF{ ##1 }{
6026            \stex_get_symbol:n { ##1 }
6027            \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6028              \l_stex_get_symbol_uri_str
6029            }
6030          }
6031        }
6032        \exp_args:Nnnx
6033        \begin{stex_annotate_env}{spfcase}{\seq_use:Nn \l_tmpa_seq {,}}
6034        \str_if_empty:NF \spftype {
6035          \stex_annotate_invisible:nnn{type}{\spftype}{}
6036        }
6037        \clist_set:No \l_tmpa_clist \spftype
6038        \tl_set:Nn \l_tmpa_tl {
6039          \item[\sproofnumber]
```

227

```
6040        \bool_set_true:N \l__stex_sproof_inc_counter_bool
6041      }
6042      \clist_map_inline:Nn \l_tmpa_clist {
6043        \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6044          \tl_clear:N \l_tmpa_tl
6045        }
6046      }
6047      \l_tmpa_tl
6048      \tl_if_empty:nF{#2}{
6049        \titleemph{#2}:~
6050      }
6051    }
6052    \__stex_sproof_add_counter:
6053    \stex_smsmode_do:
6054  }{
6055    \__stex_sproof_remove_counter:
6056    \bool_if:NT \l__stex_sproof_inc_counter_bool {
6057      \__stex_sproof_inc_counter:
6058    }
6059    \stex_if_smsmode:F{
6060      \clist_set:No \l_tmpa_clist \spftype
6061      \tl_set:Nn \l_tmpa_tl{\sproofend}
6062      \clist_map_inline:Nn \l_tmpa_clist {
6063        \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6064          \tl_clear:N \l_tmpa_tl
6065        }
6066      }
6067      \l_tmpa_tl
6068      \end{stex_annotate_env}
6069    }
6070  }
```

spfcase   similar to `spfcase`, takes a third argument.

```
6071  \newcommand\spfcasesketch[3][]{
6072    \begin{spfcase}[#1]{#2}#3\end{spfcase}
6073  }
```

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.

```
6074  \keys_define:nn { stex / just }{
6075    id        .str_set_x:N  = \l__stex_sproof_just_id_str,
6076    method    .tl_set:N     = \l__stex_sproof_just_method_tl,
6077    premises  .tl_set:N     = \l__stex_sproof_just_premises_tl,
6078    args      .tl_set:N     = \l__stex_sproof_just_args_tl
6079  }
```

The next three environments and macros are purely semantic, so we ignore the keyval
EdN:11   arguments for now and only display the content.[11]

---

[11]EDNOTE: need to do something about the premise in draft mode.

justification

6080 `\newenvironment{justification}[1][]{}{}`

\premise

6081 `\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.

6082 `\newcommand\justarg[2][]{#2}`
6083 ⟨/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

```
6084  ⟨∗package⟩
6085
6086  %%%%%%%%%%%%   others.dtx   %%%%%%%%%%%%
6087
6088  ⟨@@=stex_others⟩
```

Warnings and error messages
```
6089    % None
```

`\MSC` Math subject classifier

```
6090  \NewDocumentCommand \MSC {m} {
6091    % TODO
6092  }
```

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

Patching tikzinput, if loaded

```
6093  \@ifpackageloaded{tikzinput}{
6094    \RequirePackage{stex-tikzinput}
6095  }{}
6096
6097  \bool_if:NT \c_stex_persist_mode_bool {
6098    \input{\jobname.sms}
6099    \prop_if_exist:NT\c_stex_mathhub_main_manifest_prop{
6100      \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
6101        \l_tmpa_str
6102      \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str _manifest_prop }
6103        \c_stex_mathhub_main_manifest_prop
6104      \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
6105    }
6106  }
6107  ⟨/package⟩
```

Chapter 35

sTeX
-Metatheory Implementation

```
6108  ⟨∗package⟩
6109  ⟨@@=stex_modules⟩
6110
6111  %%%%%%%%%%%%   metatheory.dtx   %%%%%%%%%%%%
6112
6113  \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX/meta}
6114  \begingroup
6115  \stex_module_setup:nn{
6116    ns=\c_stex_metatheory_ns_str,
6117    meta=NONE
6118  }{Metatheory}
6119  \stex_reactivate_macro:N \symdecl
6120  \stex_reactivate_macro:N \notation
6121  \stex_reactivate_macro:N \symdef
6122  \ExplSyntaxOff
6123  \csname stex_suppress_html:n\endcsname{
6124    % is-a (a:A, a \in A, a is an A, etc.)
6125    \symdecl{isa}[args=ai]
6126    \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
6127    \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
6128    \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
6129
6130    % bind (\forall, \Pi, \lambda etc.)
6131    \symdecl{bind}[args=Bi]
6132    \notation{bind}[forall]{\comp\forall #1.\;#2}{##1 \comp, ##2}
6133    \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6134    \notation{bind}[depfun]{\comp( #1 \comp)\;\to\;} #2}{##1 \comp, ##2}
6135
6136    % implicit bind
6137    \symdef{implicitbind}[args=Bi]{\comp\prod_{#1}#2}{##1\comp,##2}
6138
6139    % dummy variable
6140    \symdecl{dummyvar}
6141    \notation{dummyvar}[underscore]{\comp\_}
6142    \notation{dummyvar}[dot]{\comp\cdot}
```

```
6143    \notation{dummyvar}[dash]{\comp{{\rm --}}}

6144

6145    %fromto (function space, Hom-set, implication etc.)
6146    \symdecl{fromto}[args=ai]
6147    \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
6148    \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}

6149

6150    % mapto (lambda etc.)
6151    %\symdecl{mapto}[args=Bi]
6152    %\notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
6153    %\notation{mapto}[lambda]{\comp\lambda #1 \comp.\; #2}{#1 \comp, #2}
6154    %\notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.\; #2}{#1 \comp, #2}

6155

6156    % function/operator application
6157    \symdecl{apply}[args=ia]
6158    \notation{apply}[prec=0;0x\infprec,parens]{#1 \comp( #2 \comp)}{##1 \comp, ##2}
6159    \notation{apply}[prec=0;0x\infprec,lambda]{#1 \; #2 }{##1 \; ##2}

6160

6161    % collection of propositions/booleans/truth values
6162    \symdecl{prop}[name=proposition]
6163    \notation{prop}[prop]{\comp{{\rm prop}}}
6164    \notation{prop}[BOOL]{\comp{{\rm BOOL}}}

6165

6166    \symdecl{judgmentholds}[args=1]
6167    \notation{judgmentholds}[vdash,op=\vdash]{\comp\vdash\; #1}

6168

6169    % sequences
6170    \symdecl{seqtype}[args=1]
6171    \notation{seqtype}[kleene]{#1^{\comp\ast}}

6172

6173    \symdecl{seqexpr}[args=a]
6174    \notation{seqexpr}[angle,prec=nobrackets]{\comp\langle #1\comp\rangle}{##1\comp,##2}

6175

6176    \symdef{sequence-index}[args=2,li,prec=nobrackets]{{#1}_{#2}}
6177    \notation{sequence-index}[ui,prec=nobrackets]{{#1}^{#2}}

6178

6179    \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{,\ellipses}}{##1\comp,##2}
6180    \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{,\ellipses,}#2}{##1\comp,##2}
6181    \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{,\ellipses,}#2\comp{,\ellipses,}#

6182

6183    % letin (``let'', local definitions, variable substitution)
6184    \symdecl{letin}[args=bii]
6185    \notation{letin}[let]{\comp{{\rm let}}\;#1\comp{=}#2\;\comp{{\rm in}}\;#3}
6186    \notation{letin}[subst]{#3 \comp[ #1 \comp/ #2 \comp]}
6187    \notation{letin}[frac]{#3 \comp[ \frac{#2}{#1} \comp]}

6188

6189    % structures
6190    \symdecl*{module-type}[args=1]
6191    \notation{module-type}{\comp{\mathtt{MOD}} #1}
6192    \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
6193    \notation{mathstruct}[angle,prec=nobrackets]{\comp\langle #1 \comp\rangle}{##1 \comp, ##2}

6194

6195    % objects
6196    \symdecl{object}
```

```
6197    \notation{object}{\comp{\mathtt{OBJECT}}}

6198

6199 }
6200    \ExplSyntaxOn
6201    \stex_add_to_current_module:n{
6202      \let\nappa\apply
6203      \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
6204      \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
6205      \def\livar{\csname sequence-index\endcsname[li]}
6206      \def\uivar{\csname sequence-index\endcsname[ui]}
6207      \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
6208      \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
6209      \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
6210    }
6211 \__stex_modules_end_module:
6212 \endgroup

6213 ⟨/package⟩
```

Chapter 36

Tikzinput Implementation

```
6214 ⟨@@=tikzinput⟩
6215 ⟨∗package⟩
6216
6217 %%%%%%%%%%%%%    tikzinput.dtx    %%%%%%%%%%%%%
6218
6219 \ProvidesExplPackage{tikzinput}{2022/02/26}{3.0.1}{tikzinput package}
6220 \RequirePackage{l3keys2e}
6221
6222 \keys_define:nn { tikzinput } {
6223   image    .bool_set:N   = \c_tikzinput_image_bool,
6224   image    .default:n    = false ,
6225   unknown    .code:n       = {}
6226 }
6227
6228 \ProcessKeysOptions { tikzinput }
6229
6230 \bool_if:NTF \c_tikzinput_image_bool {
6231   \RequirePackage{graphicx}
6232
6233   \providecommand\usetikzlibrary[]{}
6234   \newcommand\tikzinput[2][]{\includegraphics[#1]{#2}}
6235 }{
6236   \RequirePackage{tikz}
6237   \RequirePackage{standalone}
6238
6239   \newcommand \tikzinput [2] [] {
6240     \setkeys{Gin}{#1}
6241     \ifx \Gin@ewidth \Gin@exclamation
6242       \ifx \Gin@eheight \Gin@exclamation
6243         \input { #2 }
6244       \else
6245         \resizebox{!}{ \Gin@eheight }{
6246           \input { #2 }
6247         }
6248       \fi
6249     \else
6250       \ifx \Gin@eheight \Gin@exclamation
6251         \resizebox{ \Gin@ewidth }{!}{
```

