The sTeX3 Package Collection *

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

2022-05-23

Abstract

sTeX is a collection of LaTeX packages that allow to markup documents semantically without leaving the document format.

Running 'pdflatex' over sTeX-annotated documents formats them into normal-looking PDF. But sTeX also comes with a conversion pipeline into semantically annotated HTML5, which can host semantic added-value services that make the documents active (i.e. interactive and user-adaptive) and 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, and

- 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 3.1 (last revised 2022-05-23)

i

Contents

Part I
Manual

<table>
<tr>
<td>⚠</td>
<td>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 easily be skipped on a first read.</td>
</tr>
</table>

<table>
<tr>
<td>↪M→
—M→
⤳T↝</td>
<td>Boxes like this one explain how some S<small>T</small>E<small>X</small> concept relates to the M<small>MT</small>/OMD<small>OC</small> system, philosophy or language; see [MMT; Koh06] for introductions.</td>
</tr>
</table>

Chapter 1

What is sTeX?

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

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

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

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

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

- The sTeX package collection to use semantic annotations in LaTeX documents,

- RusTeX [RT] to convert `tex` sources to (semantically enriched) `xhtml`,

- The Mmt system [MMT], that extracts semantic information from the thus generated `xhtml` and provides semantically informed added value services. Notably, Mmt integrates the RusTeX system already.

Chapter 2

Quickstart

2.1 Setup

There are two ways of using sTeX: as a

1. way of writing LaTeX more modularly (object-oriented Math) for creating PDF documents or

2. foundation for authoring active documents in HTML5 instrumented with knowledge management services.

Both are legitimate and useful. The first requires a significantly smaller tool-chain, so we describe it first. The second requires a much more substantial (and experimental) toolchain of knowledge management systems. Both workflows profit from an integrated development environment (IDE), which (also) automates setup as far as possible (see subsection 2.1.4).

2.1.1 Minimal Setup for the PDF-only Workflow

In the best of all worlds, there is no setup, as you already have a new version of TeXLive on your system as a LaTeX enthusiast. If not now is the time to install it; see [TL]. You can usually update TeXLive via a package manager or the TeXLive manager **tlmgr**.

Alternatively, you can install sTeX from CTAN, the Comprehensive TeX Archive Network; see [ST] for details.

2.1.2 GIT-based Setup for the sTeX Development Version

If you want use the latest and greatest sTeX packages that have not even been released to CTAN, then you can directly clone them from the sTeX development repository [sTeX] by the following command-line instructions:

```
cd <stexdir>
git clone https://github.com/slatex/sTeX.git
```

and keep it updated by pulling updates via `git pull` in the cloned sTeX directory. Then update your `TEXINPUTS` environment variable, e.g. by placing the following line in your `.bashrc`:

```
export TEXINPUTS="$(TEXINPUTS):<sTeXDIR>//:"
```

2.1.3   sTeX Archives (Manual Setup)

Writing semantically annotated sTeX becomes much easier, if we can use well-designed libraries of already annotated content. sTeX provides such libraries as sTeX archives – i.e. GIT repositories at https://gl.mathhub.info – most prominently the SMGLoM libraries at https://gl.mathhub.info/smglom.

To do so, we set up a **local MathHub** by creating a MathHub directory `<mhdir>`. Every sTeX archive as an **archive path `<apath>`** and a name `<archive>`. We can clone the sTeX archive by the following command-line instructions:

```
cd <mhdir>/<apath>
git clone https://gl.mathhub.info/smglom/<archive>.git
```

Note that sTeX archives often depend on other archives, thus you should be prepared to clone these as well – e.g. if `pdflatex` reports missing files. 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).

```
export MATHHUB="<mhdir>''
```

2.1.4   The sTeX IDE

We are currently working on an sTeX IDE as an sTeX plugin for `VScode`; see [SIa]. It will feature a setup procedure that automates the setup described above (and below). For additional functionality see the (now obsolete) plugin for sTeX 1 [SLS; SIb].

2.1.5   Manual Setup for Active Documents and Knowledge Management Services

Foregoing on the sTeX IDE, we will need several additional (on top of the minimal setup above) pieces of software; namely:

- **The Mmt System** available here. 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.

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*:

```
 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.

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 to understand the respective parts of the sTeX markup infrastructure:

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

First, we open a new *module* called `GeometricSeries`. The main purpose of the `smodule` environment is to group the contents and associate it with a *globally unique* identifier (URI), which is computed from the name `GeometricSeries` and the document context.

(Depending on your pdf viewer), the URI should pop up in a tooltip if you hover over the word **geometric series**.

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

Next, we *import* two modules – `series` from the $\mathrm{S\!T\!E\!X}$ archive `smglom/calculus`, and `realarith` from the $\mathrm{S\!T\!E\!X}$ archive `smglom/arithmetics`. 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 $\mathrm{S\!T\!E\!X}$ symbols and associated semantic macros (e.g. `\infinitesum`, `\realdivide`, `\realpower`) in the imported module available to the current module `GeometricSeries`. The module `GeometricSeries` "exports" all of these symbols to all modules imports it via an `\importmodule{GeometricSeries}` instruction. Additionally it exports the local symbol `\geometricSeries`.

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}}
```

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.

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 the amsthm package.

<div style="margin-left:2em">

\symname

    ... is the \symname{?series}

The \symname-command prints the name of a symbol, highlights it (based on customizable settings) and associates the text printed with the corresponding symbol.

Note that the argument of \symref can be an imported symbol (here the series symbol is imported from the series module). sTEX tries to determine the full symbol URI from the argument. If there are name clashes in or with the imported symbols, the name of the exporting module can be prepended to the symbol name before the ? character.

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. \symref takes two arguments: the first ist the symbol name (or macro name), and the second a variant verbalization of the symbol, e.g. an inflection variant, a different language or a synonym. In our example \symname{?series} abbreviates \symref{?series}{series}.

\definame
\definiendum

    The \definame{geometricSeries} ...

The sdefinition-environment provides two additional macros, \definame and \definiendum which behave similarly 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.

</div>

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

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` while preserving all the SₜₑX markup in the result.

TODO VSCode Plugin

Using RusTₑX [RT], 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"/>
```

---

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

```
  <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>
```

...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.

2.2.2 Mmt/OMDoc Conversion

Another way to convert our document to *actual* Mmt/OMDoc is to put it in an sTeX **archive** (see section 3.2) and have Mmt take care of everything.

Assuming the above file is `source/demo.tex` in an sTeX archive `MyTest`, you can run Mmt and do `build MyTest stex-omdoc demo.tex` to convert the document to both `xhtml` (which you will find in `xhtml/demo.xhtml` in the archive) and formal Mmt/OMDoc, which you can subsequently view in the Mmt browser (see https://uniformal.github.io//doc/applications/server.html#the-mmt-web-site for details).

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 ($\langle language \rangle *$) Languages to load with the `babel` package.

mathhub ($\langle directory \rangle$) MathHub folder to search for repositories – this is not necessary if the `MATHHUB` system variable is set.

writesms ($\langle boolean \rangle$) with this package option, sTeX will write the contents of all external modules imported via `\importmodule` or `\usemodule` into a file `\jobname.sms` (analogously to the table of contents `.toc`-file).

usems ($\langle boolean \rangle$) subsequently tells sTeX to read the generated sms-file at the beginning of the document. This allows for e.g. collaborating on documents without all authors having to have all used archives and modules available – one author can load the modules with `writesms`, and the rest can use the the modules with `usesms`. Furthermore, the sms file can be submitted alongside a `tex`-file, effectively making it "standalone".

image ($\langle boolean \rangle$) passed on to `tikzinput`.

debug ($\langle log\text{-}prefix \rangle *$) 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:

1. sTeX **archives** (see section 3.2) contain individual `.tex`-files.

2. These may contain sTeX **modules**, introduced via `\begin{smodule}{ModuleName}`.

3. 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.

4. sTeX **expressions** finally are built up from usages of semantic macros.

> - 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 [RK13].
> - 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 abstract syntax (and XML encoding) of OpenMath [Bus+04].

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 four 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. This method is recommended on systems where it is difficult to set environment variables.

3.2.2 The Structure of STEX Archives

An STEX archive `group/name` is 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 the following 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 "encyclopaedic" 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

EdN:1
- `/source/pic/` – image files.[1]

3.2.3 MANIFEST.MF-Files

The `MANIFEST.MF` in the `META-INF`-directory consists of key-value-pairs, informing 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
```

---

[1]EDNOTE: MK: bisher habe ich immer PIC subdirs, soll ich das ändern?

```
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, e.g. for lazy loading.

In the majority of practical cases \inputref is likely to be preferred over \mhinput because it leads to less duplication in the generated `xhtml`.

\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. Typical invocations are

- \addmhbibresource{lib/refs.bib}, which specifies a bibliography in the `lib` folder in the local archive or

- \addmhbibresource[HW/meta-inf]{lib/refs.bib} in another.

\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`.

\libinput 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.

\libusepackage throws 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.

\libusepackage is particularly useful in `preamble.tex` when we want to use custom packages that are not part of TeXLive. In this case we commit the respective packages in one of the `lib` folders and use \libusepackage to load them.

3.3 Module, Symbol and Notation Declarations

3.3.1 The `smodule`-Environment

smodule A new module is declared using the basic syntax

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

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

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

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

type ($\langle string \rangle *$) for use in customizations.

deprecate ($\langle module \rangle$) if set, will throw a warning when loaded, urging to use $\langle module \rangle$ instead.

id ($\langle string \rangle$) for cross-referencing.

ns ($\langle URI \rangle$) 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 ($\langle language \rangle$) if not set, computed from the current file name (e.g. `foo.en.tex`).

sig ($\langle language \rangle$) 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 ($\langle string \rangle *$) names of the creators.

contributors ($\langle string \rangle *$) names of contributors.

srccite ($\langle string \rangle$) a source citation for the content of this module.

> ↪M→ An sTEX module corresponds to an MMT/OMDoc *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
```

.

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

For example:

Example 2

Input:

```
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.

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

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

> ↪M↠ `\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.

.

So far we have gained exactly ... nothing by adding the arity information: we cannot do anything with the arguments in the text.

We will now see what we can gain with more machinery.

`\notation` 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.

`\comp` For many semantic services e.g. semantic highlighting or **wikification** (linking user-visible notation components to the definition of the respective symbol they come from), we need to specify the notation components. Unfortunately, there is currently no way the sTeX engine can infer this by itself, so we have to specify it manually in the notation specification. 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 meaningful 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:

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.

But one man's meat is another man's poison: it is very subjective what the "default notation" of an operator should be. Different communities have different practices. For instance, the complex unit is written as i in Mathematics and as j in electrical engineering. So to allow modular specification and facilitate re-use of document fragments sTeX allows to re-set notation defaults.

`\setnotation`

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

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

Operator Notations

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

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

Input:

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

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 Modes

The notations so far used *simple* arguments which we call *mode-i* 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 mode-i arguments. However, there are three more argument modes which we will investigate now, namely mode-b, mode-a and mode-B arguments.

Mode-b Arguments

A mode-b 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→ Mode-b arguments behave exactly like mode-i arguments within TeX, but appli-
> —M→ cations of binding operators, i.e. symbols with mode-b arguments, are translated
> ~T↝ to `OMBIND`-terms in OMDoc/Mmt, rather than `OMA`.

For 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.

Mode-a Arguments

Mode-`a` 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. Mode-`a` 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 mode-`a` 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 mode-`a` 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, associative 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 $a \neq b \land a \neq c \land a \neq d \land b \neq c \land b \neq d \land c \neq d$

As before, at the PDF level, this annotation is invisible (and without effect), but at the level of the generated OMDoc/MMT this leads to more semantical expressions.

Mode-`B` Arguments

Finally, mode-`B` 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!)

The type and def keys correspond to the type and definiens components of OMDoc/Mmt constants.

Correspondingly, the name "type" should be taken with a grain of salt, since OMDoc/Mmt– being foundation-independent – does not a priori implement a fixed typing system.

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

Example 13

Input:

```
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 straight-forward in theory:

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 $\mathbb{N}\times\mathbb{N}\rightarrow\mathbb{N}$

.

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 $\cdot$ 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 perfectly 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 $p_{op} = 50$.

3.3.6 Variables

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

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

`\svar`

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

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

`\vardef`

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

Example 19

Input:

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

.

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

TODO: bind=forall/exists

3.3.7   Variable Sequences

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

`\varseq`

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

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:

> $a_1 + \ldots + a_n$

.

27

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:

$a_1^1, \ldots, a_n^m$ 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:

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

The sTeX features for modular document management are inherited from the OM-Doc/MMT model that organizes knowledge into a graph, where the nodes are theories (called modules in sTeX) and the edges are truth-preserving mappings (called theory morphismes in MMT). We have already seen modules/theories above.

Before we get into theory morphisms in sTeX we will see a very simple application of modules: managing multilinguality modularly.

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
\usemodule

\importmodule[Some/Archive]{path?ModuleName} is only allowed within an **smodule**-environment and makes the symbols declared in `ModuleName` available therein. Additionally the symbols 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, ꟻTᴇX 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 Mᴍᴛ does things.

Unfortunately, TᴇX 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 ꟻTᴇX can resolve them accordingly. Largely, users need not concern themselves with namespaces at all, but for completenesses sake, we describe how they are constructed:

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

- If the same statement occurs in a file `/path/to/file/bar[.⟨lang⟩].tex`, the namespace is `file://path/to/file/bar`.

In other words: outside of archives, the namespace corresponds to the file URI with the filename dropped iff it is equal to the module name, and ignoring the (optional) language suffix.

If the current file is in an archive, the procedure is the same except that the initial segment of the file path up to the archive's `source`-folder is replaced by the archive's namespace URI.

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

- `\importmodule{Foo}` outside of an archive refers to module `Foo` in the current namespace. Consequently, `Foo` must have been declared earlier in the same document or, if not, in a file `Foo[.⟨lang⟩].tex` in the same directory.

- The same statement *within* an archive refers to either the module `Foo` declared earlier in the same document, or otherwise to the module `Foo` in the archive's top-level namespace. In the latter case, is has to be declared in a file `Foo[.⟨lang⟩].tex` directly in the archive's `source`-folder.

- Similarly, in `\importmodule{some/path?Foo}` the path `some/path` refers to either the sub-directory and relative namespace path of the current directory and namespace outside of an archive, or relative to the current archive's top-level namespace and `source`-folder, respectively.

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

- Similarly, `\importmodule[Some/Archive]{some/path?Foo}` is resolved like the previous cases, but relative to the archive `Some/Archive` in the mathhub-directory.

- Finally, `\importmodule{full://uri?Foo}` naturally refers to the module `Foo` in the namespace `full://uri`. Since the file this module is declared

---

in can not be determined directly from the URI, the module must be in memory already, e.g. by being referenced earlier in the same document.

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

`\STEXexport`

`\importmodule` and `\usemodule` import all symbols, notations, semantic macros and (recursively) `\importmodule`s. If you want to additionally export e.g. convenience macros and other (sTEX) 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`.

For persistency reasons, everything in an `\STEXexport` is digested by TEXin the LATEX3-category code scheme. This means that the characters _ and : are considered *letters* and valid parts of control sequence names, and space characters are ignored entirely. For spaces, use the character ~ instead, and keep in mind, that if you want to use subscripts, you should use `\c_math_subscript_token` instead of _!

Also 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, rather, 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}{monoid}{\mathbb{Z}_{+,0}}[
2     universe = Int ,
3     op = addition ,
4     unit = zero
5 ]
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.

> **\instantiate** and `mathstructure` make use of the *Theories-as-Types* paradigm (see [MRK18]):
> `mathstructure{<name>}` simply creates 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** generates a constant whose definiens is a record term of type `Mod(<name>-structure)`, with the fields assigned based on the respective 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 \varinstantiate 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 $\varM{op}!:\funtype{\varM{universe},\varM{universe}}{\varM{universe}}$ ...
```

Output:

A monoid is a structure $M := \langle U, \circ, e \rangle$ such that $\circ : U \times U \rightarrow U$ ...

.

and

Example 28

Input:

```
1  \varinstantiate{varMb}{monoid}{M_2}[universe = Int]
2
3  Let $\varMb!:=\mathstruct{\varMb{universe},\varMb{op}!,\varMb{unit}}$
4 be a \symname{monoid} on $\Int$ ...
```

Output:

Let $M_2 := \langle \mathbb{Z}, \circ, e \rangle$ be a monoid on $\mathbb{Z}$ ...

.

We will return to these two 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 29

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:

34

.

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 30

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 31

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)

The stex-metatheory package contains sTeX 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). As such, it serves as the default meta theory for any sTeX module.

We can also see the stex-metatheory as a foundation of mathematics in the sense of [Rab15], albeit an informal one (the ones discussed there are all formal foundations). The state of the stex-metatheory is necessarily incomplete, and will stay so for a long while: It arises as a collection of empirically useful symbols that are collected as more and more mathematics are encoded in sTeX and are classified as foundational.

Formal 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.

We make this theory part of the sTeX collection rather than encoding it in sTeX itself[2]

EdN:2

---

[2]EdNote: MK: why? continue

36

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 32

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.

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

> **Example 33**
> 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` and does *not* contain a `?`, then SₜₑX checks all symbols currently in scope until it finds one, whose name is `string`. If `string` is of the form `pre?name`, SₜₑX first looks through all modules currently in scope, whose full URI ends with `pre`, and then looks for a symbol with name `name` in those. 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.

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 34**
> 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.

> ⚠ Note the difference in treating "arguments" between math mode and text mode. In math mode the (in this case two) tokens/groups following the `\addition` macro are treated as arguments to the addition function, whereas in text mode the group following `\addition` is taken to be the ad-hoc presentation. We drill in on this now.

`\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 usual syntax using `!`:

Example 35
Input:

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

Output:

```
Addition is...
```

.

Indeed, `\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).[3]

EdN:3

Example 36
Input:

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

Output:

---
[3]EDNOTE: MK: I do not understand why we have to/want to give the second arg*; I think this must be elaborated on.

> 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.[4]

The same syntax can be used in math mod as well. This allows us to spontaneously introduce new notations on the fly. We can activate it using the starred variants of semantic macros:

Example 37

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

---

[4]EDNOTE: MK: I do not understand this at all.

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 counterexamples, 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), as well as `title=` (for giving the paragraph a title), and finally `for=`.

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

Example 38

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, $2\cdot3$ 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.

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

\definiens

Additionally, sdefinition (and sparagraph with type=symdoc) introduces \definiens[<optional sym which marks up <code> as being the explicit *definiens* of <optional symbolname> (in case for= has multiple symbols).

All four statement environments – i.e. sdefinition, sassertion, sexample, and sparagraph – also take an optional parameter name= – if this one is given a value, the environment will generate a *symbol* by that name (but with no semantic macro). Not only does this allow for \symref et al, it allows us to resume our earlier example for monoids much more nicely:[5]

EdN:5

> **Example 39**
> Input:

---

[5]EDNOTE: MK: we should reference the example explicitly here.