```
6252          \input { #2 }
6253        }
6254      \else
6255        \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
6256          \input { #2 }
6257        }
6258      \fi
6259    \fi
6260   }
6261 }
6262
6263 \newcommand \ctikzinput [2] [] {
6264   \begin{center}
6265     \tikzinput [#1] {#2}
6266   \end{center}
6267 }
6268
6269 \@ifpackageloaded{stex}{
6270   \RequirePackage{stex-tikzinput}
6271 }{}
6272
6273 ⟨/package⟩
6274 ⟨*stex⟩
6275 \ProvidesExplPackage{stex-tikzinput}{2022/02/26}{3.0.1}{stex-tikzinput}
6276 \RequirePackage{stex}
6277 \RequirePackage{tikzinput}
6278
6279 \newcommand\mhtikzinput[2][]{%
6280   \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
6281   \stex_in_repository:nn\Gin@mhrepos{
6282     \tikzinput[#1]{\mhpath{##1}{#2}}
6283   }
6284 }
6285 \newcommand\cmhtikzinput[2][]{\begin{center}\mhtikzinput[#1]{#2}\end{center}}
6286
6287 \cs_new_protected:Nn \__tikzinput_usetikzlibrary:nn {
6288   \pgfkeys@spdef\pgf@temp{#1}
6289   \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
6290   \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgfut
6291   \expandafter\edef\csname tikz@library@#1atcode\endcsname{\the\catcode`\@}
6292   \expandafter\edef\csname tikz@library@#1barcode\endcsname{\the\catcode`\|}
6293   \expandafter\edef\csname tikz@library@#1dollarcode\endcsname{\the\catcode`\$}
6294   \catcode`\@=11
6295   \catcode`\|=12
6296   \catcode`\$=3
6297   \pgfutil@InputIfFileExists{#2}{}{}
6298   \catcode`\@=\csname tikz@library@#1atcode\endcsname
6299   \catcode`\|=\csname tikz@library@#1barcode\endcsname
6300   \catcode`\$=\csname tikz@library@#1dollarcode\endcsname
6301 }
6302
6303
6304 \newcommand\libusetikzlibrary[1]{
```

```
6305    \prop_if_exist:NF \l_stex_current_repository_prop {
6306      \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6307    }
6308    \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
6309      \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6310    }
6311    \seq_clear:N \l__tikzinput_libinput_files_seq
6312    \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
6313    \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
6314
6315    \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
6316      \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / tikzlibra
6317      \IfFileExists{ \l_tmpa_str }{
6318        \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6319      }{}
6320      \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
6321      \seq_put_right:No \l_tmpa_seq \l_tmpa_str
6322    }
6323
6324    \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / tikzlibrary #1 .code.t
6325    \IfFileExists{ \l_tmpa_str }{
6326      \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6327    }{}
6328
6329    \seq_if_empty:NTF \l__tikzinput_libinput_files_seq {
6330      \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .code.t
6331    }{
6332      \int_compare:nNnTF {\seq_count:N \l__tikzinput_libinput_files_seq} = 1 {
6333        \seq_map_inline:Nn \l__tikzinput_libinput_files_seq {
6334          \__tikzinput_usetikzlibrary:nn{#1}{ ##1 }
6335        }
6336      }{
6337        \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .co
6338      }
6339    }
6340  }
6341  ⟨/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

⟨6342⟩ ⟨∗package⟩
⟨6343⟩ ⟨@@=document_structure⟩
⟨6344⟩ \ProvidesExplPackage{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure}
⟨6345⟩ \RequirePackage{l3keys2e}

37.1 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).

```
6346
6347 \keys_define:nn{ document-structure }{
6348   class       .str_set_x:N  = \c_document_structure_class_str,
6349   topsect     .str_set_x:N  = \c_document_structure_topsect_str,,
6350   unknown     .code:n       = {
6351     \PassOptionsToClass{\CurrentOption}{stex}
6352     \PassOptionsToClass{\CurrentOption}{tikzinput}
6353   }
6354 % showignores .bool_set:N  = \c_document_structure_showignores_bool,
6355 }
6356 \ProcessKeysOptions{ document-structure }
6357 \str_if_empty:NT \c_document_structure_class_str {
6358   \str_set:Nn \c_document_structure_class_str {article}
6359 }
6360 \str_if_empty:NT \c_document_structure_topsect_str {
6361   \str_set:Nn \c_document_structure_topsect_str {section}
6362 }
```

Then we need to set up the packages by requiring the sref package to be loaded, and set up triggers for other languages

```
6363 \RequirePackage{xspace}
6364 \RequirePackage{comment}
6365 \RequirePackage{stex}
6366 \AddToHook{begindocument}{
```

```
6367   \ltx@ifpackageloaded{babel}{
6368       \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6369       \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6370           \makeatletter\input{document-structure-ngerman.ldf}\makeatother
6371       }
6372   }{}
6373 }
```

\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.

```
6374 \int_new:N \l_document_structure_section_level_int
6375 \str_case:VnF \c_document_structure_topsect_str {
6376   {part}{
6377       \int_set:Nn \l_document_structure_section_level_int {0}
6378   }
6379   {chapter}{
6380       \int_set:Nn \l_document_structure_section_level_int {1}
6381   }
6382 }{
6383   \str_case:VnF \c_document_structure_class_str {
6384     {book}{
6385         \int_set:Nn \l_document_structure_section_level_int {0}
6386     }
6387     {report}{
6388         \int_set:Nn \l_document_structure_section_level_int {0}
6389     }
6390   }{
6391       \int_set:Nn \l_document_structure_section_level_int {2}
6392   }
6393 }
```

37.2   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

EdN:12   element of class `omdoc_currentsectionlevel`, wich will be instantiated by CSS later.[12]

```
6394 \def\current@section@level{document}%
6395 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
6396 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

\skipomgroup

```
6397 \cs_new_protected:Npn \skipomgroup {
```

---

[12]EDNOTE: MK: we may have to experiment with the more powerful uppercasing macro from `mfirstuc.sty` once we internationalize.

```
6398      \ifcase\l_document_structure_section_level_int
6399      \or\stepcounter{part}
6400      \or\stepcounter{chapter}
6401      \or\stepcounter{section}
6402      \or\stepcounter{subsection}
6403      \or\stepcounter{subsubsection}
6404      \or\stepcounter{paragraph}
6405      \or\stepcounter{subparagraph}
6406      \fi
6407 }
```

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

blindfragment

```
6408 \newcommand\at@begin@blindomgroup[1]{}
6409 \newenvironment{blindfragment}
6410 {
6411    \int_incr:N\l_document_structure_section_level_int
6412    \at@begin@blindomgroup\l_document_structure_section_level_int
6413 }{}
```

\omgroup@nonum   convenience macro: \omgroup@nonum{⟨*level*⟩}{⟨*title*⟩} makes an unnumbered sectioning with title ⟨*title*⟩ at level ⟨*level*⟩.

```
6414 \newcommand\omgroup@nonum[2]{
6415    \ifx\hyper@anchor\@undefined\else\phantomsection\fi
6416    \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
6417 }
```

(*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.

```
6418 \newcommand\omgroup@num[2]{
6419    \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
6420       \@nameuse{#1}{#2}
6421    }{
6422       \cs_if_exist:NTF\rdfmeta@sectioning{
6423          \@nameuse{rdfmeta@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
6424       }{
6425          \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
6426       }
6427    }
6428 %\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\omgroup@id
6429 }
```

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

sfragment

```
6430 \keys_define:nn { document-structure / omgroup }{
6431    id           .str_set_x:N = \l__document_structure_omgroup_id_str,
6432    date         .str_set_x:N = \l__document_structure_omgroup_date_str,
6433    creators     .clist_set:N = \l__document_structure_omgroup_creators_clist,
```

```
6434    contributors  .clist_set:N = \l__document_structure_omgroup_contributors_clist,
6435    srccite       .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
6436    type          .tl_set:N    = \l__document_structure_omgroup_type_tl,
6437    short         .tl_set:N    = \l__document_structure_omgroup_short_tl,
6438    display       .tl_set:N    = \l__document_structure_omgroup_display_tl,
6439    intro         .tl_set:N    = \l__document_structure_omgroup_intro_tl,
6440    imports       .tl_set:N    = \l__document_structure_omgroup_imports_tl,
6441    loadmodules   .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
6442 }
6443 \cs_new_protected:Nn \__document_structure_omgroup_args:n {
6444    \str_clear:N \l__document_structure_omgroup_id_str
6445    \str_clear:N \l__document_structure_omgroup_date_str
6446    \clist_clear:N \l__document_structure_omgroup_creators_clist
6447    \clist_clear:N \l__document_structure_omgroup_contributors_clist
6448    \tl_clear:N \l__document_structure_omgroup_srccite_tl
6449    \tl_clear:N \l__document_structure_omgroup_type_tl
6450    \tl_clear:N \l__document_structure_omgroup_short_tl
6451    \tl_clear:N \l__document_structure_omgroup_display_tl
6452    \tl_clear:N \l__document_structure_omgroup_imports_tl
6453    \tl_clear:N \l__document_structure_omgroup_intro_tl
6454    \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
6455    \keys_set:nn { document-structure / omgroup } { #1 }
6456 }
```

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.

```
6457 \newif\if@mainmatter\@mainmattertrue
6458 \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.

```
6459 \keys_define:nn { document-structure / sectioning }{
6460    name    .str_set_x:N  = \l__document_structure_sect_name_str    ,
6461    ref     .str_set_x:N  = \l__document_structure_sect_ref_str     ,
6462    clear   .bool_set:N   = \l__document_structure_sect_clear_bool ,
6463    clear   .default:n    = {true}                  ,
6464    num     .bool_set:N   = \l__document_structure_sect_num_bool    ,
6465    num     .default:n    = {true}
6466 }
6467 \cs_new_protected:Nn \__document_structure_sect_args:n {
6468    \str_clear:N \l__document_structure_sect_name_str
6469    \str_clear:N \l__document_structure_sect_ref_str
6470    \bool_set_false:N \l__document_structure_sect_clear_bool
6471    \bool_set_false:N \l__document_structure_sect_num_bool
6472    \keys_set:nn { document-structure / sectioning } { #1 }
6473 }
6474 \newcommand\omdoc@sectioning[3][]{
6475    \__document_structure_sect_args:n {#1 }
6476    \let\omdoc@sect@name\l__document_structure_sect_name_str
6477    \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
6478    \if@mainmatter% numbering not overridden by frontmatter, etc.
6479      \bool_if:NTF \l__document_structure_sect_num_bool {
6480        \omgroup@num{#2}{#3}
```

240

```
6481       }{
6482         \omgroup@nonum{#2}{#3}
6483       }
6484       \def\current@section@level{\omdoc@sect@name}
6485     \else
6486       \omgroup@nonum{#2}{#3}
6487     \fi
6488 }% 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.

```
6489 \newcommand\omgroup@redefine@addtocontents[1]{%
6490 %\edef\__document_structureimport{#1}%
6491 %\@for\@I:=\__document_structureimport\do{%
6492 %\edef\@path{\csname module@\@I  @path\endcsname}%
6493 %\@ifundefined{tf@toc}\relax%
6494 %     {\protected@write\tf@toc{}{\string\@requiremodules{\@path}}}}}
6495 %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
6496 %\def\addcontentsline##1##2##3{%
6497 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}
6498 %\else% hyperref.sty not loaded
6499 %\def\addcontentsline##1##2##3{%
6500 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}{
6501 %\fi
6502 }% 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.

```
6503 \newenvironment{sfragment}[2][]% keys, title
6504 {
6505     \__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.

```
6506     \stex_csl_to_imports:No \usemodule \l__document_structure_omgroup_imports_tl
6507
6508     \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
6509       \omgroup@redefine@addtocontents{
6510         %\@ifundefined{module@id}\used@modules%
6511         %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}
6512       }
6513     }
```

now we only need to construct the right sectioning depending on the value of `\section@level`.

```
6514     \int_incr:N\l_document_structure_section_level_int
6515     \ifcase\l_document_structure_section_level_int
6516       \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
6517       \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
6518       \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
6519       \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
6520       \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
6521       \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#
6522       \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{paragr
```

```
6523    \fi
6524    \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
6525    \str_if_empty:NF \l__document_structure_omgroup_id_str {
6526      \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
6527    }
6528 }% for customization
6529 {}
```

and finally, we localize the sections

```
6530 \newcommand\omdoc@part@kw{Part}
6531 \newcommand\omdoc@chapter@kw{Chapter}
6532 \newcommand\omdoc@section@kw{Section}
6533 \newcommand\omdoc@subsection@kw{Subsection}
6534 \newcommand\omdoc@subsubsection@kw{Subsubsection}
6535 \newcommand\omdoc@paragraph@kw{paragraph}
6536 \newcommand\omdoc@subparagraph@kw{subparagraph}
```

37.3   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

```
6537 \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).

```
6538 \cs_if_exist:NTF\frontmatter{
6539    \let\__document_structure_orig_frontmatter\frontmatter
6540    \let\frontmatter\relax
6541 }{
6542    \tl_set:Nn\__document_structure_orig_frontmatter{
6543      \clearpage
6544      \@mainmatterfalse
6545      \pagenumbering{roman}
6546    }
6547 }
6548 \cs_if_exist:NTF\backmatter{
6549    \let\__document_structure_orig_backmatter\backmatter
6550    \let\backmatter\relax
6551 }{
6552    \tl_set:Nn\__document_structure_orig_backmatter{
6553      \clearpage
6554      \@mainmatterfalse
6555      \pagenumbering{roman}
6556    }
6557 }
```

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.

```
6558 \newenvironment{frontmatter}{
6559   \__document_structure_orig_frontmatter
6560 }{
6561   \cs_if_exist:NTF\mainmatter{
6562     \mainmatter
6563   }{
6564     \clearpage
6565     \@mainmattertrue
6566     \pagenumbering{arabic}
6567   }
6568 }
```

backmatter As backmatter is at the end of the document, we do nothing for `\endbackmatter`.

```
6569 \newenvironment{backmatter}{
6570   \__document_structure_orig_backmatter
6571 }{
6572   \cs_if_exist:NTF\mainmatter{
6573     \mainmatter
6574   }{
6575     \clearpage
6576     \@mainmattertrue
6577     \pagenumbering{arabic}
6578   }
6579 }
```

finally, we make sure that page numbering is arabic and we have main matter as the default

```
6580 \@mainmattertrue\pagenumbering{arabic}
```

\prematurestop We initialize `\afterprematurestop`, and provide `\prematurestop@endomgroup` which looks up `\omgroup@level` and recursively ends enough `{sfragment}`s.

```
6581 \def \c__document_structure_document_str{document}
6582 \newcommand\afterprematurestop{}
6583 \def\prematurestop@endomgroup{
6584   \unless\ifx\@currenvir\c__document_structure_document_str
6585     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafte
6586     \expandafter\prematurestop@endomgroup
6587   \fi
6588 }
6589 \providecommand\prematurestop{
6590   \message{Stopping~sTeX~processing~prematurely}
6591   \prematurestop@endomgroup
6592   \afterprematurestop
6593   \end{document}
6594 }
```

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

37.4 Global Variables

\setSGvar   set a global variable

```
6595 \RequirePackage{etoolbox}
6596 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}
```

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

\useSGvar   use a global variable

```
6597 \newrobustcmd\useSGvar[1]{%
6598   \@ifundefined{sTeX@Gvar@#1}
6599   {\PackageError{document-structure}
6600     {The sTeX Global variable #1 is undefined}
6601     {set it with \protect\setSGvar}}
6602 \@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.

```
6603 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
6604   \@ifundefined{sTeX@Gvar@#1}
6605   {\PackageError{document-structure}
6606     {The sTeX Global variable #1 is undefined}
6607     {set it with \protect\setSGvar}}
6608   {\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.

```
6609 ⟨∗cls⟩
6610 ⟨@@=notesslides⟩
6611 \ProvidesExplClass{notesslides}{2022/02/28}{3.1.0}{notesslides Class}
6612 \RequirePackage{l3keys2e}
6613
6614 \keys_define:nn{notesslides / cls}{
6615   class   .str_set_x:N = \c__notesslides_class_str,
6616   notes   .bool_set:N  = \c__notesslides_notes_bool ,
6617   slides  .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
6618   docopt  .str_set_x:N = \c__notesslides_docopt_str,
6619   unknown .code:n      = {
6620     \PassOptionsToPackage{\CurrentOption}{document-structure}
6621     \PassOptionsToClass{\CurrentOption}{beamer}
6622     \PassOptionsToPackage{\CurrentOption}{notesslides}
6623   }
6624 }
6625 \ProcessKeysOptions{ notesslides / cls }
6626
6627
6628 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{book}{
6629   \PassOptionsToPackage{defaulttopsec=part}{notesslides}
6630 }
6631 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{report}{
6632   \PassOptionsToPackage{defaulttopsec=part}{notesslides}
6633 }
6634
6635
6636
6637
6638 \bool_if:NTF \c__notesslides_notes_bool {
6639   \PassOptionsToPackage{notes=true}{notesslides}
```

```
6640 }{
6641     \PassOptionsToPackage{notes=false}{notesslides}
6642 }
6643 ⟨/cls⟩
```

now we do the same for the `notesslides` package.

```
6644 ⟨*package⟩
6645 \ProvidesExplPackage{notesslides}{2022/02/28}{3.1.0}{notesslides Package}
6646 \RequirePackage{l3keys2e}
6647
6648 \keys_define:nn{notesslides / pkg}{
6649     topsect         .str_set_x:N  = \c__notesslides_topsect_str,
6650     defaulttopsect  .str_set_x:N  = \c__notesslides_defaulttopsec_str,
6651     notes           .bool_set:N   = \c__notesslides_notes_bool ,
6652     slides          .code:n       = { \bool_set_false:N \c__notesslides_notes_bool },
6653     sectocframes    .bool_set:N   = \c__notesslides_sectocframes_bool ,
6654     frameimages     .bool_set:N   = \c__notesslides_frameimages_bool ,
6655     fiboxed         .bool_set:N   = \c__notesslides_fiboxed_bool ,
6656     noproblems      .bool_set:N   = \c__notesslides_noproblems_bool,
6657     unknown         .code:n       = {
6658         \PassOptionsToClass{\CurrentOption}{stex}
6659         \PassOptionsToClass{\CurrentOption}{tikzinput}
6660     }
6661 }
6662 \ProcessKeysOptions{ notesslides / pkg }
6663 \newif\ifnotes
6664 \bool_if:NTF \c__notesslides_notes_bool {
6665     \notestrue
6666 }{
6667     \notesfalse
6668 }
6669
```

we give ourselves a macro `\@@topsect` that needs only be evaluated once, so that the `\ifdefstring` conditionals work below.

```
6670 \str_if_empty:NTF \c__notesslides_topsect_str {
6671     \str_set_eq:NN \__notesslidestopsect \c__notesslides_defaulttopsec_str
6672 }{
6673     \str_set_eq:NN \__notesslidestopsect \c__notesslides_topsect_str
6674 }
6675 ⟨/package⟩
```

Depending on the options, we either load the `article`-based `document-structure` or the `beamer` class (and set some counters).

```
6676 ⟨*cls⟩
6677 \bool_if:NTF \c__notesslides_notes_bool {
6678     \str_if_empty:NT \c__notesslides_class_str {
6679         \str_set:Nn \c__notesslides_class_str {article}
6680     }
6681     \exp_after:wN\LoadClass\exp_after:wN[\c__notesslides_docopt_str]
6682         {\c__notesslides_class_str}
6683 }{
6684     \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
6685     \newcounter{Item}
6686     \newcounter{paragraph}
```

246

```
6687    \newcounter{subparagraph}
6688    \newcounter{Hfootnote}
6689  }
6690  \RequirePackage{document-structure}
```

now it only remains to load the `notesslides` package that does all the rest.

```
6691  \RequirePackage{notesslides}
6692  ⟨/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.