```
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...

.

The main difference to before[6] is that the two `sassertion`s now have `name=` attributes. Thus the `mathstructure` monoid now 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}{monoid}{\mathbb{Z}_{+,0}}[
2      universe = Int ,
3      op = addition ,
4      unit = zero
5  ]
```

---

[6]EDNOTE: MK: reference

...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]

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

We will go over the general intuition by way of a running example:

```
1  \begin{sproof}[id=simple-proof]
2    {We prove that $\sum_{i=1}^n{2i-1}=n^{2}$ by induction over $n$}
3   \begin{spfcases}{For the induction we have to consider three cases:}
4    \begin{spfcase}{$n=1$}
5     \begin{spfstep}[type=inline] then we compute $1=1^2$\end{spfstep}
6    \end{spfcase}
7    \begin{spfcase}{$n=2$}
8      \begin{spfcomment}[type=inline]
9        This case is not really necessary, but we do it for the
10       fun of it (and to get more intuition).
11     \end{spfcomment}
12     \begin{spfstep}[type=inline] We compute $1+3=2^{2}=4$.\end{spfstep}
13   \end{spfcase}
14   \begin{spfcase}{$n>1$}
15     \begin{spfstep}[type=assumption,id=ind-hyp]
16       Now, we assume that the assertion is true for a certain $k\geq 1$,
17       i.e. $\sum_{i=1}^k{(2i-1)}=k^{2}$.
18     \end{spfstep}
19     \begin{spfcomment}
20       We have to show that we can derive the assertion for $n=k+1$ from
21       this assumption, i.e. $\sum_{i=1}^{k+1}{(2i-1)}=(k+1)^{2}$.
22     \end{spfcomment}
23     \begin{spfstep}
24       We obtain $\sum_{i=1}^{k+1}{2i-1}=\sum_{i=1}^k{2i-1}+2(k+1)-1$
25       \spfjust[method=arith:split-sum]{by splitting the sum}.
26     \end{spfstep}
27     \begin{spfstep}
28       Thus we have $\sum_{i=1}^{k+1}{(2i-1)}=k^2+2k+1$
29       \spfjust[method=fertilize]{by inductive hypothesis}.
30     \end{spfstep}
31     \begin{spfstep}[type=conclusion]
32       We can \spfjust[method=simplify]{simplify} the right-hand side to
33       ${k+1}^2$, which proves the assertion.
34     \end{spfstep}
35   \end{spfcase}
36   \begin{spfstep}[type=conclusion]
37     We have considered all the cases, so we have proven the assertion.
38   \end{spfstep}
39  \end{spfcases}
40 \end{sproof}
```

This yields the following result:

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

---

[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

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.

□

sproof The sproof 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 spfstep, spfcomment, and spfcases environments that are used to markup the proof steps.

\spfidea 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 same optional 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.

\spfjust   This evidence is marked up with the `\spfjust` macro in the stex-proofs 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 running example 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.

     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.

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

spfcases   The `spfcases` 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 `spfcases` environment are a sequence of case proofs marked up in the `spfcase` 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 `spfcase` environment is the same as that of a `sproof`, i.e. `spfstep`s, `spfcomment`s, and `spfcases` environments.

\spfcasesketch   `\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.

spfcomment   The `spfcomment` 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`.

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:

The stex-proofs 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` 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{}`.

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 supposed to look like.

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

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

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

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

After sTeX reads and processes the optional arguments for these environments, (some of) their values are stored in the macros \s*<field> (i.e. sexampleid, \sassertionname, etc.). It then checks for all the values <type> in the type=-list, whether an \stexpatch*[<type>] for the current environment has been called. If it finds one, it uses the 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-like 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 kinds of* sdefinitions to use a predefined definition-environment irrespective of their `type=`, then we can issue the following customization patch:

```
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 SₜₑX highlights variables, notation components, \symrefs and \definiendums, respectively.

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

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

For each of the four macros, there exists an additional macro that takes the full URI of the relevant symbol currently being highlighted as a second argument. That allows us to e.g. use pdf tooltips and links. For example, this document uses[7]

```
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

7.1 Tikzinput: Treating TIKZ code as images

`image`

The behavior of the ikzinput package is determined by whether the `image` option is given. If it is not, then the `tikz` package is loaded, all other options are passed on to it and `\tikzinput{⟨file⟩}` inputs the TIKZ file ⟨file⟩`.tex`; if not, only the `graphicx` package is loaded and `\tikzinput{⟨file⟩}` loads an image file ⟨file⟩`.`⟨ext⟩ generated from ⟨file⟩`.tex`.

The selective input functionality of the `tikzinput` package assumes that the TIKZ pictures are externalized into a standalone picture file, such as the following one

```
1 \documentclass{standalone}
2 \usepackage{tikz}
3 \usetikzpackage{...}
4 \begin{document}
5   \begin{tikzpicture}
6     ...
7   \end{tikzpicture}
8 \end{document}
```

The `standalone` class is a minimal LaTeX class that when loaded in a document that uses the `standalone` package: the preamble and the `documenat` environment are disregarded during loading, so they do not pose any problems. In effect, an `\input` of the file above only sees the `tikzpicture` environment, but the file itself is standalone in the sense that we can run LaTeX over it separately, e.g. for generating an image file from it.

`\tikzinput`
`\ctikzinput`

This is exactly where the `tikzinput` package comes in: it supplies the `\tikzinput` macro, which – depending on the `image` option – either directly inputs the TIKZ picture (source) or tries to load an image file generated from it.

Concretely, if the `image` option is not set for the `tikzinput` package, then `\tikzinput[⟨opt⟩]{⟨file⟩}` disregards the optional argument ⟨opt⟩ and inputs ⟨file⟩`.tex` via `\input` and resizes it to as specified in the `width` and `height` keys. If it is, `\tikzinput[⟨opt⟩]{⟨file⟩}` expands to `\includegraphics[⟨opt⟩]{⟨file⟩}`.

`\ctizkinput` is a version of `\tikzinput` that is centered.

`\mhtikzinput`	`\mhtizkinput` is a variant of `\tikzinput` that treats its file path argument as a relative
`\cmhtikzinput`	path in a math archive in analogy to `\inputref`. To give the archive path, we use the

`\mhtikzinput` is a variant of `\tikzinput` that treats its file path argument as a relative path in a math archive in analogy to `\inputref`. To give the archive path, we use the `mhrepos=` key. Again, `\cmhtikzinput` is a version of `\mhtikzinput` that is centered.

`\libusetikzlibrary`   Sometimes, we want to supply archive-specific TIKZ libraries in the `lib` folder of the archive or the `meta-inf/lib` of the archive group. Then we need an analogon to `\libinput` for `\usetikzlibrary`. The stex-tikzinput package provides the `libusetikzlibrary` for this purpose.

7.2 Modular Document Structuring

The `document-structure` 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.

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.

The `document-structure` package accepts 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`

sfragment   The structure of the document is given by nested `sfragment` environments. In the LaTeX route, the `sfragment` environment is flexibly mapped to sectioning commands, inducing the proper sectioning level from the nesting of `sfragment` environments. Correspondingly, the `sfragment` environment takes an optional key/value argument for metadata followed by a regular argument for the (section) title of the sfragment. The optional metadata argument has the keys `id` for an identifier, `creators` and `contributors` for the Dublin Core metadata [DCM03]. The option `short` allows to give a short title for the generated

EdN:8   section. If the title contains semantic macros, they need to be protected by `\protect`[8], and we need to give the `loadmodules` key it needs no value. For instance we would have

```
1 \begin{smodule}{foo}
2   \symdef{bar}{B^a_r}
3   ...
4   \begin{sfragment}[id=sec.barderiv,loadmodules]
5     {Introducing $\protect\bar$ Derivations}
```

---

[8]EDNOTE: MK: still?

51

STEX automatically computes the sectioning level, from the nesting of `sfragment` environments.

But sometimes, we want to skip levels (e.g. to use a `\subsection*` as an introduction for a chapter).

blindfragment   Therefore the `document-structure` package provides a variant `blindfragment` 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 `blindfragment` environment is useful e.g. for creating frontmatter at the correct level. The example below shows a typical setup for the outer document structure of a book with parts and chapters.

```
1 \begin{document}
2 \begin{blindfragment}
3 \begin{blindfragment}
4 \begin{frontmatter}
5 \maketitle\newpage
6 \begin{sfragment}{Preface}
7 ... <<preface>> ...
8 \end{sfragment}
9 \clearpage\setcounter{tocdepth}{4}\tableofcontents\clearpage
10 \end{frontmatter}
11 \end{blindfragment}
12 ... <<introductory remarks>> ...
13 \end{blindfragment}
14 \begin{sfragment}{Introduction}
15 ... <<intro>> ...
16 \end{sfragment}
17 ... <<more chapters>> ...
18 \bibliographystyle{alpha}\bibliography{kwarc}
19 \end{document}
```

Here we use two levels of `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 `blindfragment` makes sure that the introductory remarks become a "chapter" instead of a "part".

- The inner one groups the frontmatter[3] and makes the preface of the book a section-level construct.[9]

EdN:9

\skipfragment   The `\skipfragment` "skips an `sfragment`", 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 `\skipfragment`.

---

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

[9] EDNOTE: MK: We need a substitute for the "Note that here the `display=flow` on the `sfragment` environment prevents numbering as is traditional for prefaces."

52

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

`\prematurestop` `\afterprematurestop`	For prematurely stopping the formatting of a document, SʇEX provides the `\prematurestop` macro. It can be used everywhere in a document and ignores all input after that – backing out of the `sfragment` 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.

$\quad$ `\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].

$\quad$ 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 SʇEX preamble of the course notes file.

`\setSGvar` `\useSGvar`	`\setSGvar{`⟨*vname*⟩`}{`⟨*text*⟩`}` to set the global variable ⟨*vname*⟩ to ⟨*text*⟩ and `\useSGvar{`⟨*vname*⟩`}` to reference it.

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

7.3 Slides and Course Notes

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

$\quad$ 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 SʇEX and 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).

$\quad$ 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.

The notesslides class takes a variety of class options:

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

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

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

Slides are represented with the `frame` environment just like in the `beamer` class, see [Tanb] for details. The notesslides class adds the `note` environment for encapsulating the course note fragments.[4]

> 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.

By interleaving the `frame` and `note` environments, we can build course notes as shown here:

```
1  \ifnotes\maketitle\else
2  \frame[noframenumbering]\maketitle\fi
3
4  \begin{note}
5    We start this course with ...
6  \end{note}
7
8  \begin{frame}
9    \frametitle{The first slide}
10   ...
11 \end{frame}
12 \begin{note}
13   ... and more explanatory text
14 \end{note}
15
16 \begin{frame}
17   \frametitle{The second slide}
18   ...
19 \end{frame}
20 ...
```

Note the use of the `\ifnotes` conditional, which allows different treatment between `notes` and `slides` mode – manually setting `\notestrue` or `\notesfalse` is strongly discouraged however.

---

[4]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.

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

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

\inputref*

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

nexample, nsproof, nassertion

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

\setslidelogo

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

\setsource

The default footer line of the notesslides package mentions copyright and licensing. In the beamer class, `\source` stores the author's name as the copyright holder . By default it is *Michael Kohlhase* in the notesslides package since he is the main user and designer of this package. `\setsource{⟨name⟩}` can change the writer's name.

\setlicensing

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.

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 STEX notes.

`\frameimage` `\mhframeimage`	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.

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

```
1 \frameimage{\MathHub{fooMH/bar/source/baz/foobar}}
```

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

```
1 \mhframeimage[fooMH/bar]{baz/foobar}
```

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

If `baz/foobar` is the "current module", i.e. if we are on the MathHub path `...MathHub/fooMH/bar...`, then stating the repository in the first optional argument is redundant, so we can just use

```
1 \mhframeimage{baz/foobar}
```

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

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:

```
1 \excursion{founif}{../ex/founif}{We will cover first-order unification in}
2 ...
3 \begin{appendix}\printexcursions\end{appendix}
```

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

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

`\activateexcursion` `\printexcursion` `\excursionref`
Here `\activateexcursion{⟨path⟩}` augments the `\printexcursions` macro by a call `\inputref{⟨path⟩}`. In this way, the `\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{⟨label⟩}` for that.

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

```
1 \begin{note}
2 \begin{sfragment}[id=<id>]{Excursions}
3   \inputref{<path>}
4   \printexcursions
5 \end{sfragment}
6 \end{note}
```

> When option `book` which uses `\pagestyle{headings}` is given and semantic macros are given in the `sfragment` 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 `document-structure` package.

7.4 Representing Problems and Solutions

The `problem` package supplies an infrastructure that allows specify problem. Problems are text fragments that come with auxiliary functions: hints, notes, and solutions[5]. 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.

solutions
notes
hints
gnotes
pts
min
boxed
test

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

---

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

Example 40

Input:

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

Output:

Problem 7.4.1 (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.

Grading: if they do not give the justification deduct 5 pts

.

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

hint,exnote,gnote The `hint` and `exnote` environments can be used in a `problem` environment to give hints and to make notes that elaborate certain aspects of the problem. The `gnote` (grading notes) environment can be used to document situations that may arise in grading.

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

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.

mcb Multiple choice blocks can be formatted using the mcb environment, in which single choices are marked up with \mcc macro.

\mcc \mcc[⟨*keyvals*⟩]{⟨*text*⟩} 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.

If we start the solutions, then we get

Example 41

Input:

```
 1 \startsolutions
 2 \begin{sproblem}[title=Functions,name=functions1]
 3   What is the keyword to introduce a function definition in python?
 4   \begin{mcb}
 5     \mcc[T]{def}
 6     \mcc[F,feedback=that is for C and C++]{function}
 7     \mcc[F,feedback=that is for Standard ML]{fun}
 8     \mcc[F,Ftext=Noooooooooo,feedback=that is for Java]{public static void}
 9   \end{mcb}
10 \end{sproblem}
```

Output:

> **Problem 7.4.2 (Functions)**
> What is the keyword to introduce a function definition in python?
>
> ☐ def
>   **(true)**
>
> ☐ function
>   **(false)** *(that is for C and C++)*
>
> ☐ fun
>   **(false)** *(that is for Standard ML)*
>
> ☐ public static void
>   **(false)** *(that is for Java)*

EdN:10 ˙without solutions (that is what the students see during the exam/quiz)[10]

---

[10]EdNote: MK: that did not work!

59

Example 42

Input:

```
 1 \stopsolutions
 2 \begin{sproblem}[title=Functions,name=functions1]
 3   What is the keyword to introduce a function definition in python?
 4   \begin{mcb}
 5     \mcc[T]{def}
 6     \mcc[F,feedback=that is for C and C++]{function}
 7     \mcc[F,feedback=that is for Standard ML]{fun}
 8     \mcc[F,Ftext=Noooooooooo,feedback=that is for Java]{public static void}
 9   \end{mcb}
10 \end{sproblem}
```

Output:

> **Problem 7.4.3 (Functions)**
> What is the keyword to introduce a function definition in python?
>
> ☐ def
>   **(true)**
>
> ☐ function
>   **(false)** *(that is for C and C++)*
>
> ☐ fun
>   **(false)** *(that is for Standard ML)*
>
> ☐ public static void
>   **(false)** *(that is for Java)*

.

\includeproblem 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.

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

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

7.5 Homeworks, Quizzes and Exams

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 roblem package. It is designed to be compatible with `problems.sty`, and inherits some of the functionality.

<div style="margin-left: 2em;">

solutions
notes
hints
gnotes
pts
min

The wexam 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).

assignment
number

This package supplies the `assignment` environment that groups problems into assignment sheets. It takes an optional KeyVal argument with the keys `number` (for the assignment number; if none is given, 1 is assumed as the default or — in multi-assignment documents — the ordinal of the `assignment` environment), `title` (for the assignment

title
type
given
due
multiple

title; this is referenced in the title of the assignment sheet), `type` (for the assignment type; e.g. "quiz", or "homework"), `given` (for the date the assignment was given), and `due` (for the date the assignment is due).

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

test

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

\testspace
\testnewpage
\testemptypage

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

testheading
duration
min
reqpts

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

</div>

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

Will result in

Name: Matriculation Number:

320101 General Computer Science (Fall 2010)

2022-05-23

You have one hour (sharp) for the test;
Write the solutions to the sheet.
The estimated time for solving this exam is 60 minutes, leaving you
0 minutes for revising your exam.
You can reach 40 points if you solve all problems. You will only need
27 points for a perfect score, i.e. 13 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.	7.4.1	7.4.2	7.4.3	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	10			4	4	6	6	4	4	2	40	
reached												

good luck

EdN:11   [11]

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

---

[11]EDNOTE: MK: The first three "problems" come from the stex examples above, how do we get rid of
this?

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:nnn {⟨property⟩} {⟨resource⟩} {⟨content⟩}`
`\stex_annotate_invisible:nnn`	
`\stex_annotate_invisible:n`	

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:NTF ⋆

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

\c_stex_pwd_seq
\c_stex_pwd_str
\c_stex_mainfile_seq
\c_stex_mainfile_str

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

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

\stex_filestack_push:n Push and pop (repsectively) 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 We determine the path to the local MathHub folder via one of four means, in order of
\c_stex_mathhub_seq precedence:
\c_stex_mathhub_str

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 ⟨*<URI>*⟩ is stored in four macros. All modifications of these macros are global:

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

name The *name* of the module,

ns the *namespace* in field `ns`,

file the *file* containing the module, as a sequence of path fragments

lang the module's *language*,

sig the language of the signature module, if the current file is a translation from some other language,

deprecate if this module is deprecated, the module that replaces it,

meta the metatheory of the module.

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

`\c_stex_module_<URI>_constants`

The names of all constants declared in the module

`\c_stex_module_<URI>_constants`

The full URIs of all modules imported in this module

`\l_stex_current_module_str`

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

`\l_stex_all_modules_seq`

Stores full URIs for all modules currently in scope.

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

Conditional for whether we are currently in a module

`\stex_if_module_exists_p:n` ⋆
`\stex_if_module_exists: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.

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}`.

\STEXInternalTermMathOMSiiii
\STEXInternalTermMathOMAiiii
\STEXInternalTermMathOMBiiii

`⟨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.

\STEXInternalTermMathArgiii

`\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*⟩.

\STEXInternalTermMathAssocArgiiii	\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

81

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

Chapter 18

sTeX-Metatheory

18.1 Symbols

Part III
Extensions

Chapter 19

Tikzinput: Treating TIKZ code as images

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

Chapter 21

NotesSlides – Slides and Course Notes

Chapter 22

`problem.sty`: An Infrastructure for formatting Problems

Chapter 23

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

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
23
24 \clist_if_empty:NT \c_stex_languages_clist {
25   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
26   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
27   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
28   \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
29     \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
30       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
```

```
31        }
32      }
33    \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
34    \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
35      \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
36      \prop_if_in:NoT \c_stex_languages_prop \l_tmpa_str {
37        \stex_debug:nn{language} {Language~\l_tmpa_str~
38          inferred~from~file~name}
39        \exp_args:NNo \stex_set_language:Nn \l_tmpa_str \l_tmpa_str
40      }
41    }
42  }
43  ⟨/cls⟩
```