```
6693  ⟨*package⟩
6694  \bool_if:NT \c__notesslides_notes_bool {
6695    \RequirePackage{a4wide}
6696    \RequirePackage{marginnote}
6697    \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
6698    \RequirePackage{mdframed}
6699    \RequirePackage[noxcolor,noamsthm]{beamerarticle}
6700    \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
6701  }
6702  \RequirePackage{stex-tikzinput}
6703  \RequirePackage{etoolbox}
6704  \RequirePackage{amssymb}
6705  \RequirePackage{amsmath}
6706  \RequirePackage{comment}
6707  \RequirePackage{textcomp}
6708  \RequirePackage{url}
6709  \RequirePackage{graphicx}
6710  \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.[13]

```
6711  \bool_if:NT \c__notesslides_notes_bool {
6712    \renewcommand\usetheme[2][]{\usepackage[#1]{beamernotestheme#2}}
6713  }
6714
6715
6716  \NewDocumentCommand \libusetheme {O{} m} {
6717    \bool_if:NTF \c__notesslides_notes_bool {
6718      \libusepackage[#1]{beamernotestheme#2}
6719    }{
6720    \libusepackage[#1]{beamertheme#2}
6721    }
6722  }
```

---

[13]EDNOTE: MK: This is not ideal, but I am not sure that I want to be able to provide the full theme functionality there.

We define the sizes of slides in the notes. Somehow, we cannot get by with the same here.

```
6723  \newcounter{slide}
6724  \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
6725  \newlength{\slideheight}\setlength{\slideheight}{9cm}
```

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.

```
6726  \bool_if:NTF \c__notesslides_notes_bool {
6727    \renewenvironment{note}{\ignorespaces}{}
6728  }{
6729    \excludecomment{note}
6730  }
```

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.

```
6731  \bool_if:NT \c__notesslides_notes_bool {
6732    \newlength{\slideframewidth}
6733    \setlength{\slideframewidth}{1.5pt}
```

frame We first define the keys.

```
6734  \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
6735    \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
6736      \bool_set_true:N #1
6737    }{
6738      \bool_set_false:N #1
6739    }
6740  }
6741  \keys_define:nn{notesslides / frame}{
6742    label                .str_set_x:N  = \l__notesslides_frame_label_str,
6743    allowframebreaks   .code:n       = {
6744      \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
6745    },
6746    allowdisplaybreaks  .code:n       = {
6747      \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
6748    },
6749    fragile            .code:n       = {
6750      \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
6751    },
6752    shrink             .code:n       = {
6753      \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
6754    },
6755    squeeze            .code:n       = {
6756      \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
6757    },
6758    t                  .code:n       = {
6759      \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
6760    },
6761  }
6762  \cs_new_protected:Nn \__notesslides_frame_args:n {
6763    \str_clear:N \l__notesslides_frame_label_str
```

```
6764        \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
6765        \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
6766        \bool_set_true:N \l__notesslides_frame_fragile_bool
6767        \bool_set_true:N \l__notesslides_frame_shrink_bool
6768        \bool_set_true:N \l__notesslides_frame_squeeze_bool
6769        \bool_set_true:N \l__notesslides_frame_t_bool
6770        \keys_set:nn { notesslides / frame }{ #1 }
6771    }
```

We define the environment, read them, and construct the slide number and label.

```
6772    \renewenvironment{frame}[1][]{
6773        \__notesslides_frame_args:n{#1}
6774        \sffamily
6775        \stepcounter{slide}
6776        \def\@currentlabel{\theslide}
6777        \str_if_empty:NF \l__notesslides_frame_label_str {
6778          \label{\l__notesslides_frame_label_str}
6779        }
```

We redefine the `itemize` environment so that it looks more like the one in `beamer`.

```
6780        \def\itemize@level{outer}
6781        \def\itemize@outer{outer}
6782        \def\itemize@inner{inner}
6783        \renewcommand\newpage{\addtocounter{framenumber}{1}}
6784        \newcommand\metakeys@show@keys[2]{\marginnote{{\scriptsize ##2}}}
6785        \renewenvironment{itemize}{
6786          \ifx\itemize@level\itemize@outer
6787            \def\itemize@label{$\rhd$}
6788          \fi
6789          \ifx\itemize@level\itemize@inner
6790            \def\itemize@label{$\scriptstyle\rhd$}
6791          \fi
6792          \begin{list}
6793          {\itemize@label}
6794          {\setlength{\labelsep}{.3em}
6795           \setlength{\labelwidth}{.5em}
6796           \setlength{\leftmargin}{1.5em}
6797          }
6798          \edef\itemize@level{\itemize@inner}
6799        }{
6800          \end{list}
6801        }
```

We create the box with the `mdframed` environment from the equinymous package.

```
6802        \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth
6803    }{
6804        \medskip\miko@slidelabel\end{mdframed}
6805    }
```

Now, we need to redefine the frametitle (we are still in course notes mode).

\frametitle

```
6806        \renewcommand{\frametitle}[1]{{\Large\bf\sf\color{blue}{#1}}\medskip}
6807    }
```

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

249

\pause

```
6808 \bool_if:NT \c__notesslides_notes_bool {
6809   \newcommand\pause{}
6810 }
```

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

nparagraph

```
6811 \bool_if:NTF \c__notesslides_notes_bool {
6812   \newenvironment{nparagraph}[1][]{\begin{sparagraph}[#1]}{\end{sparagraph}}
6813 }{
6814   \excludecomment{nparagraph}
6815 }
```

nfragment

```
6816 \bool_if:NTF \c__notesslides_notes_bool {
6817   \newenvironment{nfragment}[2][]{\begin{sfragment}[#1]{#2}}{\end{sfragment}}
6818 }{
6819   \excludecomment{nfragment}
6820 }
```

ndefinition

```
6821 \bool_if:NTF \c__notesslides_notes_bool {
6822   \newenvironment{ndefinition}[1][]{\begin{sdefinition}[#1]}{\end{sdefinition}}
6823 }{
6824   \excludecomment{ndefinition}
6825 }
```

nassertion

```
6826 \bool_if:NTF \c__notesslides_notes_bool {
6827   \newenvironment{nassertion}[1][]{\begin{sassertion}[#1]}{\end{sassertion}}
6828 }{
6829   \excludecomment{nassertion}
6830 }
```

nsproof

```
6831 \bool_if:NTF \c__notesslides_notes_bool {
6832   \newenvironment{nproof}[2][]{\begin{sproof}[#1]{#2}}{\end{sproof}}
6833 }{
6834   \excludecomment{nproof}
6835 }
```

nexample

```
6836 \bool_if:NTF \c__notesslides_notes_bool {
6837   \newenvironment{nexample}[1][]{\begin{sexample}[#1]}{\end{sexample}}
6838 }{
6839   \excludecomment{nexample}
6840 }
```

\inputref@*skip   We customize the hooks for in \inputref.

```
6841 \def\inputref@preskip{\smallskip}
6842 \def\inputref@postskip{\medskip}
```

---

[14]EDNOTE: MK: fake it in notes mode for now

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

\inputref*

```
6843  \let\orig@inputref\inputref
6844  \def\inputref{\@ifstar\ninputref\orig@inputref}
6845  \newcommand\ninputref[2][]{
6846    \bool_if:NT \c__notesslides_notes_bool {
6847      \orig@inputref[#1]{#2}
6848    }
6849  }
```

(*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⟩}`.

```
6850  \newlength{\slidelogoheight}
6851
6852  \bool_if:NTF \c__notesslides_notes_bool {
6853    \setlength{\slidelogoheight}{.4cm}
6854  }{
6855    \setlength{\slidelogoheight}{1cm}
6856  }
6857  \newsavebox{\slidelogo}
6858  \sbox{\slidelogo}{\sTeX}
6859  \newrobustcmd{\setslidelogo}[1]{
6860    \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
6861  }
```

(*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.

```
6862  \def\source{Michael Kohlhase}% customize locally
6863  \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.

```
6864  \def\copyrightnotice{\footnotesize\copyright :\hspace{.3ex}{\source}}
6865  \newsavebox{\cclogo}
6866  \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{stex-cc_somerights}}
6867  \newif\ifcchref\cchreffalse
6868  \AtBeginDocument{
6869    \@ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
6870  }
```

```
6871  \def\licensing{
6872    \ifcchref
6873      \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
6874    \else
6875      {\usebox{\cclogo}}
6876    \fi
6877  }
6878  \newrobustcmd{\setlicensing}[2][]{
6879    \def\@url{#1}
6880    \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
6881    \ifx\@url\@empty
6882      \def\licensing{{\usebox{\cclogo}}}
6883    \else
6884      \def\licensing{
6885        \ifcchref
6886        \href{#1}{\usebox{\cclogo}}
6887        \else
6888        {\usebox{\cclogo}}
6889        \fi
6890      }
6891    \fi
6892  }
```

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

EdN:15 \slidelabel Now, we set up the slide label for the `article` mode.[15]

```
6893  \newrobustcmd\miko@slidelabel{
6894    \vbox to \slidelogoheight{
6895      \vss\hbox to \slidewidth
6896      {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
6897    }
6898  }
```

(*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.

```
6899  \def\Gin@mhrepos{}
6900  \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
6901  \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
6902  \newrobustcmd\frameimage[2][]{
6903    \stepcounter{slide}
6904    \bool_if:NT \c__notesslides_frameimages_bool {
6905      \def\Gin@ewidth{}\setkeys{Gin}{#1}
6906      \bool_if:NF \c__notesslides_notes_bool { \vfill }
6907      \begin{center}
6908        \bool_if:NTF \c__notesslides_fiboxed_bool {
6909          \fbox{
6910            \ifx\Gin@ewidth\@empty
6911              \ifx\Gin@mhrepos\@empty
```

---

[15]EDNOTE: see that we can use the themes for the slides some day. This is all fake.

252

```
6912              \mhgraphics[width=\slidewidth,#1]{#2}
6913            \else
6914              \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
6915            \fi
6916          \else% Gin@ewidth empty
6917            \ifx\Gin@mhrepos\@empty
6918              \mhgraphics[#1]{#2}
6919            \else
6920              \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
6921            \fi
6922          \fi% Gin@ewidth empty
6923        }
6924      }{
6925        \ifx\Gin@ewidth\@empty
6926          \ifx\Gin@mhrepos\@empty
6927            \mhgraphics[width=\slidewidth,#1]{#2}
6928          \else
6929            \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
6930          \fi
6931          \ifx\Gin@mhrepos\@empty
6932            \mhgraphics[#1]{#2}
6933          \else
6934            \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
6935          \fi
6936        \fi% Gin@ewidth empty
6937      }
6938      \end{center}
6939      \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
6940      \bool_if:NF \c__notesslides_notes_bool { \vfill }
6941    }
6942 } % 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.

```
6943 \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

```
6944 \AddToHook{begindocument}{
6945   \definecolor{green}{rgb}{0,.5,0}
6946   \definecolor{purple}{cmyk}{.3,1,0,.17}
6947 }
```

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.