24.2   Preliminaries

```
44  ⟨*package⟩
45
46  %%%%%%%%%%%%   basics.dtx   %%%%%%%%%%%%
47
48  \RequirePackage{expl3,l3keys2e,ltxcmds}
49  \ProvidesExplPackage{stex}{2022/03/03}{3.1.0}{sTeX package}
50
51  \bool_if_exist:NF \c_stex_document_class_bool {
52    \bool_set_false:N \c_stex_document_class_bool
53    \RequirePackage{standalone}
54  }
55
56  \message{^^J
57    *****************************^^J
58    *~This~is~sTeX~version~3.1.0~*^^J
59    *****************************^^J
60  ^^J}
61
62  %\RequirePackage{morewrites}
63  %\RequirePackage{amsmath}
64
```

Package options:

```
65  \keys_define:nn { stex } {
66    debug      .clist_set:N  = \c_stex_debug_clist ,
67    lang       .clist_set:N  = \c_stex_languages_clist ,
68    mathhub    .tl_set_x:N   = \mathhub ,
69    usesms     .bool_set:N   = \c_stex_persist_mode_bool ,
70    writesms   .bool_set:N   = \c_stex_persist_write_mode_bool ,
71    image      .bool_set:N   = \c_tikzinput_image_bool,
72    unknown    .code:n       = {}
73  }
74  \ProcessKeysOptions { stex }
```

\stex   The STeXlogo:

\sTeX

```
75  \RequirePackage{stex-logo} % externalized for backwards-compatibility reasons
```

(*End definition for* \stex *and* \sTeX*. These functions are documented on page 64.*)

94

24.3 Messages and logging

76 ⟨@@=stex_log⟩

Warnings and error messages

```
77 \msg_new:nnn{stex}{error/unknownlanguage}{
78   Unknown~language:~#1
79 }
80 \msg_new:nnn{stex}{warning/nomathhub}{
81   MATHHUB~system~variable~not~found~and~no~
82   \detokenize{\mathhub}-value~set!
83 }
84 \msg_new:nnn{stex}{error/deactivated-macro}{
85   The~\detokenize{#1}~command~is~only~allowed~in~#2!
86 }
```

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

```
87 \cs_new_protected:Nn \stex_debug:nn {
88   \clist_if_in:NnTF \c_stex_debug_clist { all } {
89     \msg_set:nnn{stex}{debug / #1}{
90       \\Debug~#1:~#2\\
91     }
92     \msg_none:nn{stex}{debug / #1}
93   }{
94     \clist_if_in:NnT \c_stex_debug_clist { #1 } {
95       \msg_set:nnn{stex}{debug / #1}{
96         \\Debug~#1:~#2\\
97       }
98       \msg_none:nn{stex}{debug / #1}
99     }
100   }
101 }
```

(*End definition for* \stex_debug:nn. *This function is documented on page 64.*)

Redirecting messages:

```
102 \clist_if_in:NnTF \c_stex_debug_clist {all} {
103     \msg_redirect_module:nnn{ stex }{ none }{ term }
104 }{
105   \clist_map_inline:Nn \c_stex_debug_clist {
106     \msg_redirect_name:nnn{ stex }{ debug / ##1 }{ term }
107   }
108 }
109
110 \stex_debug:nn{log}{debug~mode~on}
```

24.4 HTML Annotations

111 ⟨@@=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.

```
112 \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`

```
113 \cs_new_protected:Nn \_stex_html_checkempty:n {
114   \tl_set:Nn \l_stex_html_arg_tl { #1 }
115   \tl_if_empty:NT \l_stex_html_arg_tl {
116     \tl_set_eq:NN \l_stex_html_arg_tl \c_stex_html_emptyarg_tl
117   }
118 }
```

(*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

```
119 \bool_new:N \_stex_html_do_output_bool
120 \bool_set_true:N \_stex_html_do_output_bool
121
122 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
123   \bool_if:nTF \_stex_html_do_output_bool
124     \prg_return_true: \prg_return_false:
125 }
```

(*End definition for* `\stex_if_do_html:TF`*. This function is documented on page* *64.*)

`\stex_suppress_html:n`

Whether to (locally) produce HTML output

```
126 \cs_new_protected:Nn \stex_suppress_html:n {
127   \exp_args:Nne \use:nn {
128     \bool_set_false:N \_stex_html_do_output_bool
129     #1
130   }{
131     \stex_if_do_html:T {
132       \bool_set_true:N \_stex_html_do_output_bool
133     }
134   }
135 }
```

(*End definition for* `\stex_suppress_html:n`*. This function is documented on page* *64.*)

`\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.

```
136 \tl_if_exist:NF\stex@backend{
137   \ifcsname if@rustex\endcsname
138     \def\stex@backend{rustex}
139   \else
140     \ifcsname if@latexml\endcsname
141       \def\stex@backend{latexml}
142     \else
143       \def\stex@backend{pdflatex}
144     \fi
145   \fi
146 }
147 \input{stex-backend-\stex@backend.cfg}
148
149 \newif\ifstexhtml
150 \stex_html_backend:TF\stexhtmltrue\stexhtmlfalse
151
```

(*End definition for* `\stex_annotate:nnn` *,* `\stex_annotate_invisible:n` *, and* `\stex_annotate_invisible:nnn`.
These functions are documented on page 65.)

24.5 Babel Languages

152 ⟨@@=stex_language⟩

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

```
153 \exp_args:NNx \prop_const_from_keyval:Nn \c_stex_languages_prop { \tl_to_str:n {
154     en = english ,
155     de = ngerman ,
156     ar = arabic ,
157     bg = bulgarian ,
158     ru = russian ,
159     fi = finnish ,
160     ro = romanian ,
161     tr = turkish ,
162     fr = french
163 }}
164
165 \exp_args:NNx \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop { \tl_to_str:n {
166     english  = en ,
167     ngerman  = de ,
168     arabic   = ar ,
169     bulgarian = bg ,
170     russian  = ru ,
171     finnish  = fi ,
172     romanian = ro ,
173     turkish  = tr ,
174     french   = fr
175 }}
176 % todo: chinese simplified (zhs)
177 %       chinese traditional (zht)
```

(*End definition for* `\c_stex_languages_prop` *and* `\c_stex_language_abbrevs_prop`. *These variables are
documented on page* 65.)

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

```
178 \cs_new_protected:Nn \stex_set_language:Nn {
179     \str_set:Nx \l_tmpa_str {#2}
180     \prop_get:NoNT \c_stex_languages_prop \l_tmpa_str #1 {
181         \ifx\@onlypreamble\@notprerr
182             \ltx@ifpackageloaded{babel}{
183                 \exp_args:No \selectlanguage #1
184             }{}
185         \else
186             \exp_args:No \str_if_eq:nnTF #1 {turkish} {
187                 \RequirePackage[#1,shorthands=:!]{babel}
188             }{
189                 \RequirePackage[#1]{babel}
190             }
191         \fi
192     }
193 }
194
```

```
195 \clist_if_empty:NF \c_stex_languages_clist {
196   \bool_set_false:N \l_tmpa_bool
197   \clist_clear:N \l_tmpa_clist
198   \clist_map_inline:Nn \c_stex_languages_clist {
199     \str_set:Nx \l_tmpa_str {#1}
200     \str_if_eq:nnT {#1}{tr}{
201       \bool_set_true:N \l_tmpa_bool
202     }
203     \prop_get:NoNTF \c_stex_languages_prop \l_tmpa_str \l_tmpa_str {
204       \clist_put_right:No \l_tmpa_clist \l_tmpa_str
205     } {
206       \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
207     }
208   }
209   \stex_debug:nn{lang} {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
210   \bool_if:NTF \l_tmpa_bool {
211     \RequirePackage[\clist_use:Nn \l_tmpa_clist,,shorthands=:!]{babel}
212   }{
213     \RequirePackage[\clist_use:Nn \l_tmpa_clist,]{babel}
214   }
215 }
216
217 \AtBeginDocument{
218   \stex_html_backend:T {
219     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
220     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
221     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
222     \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
223     \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
224       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
225       \stex_debug:nn{basics} {Language~\l_tmpa_str~
226         inferred~from~file~name}
227       \stex_annotate_invisible:nnn{language}{ \l_tmpa_str }{}
228     }
229   }
230 }
```

24.6 Persistence

```
231 ⟨@@=stex_persist⟩
232 \bool_if:NTF \c_stex_persist_mode_bool {
233   \def \stex_persist:n #1 {}
234   \def \stex_persist:x #1 {}
235 }{
236   \bool_if:NTF \c_stex_persist_write_mode_bool {
237   \iow_new:N \c__stex_persist_iow
238   \iow_open:Nn \c__stex_persist_iow{\jobname.sms}
239   \AtEndDocument{
240     \iow_close:N \c__stex_persist_iow
241   }
242   \cs_new_protected:Nn \stex_persist:n {
243     \tl_set:Nn \l_tmpa_tl { #1 }
244     \regex_replace_all:nnN { \cP\# } { \cO\# } \l_tmpa_tl
```

```
245    \regex_replace_all:nnN { \  } { \~ } \l_tmpa_tl
246    \exp_args:NNo \iow_now:Nn \c__stex_persist_iow \l_tmpa_tl
247  }
248  \cs_generate_variant:Nn \stex_persist:n {x}
249  }{
250    \def \stex_persist:n #1 {}
251    \def \stex_persist:x #1 {}
252  }
253 }
```

24.7 Auxiliary Methods

```
254 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
255  \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
256  \def#1{
257    \msg_error:nnnn{stex}{error/deactivated-macro}{\detokenize{#1}}{#2}
258  }
259 }
```

(*End definition for* \stex_deactivate_macro:Nn. *This function is documented on page 65.*)

```
260 \cs_new_protected:Nn \stex_reactivate_macro:N {
261  \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname
262 }
```

(*End definition for* \stex_reactivate_macro:N. *This function is documented on page 65.*)

```
263 \protected\def\ignorespacesandpars{
264  \begingroup\catcode13=10\relax
265  \@ifnextchar\par{
266    \endgroup\expandafter\ignorespacesandpars\@gobble
267  }{
268    \endgroup
269  }
270 }
271
272 \cs_new_protected:Nn \stex_copy_control_sequence:NNN {
273  \tl_set:Nx \_tmp_args_tl {\cs_argument_spec:N #2}
274  \exp_args:NNo \tl_remove_all:Nn \_tmp_args_tl \c_hash_str
275  \int_set:Nn \l_tmpa_int {\tl_count:N \_tmp_args_tl}
276
277  \tl_clear:N \_tmp_args_tl
278  \int_step_inline:nn \l_tmpa_int {
279    \tl_put_right:Nx \_tmp_args_tl {{\exp_not:n{####}\exp_not:n{##1}}}
280  }
281
282  \tl_set:Nn #3 {\cs_generate_from_arg_count:NNnn #1 \cs_set:Npn}
283  \tl_put_right:Nx #3 { {\int_use:N \l_tmpa_int}{
284    \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
285    \exp_after:wN\exp_after:wN\exp_after:wN {
286      \exp_after:wN #2 \_tmp_args_tl
```

99

```
287          }
288        }}
289    }
290    \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {cNN}
291    \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {NcN}
292    \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {ccN}
293
294    \cs_new_protected:Nn \stex_copy_control_sequence_ii:NNN {
295      \tl_set:Nx \_tmp_args_tl {\cs_argument_spec:N #2}
296      \exp_args:NNo \tl_remove_all:Nn \_tmp_args_tl \c_hash_str
297      \int_set:Nn \l_tmpa_int {\tl_count:N \_tmp_args_tl}
298
299      \tl_clear:N \_tmp_args_tl
300      \int_step_inline:nn \l_tmpa_int {
301        \tl_put_right:Nx \_tmp_args_tl {{\exp_not:n{########}\exp_not:n{##1}}}
302      }
303
304      \edef \_tmp_args_tl {
305        \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
306        \exp_after:wN\exp_after:wN\exp_after:wN {
307          \exp_after:wN #2 \_tmp_args_tl
308        }
309      }
310
311      \exp_after:wN \def \exp_after:wN \_tmp_args_tl
312      \exp_after:wN ##\exp_after:wN 1 \exp_after:wN ##\exp_after:wN 2
313      \exp_after:wN  { \_tmp_args_tl }
314
315      \edef \_tmp_args_tl {
316        \exp_after:wN \exp_not:n \exp_after:wN {
317          \_tmp_args_tl {####1}{####2}
318        }
319      }
320
321      \tl_set:Nn #3 {\cs_generate_from_arg_count:NNnn #1 \cs_set:Npn}
322      \tl_put_right:Nx #3 { {\int_use:N \l_tmpa_int}{
323        \exp_after:wN\exp_not:n\exp_after:wN{\_tmp_args_tl}
324      }}
325    }
326
327    \cs_generate_variant:Nn \stex_copy_control_sequence_ii:NNN {cNN}
328    \cs_generate_variant:Nn \stex_copy_control_sequence_ii:NNN {NcN}
329    \cs_generate_variant:Nn \stex_copy_control_sequence_ii:NNN {ccN}
```

(*End definition for* \ignorespacesandpars. *This function is documented on page 65.*)

\MMTrule

```
330    \NewDocumentCommand \MMTrule {m m}{
331      \seq_set_split:Nnn \l_tmpa_seq , {#2}
332      \int_zero:N \l_tmpa_int
333      \stex_annotate_invisible:nnn{mmtrule}{scala://#1}{
334        \seq_if_empty:NF \l_tmpa_seq {
335          $\seq_map_inline:Nn \l_tmpa_seq {
336            \int_incr:N \l_tmpa_int
```

```
337        \stex_annotate:nnn{arg}{i\int_use:N \l_tmpa_int}{##1}
338      }$
339    }
340  }
341 }

343 \NewDocumentCommand \MMTinclude {m}{
344   \stex_annotate_invisible:nnn{import}{#1}{}
345 }

347 \tl_new:N \g_stex_document_title
348 \cs_new_protected:Npn \STEXtitle #1 {
349   \tl_if_empty:NT \g_stex_document_title {
350     \tl_gset:Nn \g_stex_document_title { #1 }
351   }
352 }
353 \cs_new_protected:Nn \stex_document_title:n {
354   \tl_if_empty:NT \g_stex_document_title {
355     \tl_gset:Nn \g_stex_document_title { #1 }
356     \stex_annotate_invisible:n{\noindent
357       \stex_annotate:nnn{doctitle}{}{ #1 }
358     \par}
359   }
360 }
361 \AtBeginDocument {
362   \let \STEXtitle \stex_document_title:n
363   \tl_if_empty:NF \g_stex_document_title {
364     \stex_annotate_invisible:n{\noindent
365       \stex_annotate:nnn{doctitle}{}{ \g_stex_document_title }
366     \par}
367   }
368   \let\_stex_maketitle:\maketitle
369   \def\maketitle{
370     \tl_if_empty:NF \@title {
371       \exp_args:No \stex_document_title:n \@title
372     }
373     \_stex_maketitle:
374   }
375 }

377 \cs_new_protected:Nn \stex_par: {
378   \mode_if_vertical:F{
379     \if@minipage\else\if@nobreak\else\par\fi\fi
380   }
381 }

383 ⟨/package⟩
```

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

Chapter 25

sTEX
-MathHub Implementation

```
384 ⟨∗package⟩
385
386 %%%%%%%%%%%    mathhub.dtx    %%%%%%%%%%%
387
388 ⟨@@=stex_path⟩
```

Warnings and error messages
```
389 \msg_new:nnn{stex}{error/norepository}{
390   No~archive~#1~found~in~#2
391 }
392 \msg_new:nnn{stex}{error/notinarchive}{
393   Not~currently~in~an~archive,~but~\detokenize{#1}~
394   needs~one!
395 }
396 \msg_new:nnn{stex}{error/nofile}{
397   \detokenize{#1}~could~not~find~file~#2
398 }
399 \msg_new:nnn{stex}{error/twofiles}{
400   \detokenize{#1}~found~two~candidates~for~#2
401 }
```

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

```
402 \cs_new_protected:Nn \stex_path_from_string:Nn {
403   \str_set:Nx \l_tmpa_str { #2 }
404   \str_if_empty:NTF \l_tmpa_str {
405     \seq_clear:N #1
406   }{
407     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
408     \sys_if_platform_windows:T{
409       \seq_clear:N \l_tmpa_tl
```

```
410      \seq_map_inline:Nn #1 {
411        \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
412        \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
413      }
414      \seq_set_eq:NN #1 \l_tmpa_tl
415    }
416    \stex_path_canonicalize:N #1
417  }
418 }
419
```

(*End definition for* `\stex_path_from_string:Nn`*. This function is documented on page* *66*.)

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

```
420 \cs_new_protected:Nn \stex_path_to_string:NN {
421   \exp_args:NNe \str_set:Nn #2 { \seq_use:Nn #1 / }
422 }
423
424 \cs_new:Nn \stex_path_to_string:N {
425   \seq_use:Nn #1 /
426 }
```

(*End definition for* `\stex_path_to_string:NN` *and* `\stex_path_to_string:N`*. These functions are documented on page* *66*.)

`\c__stex_path_dot_str`
`\c__stex_path_up_str`

. and .., respectively.

```
427 \str_const:Nn \c__stex_path_dot_str {.}
428 \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.

```
429 \cs_new_protected:Nn \stex_path_canonicalize:N {
430   \seq_if_empty:NF #1 {
431     \seq_clear:N \l_tmpa_seq
432     \seq_get_left:NN #1 \l_tmpa_tl
433     \str_if_empty:NT \l_tmpa_tl {
434       \seq_put_right:Nn \l_tmpa_seq {}
435     }
436     \seq_map_inline:Nn #1 {
437       \str_set:Nn \l_tmpa_tl { ##1 }
438       \str_if_eq:NNF \l_tmpa_tl \c__stex_path_dot_str {
439         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
440           \seq_if_empty:NTF \l_tmpa_seq {
441             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
442               \c__stex_path_up_str
443             }
444           }{
445             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
446             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
447               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
448                 \c__stex_path_up_str
449               }
450             }{
```

```
451                    \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
452                }
453            }
454        }{
455            \str_if_empty:NF \l_tmpa_tl {
456                \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
457            }
458        }
459        }
460    }
461    \seq_gset_eq:NN #1 \l_tmpa_seq
462 }
463 }
```

(*End definition for* `\stex_path_canonicalize:N`*. This function is documented on page 66.*)

```
464 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
465    \seq_if_empty:NTF #1 {
466        \prg_return_false:
467    }{
468        \seq_get_left:NN #1 \l_tmpa_tl
469        \sys_if_platform_windows:TF{
470            \str_if_in:NnTF \l_tmpa_tl {:}{
471                \prg_return_true:
472            }{
473                \prg_return_false:
474            }
475        }{
476            \str_if_empty:NTF \l_tmpa_tl {
477                \prg_return_true:
478            }{
479                \prg_return_false:
480            }
481        }
482    }
483 }
```

(*End definition for* `\stex_path_if_absolute:NTF`*. This function is documented on page 66.*)

25.2 PWD and kpsewhich

```
484 \str_new:N\l_stex_kpsewhich_return_str
485 \cs_new_protected:Nn \stex_kpsewhich:n {\begingroup
486    \catcode`\ =12
487    \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
488    \tl_gset_eq:NN \l_tmpa_tl \l_tmpa_tl
489    \endgroup
490    \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
491    \tl_trim_spaces:N \l_stex_kpsewhich_return_str
492 }
```

(*End definition for* `\stex_kpsewhich:n`*. This function is documented on page 66.*)

We determine the PWD

`\c_stex_pwd_seq`
`\c_stex_pwd_str`

```
493 \sys_if_platform_windows:TF{
494   \begingroup\escapechar=-1\catcode`\\=12
495   \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
496   \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
497   \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_
498 }{
499   \stex_kpsewhich:n{-var-value~PWD}
500 }
501
502 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
503 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
504 \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 66.*)

25.3   File Hooks and Tracking

505 ⟨@@=stex_files⟩

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

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

`\g__stex_files_stack`   keeps track of file changes

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

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

`\c_stex_mainfile_seq`
`\c_stex_mainfile_str`

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

(*End definition for* `\c_stex_mainfile_seq` *and* `\c_stex_mainfile_str`*. These variables are documented on page 66.*)

`\g_stex_currentfile_seq`

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

(*End definition for* `\g_stex_currentfile_seq`*. This variable is documented on page 67.*)

`\stex_filestack_push:n`

```
511 \cs_new_protected:Nn \stex_filestack_push:n {
512   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
513   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
514     \stex_path_from_string:Nn\g_stex_currentfile_seq{
515       \c_stex_pwd_str/#1
516     }
```

```
517    }
518    \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
519    \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
520 }
```

(*End definition for* \stex_filestack_push:n*. This function is documented on page 67.*)

\stex_filestack_pop:

```
521 \cs_new_protected:Nn \stex_filestack_pop: {
522    \seq_if_empty:NF\g__stex_files_stack{
523       \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
524    }
525    \seq_if_empty:NTF\g__stex_files_stack{
526       \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
527    }{
528       \seq_get:NN\g__stex_files_stack\l_tmpa_seq
529       \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
530    }
531 }
```

(*End definition for* \stex_filestack_pop:*. This function is documented on page 67.*)

Hooks for the current file:

```
532 \AddToHook{file/before}{
533    \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
534 }
535 \AddToHook{file/after}{
536    \stex_filestack_pop:
537 }
```

25.4 MathHub Repositories

```
538 ⟨@@=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.

```
539 \str_if_empty:NTF\mathhub{
540    \sys_if_platform_windows:TF{
541       \begingroup\escapechar=-1\catcode`\\=12
542       \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str MATHHUB\c_percent_str}
543       \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
544       \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_ste
545    }{
546       \stex_kpsewhich:n{-var-value~MATHHUB}
547    }
548    \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
549
550    \str_if_empty:NT \c_stex_mathhub_str {
551       \sys_if_platform_windows:TF{
552          \begingroup\escapechar=-1\catcode`\\=12
553          \exp_args:Nx\stex_kpsewhich:n{-var-value~HOME}
554          \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
555          \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_s
556       }{
```

106

```
557        \stex_kpsewhich:n{-var-value~HOME}
558      }
559      \ior_open:NnT \l_tmpa_ior{\l_stex_kpsewhich_return_str / .stex / mathhub.path}{
560        \begingroup\escapechar=-1\catcode`\\=12
561        \ior_str_get:NN \l_tmpa_ior \l_tmpa_str
562        \sys_if_platform_windows:T{
563          \exp_args:NNx\str_replace_all:Nnn\l_tmpa_str{\c_backslash_str}/
564        }
565        \str_gset_eq:NN \c_stex_mathhub_str\l_tmpa_str
566        \endgroup
567        \ior_close:N \l_tmpa_ior
568      }
569    }
570    \str_if_empty:NTF\c_stex_mathhub_str{
571      \msg_warning:nn{stex}{warning/nomathhub}
572    }{
573      \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
574      \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
575    }
576  }{
577    \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
578    \stex_path_if_absolute:NF \c_stex_mathhub_seq {
579      \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
580        \c_stex_pwd_str/\mathhub
581      }
582    }
583    \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
584    \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
585  }
```

(*End definition for* \mathhub, \c_stex_mathhub_seq, *and* \c_stex_mathhub_str. *These variables are documented on page* 67.)

__stex_mathhub_do_manifest:n   Checks whether the manifest for archive #1 already exists, and if not, finds and parses the corresponding manifest file

```
586  \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
587    \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
588      \str_set:Nx \l_tmpa_str { #1 }
589      \prop_new:c { c_stex_mathhub_#1_manifest_prop }
590      \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
591      \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
592      \__stex_mathhub_find_manifest:N \l_tmpa_seq
593      \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
594        \msg_error:nnxx{stex}{error/norepository}{#1}{
595          \stex_path_to_string:N \c_stex_mathhub_str
596        }
597        \input{Fatal~Error!}
598      } {
599        \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
600      }
601    }
602  }
```

(*End definition for* __stex_mathhub_do_manifest:n.)

603 `\seq_new:N\l__stex_mathhub_manifest_file_seq`

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

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

```
604 \cs_new_protected:Nn \__stex_mathhub_find_manifest:N {
605   \seq_set_eq:NN\l_tmpa_seq #1
606   \bool_set_true:N\l_tmpa_bool
607   \bool_while_do:Nn \l_tmpa_bool {
608     \seq_if_empty:NTF \l_tmpa_seq {
609       \bool_set_false:N\l_tmpa_bool
610     }{
611       \file_if_exist:nTF{
612         \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
613       }{
614         \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
615         \bool_set_false:N\l_tmpa_bool
616       }{
617         \file_if_exist:nTF{
618           \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
619         }{
620           \seq_put_right:Nn\l_tmpa_seq{META-INF}
621           \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
622           \bool_set_false:N\l_tmpa_bool
623         }{
624           \file_if_exist:nTF{
625             \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
626           }{
627             \seq_put_right:Nn\l_tmpa_seq{meta-inf}
628             \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
629             \bool_set_false:N\l_tmpa_bool
630           }{
631             \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
632           }
633         }
634       }
635     }
636   }
637   \seq_set_eq:NN\l__stex_mathhub_manifest_file_seq\l_tmpa_seq
638 }
```

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

\c__stex_mathhub_manifest_ior File variable used for `MANIFEST`-files

639 `\ior_new:N \c__stex_mathhub_manifest_ior`

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

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

```
640 \cs_new_protected:Nn \__stex_mathhub_parse_manifest:n {
641   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
642   \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
```

```
643    \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
644      \str_set:Nn \l_tmpa_str {##1}
645      \exp_args:NNoo \seq_set_split:Nnn
646          \l_tmpb_seq \c_colon_str \l_tmpa_str
647      \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
648        \exp_args:NNe \str_set:Nn \l_tmpb_tl {
649          \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
650        }
651        \exp_args:No \str_case:nnTF \l_tmpa_tl {
652          {id} {
653            \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
654              { id } \l_tmpb_tl
655          }
656          {narration-base} {
657            \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
658              { narr } \l_tmpb_tl
659          }
660          {url-base} {
661            \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
662              { docurl } \l_tmpb_tl
663          }
664          {source-base} {
665            \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
666              { ns } \l_tmpb_tl
667          }
668          {ns} {
669            \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
670              { ns } \l_tmpb_tl
671          }
672          {dependencies} {
673            \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
674              { deps } \l_tmpb_tl
675          }
676        }{}{}
677      }{}
678    }
679    \ior_close:N \c__stex_mathhub_manifest_ior
680    \stex_persist:x {
681      \prop_set_from_keyval:cn{ c_stex_mathhub_#1_manifest_prop }{
682        \exp_after:wN \prop_to_keyval:N \csname c_stex_mathhub_#1_manifest_prop\endcsname
683      }
684    }
685 }
```

(*End definition for* __stex_mathhub_parse_manifest:n.)

\stex_set_current_repository:n

```
686 \cs_new_protected:Nn \stex_set_current_repository:n {
687   \stex_require_repository:n { #1 }
688   \prop_set_eq:Nc \l_stex_current_repository_prop {
689     c_stex_mathhub_#1_manifest_prop
690   }
691 }
```

(*End definition for* \stex_set_current_repository:n. *This function is documented on page 67.*)

109

```
692 \cs_new_protected:Nn \stex_require_repository:n {
693   \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
694     \stex_debug:nn{mathhub}{Opening~archive:~#1}
695     \__stex_mathhub_do_manifest:n { #1 }
696   }
697 }
```

(*End definition for* \stex_require_repository:n*. This function is documented on page* *67.*)

Current MathHub repository

```
698 %\prop_new:N \l_stex_current_repository_prop
699 \bool_if:NF \c_stex_persist_mode_bool {
700   \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
701   \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
702     \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
703   } {
704     \__stex_mathhub_parse_manifest:n { main }
705     \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
706       \l_tmpa_str
707     \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str _manifest_prop }
708       \c_stex_mathhub_main_manifest_prop
709     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
710     \stex_debug:nn{mathhub}{Current~repository:~
711       \prop_item:Nn \l_stex_current_repository_prop {id}
712   }
713   }
714 }
```

(*End definition for* \l_stex_current_repository_prop*. This variable is documented on page* *67.*)

Executes the code in the second argument in the context of the repository whose ID is provided as the first argument.

```
715 \cs_new_protected:Nn \stex_in_repository:nn {
716   \str_set:Nx \l_tmpa_str { #1 }
717   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
718   \str_if_empty:NTF \l_tmpa_str {
719     \prop_if_exist:NTF \l_stex_current_repository_prop {
720       \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
721       \exp_args:Ne \l_tmpa_cs{
722         \prop_item:Nn \l_stex_current_repository_prop { id }
723       }
724     }{
725       \l_tmpa_cs{}
726     }
727   }{
728     \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}
729     \stex_require_repository:n \l_tmpa_str
730     \str_set:Nx \l_tmpa_str { #1 }
731     \exp_args:Nne \use:nn {
732       \stex_set_current_repository:n \l_tmpa_str
733       \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
734     }{
735       \stex_debug:nn{mathhub}{switching~back~to:~
```

```
736        \prop_if_exist:NTF \l_stex_current_repository_prop {
737          \prop_item:Nn \l_stex_current_repository_prop { id }:~
738          \meaning\l_stex_current_repository_prop
739        }{
740          no~repository
741        }
742      }
743      \prop_if_exist:NTF \l_stex_current_repository_prop {
744       \stex_set_current_repository:n {
745        \prop_item:Nn \l_stex_current_repository_prop { id }
746       }
747      }{
748        \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
749      }
750    }
751  }
752 }
```

(End definition for `\stex_in_repository:nn`*. This function is documented on page 67.)*

25.5   Using Content in Archives

```
753 \def \mhpath #1 #2 {
754   \exp_args:Ne \tl_if_empty:nTF{#1}{
755     \c_stex_mathhub_str /
756       \prop_item:Nn \l_stex_current_repository_prop { id }
757       / source / #2
758   }{
759     \c_stex_mathhub_str / #1 / source / #2
760   }
761 }
```

(End definition for `\mhpath`*. This function is documented on page 68.)*

```
762 \newif \ifinputref \inputreffalse
763
764 \cs_new_protected:Nn \__stex_mathhub_mhinput:nn {
765   \stex_in_repository:nn {#1} {
766     \ifinputref
767       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
768     \else
769       \inputreftrue
770       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
771       \inputreffalse
772     \fi
773   }
774 }
775 \NewDocumentCommand \mhinput { O{} m}{
776   \__stex_mathhub_mhinput:nn{ #1 }{ #2 }
777 }
778
```

```
779 \cs_new_protected:Nn \__stex_mathhub_inputref:nn {
780   \stex_in_repository:nn {#1} {
781     \stex_html_backend:TF {
782       \str_clear:N \l_tmpa_str
783       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
784         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
785       }
786
787       \tl_if_empty:nTF{ ##1 }{
788         \IfFileExists{#2}{
789           \stex_annotate_invisible:nnn{inputref}{
790             \l_tmpa_str / #2
791           }{}
792         }{
793           \input{#2}
794         }
795       }{
796         \IfFileExists{ \c_stex_mathhub_str / ##1 / source / #2 }{
797           \stex_annotate_invisible:nnn{inputref}{
798             \l_tmpa_str / #2
799           }{}
800         }{
801           \input{ \c_stex_mathhub_str / ##1 / source / #2 }
802         }
803       }
804
805     }{
806       \begingroup
807       \inputreftrue
808       \tl_if_empty:nTF{ ##1 }{
809         \input{#2}
810       }{
811         \input{ \c_stex_mathhub_str / ##1 / source / #2 }
812       }
813       \endgroup
814     }
815   }
816 }
817 \NewDocumentCommand \inputref { O{} m }{
818   \__stex_mathhub_inputref:nn{ #1 }{ #2 }
819 }
```

(End definition for `\inputref` *and* `\mhinput`*. These functions are documented on page 68.)*

```
820 \cs_new_protected:Nn \__stex_mathhub_mhbibresource:nn {
821   \stex_in_repository:nn {#1} {
822     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
823   }
824 }
825 \newcommand\addmhbibresource[2][]{
826   \__stex_mathhub_mhbibresource:nn{ #1 }{ #2 }
827 }
```

(End definition for `\addmhbibresource`*. This function is documented on page 68.)*

\libinput

```
828 \cs_new_protected:Npn \libinput #1 {
829   \prop_if_exist:NF \l_stex_current_repository_prop {
830     \msg_error:nnn{stex}{error/notinarchive}\libinput
831   }
832   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
833     \msg_error:nnn{stex}{error/notinarchive}\libinput
834   }
835   \seq_clear:N \l__stex_mathhub_libinput_files_seq
836   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
837   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
838
839   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
840     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #1.tex}
841     \IfFileExists{ \l_tmpa_str }{
842       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
843     }{}
844     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
845     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
846   }
847
848   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}
849   \IfFileExists{ \l_tmpa_str }{
850     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
851   }{}
852
853   \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
854     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
855   }{
856     \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
857       \input{ ##1 }
858     }
859   }
860 }
```

(End definition for \libinput*. This function is documented on page 68.)*

\libusepackage

```
861 \NewDocumentCommand \libusepackage {O{} m} {
862   \prop_if_exist:NF \l_stex_current_repository_prop {
863     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
864   }
865   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
866     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
867   }
868   \seq_clear:N \l__stex_mathhub_libinput_files_seq
869   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
870   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
871
872   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
873     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #2}
874     \IfFileExists{ \l_tmpa_str.sty }{
875       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
876     }{}
```

113

```
877      \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
878      \seq_put_right:No \l_tmpa_seq \l_tmpa_str
879    }
880
881    \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2}
882    \IfFileExists{ \l_tmpa_str.sty }{
883      \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
884    }{}
885
886    \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
887      \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
888    }{
889      \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
890        \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
891          \usepackage[#1]{ ##1 }
892        }
893      }{
894        \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
895      }
896    }
897 }
```

(*End definition for* `\libusepackage`*. This function is documented on page 68.*)

\cmhgraphics

```
898
899 \AddToHook{begindocument}{
900 \ltx@ifpackageloaded{graphicx}{
901    \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
902    \providecommand\mhgraphics[2][]{%
903      \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
904      \includegraphics[#1]{\mhpath\Gin@mhrepos{#2}}}
905    \providecommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
906 }{}
```

(*End definition for* `\mhgraphics` *and* `\cmhgraphics`*. These functions are documented on page 68.*)

\lstinputmhlisting
\clstinputmhlisting

```
907 \ltx@ifpackageloaded{listings}{
908    \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
909    \newcommand\lstinputmhlisting[2][]{%
910      \def\lst@mhrepos{}\setkeys{lst}{#1}%
911      \lstinputlisting[#1]{\mhpath\lst@mhrepos{#2}}}
912    \newcommand\clstinputmhlisting[2][]{\begin{center}\lstinputmhlisting[#1]{#2}\end{center}
913 }{}
914 }
915
916 ⟨/package⟩
```

(*End definition for* `\lstinputmhlisting` *and* `\clstinputmhlisting`*. These functions are documented on page 68.*)

Chapter 26

sTEX
-References Implementation

```
917  ⟨∗package⟩
918
919  %%%%%%%%%%%%   references.dtx   %%%%%%%%%%%%
920
921  ⟨@@=stex_refs⟩
```

Warnings and error messages

```
922
```

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

```
923  %\iow_new:N \c__stex_refs_refs_iow
924  \AtBeginDocument{
925  %  \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
926  }
927  \AtEndDocument{
928  %  \iow_close:N \c__stex_refs_refs_iow
929  }
```

\STEXreftitle

```
930  \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
931
932  \NewDocumentCommand \STEXreftitle { m } {
933    \tl_gset:Nx \g__stex_refs_title_tl { #1 }
934  }
```

(*End definition for* \STEXreftitle*. This function is documented on page 69.*)

26.1 Document URIs and URLs

\l_stex_current_docns_str

```
935  \str_new:N \l_stex_current_docns_str
```

(*End definition for* \l_stex_current_docns_str*. This variable is documented on page 69.*)

115

```
936 \cs_new_protected:Nn \stex_get_document_uri: {
937   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
938   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
939   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
940   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
941   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
942
943   \str_clear:N \l_tmpa_str
944   \prop_if_exist:NT \l_stex_current_repository_prop {
945     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
946       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
947     }
948   }
949
950   \str_if_empty:NTF \l_tmpa_str {
951     \str_set:Nx \l_stex_current_docns_str {
952       file:/\stex_path_to_string:N \l_tmpa_seq
953     }
954   }{
955     \bool_set_true:N \l_tmpa_bool
956     \bool_while_do:Nn \l_tmpa_bool {
957       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
958       \exp_args:No \str_case:nnTF { \l_tmpb_str } {
959         {source} { \bool_set_false:N \l_tmpa_bool }
960       }{}{
961         \seq_if_empty:NT \l_tmpa_seq {
962           \bool_set_false:N \l_tmpa_bool
963         }
964       }
965     }
966
967     \seq_if_empty:NTF \l_tmpa_seq {
968       \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
969     }{
970       \str_set:Nx \l_stex_current_docns_str {
971         \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
972       }
973     }
974   }
975 }
```