```
6948 % \def\STpresent#1{\textcolor{blue}{#1}}
6949 \def\defemph#1{{\textcolor{magenta}{#1}}}
6950 \def\symrefemph#1{{\textcolor{cyan}{#1}}}
```

```
6951  \def\compemph#1{{\textcolor{blue}{#1}}}
6952  \def\titleemph#1{{\textcolor{blue}{#1}}}
6953  \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.

```
6954  \pgfdeclareimage[width=.8em]{miko@small@dbend}{stex-dangerous-bend}
6955  \def\smalltextwarning{
6956    \pgfuseimage{miko@small@dbend}
6957    \xspace
6958  }
6959  \pgfdeclareimage[width=1.2em]{miko@dbend}{stex-dangerous-bend}
6960  \newrobustcmd\textwarning{
6961    \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
6962    \xspace
6963  }
6964  \pgfdeclareimage[width=2.5em]{miko@big@dbend}{stex-dangerous-bend}
6965  \newrobustcmd\bigtextwarning{
6966    \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
6967    \xspace
6968  }
```

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

```
6969  \newrobustcmd\putgraphicsat[3]{
6970    \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
6971  }
6972  \newrobustcmd\putat[2]{
6973    \begin{picture}(0,0)\put(#1){#2}\end{picture}
6974  }
```

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`.

```
6975  \bool_if:NT \c__notesslides_sectocframes_bool {
6976    \str_if_eq:VnTF \__notesslidestopsect{part}{
6977      \newcounter{chapter}\counterwithin*{section}{chapter}
6978    }{
6979      \str_if_eq:VnT\__notesslidestopsect{chapter}{
6980        \newcounter{chapter}\counterwithin*{section}{chapter}
6981      }
6982    }
6983  }
```

\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

```
6984  \def\part@prefix{}
6985  \@ifpackageloaded{document-structure}{}{
6986    \str_case:VnF \__notesslidestopsect {
```

```
6987      {part}{
6988        \int_set:Nn \l_document_structure_section_level_int {0}
6989        \def\thesection{\arabic{chapter}.\arabic{section}}
6990        \def\part@prefix{\arabic{chapter}.}
6991      }
6992      {chapter}{
6993        \int_set:Nn \l_document_structure_section_level_int {1}
6994        \def\thesection{\arabic{chapter}.\arabic{section}}
6995        \def\part@prefix{\arabic{chapter}.}
6996      }
6997    }{
6998      \int_set:Nn \l_document_structure_section_level_int {2}
6999      \def\part@prefix{}
7000    }
7001 }
7002
7003 \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

```
7004    \renewenvironment{sfragment}[2][]{
7005      \__document_structure_omgroup_args:n { #1 }
7006      \int_incr:N \l_document_structure_section_level_int
7007      \bool_if:NT \c__notesslides_sectocframes_bool {
7008        \stepcounter{slide}
7009        \begin{frame}[noframenumbering]
7010        \vfill\Large\centering
7011        \red{
7012          \ifcase\l_document_structure_section_level_int\or
7013            \stepcounter{part}
7014            \def\__notesslideslabel{\omdoc@part@kw~\Roman{part}}
7015            \def\currentsectionlevel{\omdoc@part@kw}
7016          \or
7017            \stepcounter{chapter}
7018            \def\__notesslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
7019            \def\currentsectionlevel{\omdoc@chapter@kw}
7020          \or
7021            \stepcounter{section}
7022            \def\__notesslideslabel{\part@prefix\arabic{section}}
7023            \def\currentsectionlevel{\omdoc@section@kw}
7024          \or
7025            \stepcounter{subsection}
7026            \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
7027            \def\currentsectionlevel{\omdoc@subsection@kw}
7028          \or
7029            \stepcounter{subsubsection}
7030            \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
7031            \def\currentsectionlevel{\omdoc@subsubsection@kw}
7032          \or
7033            \stepcounter{paragraph}
7034            \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
```

255

```
7035        \def\currentsectionlevel{\omdoc@paragraph@kw}
7036      \else
7037        \def\__notesslideslabel{}
7038        \def\currentsectionlevel{\omdoc@paragraph@kw}
7039      \fi% end ifcase
7040      \__notesslideslabel%\sref@label@id\__notesslideslabel
7041      \quad #2%
7042    }%
7043    \vfill%
7044    \end{frame}%
7045  }
7046  \str_if_empty:NF \l__document_structure_omgroup_id_str {
7047    \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
7048  }
7049  }{}
7050 }
```

We set up a `beamer` template for theorems like ams style, but without a block environment.

```
7051 \def\inserttheorembodyfont{\normalfont}
7052 %\bool_if:NF \c__notesslides_notes_bool {
7053 %  \defbeamertemplate{theorem begin}{miko}
7054 %  {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
7055 %    \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
7056 %    \inserttheorempunctuation\inserttheorembodyfont\xspace}
7057 %  \defbeamertemplate{theorem end}{miko}{}
```

and we set it as the default one.

```
7058 %  \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.

```
7059 %  \expandafter\def\csname Parent2\endcsname{}
7060 %}
7061
7062 \AddToHook{begindocument}{ % this does not work for some reasone
7063   \setbeamertemplate{theorems}[ams style]
7064 }
7065 \bool_if:NT \c__notesslides_notes_bool {
7066   \renewenvironment{columns}[1][]{%
7067     \par\noindent%
7068     \begin{minipage}%
7069     \slidewidth\centering\leavevmode%
7070   }{%
7071     \end{minipage}\par\noindent%
7072   }%
7073   \newsavebox\columnbox%
7074   \renewenvironment<>{column}[2][]{%
7075     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
7076   }{%
7077     \end{minipage}\end{lrbox}\usebox\columnbox%
7078   }%
7079 }
```

```
7080  \bool_if:NTF \c__notesslides_noproblems_bool {
7081    \newenvironment{problems}{}{}
7082  }{
7083    \excludecomment{problems}
7084  }
```

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.

```
7085  \gdef\printexcursions{}
7086  \newcommand\excursionref[2]{% label, text
7087    \bool_if:NT \c__notesslides_notes_bool {
7088      \begin{sparagraph}[title=Excursion]
7089        #2 \sref[fallback=the appendix]{#1}.
7090      \end{sparagraph}
7091    }
7092  }
7093  \newcommand\activate@excursion[2][]{
7094    \gappto\printexcursions{\inputref[#1]{#2}}
7095  }
7096  \newcommand\excursion[4][]{% repos, label, path, text
7097    \bool_if:NT \c__notesslides_notes_bool {
7098      \activate@excursion[#1]{#3}\excursionref{#2}{#4}
7099    }
7100  }
```

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

\excursiongroup

```
7101  \keys_define:nn{notesslides / excursiongroup }{
7102    id        .str_set_x:N  = \l__notesslides_excursion_id_str,
7103    intro     .tl_set:N     = \l__notesslides_excursion_intro_tl,
7104    mhrepos    .str_set_x:N  = \l__notesslides_excursion_mhrepos_str
7105  }
7106  \cs_new_protected:Nn \__notesslides_excursion_args:n {
7107    \tl_clear:N \l__notesslides_excursion_intro_tl
7108    \str_clear:N \l__notesslides_excursion_id_str
7109    \str_clear:N \l__notesslides_excursion_mhrepos_str
7110    \keys_set:nn {notesslides / excursiongroup }{ #1 }
7111  }
7112  \newcommand\excursiongroup[1][]{
7113    \__notesslides_excursion_args:n{ #1 }
7114    \ifdefempty\printexcursions{}% only if there are excursions
7115    {\begin{note}
7116      \begin{sfragment}[#1]{Excursions}%
7117        \ifdefempty\l__notesslides_excursion_intro_tl{}{
7118          \inputref[\l__notesslides_excursion_mhrepos_str]{
7119            \l__notesslides_excursion_intro_tl
7120          }
7121        }
7122        \printexcursions%
```

```
7123        \end{sfragment}
7124      \end{note}}
7125 }
7126 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
7127 ⟨/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.

```
7128 ⟨∗package⟩
7129 ⟨@@=problems⟩
7130 \ProvidesExplPackage{problem}{2022/02/26}{3.0.1}{Semantic Markup for Problems}
7131 \RequirePackage{l3keys2e,stex}
7132
7133 \keys_define:nn { problem / pkg }{
7134   notes    .default:n   = { true },
7135   notes    .bool_set:N  = \c__problems_notes_bool,
7136   gnotes   .default:n   = { true },
7137   gnotes   .bool_set:N  = \c__problems_gnotes_bool,
7138   hints    .default:n   = { true },
7139   hints    .bool_set:N  = \c__problems_hints_bool,
7140   solutions .default:n  = { true },
7141   solutions .bool_set:N = \c__problems_solutions_bool,
7142   pts      .default:n   = { true },
7143   pts      .bool_set:N  = \c__problems_pts_bool,
7144   min      .default:n   = { true },
7145   min      .bool_set:N  = \c__problems_min_bool,
7146   boxed    .default:n   = { true },
7147   boxed    .bool_set:N  = \c__problems_boxed_bool,
7148   unknown  .code:n      = {}
7149 }
7150 \newif\ifsolutions
7151
7152 \ProcessKeysOptions{ problem / pkg }
7153 \bool_if:NTF \c__problems_solutions_bool {
7154   \solutionstrue
7155 }{
7156   \solutionsfalse
7157 }
```

Then we make sure that the necessary packages are loaded (in the right versions).

```
7158  \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.

```
7159  \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.