(End definition for \stex_get_document_uri:. *This function is documented on page 69.)*

```
976 \str_new:N \l_stex_current_docurl_str
```

(End definition for \l_stex_current_docurl_str. *This variable is documented on page 69.)*

```
977 \cs_new_protected:Nn \stex_get_document_url: {
978   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
979   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
980   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
```

```
981    \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
982    \seq_put_right:No \l_tmpa_seq \l_tmpb_str
983
984    \str_clear:N \l_tmpa_str
985    \prop_if_exist:NT \l_stex_current_repository_prop {
986      \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
987        \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
988          \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
989        }
990      }
991    }
992
993    \str_if_empty:NTF \l_tmpa_str {
994      \str_set:Nx \l_stex_current_docurl_str {
995        file:/\stex_path_to_string:N \l_tmpa_seq
996      }
997    }{
998      \bool_set_true:N \l_tmpa_bool
999      \bool_while_do:Nn \l_tmpa_bool {
1000         \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1001         \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1002           {source} { \bool_set_false:N \l_tmpa_bool }
1003         }{}{
1004           \seq_if_empty:NT \l_tmpa_seq {
1005             \bool_set_false:N \l_tmpa_bool
1006           }
1007         }
1008      }
1009
1010      \seq_if_empty:NTF \l_tmpa_seq {
1011         \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
1012      }{
1013         \str_set:Nx \l_stex_current_docurl_str {
1014           \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
1015         }
1016      }
1017    }
1018 }
```

(*End definition for* `\stex_get_document_url:`*. This function is documented on page 69.*)

26.2   Setting Reference Targets

```
1019 \str_const:Nn \c__stex_refs_url_str{URL}
1020 \str_const:Nn \c__stex_refs_ref_str{REF}
1021 \str_new:N \l__stex_refs_curr_label_str
1022 % @currentlabel -> number
1023 % @currentlabelname -> title
1024 % @currentHref -> name.number <- id of some kind
1025 % \theH# -> \arabic{section}
1026 % \the#  -> number
1027 % \hyper@makecurrent{#}
1028 \int_new:N \l__stex_refs_unnamed_counter_int
```

```
1029 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
1030   \stex_get_document_uri:
1031   \str_clear:N \l__stex_refs_curr_label_str
1032   \str_set:Nx \l_tmpa_str { #1 }
1033   \str_if_empty:NT \l_tmpa_str {
1034     \int_incr:N \l__stex_refs_unnamed_counter_int
1035     \str_set:Nx \l_tmpa_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
1036   }
1037   \str_set:Nx \l__stex_refs_curr_label_str {
1038     \l_stex_current_docns_str?\l_tmpa_str
1039   }
1040   \seq_if_exist:cF{g__stex_refs_labels_\l_tmpa_str _seq}{
1041     \seq_new:c {g__stex_refs_labels_\l_tmpa_str _seq}
1042   }
1043   \seq_if_in:coF{g__stex_refs_labels_\l_tmpa_str _seq}\l__stex_refs_curr_label_str {
1044     \seq_gput_right:co{g__stex_refs_labels_\l_tmpa_str _seq}\l__stex_refs_curr_label_str
1045   }
1046   \stex_if_smsmode:TF {
1047     \stex_get_document_url:
1048     \str_gset_eq:cN {sref_url_\l__stex_refs_curr_label_str _str}\l_stex_current_docurl_str
1049     \str_gset_eq:cN {sref_\l__stex_refs_curr_label_str _type}\c__stex_refs_url_str
1050   }{
1051     %\iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=~\expandafter\unexpanded\expandafter{
1052     \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
1053     \immediate\write\@auxout{\stexauxadddocref{\l_stex_current_docns_str}{\l_tmpa_str}}
1054     \str_gset:cx {sref_\l__stex_refs_curr_label_str _type}\c__stex_refs_ref_str
1055   }
1056 }
```

(*End definition for* \stex_ref_new_doc_target:n. *This function is documented on page 69.*)

The following is used to set the necessary macros in the .aux-file.

```
1057 \cs_new_protected:Npn \stexauxadddocref #1 #2 {
1058   \str_set:Nn \l_tmpa_str {#1?#2}
1059   \str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
1060   \seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
1061     \seq_new:c {g__stex_refs_labels_#2_seq}
1062   }
1063   \seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
1064     \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
1065   }
1066 }
```

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

```
1067 \AtEndDocument{
1068   \def\stexauxadddocref#1 #2 {}{}
1069 }
```

```
1070 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
1071   \stex_if_smsmode:TF {
1072     \str_if_exist:cF{sref_sym_#1_type}{
1073       \stex_get_document_url:
1074       \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str
```

```
1075        \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
1076      }
1077    }{
1078      \str_if_empty:NF \l__stex_refs_curr_label_str {
1079        \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
1080        \immediate\write\@auxout{
1081          \exp_not:N\expandafter\def\exp_not:N\csname \exp_not:N\detokenize{sref_sym_#1_label_
1082            \l__stex_refs_curr_label_str
1083          }
1084        }
1085      }
1086    }
1087 }
```

(*End definition for* `\stex_ref_new_sym_target:n`. *This function is documented on page* *69.*)

26.3 Using References

```
1088 \str_new:N \l__stex_refs_indocument_str
```

\sref   Optional arguments:

```
1089
1090 \keys_define:nn { stex / sref } {
1091    linktext      .tl_set:N  = \l__stex_refs_linktext_tl ,
1092    fallback      .tl_set:N  = \l__stex_refs_fallback_tl ,
1093    pre           .tl_set:N  = \l__stex_refs_pre_tl ,
1094    post          .tl_set:N  = \l__stex_refs_post_tl ,
1095 }
1096 \cs_new_protected:Nn \__stex_refs_args:n {
1097    \tl_clear:N \l__stex_refs_linktext_tl
1098    \tl_clear:N \l__stex_refs_fallback_tl
1099    \tl_clear:N \l__stex_refs_pre_tl
1100    \tl_clear:N \l__stex_refs_post_tl
1101    \str_clear:N \l__stex_refs_repo_str
1102    \keys_set:nn { stex / sref } { #1 }
1103 }
```

The actual macro:

```
1104 \NewDocumentCommand \sref { O{} m}{
1105    \__stex_refs_args:n { #1 }
1106    \str_if_empty:NTF \l__stex_refs_indocument_str {
1107      \str_set:Nx \l_tmpa_str { #2 }
1108      \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
1109      \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
1110        \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
1111          \seq_get_left:cNF {g__stex_refs_labels_\l_tmpa_str _seq} \l_tmpa_str {
1112            \str_clear:N \l_tmpa_str
1113          }
1114        }{
1115          \str_clear:N \l_tmpa_str
1116        }
1117      }{
1118        \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1119        \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
```

```
1120          \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
1121          \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
1122            \str_set_eq:NN \l_tmpc_str \l_tmpa_str
1123            \str_clear:N \l_tmpa_str
1124            \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str _seq} {
1125              \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
1126                \str_range:nnn { ##1 }{ -\l_tmpa_int}{ -1 }
1127              }{
1128                \seq_map_break:n {
1129                  \str_set:Nn \l_tmpa_str { ##1 }
1130                }
1131              }
1132            }
1133          }{
1134            \str_clear:N \l_tmpa_str
1135          }
1136        }
1137        \str_if_empty:NTF \l_tmpa_str {
1138          \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_lin
1139        }{
1140          \str_if_eq:cNTF {sref_\l_tmpa_str _type} \c__stex_refs_ref_str {
1141            \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1142              \cs_if_exist:cTF{autoref}{
1143                \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1144              }{
1145                \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1146              }
1147            }{
1148              \ltx@ifpackageloaded{hyperref}{
1149                \hyperref[sref_\l_tmpa_str]\l__stex_refs_linktext_tl
1150              }{
1151                \l__stex_refs_linktext_tl
1152              }
1153            }
1154          }{
1155            \ltx@ifpackageloaded{hyperref}{
1156              \href{\use:c{sref_url_\l_tmpa_str _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_t
1157            }{
1158              \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs
1159            }
1160          }
1161        }
1162      }{
1163        % TODO
1164      }
1165 }
```

(*End definition for* \sref. *This function is documented on page 70.*)

\srefsym

```
1166 \NewDocumentCommand \srefsym { O{} m}{
1167   \stex_get_symbol:n { #2 }
1168   \__stex_refs_sym_aux:nn{#1}{\l_stex_get_symbol_uri_str}
1169 }
```

120

```
1170
1171 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1172   \str_if_exist:cTF {sref_sym_#2 _label_str }{
1173     \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1174   }{
1175     \__stex_refs_args:n { #1 }
1176     \str_if_empty:NTF \l__stex_refs_indocument_str {
1177       \tl_if_exist:cTF{sref_sym_#2 _type}{
1178         % doc uri in \l_tmpb_str
1179         \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1180         \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1181           % reference
1182           \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1183             \cs_if_exist:cTF{autoref}{
1184               \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1185             }{
1186               \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1187             }
1188           }{
1189             \ltx@ifpackageloaded{hyperref}{
1190               \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1191             }{
1192               \l__stex_refs_linktext_tl
1193             }
1194           }
1195         }{
1196           % URL
1197           \ltx@ifpackageloaded{hyperref}{
1198             \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl \
1199           }{
1200             \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_re
1201           }
1202         }
1203       }{
1204         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_l
1205       }
1206     }{
1207       % TODO
1208     }
1209   }
1210 }
```

(*End definition for* \srefsym. *This function is documented on page 70.*)

```
1211 \cs_new_protected:Npn \srefsymuri #1 #2 {
1212   \__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1213 }
```

(*End definition for* \srefsymuri. *This function is documented on page 70.*)

```
1214 ⟨/package⟩
```

Chapter 27

ST_EX
-Modules Implementation

```
1215 ⟨∗package⟩
1216
1217 %%%%%%%%%%%%    modules.dtx    %%%%%%%%%%%%
1218
1219 ⟨@@=stex_modules⟩
```

Warnings and error messages
```
1220 \msg_new:nnn{stex}{error/unknownmodule}{
1221   No~module~#1~found
1222 }
1223 \msg_new:nnn{stex}{error/syntax}{
1224   Syntax~error:~#1
1225 }
1226 \msg_new:nnn{stex}{error/siglanguage}{
1227   Module~#1~declares~signature~#2,~but~does~not~
1228   declare~its~language
1229 }
1230 \msg_new:nnn{stex}{warning/deprecated}{
1231   #1~is~deprecated;~please~use~#2~instead!
1232 }
1233
1234 \msg_new:nnn{stex}{error/conflictingmodules}{
1235   Conflicting~imports~for~module~#1
1236 }
```

\l_stex_current_module_str   The current module:
```
1237 \str_new:N \l_stex_current_module_str
```

(*End definition for* \l_stex_current_module_str. *This variable is documented on page* 72.)

\l_stex_all_modules_seq   Stores all available modules
```
1238 \seq_new:N \l_stex_all_modules_seq
```

(*End definition for* \l_stex_all_modules_seq. *This variable is documented on page* 72.)

`\stex_if_in_module_p:`
`\stex_if_in_module:TF`

```
1239 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1240     \str_if_empty:NTF \l_stex_current_module_str
1241         \prg_return_false: \prg_return_true:
1242 }
```

(*End definition for* `\stex_if_in_module:TF`*. This function is documented on page* *72.*)

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

```
1243 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1244     \prop_if_exist:cTF { c_stex_module_#1_prop }
1245         \prg_return_true: \prg_return_false:
1246 }
```

(*End definition for* `\stex_if_module_exists:nTF`*. This function is documented on page* *72.*)

`\stex_add_to_current_module:n`
`\STEXexport`

Only allowed within modules:

```
1247 \cs_new_protected:Nn \stex_execute_in_module:n { \stex_if_in_module:T {
1248     \stex_add_to_current_module:n { #1 }
1249     \stex_do_up_to_module:n { #1 }
1250 }}
1251 \cs_generate_variant:Nn \stex_execute_in_module:n {x}
1252
1253 \cs_new_protected:Nn \stex_add_to_current_module:n {
1254     \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1255 }
1256 \cs_generate_variant:Nn \stex_add_to_current_module:n {x}
1257 \cs_new_protected:Npn \STEXexport {
1258     \ExplSyntaxOn
1259     \__stex_modules_export:n
1260 }
1261 \cs_new_protected:Nn \__stex_modules_export:n {
1262     \ignorespacesandpars#1\ExplSyntaxOff
1263     \stex_add_to_current_module:n { \ignorespacesandpars#1}
1264     \stex_smsmode_do:
1265 }
1266 \let \stex_module_export_helper:n \use:n
1267 \stex_deactivate_macro:Nn \STEXexport {module~environments}
```

(*End definition for* `\stex_add_to_current_module:n` *and* `\STEXexport`*. These functions are documented on page* *72.*)

`\stex_add_constant_to_current_module:n`

```
1268 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1269     \str_set:Nx \l_tmpa_str { #1 }
1270     \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
1271 }
```

(*End definition for* `\stex_add_constant_to_current_module:n`*. This function is documented on page* *72.*)

`\stex_add_import_to_current_module:n`

```
1272 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1273     \str_set:Nx \l_tmpa_str { #1 }
1274     \exp_args:Nno
```

123

```
1275    \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1276      \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1277    }
1278  }
```

(End definition for `\stex_add_import_to_current_module:n`*. This function is documented on page 72.)*

`\stex_collect_imports:n`

```
1279  \cs_new_protected:Nn \stex_collect_imports:n {
1280    \seq_clear:N \l_stex_collect_imports_seq
1281    \__stex_modules_collect_imports:n {#1}
1282  }
1283  \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1284    \seq_map_inline:cn {c_stex_module_#1_imports} {
1285      \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1286        \__stex_modules_collect_imports:n { ##1 }
1287      }
1288    }
1289    \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1290      \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1291    }
1292  }
```

(End definition for `\stex_collect_imports:n`*. This function is documented on page 72.)*

`\stex_do_up_to_module:n`

```
1293  \int_new:N \l__stex_modules_group_depth_int
1294  \cs_new_protected:Nn \stex_do_up_to_module:n {
1295    \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1296      #1
1297    }{
1298      #1
1299      \expandafter \tl_gset:Nn
1300      \csname l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1301      \expandafter\expandafter\expandafter\endcsname
1302      \expandafter\expandafter\expandafter { \csname
1303        l__stex_modules_aftergroup_\l_stex_current_module_str _tl\endcsname #1 }
1304      \aftergroup\__stex_modules_aftergroup_do:
1305    }
1306  }
1307  \cs_generate_variant:Nn \stex_do_up_to_module:n {x}
1308  \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1309    \stex_debug:nn{aftergroup}{\cs_meaning:c{
1310      l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1311    }}
1312    \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1313      \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1314      \tl_gclear:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1315    }{
1316      \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1317      \aftergroup\__stex_modules_aftergroup_do:
1318    }
1319  }
1320  \cs_new_protected:Nn \_stex_reset_up_to_module:n {
1321    \expandafter\let\csname l__stex_modules_aftergroup_#1_tl\endcsname\undefined
```

```
1322 }
```

(End definition for `\stex_do_up_to_module:n`*. This function is documented on page 72.)*

`\stex_modules_compute_namespace:nN` Computes the appropriate namespace from the top-level namespace of a repository (`#1`) and a file path (`#2`).

```
1323
```

(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.

```
1324 \str_new:N \l_stex_module_ns_str
1325 \str_new:N \l_stex_module_subpath_str
1326 \cs_new_protected:Nn \__stex_modules_compute_namespace:nN {
1327   \seq_set_eq:NN \l_tmpa_seq #2
1328   % split off file extension
1329   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str % <- filename
1330   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1331   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str % <- filename without suffixes
1332   \seq_put_right:No \l_tmpa_seq \l_tmpb_str % <- file path including name without suffixes
1333
1334   \bool_set_true:N \l_tmpa_bool
1335   \bool_while_do:Nn \l_tmpa_bool {
1336     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1337     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1338       {source} { \bool_set_false:N \l_tmpa_bool }
1339     }{}{
1340       \seq_if_empty:NT \l_tmpa_seq {
1341         \bool_set_false:N \l_tmpa_bool
1342       }
1343     }
1344   }
1345
1346   \stex_path_to_string:NN \l_tmpa_seq \l_stex_module_subpath_str
1347   % \l_tmpa_seq <- sub-path relative to archive
1348   \str_if_empty:NTF \l_stex_module_subpath_str {
1349     \str_set:Nx \l_stex_module_ns_str {#1}
1350   }{
1351     \str_set:Nx \l_stex_module_ns_str {
1352       #1/\l_stex_module_subpath_str
1353     }
1354   }
1355 }
1356
1357 \cs_new_protected:Nn \stex_modules_current_namespace: {
1358   \str_clear:N \l_stex_module_subpath_str
1359   \prop_if_exist:NTF \l_stex_current_repository_prop {
1360     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1361     \__stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1362   }{
1363     % split off file extension
1364     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1365     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
```

```
1366        \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1367        \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1368        \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1369        \str_set:Nx \l_stex_module_ns_str {
1370          file:/\stex_path_to_string:N \l_tmpa_seq
1371        }
1372      }
1373    }
```

(*End definition for* \stex_modules_current_namespace:. *This function is documented on page* *73*.)