```
7160  \def\prob@problem@kw{Problem}
7161  \def\prob@solution@kw{Solution}
7162  \def\prob@hint@kw{Hint}
7163  \def\prob@note@kw{Note}
7164  \def\prob@gnote@kw{Grading}
7165  \def\prob@pt@kw{pt}
7166  \def\prob@min@kw{min}
```

(*End definition for* \prob@*@kw*. This function is documented on page* **??**.)

For the other languages, we set up triggers

```
7167  \AddToHook{begindocument}{
7168    \ltx@ifpackageloaded{babel}{
7169        \makeatletter
7170        \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7171        \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7172          \input{problem-ngerman.ldf}
7173        }
7174        \clist_if_in:NnT \l_tmpa_clist {finnish}{
7175          \input{problem-finnish.ldf}
7176        }
7177        \clist_if_in:NnT \l_tmpa_clist {french}{
7178          \input{problem-french.ldf}
7179        }
7180        \clist_if_in:NnT \l_tmpa_clist {russian}{
7181          \input{problem-russian.ldf}
7182        }
7183        \makeatother
7184    }{}
7185  }
```

39.2   Problems and Solutions

We now prepare the KeyVal support for problems. The key macros just set appropriate internal macros.

```
7186  \keys_define:nn{ problem / problem }{
7187    id      .str_set_x:N  = \l__problems_prob_id_str,
7188    pts     .tl_set:N     = \l__problems_prob_pts_tl,
7189    min     .tl_set:N     = \l__problems_prob_min_tl,
7190    title   .tl_set:N     = \l__problems_prob_title_tl,
7191    type    .tl_set:N     = \l__problems_prob_type_tl,
7192    imports .tl_set:N     = \l__problems_prob_imports_tl,
7193    name    .str_set_x:N  = \l__problems_prob_name_str,
7194    refnum  .int_set:N    = \l__problems_prob_refnum_int
```

```
7195 }
7196 \cs_new_protected:Nn \__problems_prob_args:n {
7197   \str_clear:N \l__problems_prob_id_str
7198   \str_clear:N \l__problems_prob_name_str
7199   \tl_clear:N \l__problems_prob_pts_tl
7200   \tl_clear:N \l__problems_prob_min_tl
7201   \tl_clear:N \l__problems_prob_title_tl
7202   \tl_clear:N \l__problems_prob_type_tl
7203   \tl_clear:N \l__problems_prob_imports_tl
7204   \int_zero_new:N \l__problems_prob_refnum_int
7205   \keys_set:nn { problem / problem }{ #1 }
7206   \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
7207     \let\l__problems_prob_refnum_int\undefined
7208   }
7209 }
```

Then we set up a counter for problems.

\numberproblemsin

```
7210 \newcounter{problem}
7211 \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.

```
7212 \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

```
7213 \newcommand\prob@number{
7214   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
7215     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
7216   }{
7217     \int_if_exist:NTF \l__problems_prob_refnum_int {
7218       \prob@label{\int_use:N \l__problems_prob_refnum_int }
7219     }{
7220       \prob@label\theproblem
7221     }
7222   }
7223 }
```

(*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.

```
7224 \newcommand\prob@title[3]{%
7225   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
7226     #2 \l__problems_inclprob_title_tl #3
7227   }{
7228     \tl_if_exist:NTF \l__problems_prob_title_tl {
7229       #2 \l__problems_prob_title_tl #3
7230     }{
7231       #1
```

261

```
7232        }
7233      }
7234   }
```

(*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.

```
7235   \def\prob@heading{
7236     {\prob@problem@kw}\ \prob@number\prob@title{~}{~(}{)\strut}
7237     %\sref@label@id{\prob@problem@kw~\prob@number}{}
7238   }
```

(*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`

```
7239   \newenvironment{sproblem}[1][]{
7240     \__problems_prob_args:n{#1}%\sref@target%
7241     \@in@omtexttrue% we are in a statement (for inline definitions)
7242     \stepcounter{problem}\record@problem
7243     \def\current@section@level{\prob@problem@kw}
7244
7245     \str_if_empty:NT \l__problems_prob_name_str {
7246       \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
7247       \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
7248       \seq_get_left:NN \l_tmpa_seq \l__problems_prob_name_str
7249     }
7250     \exp_args:Nno\stex_module_setup:nn{type=problem}\l__problems_prob_name_str
7251
7252     \stex_reactivate_macro:N \STEXexport
7253     \stex_reactivate_macro:N \importmodule
7254     \stex_reactivate_macro:N \symdecl
7255     \stex_reactivate_macro:N \notation
7256     \stex_reactivate_macro:N \symdef
7257
7258     \stex_csl_to_imports:No \importmodule \l__problems_prob_imports_tl
7259
7260
7261     \tl_if_exist:NTF \l__problems_inclprob_type_tl {
7262       \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
7263     }{
7264       \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
7265     }
7266     \str_if_exist:NTF \l__problems_inclprob_id_str {
7267       \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
7268     }{
7269       \str_set_eq:NN \sproblemid \l__problems_prob_id_str
7270     }
7271
```

```
7272
7273    \clist_set:No \l_tmpa_clist \sproblemtype
7274    \tl_clear:N \l_tmpa_tl
7275    \clist_map_inline:Nn \l_tmpa_clist {
7276      \tl_if_exist:cT {__problems_sproblem_##1_start:}{
7277        \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_start:}}
7278      }
7279    }
7280    \tl_if_empty:NTF \l_tmpa_tl {
7281      \__problems_sproblem_start:
7282    }{
7283      \l_tmpa_tl
7284    }
7285    \stex_ref_new_doc_target:n \sproblemid
7286 }{
7287    \__stex_modules_end_module:
7288    \clist_set:No \l_tmpa_clist \sproblemtype
7289    \tl_clear:N \l_tmpa_tl
7290    \clist_map_inline:Nn \l_tmpa_clist {
7291      \tl_if_exist:cT {__problems_sproblem_##1_end:}{
7292        \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_end:}}
7293      }
7294    }
7295    \tl_if_empty:NTF \l_tmpa_tl {
7296      \__problems_sproblem_end:
7297    }{
7298      \l_tmpa_tl
7299    }
7300
7301
7302    \smallskip
7303 }
7304
7305
7306 \cs_new_protected:Nn \__problems_sproblem_start: {
7307    \par\noindent\textbf\prob@heading\show@pts\show@min\\\ignorespacesandpars
7308 }
7309 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
7310
7311 \newcommand\stexpatchproblem[3][] {
7312      \str_set:Nx \l_tmpa_str{ #1 }
7313      \str_if_empty:NTF \l_tmpa_str {
7314        \tl_set:Nn \__problems_sproblem_start: { #2 }
7315        \tl_set:Nn \__problems_sproblem_end: { #3 }
7316      }{
7317        \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
7318        \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
7319      }
7320 }
7321
7322
7323 \bool_if:NT \c__problems_boxed_bool {
7324    \surroundwithmdframed{problem}
7325 }
```

263

`\record@problem` This macro records information about the problems in the `*.aux` file.

```
7326 \def\record@problem{
7327   \protected@write\@auxout{}
7328   {
7329     \string\@problem{\prob@number}
7330     {
7331       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7332         \l__problems_inclprob_pts_tl
7333       }{
7334         \l__problems_prob_pts_tl
7335       }
7336     }%
7337     {
7338       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7339         \l__problems_inclprob_min_tl
7340       }{
7341         \l__problems_prob_min_tl
7342       }
7343     }
7344   }
7345 }
```

(*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).

```
7346 \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.

```
7347 \keys_define:nn { problem / solution }{
7348   id            .str_set_x:N  = \l__problems_solution_id_str ,
7349   for           .tl_set:N     = \l__problems_solution_for_tl ,
7350   height        .dim_set:N    = \l__problems_solution_height_dim ,
7351   creators      .clist_set:N  = \l__problems_solution_creators_clist ,
7352   contributors  .clist_set:N  = \l__problems_solution_contributors_clist ,
7353   srccite       .tl_set:N     = \l__problems_solution_srccite_tl
7354 }
7355 \cs_new_protected:Nn \__problems_solution_args:n {
7356   \str_clear:N \l__problems_solution_id_str
7357   \tl_clear:N \l__problems_solution_for_tl
7358   \tl_clear:N \l__problems_solution_srccite_tl
7359   \clist_clear:N \l__problems_solution_creators_clist
7360   \clist_clear:N \l__problems_solution_contributors_clist
7361   \dim_zero:N \l__problems_solution_height_dim
7362   \keys_set:nn { problem / solution }{ #1 }
7363 }
```

the next step is to define a helper macro that does what is needed to start a solution.

```
7364 \newcommand\@startsolution[1][]{
7365   \__problems_solution_args:n { #1 }
7366   \@in@omtexttrue% we are in a statement.
7367   \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`

```
7407    }
7408  \newenvironment{exhint}[1][]{
7409     \par\smallskip\hrule\smallskip
7410     \noindent\textbf{\prob@hint@kw :~ }\small
7411  }{
7412     \smallskip\hrule
7413  }
7414 }{
7415   \excludecomment{hint}
7416   \excludecomment{exhint}
7417 }
```

gnote

```
7418 \bool_if:NTF \c__problems_notes_bool {
7419   \newenvironment{gnote}[1][]{
7420     \par\smallskip\hrule\smallskip
7421     \noindent\textbf{\prob@gnote@kw : }\small
7422   }{
7423     \smallskip\hrule
7424   }
7425 }{
7426   \excludecomment{gnote}
7427 }
```

39.3   Multiple Choice Blocks

mcb [16]

```
7428 \newenvironment{mcb}{
7429   \begin{enumerate}
7430 }{
7431   \end{enumerate}
7432 }
```

we define the keys for the mcc macro

```
7433 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
7434   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
7435     \bool_set_true:N #1
7436   }{
7437     \bool_set_false:N #1
7438   }
7439 }
7440 \keys_define:nn { problem / mcc }{
7441   id        .str_set_x:N  = \l__problems_mcc_id_str ,
7442   feedback  .tl_set:N     = \l__problems_mcc_feedback_tl ,
7443   T         .default:n    = { true } ,
7444   T         .bool_set:N   = \l__problems_mcc_t_bool ,
7445   F         .default:n    = { true } ,
7446   F         .bool_set:N   = \l__problems_mcc_f_bool ,
7447   Ttext     .code:n       = {
7448     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
7449   } ,
```

---

[16]EDNOTE: MK: maybe import something better here from a dedicated MC package

266

```
7450    Ftext       .code:n        = {
7451      \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
7452    }
7453 }
7454 \cs_new_protected:Nn \l__problems_mcc_args:n {
7455    \str_clear:N \l__problems_mcc_id_str
7456    \tl_clear:N \l__problems_mcc_feedback_tl
7457    \bool_set_true:N \l__problems_mcc_t_bool
7458    \bool_set_true:N \l__problems_mcc_f_bool
7459    \bool_set_true:N \l__problems_mcc_Ttext_bool
7460    \bool_set_false:N \l__problems_mcc_Ftext_bool
7461    \keys_set:nn { problem / mcc }{ #1 }
7462 }
```

\mcc

```
7463 \newcommand\mcc[2][]{
7464    \l__problems_mcc_args:n{ #1 }
7465    \item #2
7466    \ifsolutions
7467      \\
7468      \bool_if:NT \l__problems_mcc_t_bool {
7469        % TODO!
7470        % \ifcsstring{mcc@T}{T}{}{\mcc@Ttext}%
7471      }
7472      \bool_if:NT \l__problems_mcc_f_bool {
7473        % TODO!
7474        % \ifcsstring{mcc@F}{F}{}{\mcc@Ftext}%
7475      }
7476      \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
7477        !
7478      }{
7479        \l__problems_mcc_feedback_tl
7480      }
7481    \fi
7482 } %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.

```
7483
7484 \keys_define:nn{ problem / inclproblem }{
7485    id      .str_set_x:N  = \l__problems_inclprob_id_str,
7486    pts     .tl_set:N     = \l__problems_inclprob_pts_tl,
7487    min     .tl_set:N     = \l__problems_inclprob_min_tl,
7488    title   .tl_set:N     = \l__problems_inclprob_title_tl,
7489    refnum  .int_set:N    = \l__problems_inclprob_refnum_int,
7490    type    .tl_set:N     = \l__problems_inclprob_type_tl,
7491    mhrepos .str_set_x:N  = \l__problems_inclprob_mhrepos_str
7492 }
```

267

```
7493 \cs_new_protected:Nn \__problems_inclprob_args:n {
7494   \str_clear:N \l__problems_prob_id_str
7495   \tl_clear:N \l__problems_inclprob_pts_tl
7496   \tl_clear:N \l__problems_inclprob_min_tl
7497   \tl_clear:N \l__problems_inclprob_title_tl
7498   \tl_clear:N \l__problems_inclprob_type_tl
7499   \int_zero_new:N \l__problems_inclprob_refnum_int
7500   \str_clear:N \l__problems_inclprob_mhrepos_str
7501   \keys_set:nn { problem / inclproblem }{ #1 }
7502   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
7503     \let\l__problems_inclprob_pts_tl\undefined
7504   }
7505   \tl_if_empty:NT \l__problems_inclprob_min_tl {
7506     \let\l__problems_inclprob_min_tl\undefined
7507   }
7508   \tl_if_empty:NT \l__problems_inclprob_title_tl {
7509     \let\l__problems_inclprob_title_tl\undefined
7510   }
7511   \tl_if_empty:NT \l__problems_inclprob_type_tl {
7512     \let\l__problems_inclprob_type_tl\undefined
7513   }
7514   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
7515     \let\l__problems_inclprob_refnum_int\undefined
7516   }
7517 }
7518
7519 \cs_new_protected:Nn \__problems_inclprob_clear: {
7520   \let\l__problems_inclprob_id_str\undefined
7521   \let\l__problems_inclprob_pts_tl\undefined
7522   \let\l__problems_inclprob_min_tl\undefined
7523   \let\l__problems_inclprob_title_tl\undefined
7524   \let\l__problems_inclprob_type_tl\undefined
7525   \let\l__problems_inclprob_refnum_int\undefined
7526   \let\l__problems_inclprob_mhrepos_str\undefined
7527 }
7528 \__problems_inclprob_clear:
7529
7530 \newcommand\includeproblem[2][]{
7531   \__problems_inclprob_args:n{ #1 }
7532   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
7533     \input{#2}
7534   }{
7535     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
7536       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
7537     }
7538   }
7539   \__problems_inclprob_clear:
7540 }
```

(*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.

```
7541  \AddToHook{enddocument}{
7542    \bool_if:NT \c__problems_pts_bool {
7543      \message{Total:~\arabic{pts}~points}
7544    }
7545    \bool_if:NT \c__problems_min_bool {
7546      \message{Total:~\arabic{min}~minutes}
7547    }
7548  }
```

The margin pars are reader-visible, so we need to translate

```
7549  \def\pts#1{
7550    \bool_if:NT \c__problems_pts_bool {
7551      \marginpar{#1~\prob@pt@kw}
7552    }
7553  }
7554  \def\min#1{
7555    \bool_if:NT \c__problems_min_bool {
7556      \marginpar{#1~\prob@min@kw}
7557    }
7558  }
```

\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.

```
7559  \newcounter{pts}
7560  \def\show@pts{
7561    \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7562      \bool_if:NT \c__problems_pts_bool {
7563        \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
7564        \addtocounter{pts}{\l__problems_inclprob_pts_tl}
7565      }
7566    }{
7567      \tl_if_exist:NT \l__problems_prob_pts_tl {
7568        \bool_if:NT \c__problems_pts_bool {
7569          \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
7570          \addtocounter{pts}{\l__problems_prob_pts_tl}
7571        }
7572      }
7573    }
7574  }
```

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

and now the same for the minutes

\show@min

```
7575  \newcounter{min}
7576  \def\show@min{
7577    \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7578      \bool_if:NT \c__problems_min_bool {
7579        \marginpar{\l__problems_inclprob_pts_tl\ min}
```

```
7580        \addtocounter{min}{\l__problems_inclprob_min_tl}
7581      }
7582    }{
7583      \tl_if_exist:NT \l__problems_prob_min_tl {
7584        \bool_if:NT \c__problems_min_bool {
7585          \tl_if_empty:NT\l__problems_prob_min_tl{
7586            \tl_set:Nn \l__problems_prob_min_tl {0}
7587          }
7588          \marginpar{\l__problems_prob_min_tl\ min}
7589          \addtocounter{min}{\l__problems_prob_min_tl}
7590        }
7591      }
7592    }
7593  }
7594  ⟨/package⟩
```

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

Chapter 40

Implementation: The hwexam Package

40.1 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.

```
7595  ⟨∗package⟩
7596  \ProvidesExplPackage{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
7597  \RequirePackage{l3keys2e}
7598
7599  \newif\iftest\testfalse
7600  \DeclareOption{test}{\testtrue}
7601  \newif\ifmultiple\multiplefalse
7602  \DeclareOption{multiple}{\multipletrue}
7603  \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
7604  \ProcessOptions
```

Then we make sure that the necessary packages are loaded (in the right versions).

```
7605  \RequirePackage{keyval}[1997/11/10]
7606  \RequirePackage{problem}
```

`\hwexam@*@kw`   For multilinguality, we define internal macros for keywords that can be specialized in `*.ldf` files.

```
7607  \newcommand\hwexam@assignment@kw{Assignment}
7608  \newcommand\hwexam@given@kw{Given}
7609  \newcommand\hwexam@due@kw{Due}
7610  \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
7611  blank~for~extra~space}
7612  \def\hwexam@minutes@kw{minutes}
7613  \newcommand\correction@probs@kw{prob.}
7614  \newcommand\correction@pts@kw{total}
7615  \newcommand\correction@reached@kw{reached}
7616  \newcommand\correction@sum@kw{Sum}
7617  \newcommand\correction@grade@kw{grade}
7618  \newcommand\correction@forgrading@kw{To~be~used~for~grading,~do~not~write~here}
```

271

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

For the other languages, we set up triggers

```
7619 \AddToHook{begindocument}{
7620 \ltx@ifpackageloaded{babel}{
7621 \makeatletter
7622 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7623 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7624   \input{hwexam-ngerman.ldf}
7625 }
7626 \clist_if_in:NnT \l_tmpa_clist {finnish}{
7627   \input{hwexam-finnish.ldf}
7628 }
7629 \clist_if_in:NnT \l_tmpa_clist {french}{
7630   \input{hwexam-french.ldf}
7631 }
7632 \clist_if_in:NnT \l_tmpa_clist {russian}{
7633   \input{hwexam-russian.ldf}
7634 }
7635 \makeatother
7636 }{}
7637 }
7638
```

40.2   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.

```
7639 \newcounter{assignment}
7640 \numberproblemsin{assignment}
7641 \renewcommand\prob@label[1]{\assignment@number.#1}
```

We will prepare the keyval support for the `assignment` environment.

```
7642 \keys_define:nn { hwexam / assignment } {
7643 id   .str_set_x:N = \l_@@_assign_id_str,
7644 number  .int_set:N  = \l_@@_assign_number_int,
7645 title  .tl_set:N  = \l_@@_assign_title_tl,
7646 type  .tl_set:N  = \l_@@_assign_type_tl,
7647 given .tl_set:N  = \l_@@_assign_given_tl,
7648 due .tl_set:N  = \l_@@_assign_due_tl,
7649 loadmodules .code:n  = {
7650 \bool_set_true:N \l_@@_assign_loadmodules_bool
7651 }
7652 }
7653 \cs_new_protected:Nn \_@@_assignment_args:n {
7654 \str_clear:N \l_@@_assign_id_str
7655 \int_set:Nn \l_@@_assign_number_int {-1}
7656 \tl_clear:N \l_@@_assign_title_tl
7657 \tl_clear:N \l_@@_assign_type_tl
7658 \tl_clear:N \l_@@_assign_given_tl
7659 \tl_clear:N \l_@@_assign_due_tl
7660 \bool_set_false:N \l_@@_assign_loadmodules_bool
```

```
7661  \keys_set:nn { hwexam / assignment }{ #1 }
7662  }
```