27.1   The `smodule` environment

`smodule` arguments:

```
1374  \keys_define:nn { stex / module } {
1375    title         .tl_set:N    = \smoduletitle ,
1376    type          .str_set_x:N = \smoduletype ,
1377    id            .str_set_x:N = \smoduleid ,
1378    deprecate     .str_set_x:N = \l_stex_module_deprecate_str ,
1379    ns            .str_set_x:N = \l_stex_module_ns_str ,
1380    lang          .str_set_x:N = \l_stex_module_lang_str ,
1381    sig           .str_set_x:N = \l_stex_module_sig_str ,
1382    creators      .str_set_x:N = \l_stex_module_creators_str ,
1383    contributors  .str_set_x:N = \l_stex_module_contributors_str ,
1384    meta          .str_set_x:N = \l_stex_module_meta_str ,
1385    srccite       .str_set_x:N = \l_stex_module_srccite_str
1386  }
1387
1388  \cs_new_protected:Nn \__stex_modules_args:n {
1389    \str_clear:N \smoduletitle
1390    \str_clear:N \smoduletype
1391    \str_clear:N \smoduleid
1392    \str_clear:N \l_stex_module_ns_str
1393    \str_clear:N \l_stex_module_deprecate_str
1394    \str_clear:N \l_stex_module_lang_str
1395    \str_clear:N \l_stex_module_sig_str
1396    \str_clear:N \l_stex_module_creators_str
1397    \str_clear:N \l_stex_module_contributors_str
1398    \str_clear:N \l_stex_module_meta_str
1399    \str_clear:N \l_stex_module_srccite_str
1400    \keys_set:nn { stex / module } { #1 }
1401  }
1402
1403  % module parameters here? In the body?
1404
```

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

```
1405  \cs_new_protected:Nn \stex_module_setup:nn {
1406    \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1407    \str_set:Nx \l_stex_module_name_str { #2 }
1408    \__stex_modules_args:n { #1 }
```

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

```
1409    \stex_if_in_module:TF {
1410      % Nested module
1411      \prop_get:cnN {c_stex_module_\l_stex_current_module_str _prop}
1412        { ns } \l_stex_module_ns_str
1413      \str_set:Nx \l_stex_module_name_str {
1414        \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
1415          { name } / \l_stex_module_name_str
1416      }
1417      \str_if_empty:NT \l_stex_module_lang_str {
1418        \str_set:Nx \l_stex_module_lang_str {
1419          \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
1420            { lang }
1421        }
1422      }
1423    }{
1424      % not nested:
1425      \str_if_empty:NT \l_stex_module_ns_str {
1426        \stex_modules_current_namespace:
1427        \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1428          / {\l_stex_module_ns_str}
1429        \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1430        \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1431          \str_set:Nx \l_stex_module_ns_str {
1432            \stex_path_to_string:N \l_tmpa_seq
1433          }
1434        }
1435      }
1436    }
```

Next, we determine the language of the module:

```
1437    \str_if_empty:NT \l_stex_module_lang_str {
1438      \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1439      \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1440      \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1441      \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
1442        \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
1443          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
1444        }
1445      }
1446      \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1447      \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
1448        \seq_pop_right:NN \l_tmpa_seq \l_stex_module_lang_str
1449        \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1450          inferred~from~file~name}
1451      }
1452    }
1453
1454    \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
1455      \exp_args:NNo \stex_set_language:Nn \l_tmpa_str \l_stex_module_lang_str
1456    }}
```

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

```
1457    \str_if_empty:NTF \l_stex_module_sig_str {
1458      \exp_args:Nnx \prop_gset_from_keyval:cn {
1459        c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1460      } {
1461        name      = \l_stex_module_name_str ,
1462        ns        = \l_stex_module_ns_str ,
1463        file      = \exp_not:o { \g_stex_currentfile_seq } ,
1464        lang      = \l_stex_module_lang_str ,
1465        sig       = \l_stex_module_sig_str ,
1466        deprecate = \l_stex_module_deprecate_str ,
1467        meta      = \l_stex_module_meta_str
1468      }
1469      \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1470      \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1471      \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _copymodules}
1472      \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1473      \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
```

We load the metatheory:

```
1474      \str_if_empty:NT \l_stex_module_meta_str {
1475        \str_set:Nx \l_stex_module_meta_str {
1476          \c_stex_metatheory_ns_str ? Metatheory
1477        }
1478      }
1479      \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1480        \bool_set_true:N \l_stex_in_meta_bool
1481        \exp_args:Nx \stex_add_to_current_module:n {
1482          \bool_set_true:N \l_stex_in_meta_bool
1483          \stex_activate_module:n {\l_stex_module_meta_str}
1484          \bool_set_false:N \l_stex_in_meta_bool
1485        }
1486        \stex_activate_module:n {\l_stex_module_meta_str}
1487        \bool_set_false:N \l_stex_in_meta_bool
1488      }
1489    }{
1490      \str_if_empty:NT \l_stex_module_lang_str {
1491        \msg_error:nnxx{stex}{error/siglanguage}{
1492          \l_stex_module_ns_str?\l_stex_module_name_str
1493        }{\l_stex_module_sig_str}
1494      }
1495      \stex_debug:nn{modules}{Signature~\l_stex_module_sig_str~for~\l_stex_module_ns_str?\l_st
1496      \stex_if_module_exists:nTF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1497        \stex_debug:nn{modules}{(already exists)}
1498      }{
1499        \stex_debug:nn{modules}{(needs loading)}
1500        \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1501        \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1502        \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1503        \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1504        \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1505        \str_set:Nx \l_tmpa_str {
1506          \stex_path_to_string:N \l_tmpa_seq /
```

```
1507          \l_tmpa_str . \l_stex_module_sig_str .tex
1508        }
1509        \IfFileExists \l_tmpa_str {
1510          \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1511            \str_clear:N \l_stex_current_module_str
1512            \seq_clear:N \l_stex_all_modules_seq
1513            \stex_debug:nn{modules}{Loading~signature}
1514          }
1515        }{
1516          \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1517        }
1518      }
1519      \stex_if_smsmode:F {
1520        \stex_activate_module:n {
1521          \l_stex_module_ns_str ? \l_stex_module_name_str
1522        }
1523      }
1524      \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1525    }
1526    \str_if_empty:NF \l_stex_module_deprecate_str {
1527      \msg_warning:nnxx{stex}{warning/deprecated}{
1528        Module~\l_stex_current_module_str
1529      }{
1530        \l_stex_module_deprecate_str
1531      }
1532    }
1533    \seq_put_right:Nx \l_stex_all_modules_seq {
1534      \l_stex_module_ns_str ? \l_stex_module_name_str
1535    }
1536    \tl_clear:c{l__stex_modules_aftergroup_\l_stex_module_ns_str ? \l_stex_module_name_str _tl
1537 }
```

(*End definition for* \stex_module_setup:nn. *This function is documented on page 73.*)

smodule   The module environment.

__stex_modules_begin_module: implements \begin{smodule}

```
1538 \cs_new_protected:Nn \__stex_modules_begin_module: {
1539    \stex_reactivate_macro:N \STEXexport
1540    \stex_reactivate_macro:N \importmodule
1541    \stex_reactivate_macro:N \symdecl
1542    \stex_reactivate_macro:N \notation
1543    \stex_reactivate_macro:N \symdef
1544
1545    \stex_debug:nn{modules}{
1546      New~module:\\
1547      Namespace:~\l_stex_module_ns_str\\
1548      Name:~\l_stex_module_name_str\\
1549      Language:~\l_stex_module_lang_str\\
1550      Signature:~\l_stex_module_sig_str\\
1551      Metatheory:~\l_stex_module_meta_str\\
1552      File:~\stex_path_to_string:N \g_stex_currentfile_seq
1553    }
1554
```

```
1555    \stex_if_do_html:T{
1556      \begin{stex_annotate_env} {theory} {
1557        \l_stex_module_ns_str ? \l_stex_module_name_str
1558      }
1559
1560      \stex_annotate_invisible:nnn{header}{} {
1561        \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1562        \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1563        \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1564          \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1565        }
1566        \str_if_empty:NF \smoduletype {
1567          \stex_annotate:nnn{type}{\smoduletype}{}
1568        }
1569      }
1570    }
1571    % TODO: Inherit metatheory for nested modules?
1572 }
1573 \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}`

```
1574 \cs_new_protected:Nn \__stex_modules_end_module: {
1575    \stex_debug:nn{modules}{Closing~module~\prop_item:cn {c_stex_module_\l_stex_current_module
1576    \_stex_reset_up_to_module:n \l_stex_current_module_str
1577    \stex_if_smsmode:T {
1578      \stex_persist:x {
1579        \prop_set_from_keyval:cn{c_stex_module_\l_stex_current_module_str _prop}{
1580          \exp_after:wN \prop_to_keyval:N \csname c_stex_module_\l_stex_current_module_str _pr
1581        }
1582        \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _constants}{
1583          \seq_use:cn{c_stex_module_\l_stex_current_module_str _constants},
1584        }
1585        \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _imports}{
1586          \seq_use:cn{c_stex_module_\l_stex_current_module_str _imports},
1587        }
1588        \tl_set:cn {c_stex_module_\l_stex_current_module_str _code}
1589      }
1590      \exp_after:wN \let \exp_after:wN \l_tmpa_tl \csname c_stex_module_\l_stex_current_module
1591      \exp_after:wN \stex_persist:n \exp_after:wN { \exp_after:wN { \l_tmpa_tl } }
1592    }
1593 }
```

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

The core environment

```
1594 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1595 \NewDocumentEnvironment { smodule } { O{} m } {
1596    \stex_module_setup:nn{#1}{#2}
1597    %\par
1598    \stex_if_smsmode:F{
1599      \tl_if_empty:NF \smoduletitle {
1600        \exp_args:No \stex_document_title:n \smoduletitle
1601      }
```

130

```
1602     \tl_clear:N \l_tmpa_tl
1603     \clist_map_inline:Nn \smoduletype {
1604       \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1605         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1606       }
1607     }
1608     \tl_if_empty:NTF \l_tmpa_tl {
1609       \__stex_modules_smodule_start:
1610     }{
1611       \l_tmpa_tl
1612     }
1613   }
1614   \__stex_modules_begin_module:
1615   \str_if_empty:NF \smoduleid {
1616     \stex_ref_new_doc_target:n \smoduleid
1617   }
1618   \stex_smsmode_do:
1619 } {
1620   \__stex_modules_end_module:
1621   \stex_if_smsmode:F {
1622     \end{stex_annotate_env}
1623     \clist_set:No \l_tmpa_clist \smoduletype
1624     \tl_clear:N \l_tmpa_tl
1625     \clist_map_inline:Nn \l_tmpa_clist {
1626       \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1627         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1628       }
1629     }
1630     \tl_if_empty:NTF \l_tmpa_tl {
1631       \__stex_modules_smodule_end:
1632     }{
1633       \l_tmpa_tl
1634     }
1635   }
1636 }
```

\stexpatchmodule

```
1637 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1638 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1639
1640 \newcommand\stexpatchmodule[3][] {
1641     \str_set:Nx \l_tmpa_str{ #1 }
1642     \str_if_empty:NTF \l_tmpa_str {
1643       \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1644       \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1645     }{
1646       \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1647       \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1648     }
1649 }
```

(*End definition for* \stexpatchmodule. *This function is documented on page 73.*)

27.2 Invoking modules

```
1650 \NewDocumentCommand \STEXModule { m } {
1651   \exp_args:NNx \str_set:Nn \l_tmpa_str { # 1 }
1652   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1653   \tl_set:Nn \l_tmpa_tl {
1654     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1655   }
1656   \seq_map_inline:Nn \l_stex_all_modules_seq {
1657     \str_set:Nn \l_tmpb_str { ##1 }
1658     \str_if_eq:eeT { \l_tmpa_str } {
1659       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1660     } {
1661       \seq_map_break:n {
1662         \tl_set:Nn \l_tmpa_tl {
1663           \stex_invoke_module:n { ##1 }
1664         }
1665       }
1666     }
1667   }
1668   \l_tmpa_tl
1669 }
1670
1671 \cs_new_protected:Nn \stex_invoke_module:n {
1672   \stex_debug:nn{modules}{Invoking~module~#1}
1673   \peek_charcode_remove:NTF ! {
1674     \__stex_modules_invoke_uri:nN { #1 }
1675   } {
1676     \peek_charcode_remove:NTF ? {
1677       \__stex_modules_invoke_symbol:nn { #1 }
1678     } {
1679       \msg_error:nnx{stex}{error/syntax}{
1680         ?~or~!~expected~after~
1681         \c_backslash_str STEXModule{#1}
1682       }
1683     }
1684   }
1685 }
1686
1687 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1688   \str_set:Nn #2 { #1 }
1689 }
1690
1691 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1692   \stex_invoke_symbol:n{#1?#2}
1693 }
```

(*End definition for* `\STEXModule` *and* `\stex_invoke_module:n`*. These functions are documented on page 73.*)

```
1694 \bool_new:N \l_stex_in_meta_bool
1695 \bool_set_false:N \l_stex_in_meta_bool
```

```
1696 \cs_new_protected:Nn \stex_activate_module:n {
1697   \stex_debug:nn{modules}{Activating~module~#1}
1698   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1699     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1700     \use:c{ c_stex_module_#1_code }
1701   }
1702 }
```

(*End definition for* `\stex_activate_module:n`*. This function is documented on page* *74.*)

```
1703 ⟨/package⟩
```

Chapter 28

SτεX -Module Inheritance Implementation

```
1704 ⟨∗package⟩
1705
1706 %%%%%%%%%%%%%    inheritance.dtx    %%%%%%%%%%%%%
1707
```

28.1   SMS Mode

```
1708 ⟨@@=stex_smsmode⟩
```

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

```
1709 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1710 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1711 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1712
1713 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1714   \makeatletter
1715   \makeatother
1716   \ExplSyntaxOn
1717   \ExplSyntaxOff
1718   \rustexBREAK
1719 }
1720
1721 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1722   \symdef
1723   \importmodule
1724   \notation
1725   \symdecl
1726   \STEXexport
1727   \inlineass
1728   \inlinedef
1729   \inlineex
1730   \endinput
1731   \setnotation
```

134

```
1732     \copynotation
1733     \assign
1734     \renamedecl
1735     \donotcopy
1736     \instantiate
1737     \textsymdecl
1738 }
1739
1740 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1741    \tl_to_str:n {
1742        smodule,
1743        copymodule,
1744        interpretmodule,
1745        realization,
1746        sdefinition,
1747        sexample,
1748        sassertion,
1749        sparagraph,
1750        mathstructure
1751    }
1752 }
```

(*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* 75.)

\stex_if_smsmode_p:
\stex_if_smsmode:*TF*

```
1753 \bool_new:N \g__stex_smsmode_bool
1754 \bool_set_false:N \g__stex_smsmode_bool
1755 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1756    \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1757 }
```

(*End definition for* \stex_if_smsmode:TF. *This function is documented on page* 75.)

__stex_smsmode_in_smsmode:nn

```
1758 \cs_new_protected:Nn \__stex_smsmode_in_smsmode:nn { \stex_suppress_html:n {
1759    \vbox_set:Nn \l_tmpa_box {
1760        \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1761        \bool_gset_true:N \g__stex_smsmode_bool
1762        #2
1763        \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1764    }
1765    \box_clear:N \l_tmpa_box
1766 } }
```

(*End definition for* __stex_smsmode_in_smsmode:nn.)

\stex_file_in_smsmode:nn

```
1767 \quark_new:N \q__stex_smsmode_break
1768
1769 \NewDocumentCommand \__stex_smsmode_importmodule: { O{} m} {
1770    \seq_gput_right:Nn \l__stex_smsmode_importmodules_seq {{#1}{#2}}
1771    \stex_smsmode_do:
1772 }
1773
```

```
1774 \cs_new_protected:Nn \__stex_smsmode_module:nn {
1775   \__stex_modules_args:n{#1}
1776   \stex_if_in_module:F {
1777     \str_if_empty:NF \l_stex_module_sig_str {
1778       \stex_modules_current_namespace:
1779       \str_set:Nx \l_stex_module_name_str { #2 }
1780       \stex_if_module_exists:nF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1781         \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1782         \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1783         \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1784         \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1785         \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1786         \str_set:Nx \l_tmpa_str {
1787           \stex_path_to_string:N \l_tmpa_seq /
1788           \l_tmpa_str . \l_stex_module_sig_str .tex
1789         }
1790         \IfFileExists \l_tmpa_str {
1791           \exp_args:NNx \seq_gput_right:Nn \l__stex_smsmode_sigmodules_seq \l_tmpa_str
1792         }{
1793           \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1794         }
1795       }
1796     }
1797   }
1798 }
1799
1800 \prg_new_conditional:Nnn \__stex_smsmode_check_import_pair:nn {T,F,TF} {
1801   %\stex_debug:nn{import-pair}{\detokenize{{#1}~{#2}}}
1802   \tl_if_empty:nTF{#1}{
1803     \prop_if_exist:NTF \l_stex_current_repository_prop
1804       {
1805         %\stex_debug:nn{import-pair}{in repository \prop_item:Nn \l_stex_current_repository_
1806         \prg_return_true:
1807       } {
1808         \seq_set_split:Nnn \l_tmpa_seq ? {#2}
1809         \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
1810         \tl_if_empty:NT \l_tmpa_tl {
1811           \seq_pop_left:NN \l_tmpa_seq \l_tmpa_tl
1812         }
1813         %\stex_debug:nn{import-pair}{\seq_use:Nn \l_tmpa_seq,~of~length~\seq_count:N \l_tmpa
1814         \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1
1815           \prg_return_true: \prg_return_false:
1816       }
1817   }\prg_return_true:
1818 }
1819
1820 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1821   \stex_filestack_push:n{#1}
1822   \seq_gclear:N \l__stex_smsmode_importmodules_seq
1823   \seq_gclear:N \l__stex_smsmode_sigmodules_seq
1824   % ----- new ---------------------------
1825   \__stex_smsmode_in_smsmode:nn{#1}{
1826     \let\importmodule\__stex_smsmode_importmodule:
1827     \let\stex_module_setup:nn\__stex_smsmode_module:nn
```

136

```
1828       \let\__stex_modules_begin_module:\relax
1829       \let\__stex_modules_end_module:\relax
1830       \seq_clear:N \g_stex_smsmode_allowedenvs_seq
1831       \exp_args:NNx \seq_put_right:Nn \g_stex_smsmode_allowedenvs_seq {\tl_to_str:n{smodule}}
1832       \tl_clear:N \g_stex_smsmode_allowedmacros_tl
1833       \tl_clear:N \g_stex_smsmode_allowedmacros_escape_tl
1834       \tl_put_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\importmodule}
1835       \everyeof{\q__stex_smsmode_break\noexpand}
1836       \expandafter\expandafter\expandafter
1837       \stex_smsmode_do:
1838       \csname @ @ input\endcsname "#1"\relax
1839
1840       \seq_map_inline:Nn \l__stex_smsmode_sigmodules_seq {
1841         \stex_filestack_push:n{##1}
1842         \expandafter\expandafter\expandafter
1843         \stex_smsmode_do:
1844         \csname @ @ input\endcsname "##1"\relax
1845         \stex_filestack_pop:
1846       }
1847     }
1848     % ----- new ---------------------------
1849     \__stex_smsmode_in_smsmode:nn{#1} {
1850       #2
1851       % ----- new ---------------------------
1852       \begingroup
1853       %\stex_debug:nn{smsmode}{Here:~\seq_use:Nn\l__stex_smsmode_importmodules_seq, }
1854       \seq_map_inline:Nn \l__stex_smsmode_importmodules_seq {
1855         \__stex_smsmode_check_import_pair:nnT ##1 { \begingroup
1856           \stex_import_module_uri:nn ##1
1857           \stex_import_require_module:nnnn
1858             \l_stex_import_ns_str
1859             \l_stex_import_archive_str
1860             \l_stex_import_path_str
1861             \l_stex_import_name_str \endgroup
1862       }
1863     }
1864     \endgroup
1865     \stex_debug:nn{smsmode}{Actually~loading~file~#1}
1866     % ----- new ---------------------------
1867     \everyeof{\q__stex_smsmode_break\noexpand}
1868     \expandafter\expandafter\expandafter
1869     \stex_smsmode_do:
1870     \csname @ @ input\endcsname "#1"\relax
1871   }
1872   \stex_filestack_pop:
1873 }
```

(*End definition for* `\stex_file_in_smsmode:nn`. *This function is documented on page 76.*)

`\stex_smsmode_do:` is executed on encountering \ in smsmode. It checks whether the corresponding command is allowed and executes or ignores it accordingly:

```
1874 \cs_new_protected:Npn \stex_smsmode_do: {
1875   \stex_if_smsmode:T {
1876     \__stex_smsmode_do:w
```

```
1877      }
1878  }
1879  \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1880    \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 }}{
1881      \expandafter\if\expandafter\relax\noexpand#1
1882        \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1883      \else\expandafter\__stex_smsmode_do:w\fi
1884    }{
1885      \__stex_smsmode_do:w %#1
1886    }
1887  }
1888  \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1889    \cs_if_eq:NNF #1 \q__stex_smsmode_break {
1890      \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1891        #1\__stex_smsmode_do:w
1892      }{
1893        \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1894          #1
1895        }{
1896          \cs_if_eq:NNTF \begin #1 {
1897            \__stex_smsmode_check_begin:n
1898          }{
1899            \cs_if_eq:NNTF \end #1 {
1900              \__stex_smsmode_check_end:n
1901            }{
1902              \__stex_smsmode_do:w
1903            }
1904          }
1905        }
1906      }
1907    }
1908  }
1909
1910  \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1911    \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1912      \begin{#1}
1913    }{
1914      \__stex_smsmode_do:w
1915    }
1916  }
1917  \cs_new_protected:Nn \__stex_smsmode_check_end:n {
1918    \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1919      \end{#1}\__stex_smsmode_do:w
1920    }{
1921      \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1922    }
1923  }
```

(*End definition for* `\stex_smsmode_do:`. *This function is documented on page 76.*)

28.2 Inheritance

```
1924  ⟨@@=stex_importmodule⟩
```

138

```
1925 \cs_new_protected:Nn \stex_import_module_uri:nn {
1926     \str_set:Nx \l_stex_import_archive_str { #1 }
1927     \str_set:Nn \l_stex_import_path_str { #2 }
1928
1929     \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1930     \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1931     \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1932
1933     \stex_modules_current_namespace:
1934     \bool_lazy_all:nTF {
1935         {\str_if_empty_p:N \l_stex_import_archive_str}
1936         {\str_if_empty_p:N \l_stex_import_path_str}
1937         {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1938     }{
1939         \str_set_eq:NN \l_stex_import_path_str \l_stex_module_subpath_str
1940         \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1941     }{
1942         \str_if_empty:NT \l_stex_import_archive_str {
1943             \prop_if_exist:NT \l_stex_current_repository_prop {
1944                 \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1945             }
1946         }
1947         \str_if_empty:NTF \l_stex_import_archive_str {
1948             \str_if_empty:NF \l_stex_import_path_str {
1949                 \stex_path_from_string:Nn \l_tmpb_seq {
1950                     \l_stex_module_ns_str  / .. / \l_stex_import_path_str
1951                 }
1952                 \str_set:Nx \l_stex_import_ns_str {\stex_path_to_string:N \l_tmpb_seq}
1953                 \str_replace_once:Nnn \l_stex_import_ns_str {file:/} {file://}
1954             }
1955         }{
1956             \stex_require_repository:n \l_stex_import_archive_str
1957             \prop_get:cnN { c_stex_mathhub_\l_stex_import_archive_str _manifest_prop } { ns }
1958                 \l_stex_import_ns_str
1959             \str_if_empty:NF \l_stex_import_path_str {
1960                 \str_set:Nx \l_stex_import_ns_str {
1961                     \l_stex_import_ns_str / \l_stex_import_path_str
1962                 }
1963             }
1964         }
1965     }
1966 }
```

(*End definition for* \stex_import_module_uri:nn*. This function is documented on page 77.*)

Store the return values of \stex_import_module_uri:nn.

```
1967 \str_new:N \l_stex_import_name_str
1968 \str_new:N \l_stex_import_archive_str
1969 \str_new:N \l_stex_import_path_str
1970 \str_new:N \l_stex_import_ns_str
```

(*End definition for* \l_stex_import_name_str *and others. These variables are documented on page 77.*)

`{⟨ns⟩} {⟨archive-ID⟩} {⟨path⟩} {⟨name⟩}`

```
1971 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1972   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1973
1974     \stex_debug:nn{requiremodule}{Here:\\~~1:~#1\\~~2:~#2\\~~3:~#3\\~~4:~#4}
1975
1976     \exp_args:NNxx \seq_set_split:Nnn \l_tmpa_seq {\tl_to_str:n{/}} {#4}
1977     \seq_get_left:NN \l_tmpa_seq \l_tmpc_str
1978
1979     %\stex_debug:nn{requiremodule}{Top~module:\l_tmpc_str}
1980
1981     % archive
1982     \str_set:Nx \l_tmpa_str { #2 }
1983     \str_if_empty:NTF \l_tmpa_str {
1984       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1985       \seq_put_right:Nn \l_tmpa_seq {..}
1986     } {
1987       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1988       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1989       \seq_put_right:Nn \l_tmpa_seq { source }
1990     }
1991
1992     % path
1993     \str_set:Nx \l_tmpb_str { #3 }
1994     \str_if_empty:NTF \l_tmpb_str {
1995       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / \l_tmpc_str }
1996
1997       \ltx@ifpackageloaded{babel} {
1998         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1999           { \languagename } \l_tmpb_str {
2000             \msg_error:nnx{stex}{error/unknownlanguage}{\languagename}
2001           }
2002       } {
2003         \str_clear:N \l_tmpb_str
2004       }
2005
2006       \stex_debug:nn{modules}{Checking~a1~\l_tmpa_str.\l_tmpb_str.tex}
2007       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2008         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2009       }{
2010         \stex_debug:nn{modules}{Checking~a2~\l_tmpa_str.tex}
2011         \IfFileExists{ \l_tmpa_str.tex }{
2012           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2013         }{
2014           % try english as default
2015           \stex_debug:nn{modules}{Checking~a3~\l_tmpa_str.en.tex}
2016           \IfFileExists{ \l_tmpa_str.en.tex }{
2017             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2018           }{
2019             \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2020           }
2021         }
2022       }
2023
```

140

```
2024        } {
2025          \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
2026          \seq_concat:NNN \l_tmpb_seq \l_tmpa_seq \l_tmpb_seq
2027
2028          \ltx@ifpackageloaded{babel} {
2029            \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
2030                { \languagename } \l_tmpb_str {
2031                  \msg_error:nnx{stex}{error/unknownlanguage}{\languagename}
2032                }
2033          } {
2034            \str_clear:N \l_tmpb_str
2035          }
2036
2037          \stex_path_canonicalize:N \l_tmpb_seq
2038          \stex_path_to_string:NN \l_tmpb_seq \l_tmpa_str
2039
2040          \stex_debug:nn{modules}{Checking~b1~\l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex}
2041          \IfFileExists{ \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }{
2042            \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.te
2043          }{
2044            \stex_debug:nn{modules}{Checking~b2~\l_tmpa_str/\l_tmpc_str.tex}
2045            \IfFileExists{ \l_tmpa_str/\l_tmpc_str.tex }{
2046              \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.tex }
2047            }{
2048              % try english as default
2049              \stex_debug:nn{modules}{Checking~b3~\l_tmpa_str/\l_tmpc_str.en.tex}
2050              \IfFileExists{ \l_tmpa_str/\l_tmpc_str.en.tex }{
2051                \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.en.tex }
2052              }{
2053                \stex_debug:nn{modules}{Checking~b4~\l_tmpa_str.\l_tmpb_str.tex}
2054                \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2055                  \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2056                }{
2057                  \stex_debug:nn{modules}{Checking~b4~\l_tmpa_str.tex}
2058                  \IfFileExists{ \l_tmpa_str.tex }{
2059                    \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2060                  }{
2061                    % try english as default
2062                    \stex_debug:nn{modules}{Checking~b5~\l_tmpa_str.en.tex}
2063                    \IfFileExists{ \l_tmpa_str.en.tex }{
2064                      \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2065                    }{
2066                      \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2067                    }
2068                  }
2069                }
2070              }
2071            }
2072          }
2073        }
2074
2075        \str_if_eq:eeF{\g__stex_importmodule_file_str}{\seq_use:Nn \g_stex_currentfile_seq /}{
2076          \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
2077            \seq_clear:N \l_stex_all_modules_seq
```

141

```
2078          \str_clear:N \l_stex_current_module_str
2079          \str_set:Nx \l_tmpb_str { #2 }
2080          \str_if_empty:NF \l_tmpb_str {
2081            \stex_set_current_repository:n { #2 }
2082          }
2083          \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
2084        }
2085
2086        \stex_if_module_exists:nF { #1 ? #4 } {
2087          \msg_error:nnx{stex}{error/unknownmodule}{
2088            #1?#4~(in~file~\g__stex_importmodule_file_str)
2089          }
2090        }
2091      }
2092
2093    }
2094    \stex_activate_module:n { #1 ? #4 }
2095 }
```

(End definition for \stex_import_require_module:nnnn. *This function is documented on page 77.)*

\importmodule

```
2096 \NewDocumentCommand \importmodule { O{} m } {
2097    \stex_import_module_uri:nn { #1 } { #2 }
2098    \stex_debug:nn{modules}{Importing~module:~
2099      \l_stex_import_ns_str ? \l_stex_import_name_str
2100    }
2101    \stex_import_require_module:nnnn
2102    { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2103    { \l_stex_import_path_str } { \l_stex_import_name_str }
2104    \stex_if_smsmode:F {
2105      \stex_annotate_invisible:nnn
2106        {import} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
2107    }
2108    \exp_args:Nx \stex_add_to_current_module:n {
2109      \stex_import_require_module:nnnn
2110      { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2111      { \l_stex_import_path_str } { \l_stex_import_name_str }
2112    }
2113    \exp_args:Nx \stex_add_import_to_current_module:n {
2114      \l_stex_import_ns_str ? \l_stex_import_name_str
2115    }
2116    \stex_smsmode_do:
2117    \ignorespacesandpars
2118 }
2119 \stex_deactivate_macro:Nn \importmodule {module~environments}
```

(End definition for \importmodule. *This function is documented on page 76.)*

\usemodule

```
2120 \NewDocumentCommand \usemodule { O{} m } {
2121    \stex_if_smsmode:F {
2122      \stex_import_module_uri:nn { #1 } { #2 }
2123      \stex_import_require_module:nnnn
2124      { \l_stex_import_ns_str } { \l_stex_import_archive_str }
```

```
2125      { \l_stex_import_path_str } { \l_stex_import_name_str }
2126      \stex_annotate_invisible:nnn
2127        {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
2128    }
2129    \stex_smsmode_do:
2130    \ignorespacesandpars
2131 }
```

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

```
2132 \cs_new_protected:Nn \stex_csl_to_imports:Nn {
2133   \tl_if_empty:nF{#2}{
2134     \clist_set:Nn \l_tmpa_clist {#2}
2135     \clist_map_inline:Nn \l_tmpa_clist {
2136       \tl_if_head_eq_charcode:nNTF {##1}[{
2137         #1 ##1
2138       }{
2139         #1{##1}
2140       }
2141     }
2142   }
2143 }
2144 \cs_generate_variant:Nn \stex_csl_to_imports:Nn {No}
2145
2146
2147 ⟨/package⟩
```

Chapter 29

sTEX
-Symbols Implementation

2148 ⟨∗package⟩
2149
2150 %%%%%%%%%%%%%   symbols.dtx   %%%%%%%%%%%%%
2151

Warnings and error messages

2152 \msg_new:nnn{stex}{error/wrongargs}{
2153   args~value~in~symbol~declaration~for~#1~
2154   needs~to~be~i,~a,~b~or~B,~but~#2~given
2155 }
2156 \msg_new:nnn{stex}{error/unknownsymbol}{
2157   No~symbol~#1~found!
2158 }
2159 \msg_new:nnn{stex}{error/seqlength}{
2160   Expected~#1~arguments;~got~#2!
2161 }
2162 \msg_new:nnn{stex}{error/unknownnotation}{
2163   Unknown~notation~#1~for~#2!
2164 }

29.1 Symbol Declarations

2165 ⟨@@=stex_symdecl⟩

\stex_all_symbols:n Map over all available symbols

2166 \cs_new_protected:Nn \stex_all_symbols:n {
2167   \def __stex_symdecl_all_symbols_cs ##1 {#1}
2168   \seq_map_inline:Nn \l_stex_all_modules_seq {
2169   \seq_map_inline:cn{c_stex_module_##1_constants}{
2170   __stex_symdecl_all_symbols_cs{##1?####1}
2171   }
2172   }
2173 }

(*End definition for* \stex_all_symbols:n. *This function is documented on page* *79.*)

144

```
2174 \NewDocumentCommand \STEXsymbol { m } {
2175   \stex_get_symbol:n { #1 }
2176   \exp_args:No
2177   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
2178 }
```

(*End definition for* \STEXsymbol*. This function is documented on page 80.*)

symdecl arguments:

```
2179 \keys_define:nn { stex / symdecl } {
2180   name        .str_set_x:N  = \l_stex_symdecl_name_str ,
2181   local       .bool_set:N   = \l_stex_symdecl_local_bool ,
2182   args        .str_set_x:N  = \l_stex_symdecl_args_str ,
2183   type        .tl_set:N     = \l_stex_symdecl_type_tl ,
2184   deprecate   .str_set_x:N  = \l_stex_symdecl_deprecate_str ,
2185   align       .str_set:N    = \l_stex_symdecl_align_str , % TODO(?)
2186   gfc         .str_set:N    = \l_stex_symdecl_gfc_str , % TODO(?)
2187   specializes .str_set:N    = \l_stex_symdecl_specializes_str , % TODO(?)
2188   def         .tl_set:N     = \l_stex_symdecl_definiens_tl ,
2189   reorder     .str_set_x:N  = \l_stex_symdecl_reorder_str ,
2190   assoc       .choices:nn   =
2191     {bin,binl,binr,pre,conj,pwconj}
2192     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}
2193 }
2194
2195 \bool_new:N \l_stex_symdecl_make_macro_bool
2196
2197 \cs_new_protected:Nn \__stex_symdecl_args:n {
2198   \str_clear:N \l_stex_symdecl_name_str
2199   \str_clear:N \l_stex_symdecl_args_str
2200   \str_clear:N \l_stex_symdecl_deprecate_str
2201   \str_clear:N \l_stex_symdecl_reorder_str
2202   \str_clear:N \l_stex_symdecl_assoctype_str
2203   \bool_set_false:N \l_stex_symdecl_local_bool
2204   \tl_clear:N \l_stex_symdecl_type_tl
2205   \tl_clear:N \l_stex_symdecl_definiens_tl
2206
2207   \keys_set:nn { stex / symdecl } { #1 }
2208 }
```

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

```
2209
2210 \NewDocumentCommand \symdecl { s m O{}} {
2211   \__stex_symdecl_args:n { #3 }
2212   \IfBooleanTF #1 {
2213     \bool_set_false:N \l_stex_symdecl_make_macro_bool
2214   } {
2215     \bool_set_true:N \l_stex_symdecl_make_macro_bool
2216   }
2217   \stex_symdecl_do:n { #2 }
2218   \stex_smsmode_do:
2219 }
```

145

```
2220
2221 \cs_new_protected:Nn \stex_symdecl_do:nn {
2222   \__stex_symdecl_args:n{#1}
2223   \bool_set_false:N \l_stex_symdecl_make_macro_bool
2224   \stex_symdecl_do:n{#2}
2225 }
2226
2227 \stex_deactivate_macro:Nn \symdecl {module~environments}
```

(*End definition for* \symdecl. *This function is documented on page 78.*)

```
2228 \cs_new_protected:Nn \stex_symdecl_do:n {
2229   \stex_if_in_module:F {
2230     % TODO throw error? some default namespace?
2231   }
2232
2233   \str_if_empty:NT \l_stex_symdecl_name_str {
2234     \str_set:Nx \l_stex_symdecl_name_str { #1 }
2235   }
2236
2237   \prop_if_exist:cT { l_stex_symdecl_
2238       \l_stex_current_module_str ?
2239       \l_stex_symdecl_name_str
2240     _prop
2241   }{
2242     % TODO throw error (beware of circular dependencies)
2243   }
2244
2245   \prop_clear:N \l_tmpa_prop
2246   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
2247   \seq_clear:N \l_tmpa_seq
2248   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
2249   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
2250
2251   \str_if_empty:NT \l_stex_symdecl_deprecate_str {
2252     \str_if_empty:NF \l_stex_module_deprecate_str {
2253       \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
2254     }
2255   }
2256   \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
2257
2258   \exp_args:No \stex_add_constant_to_current_module:n {
2259     \l_stex_symdecl_name_str
2260   }
2261
2262   % arity/args
2263   \int_zero:N \l_tmpb_int
2264
2265   \bool_set_true:N \l_tmpa_bool
2266   \str_map_inline:Nn \l_stex_symdecl_args_str {
2267     \token_case_meaning:NnF ##1 {
2268       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2269       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
```

146

```
2270        {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2271        {\tl_to_str:n a} {
2272          \bool_set_false:N \l_tmpa_bool
2273          \int_incr:N \l_tmpb_int
2274        }
2275        {\tl_to_str:n B} {
2276          \bool_set_false:N \l_tmpa_bool
2277          \int_incr:N \l_tmpb_int
2278        }
2279      }{
2280        \msg_error:nnxx{stex}{error/wrongargs}{
2281          \l_stex_current_module_str ?
2282          \l_stex_symdecl_name_str
2283        }{##1}
2284      }
2285    }
2286    \bool_if:NTF \l_tmpa_bool {
2287      % possibly numeric
2288      \str_if_empty:NTF \l_stex_symdecl_args_str {
2289        \prop_put:Nnn \l_tmpa_prop { args } {}
2290        \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2291      }{
2292        \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
2293        \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2294        \str_clear:N \l_tmpa_str
2295        \int_step_inline:nn \l_tmpa_int {
2296          \str_put_right:Nn \l_tmpa_str i
2297        }
2298        \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
2299      }
2300    } {
2301      \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
2302      \prop_put:Nnx \l_tmpa_prop { arity }
2303        { \str_count:N \l_stex_symdecl_args_str }
2304    }
2305    \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2306
2307    \tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
2308      \prop_put:Nnx \l_tmpa_prop { defined }{ false }
2309    }{
2310      \prop_put:Nnx \l_tmpa_prop { defined }{ true }
2311    }
2312
2313    % semantic macro
2314
2315    \bool_if:NT \l_stex_symdecl_make_macro_bool {
2316      \exp_args:Nx \stex_do_up_to_module:n {
2317        \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2318          \l_stex_current_module_str ? \l_stex_symdecl_name_str
2319        }}
2320      }
2321    }
2322
2323    \stex_debug:nn{symbols}{New~symbol:~
```