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.

```
7663  \newcommand\given@due[2]{
7664  \bool_lazy_all:nF {
7665  {\tl_if_empty_p:V \l_@@_inclassign_given_tl}
7666  {\tl_if_empty_p:V \l_@@_assign_given_tl}
7667  {\tl_if_empty_p:V \l_@@_inclassign_due_tl}
7668  {\tl_if_empty_p:V \l_@@_assign_due_tl}
7669  }{ #1 }
7670
7671  \tl_if_empty:NTF \l_@@_inclassign_given_tl {
7672  \tl_if_empty:NF \l_@@_assign_given_tl {
7673  \hwexam@given@kw\xspace\l_@@_assign_given_tl
7674  }
7675  }{
7676  \hwexam@given@kw\xspace\l_@@_inclassign_given_tl
7677  }
7678
7679  \bool_lazy_or:nnF {
7680  \bool_lazy_and_p:nn {
7681  \tl_if_empty_p:V \l_@@_inclassign_due_tl
7682  }{
7683  \tl_if_empty_p:V \l_@@_assign_due_tl
7684  }
7685  }{
7686  \bool_lazy_and_p:nn {
7687  \tl_if_empty_p:V \l_@@_inclassign_due_tl
7688  }{
7689  \tl_if_empty_p:V \l_@@_assign_due_tl
7690  }
7691  }{ ,~ }
7692
7693  \tl_if_empty:NTF \l_@@_inclassign_due_tl {
7694  \tl_if_empty:NF \l_@@_assign_due_tl {
7695  \hwexam@due@kw\xspace \l_@@_assign_due_tl
7696  }
7697  }{
7698  \hwexam@due@kw\xspace \l_@@_inclassign_due_tl
7699  }
7700
7701  \bool_lazy_all:nF {
7702  { \tl_if_empty_p:V \l_@@_inclassign_given_tl }
7703  { \tl_if_empty_p:V \l_@@_assign_given_tl }
7704  { \tl_if_empty_p:V \l_@@_inclassign_due_tl }
7705  { \tl_if_empty_p:V \l_@@_assign_due_tl }
7706  }{ #2 }
7707  }
```

\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.

```
7708 \newcommand\assignment@title[3]{
7709 \tl_if_empty:NTF \l_@@_inclassign_title_tl {
7710 \tl_if_empty:NTF \l_@@_assign_title_tl {
7711 #1
7712 }{
7713 #2\l_@@_assign_title_tl#3
7714 }
7715 }{
7716 #2\l_@@_inclassign_title_tl#3
7717 }
7718 }
```

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

\assignment@number   Like `\assignment@title` only for the number, and no around part.

```
7719 \newcommand\assignment@number{
7720 \int_compare:nNnTF \l_@@_inclassign_number_int = {-1} {
7721 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
7722 \arabic{assignment}
7723 } {
7724 \int_use:N \l_@@_assign_number_int
7725 }
7726 }{
7727 \int_use:N \l_@@_inclassign_number_int
7728 }
7729 }
```

(*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.

```
7730 \newenvironment{assignment}[1][]{
7731 \_@@_assignment_args:n { #1 }
7732 %\sref@target
7733 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
7734 \global\stepcounter{assignment}
7735 }{
7736 \global\setcounter{assignment}{\int_use:N\l_@@_assign_number_int}
7737 }
7738 \setcounter{problem}{0}
7739 \def\current@section@level{\document@hwexamtype}
7740 %\sref@label@id{\document@hwexamtype \thesection}
7741 \begin{@assignment}
7742 }{
7743 \end{@assignment}
7744 }
```

In the multi-assignment case we just use the `omdoc` environment for suitable sectioning.

```
7745  \def\ass@title{
7746  \protect\document@hwexamtype~\arabic{assignment}
7747  \assignment@title{}{\;(}{)\;} -- \given@due{}{}
7748  }
7749  \ifmultiple
7750  \newenvironment{@assignment}{
7751  \bool_if:NTF \l_@@_assign_loadmodules_bool {
7752  \begin{sfragment}[loadmodules]{\ass@title}
7753  }{
7754  \begin{sfragment}{\ass@title}
7755  }
7756  }{
7757  \end{sfragment}
7758  }
```

for the single-page case we make a title block from the same components.

```
7759  \else
7760  \newenvironment{@assignment}{
7761  \begin{center}\bf
7762  \Large\@title\strut\\
7763  \document@hwexamtype~\arabic{assignment}\assignment@title{\;}{:\;}{\\}
7764  \large\given@due{--\;}{\;--}
7765  \end{center}
7766  }{}
7767  \fi% multiple
```

40.3   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.

```
7768  \keys_define:nn { hwexam / inclassignment } {
7769  %id   .str_set_x:N = \l_@@_assign_id_str,
7770  number   .int_set:N  = \l_@@_inclassign_number_int,
7771  title  .tl_set:N  = \l_@@_inclassign_title_tl,
7772  type  .tl_set:N  = \l_@@_inclassign_type_tl,
7773  given .tl_set:N  = \l_@@_inclassign_given_tl,
7774  due .tl_set:N  = \l_@@_inclassign_due_tl,
7775  mhrepos   .str_set_x:N = \l_@@_inclassign_mhrepos_str
7776  }
7777  \cs_new_protected:Nn \_@@_inclassignment_args:n {
7778  \int_set:Nn \l_@@_inclassign_number_int {-1}
7779  \tl_clear:N \l_@@_inclassign_title_tl
7780  \tl_clear:N \l_@@_inclassign_type_tl
7781  \tl_clear:N \l_@@_inclassign_given_tl
7782  \tl_clear:N \l_@@_inclassign_due_tl
7783  \str_clear:N \l_@@_inclassign_mhrepos_str
7784  \keys_set:nn { hwexam / inclassignment }{ #1 }
7785  }
7786  \_@@_inclassignment_args:n {}
7787
7788  \newcommand\inputassignment[2][]{
```

```
7789  \_@@_inclassignment_args:n { #1 }
7790  \str_if_empty:NTF \l_@@_inclassign_mhrepos_str {
7791  \input{#2}
7792  }{
7793  \stex_in_repository:nn{\l_@@_inclassign_mhrepos_str}{
7794  \input{\mhpath{\l_@@_inclassign_mhrepos_str}{#2}}}
7795  }
7796  }
7797  \_@@_inclassignment_args:n {}
7798  }
7799  \newcommand\includeassignment[2][]{
7800  \newpage
7801  \inputassignment[#1]{#2}
7802  }
```

(*End definition for* \in*assignment. *This function is documented on page* **??**.)

40.4 Typesetting Exams

\quizheading

```
7803  \ExplSyntaxOff
7804  \newcommand\quizheading[1]{%
7805  \def\@tas{#1}%
7806  \large\noindent NAME: \hspace{8cm}  MAILBOX:\\[2ex]%
7807  \ifx\@tas\@empty\else%
7808  \noindent TA:~\@for\@I:=\@tas\do{{\Large$\Box$}\@I\hspace*{1em}}\\[2ex]%
7809  \fi%
7810  }
7811  \ExplSyntaxOn
```

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

\testheading

```
7812
7813  \def\hwexamheader{\input{hwexam-default.header}}
7814
7815  \def\hwexamminutes{
7816  \tl_if_empty:NTF \testheading@duration {
7817  {\testheading@min}~\hwexam@minutes@kw
7818  }{
7819  \testheading@duration
7820  }
7821  }
7822
7823  \keys_define:nn { hwexam / testheading } {
7824  min   .tl_set:N  = \testheading@min,
7825  duration .tl_set:N  = \testheading@duration,
7826  reqpts .tl_set:N  = \testheading@reqpts,
7827  tools .tl_set:N  = \testheading@tools
7828  }
7829  \cs_new_protected:Nn \_@@_testheading_args:n {
7830  \tl_clear:N \testheading@min
7831  \tl_clear:N \testheading@duration
```

```
7832    \tl_clear:N \testheading@reqpts
7833    \tl_clear:N \testheading@tools
7834    \keys_set:nn { hwexam / testheading }{ #1 }
7835  }
7836  \newenvironment{testheading}[1][]{
7837    \_@@_testheading_args:n{ #1 }
7838    \newcount\check@time\check@time=\testheading@min
7839    \advance\check@time by -\theassignment@totalmin
7840    \newif\if@bonuspoints
7841    \tl_if_empty:NTF \testheading@reqpts {
7842    \@bonuspointsfalse
7843    }{
7844    \newcount\bonus@pts
7845    \bonus@pts=\theassignment@totalpts
7846    \advance\bonus@pts by -\testheading@reqpts
7847    \edef\bonus@pts{\the\bonus@pts}
7848    \@bonuspointstrue
7849    }
7850    \edef\check@time{\the\check@time}
7851
7852    \makeatletter\hwexamheader\makeatother
7853  }{
7854    \newpage
7855  }
```

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

`\testspace`

```
7856  \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}
```

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

`\testnewpage`

```
7857  \newcommand\testnewpage{\iftest\newpage\fi}
```

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

`\testemptypage`

```
7858  \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.

```
7859  ⟨@@=problems⟩
7860  \renewcommand\@problem[3]{
7861    \stepcounter{assignment@probs}
7862    \def\__problemspts{#2}
7863    \ifx\__problemspts\@empty\else
7864    \addtocounter{assignment@totalpts}{#2}
7865    \fi
7866    \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\f
7867    \xdef\correction@probs{\correction@probs & #1}%
7868    \xdef\correction@pts{\correction@pts & #2}
7869    \xdef\correction@reached{\correction@reached &}
```

7870 `}`
7871 ⟨@@=hwexam⟩

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

`\correction@table` This macro generates the correction table

7872 `\newcounter{assignment@probs}`
7873 `\newcounter{assignment@totalpts}`
7874 `\newcounter{assignment@totalmin}`
7875 `\def\correction@probs{\correction@probs@kw}`
7876 `\def\correction@pts{\correction@pts@kw}`
7877 `\def\correction@reached{\correction@reached@kw}`
7878 `\stepcounter{assignment@probs}`
7879 `\newcommand\correction@table{`
7880 `\resizebox{\textwidth}{!}{%`
7881 `\begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%`
7882 `&\multicolumn{\theassignment@probs}{c||}%|`
7883 `{\footnotesize\correction@forgrading@kw} &\\\hline`
7884 `\correction@probs & \correction@sum@kw & \correction@grade@kw\\\hline`
7885 `\correction@pts &\theassignment@totalpts & \\\hline`
7886 `\correction@reached & & \\[.7cm]\hline`
7887 `\end{tabular}}}`
7888 ⟨/package⟩

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

40.5   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}
```