```
2324      \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2325      Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2326      Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
2327      Definiens:~\exp_not:o {\l_stex_symdecl_definiens_tl}
2328    }
2329
2330    % circular dependencies require this:
2331    \stex_if_do_html:T {
2332      \stex_annotate_invisible:nnn {symdecl} {
2333        \l_stex_current_module_str ? \l_stex_symdecl_name_str
2334      } {
2335        \tl_if_empty:NF \l_stex_symdecl_type_tl {
2336          \stex_annotate_invisible:nnn{type}{}{$\l_stex_symdecl_type_tl$}
2337        }
2338        \stex_annotate_invisible:nnn{args}{\prop_item:Nn \l_tmpa_prop { args }}{}
2339        \stex_annotate_invisible:nnn{macroname}{#1}{}
2340        \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2341          \stex_annotate_invisible:nnn{definiens}{}
2342            {$\l_stex_symdecl_definiens_tl$}
2343        }
2344        \str_if_empty:NF \l_stex_symdecl_assoctype_str {
2345          \stex_annotate_invisible:nnn{assoctype}{\l_stex_symdecl_assoctype_str}{}
2346        }
2347        \str_if_empty:NF \l_stex_symdecl_reorder_str {
2348          \stex_annotate_invisible:nnn{reorderargs}{\l_stex_symdecl_reorder_str}{}
2349        }
2350      }
2351    }
2352    \prop_if_exist:cF {
2353      l_stex_symdecl_
2354      \l_stex_current_module_str ? \l_stex_symdecl_name_str
2355      _prop
2356    } {
2357      \bool_if:NTF \l_stex_symdecl_local_bool \stex_do_up_to_module:x \stex_execute_in_module:
2358        \__stex_symdecl_restore_symbol:nnnnnnn
2359          {\l_stex_symdecl_name_str}
2360          { \prop_item:Nn \l_tmpa_prop {args} }
2361          { \prop_item:Nn \l_tmpa_prop {arity} }
2362          { \prop_item:Nn \l_tmpa_prop {assocs} }
2363          { \prop_item:Nn \l_tmpa_prop {defined} }
2364          {\bool_if:NT \l_stex_symdecl_make_macro_bool {#1} }
2365          {\l_stex_current_module_str}
2366      }
2367    }
2368 }
2369 \cs_new_protected:Nn \__stex_symdecl_restore_symbol:nnnnnnn {
2370    \prop_clear:N \l_tmpa_prop
2371    \prop_put:Nnn \l_tmpa_prop { module } { #7 }
2372    \prop_put:Nnn \l_tmpa_prop { name } { #1}
2373    \prop_put:Nnn \l_tmpa_prop { args } {#2}
2374    \prop_put:Nnn \l_tmpa_prop { arity } { #3 }
2375    \prop_put:Nnn \l_tmpa_prop { assocs } { #4 }
2376    \prop_put:Nnn \l_tmpa_prop { defined } { #5 }
2377    \tl_if_empty:nF{#6}{
```

148

```
2378        \tl_set:cx{#6}{\stex_invoke_symbol:n{\detokenize{#7 ? #1}}}
2379    }
2380    \prop_set_eq:cN{l_stex_symdecl_ \detokenize{#7 ? #1} _prop}\l_tmpa_prop
2381    \seq_clear:c{l_stex_symdecl_ \detokenize{#7 ? #1} _notations}
2382 }
```

(*End definition for* `\stex_symdecl_do:n`*. This function is documented on page* *79.*)

`\textsymdecl`

```
2383
2384 \keys_define:nn { stex / textsymdecl } {
2385    name     .str_set_x:N = \l__stex_symdecl_name_str ,
2386    type     .tl_set:N    = \l__stex_symdecl_type_tl
2387 }
2388
2389 \cs_new_protected:Nn \_stex_textsymdecl_args:n {
2390    \str_clear:N \l__stex_symdecl_name_str
2391    \tl_clear:N \l__stex_symdecl_type_tl
2392    \keys_set:nn { stex / textsymdecl } { #1 }
2393 }
2394
2395 \NewDocumentCommand \textsymdecl {m O{} m} {
2396    \_stex_textsymdecl_args:n { #2 }
2397    \str_if_empty:NTF \l__stex_symdecl_name_str {
2398       \__stex_symdecl_args:n{name=#1,#2}
2399    }{
2400       \__stex_symdecl_args:n{#2}
2401    }
2402    \bool_set_true:N \l_stex_symdecl_make_macro_bool
2403    \stex_symdecl_do:n{#1-sym}
2404    \stex_execute_in_module:n{
2405       \cs_set_nopar:cpn{#1name}{
2406          \ifvmode\hbox_unpack:N\c_empty_box\fi
2407          \hbox{#3}\xspace
2408       }
2409       \cs_set_nopar:cpn{#1}{
2410          \ifmmode\csname#1-sym\expandafter\endcsname\else
2411          \ifvmode\hbox_unpack:N\c_empty_box\fi
2412          \symref{#1-sym}{\hbox{#3}}\expandafter\xspace
2413          \fi
2414       }
2415    }
2416    \stex_execute_in_module:x{
2417       \__stex_notation_restore_notation:nnnnn
2418       {\l_stex_current_module_str?\tl_if_empty:NTF\l__stex_symdecl_name_str{#1}\l__stex_symdec
2419       {}{0}
2420       {\exp_not:n{\STEXInternalTermMathOMSiiii{\STEXInternalCurrentSymbolStr}{}{\neginfprec}{
2421          \comp{\hbox{#3}}\STEXInternalSymbolAfterInvokationTL
2422       }}}
2423       {}
2424    }
2425    \stex_smsmode_do:
2426 }
```

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

149

```
2427  \str_new:N \l_stex_get_symbol_uri_str

2428
2429  \cs_new_protected:Nn \stex_get_symbol:n {
2430    \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2431      \tl_set:Nn \l_tmpa_tl { #1 }
2432      \__stex_symdecl_get_symbol_from_cs:
2433    }{
2434      % argument is a string
2435      % is it a command name?
2436      \cs_if_exist:cTF { #1 }{
2437        \cs_set_eq:Nc \l_tmpa_tl { #1 }
2438        \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2439        \str_if_empty:NTF \l_tmpa_str {
2440          \exp_args:Nx \cs_if_eq:NNTF {
2441            \tl_head:N \l_tmpa_tl
2442          } \stex_invoke_symbol:n {
2443            \__stex_symdecl_get_symbol_from_cs:
2444          }{
2445            \__stex_symdecl_get_symbol_from_string:n { #1 }
2446          }
2447        } {
2448          \__stex_symdecl_get_symbol_from_string:n { #1 }
2449        }
2450      }{
2451        % argument is not a command name
2452        \__stex_symdecl_get_symbol_from_string:n { #1 }
2453        % \l_stex_all_symbols_seq
2454      }
2455    }
2456    \str_if_eq:eeF {
2457      \prop_item:cn {
2458        l_stex_symdecl_\l_stex_get_symbol_uri_str _prop
2459      }{ deprecate }
2460    }{}{
2461      \msg_warning:nnxx{stex}{warning/deprecated}{
2462        Symbol~\l_stex_get_symbol_uri_str
2463      }{
2464        \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{ deprecate }
2465      }
2466    }
2467  }

2468
2469  \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2470    \tl_set:Nn \l_tmpa_tl {
2471      \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2472    }
2473    \str_set:Nn \l_tmpa_str { #1 }

2474
2475    %\int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }

2476
2477    \str_if_in:NnTF \l_tmpa_str ? {
2478      \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
2479      \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
```

```
2480      \str_set:Nx \l_tmpb_str {\seq_use:Nn \l_tmpa_seq ?}
2481   }{
2482      \str_clear:N \l_tmpb_str
2483   }
2484   \str_if_empty:NTF \l_tmpb_str {
2485      \seq_map_inline:Nn \l_stex_all_modules_seq {
2486         \seq_map_inline:cn{c_stex_module_##1_constants}{
2487            \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2488               \seq_map_break:n{\seq_map_break:n{
2489                  \tl_set:Nn \l_tmpa_tl {
2490                     \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2491                  }
2492               }}
2493            }
2494         }
2495      }
2496   }{
2497      \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpb_str }
2498      \seq_map_inline:Nn \l_stex_all_modules_seq {
2499         \str_if_eq:eeT{ \l_tmpb_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2500            \seq_map_inline:cn{c_stex_module_##1_constants}{
2501               \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2502                  \seq_map_break:n{\seq_map_break:n{
2503                     \tl_set:Nn \l_tmpa_tl {
2504                        \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2505                     }
2506                  }}
2507               }
2508            }
2509         }
2510      }
2511   }

2513   \l_tmpa_tl
2514 }

2516 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
2517   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2518      { \tl_tail:N \l_tmpa_tl }
2519   \tl_if_single:NTF \l_tmpa_tl {
2520      \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2521         \exp_after:wN \str_set:Nn \exp_after:wN
2522            \l_stex_get_symbol_uri_str \l_tmpa_tl
2523      }{
2524         % TODO
2525         % tail is not a single group
2526      }
2527   }{
2528      % TODO
2529      % tail is not a single group
2530   }
2531 }
```

(*End definition for* \stex_get_symbol:n. *This function is documented on page 79.*)

29.2   Notations

⟨@@=stex_notation⟩

notation arguments:

```
2533 \keys_define:nn { stex / notation } {
2534 %  lang    .tl_set_x:N  = \l__stex_notation_lang_str ,
2535   variant .tl_set_x:N  = \l__stex_notation_variant_str ,
2536   prec    .str_set_x:N = \l__stex_notation_prec_str ,
2537   op      .tl_set:N    = \l__stex_notation_op_tl ,
2538   primary .bool_set:N  = \l__stex_notation_primary_bool ,
2539   primary .default:n   = {true} ,
2540   unknown .code:n      = \str_set:Nx
2541       \l__stex_notation_variant_str \l_keys_key_str
2542 }
2543
2544 \cs_new_protected:Nn \_stex_notation_args:n {
2545 %  \str_clear:N \l__stex_notation_lang_str
2546   \str_clear:N \l__stex_notation_variant_str
2547   \str_clear:N \l__stex_notation_prec_str
2548   \tl_clear:N \l__stex_notation_op_tl
2549   \bool_set_false:N \l__stex_notation_primary_bool
2550
2551   \keys_set:nn { stex / notation } { #1 }
2552 }
```

\notation

```
2553 \NewDocumentCommand \notation { s m O{}} {
2554   \_stex_notation_args:n { #3 }
2555   \tl_clear:N \l_stex_symdecl_definiens_tl
2556   \stex_get_symbol:n { #2 }
2557   \tl_set:Nn \l_stex_notation_after_do_tl {
2558     \__stex_notation_final:
2559     \IfBooleanTF#1{
2560       \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2561     }{}
2562     \stex_smsmode_do:\ignorespacesandpars
2563   }
2564   \stex_notation_do:nnnnn
2565     { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } }
2566     { \prop_item:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } }
2567     { \l__stex_notation_variant_str }
2568     { \l__stex_notation_prec_str}
2569 }
2570 \stex_deactivate_macro:Nn \notation {module~environments}
```

(*End definition for* \notation. *This function is documented on page* *79.*)

\stex_notation_do:nnnnn

```
2571 \seq_new:N \l__stex_notation_precedences_seq
2572 \tl_new:N \l__stex_notation_opprec_tl
2573 \int_new:N \l__stex_notation_currarg_int
2574 \tl_new:N \STEXInternalSymbolAfterInvokationTL
2575
2576 \cs_new_protected:Nn \stex_notation_do:nnnnn {
```

```
2577    \let\STEXInternalCurrentSymbolStr\relax
2578    \seq_clear:N \l__stex_notation_precedences_seq
2579    \tl_clear:N \l__stex_notation_opprec_tl
2580    \str_set:Nx \l__stex_notation_args_str { #1 }
2581    \str_set:Nx \l__stex_notation_arity_str { #2 }
2582    \str_set:Nx \l__stex_notation_suffix_str { #3 }
2583    \str_set:Nx \l__stex_notation_prec_str { #4 }
2584
2585    % precedences
2586    \str_if_empty:NTF \l__stex_notation_prec_str {
2587      \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2588        \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2589      }{
2590        \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2591      }
2592    } {
2593      \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2594        \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2595        \int_step_inline:nn { \l__stex_notation_arity_str } {
2596          \exp_args:NNo
2597          \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2598        }
2599      }{
2600        \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2601        \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2602          \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2603          \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2604            \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2605              \l_tmpa_seq {\tl_to_str:n{x} } { \l_tmpa_str }
2606            \seq_map_inline:Nn \l_tmpa_seq {
2607              \seq_put_right:Nn \l_tmpb_seq { ##1 }
2608            }
2609          }
2610        }{
2611          \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2612            \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2613          }{
2614            \tl_set:No \l__stex_notation_opprec_tl { 0 }
2615          }
2616        }
2617      }
2618    }
2619
2620    \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2621    \int_step_inline:nn { \l__stex_notation_arity_str } {
2622      \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
2623        \exp_args:NNo
2624        \seq_put_right:No \l__stex_notation_precedences_seq {
2625          \l__stex_notation_opprec_tl
2626        }
2627      }
2628    }
2629    \tl_clear:N \l_stex_notation_dummyargs_tl
2630
```

```
2631    \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2632      \exp_args:NNe
2633      \cs_set:Npn \l_stex_notation_macrocode_cs {
2634        \STEXInternalTermMathOMSiiii { \STEXInternalCurrentSymbolStr }
2635          { \l__stex_notation_suffix_str }
2636          { \l__stex_notation_opprec_tl }
2637          { \exp_not:n { #5 } }
2638      }
2639      \l_stex_notation_after_do_tl
2640    }{
2641      \str_if_in:NnTF \l__stex_notation_args_str b {
2642        \exp_args:Nne \use:nn
2643        {
2644        \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2645        \cs_set:Npn \l__stex_notation_arity_str } { {
2646          \STEXInternalTermMathOMBiiii { \STEXInternalCurrentSymbolStr }
2647            { \l__stex_notation_suffix_str }
2648            { \l__stex_notation_opprec_tl }
2649            { \exp_not:n { #5 } }
2650      }}
2651    }{
2652      \str_if_in:NnTF \l__stex_notation_args_str B {
2653        \exp_args:Nne \use:nn
2654        {
2655        \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2656        \cs_set:Npn \l__stex_notation_arity_str } { {
2657          \STEXInternalTermMathOMBiiii { \STEXInternalCurrentSymbolStr }
2658            { \l__stex_notation_suffix_str }
2659            { \l__stex_notation_opprec_tl }
2660            { \exp_not:n { #5 } }
2661        } }
2662      }{
2663        \exp_args:Nne \use:nn
2664        {
2665        \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2666        \cs_set:Npn \l__stex_notation_arity_str } { {
2667          \STEXInternalTermMathOMAiiii { \STEXInternalCurrentSymbolStr }
2668            { \l__stex_notation_suffix_str }
2669            { \l__stex_notation_opprec_tl }
2670            { \exp_not:n { #5 } }
2671        } }
2672      }
2673    }
2674
2675    \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2676    \int_zero:N \l__stex_notation_currarg_int
2677    \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2678    \__stex_notation_arguments:
2679  }
2680 }
```

(*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

154

```
2681 \cs_new_protected:Nn \__stex_notation_arguments: {
2682   \int_incr:N \l__stex_notation_currarg_int
2683   \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2684     \l_stex_notation_after_do_tl
2685   }{
2686     \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2687     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2688     \str_if_eq:VnTF \l_tmpa_str a {
2689       \__stex_notation_argument_assoc:nn{a}
2690     }{
2691       \str_if_eq:VnTF \l_tmpa_str B {
2692         \__stex_notation_argument_assoc:nn{B}
2693       }{
2694         \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpb_str
2695         \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2696           { \STEXInternalTermMathArgiii
2697             { \l_tmpa_str\int_use:N \l__stex_notation_currarg_int }
2698             { \l_tmpb_str }
2699             { ####\int_use:N \l__stex_notation_currarg_int }
2700         }
2701       }
2702       \__stex_notation_arguments:
2703     }
2704   }
2705 }
2706 }
```

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

__stex_notation_argument_assoc:nn

```
2707 \cs_new_protected:Nn \__stex_notation_argument_assoc:nn {
2708
2709   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2710     {\l__stex_notation_arity_str}{
2711     #2
2712   }
2713   \int_zero:N \l_tmpa_int
2714   \tl_clear:N \l_tmpa_tl
2715   \str_map_inline:Nn \l__stex_notation_args_str {
2716     \int_incr:N \l_tmpa_int
2717     \tl_put_right:Nx \l_tmpa_tl {
2718       \str_if_eq:nnTF {##1}{a}{ {} }{
2719         \str_if_eq:nnTF {##1}{B}{ {} }{
2720           {\_stex_term_arg:nn{##1\int_use:N \l_tmpa_int}{############### \int_use:N \l_tmpa
2721         }
2722       }
2723     }
2724   }
2725   \exp_after:wN\exp_after:wN\exp_after:wN \def
2726   \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2727   \exp_after:wN\exp_after:wN\exp_after:wN ##
2728   \exp_after:wN\exp_after:wN\exp_after:wN 1
2729   \exp_after:wN\exp_after:wN\exp_after:wN ##
2730   \exp_after:wN\exp_after:wN\exp_after:wN 2
```

```
2731    \exp_after:wN\exp_after:wN\exp_after:wN {
2732      \exp_after:wN \exp_after:wN \exp_after:wN
2733      \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2734        \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2735      }
2736    }
2737
2738    \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2739    \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
2740      \STEXInternalTermMathAssocArgiiii
2741        { #1\int_use:N \l__stex_notation_currarg_int }
2742        { \l_tmpa_str }
2743        { ####\int_use:N \l__stex_notation_currarg_int }
2744        { \l_tmpa_cs {####1} {####2} }
2745    } }
2746    \__stex_notation_arguments:
2747  }
```

(End definition for __stex_notation_argument_assoc:nn.*)*

__stex_notation_final: Called after processing all notation arguments

```
2748  \cs_new_protected:Nn \__stex_notation_restore_notation:nnnnn {
2749    \cs_generate_from_arg_count:cNnn{stex_notation_\detokenize{#1} \c_hash_str \detokenize{#2}
2750    \cs_set_nopar:Npn {#3}{#4}
2751    \tl_if_empty:nF {#5}{
2752      \tl_set:cn{stex_op_notation_\detokenize{#1} \c_hash_str \detokenize{#2}_cs}{ \comp{ #5 }
2753    }
2754    \seq_if_exist:cT { l_stex_symdecl_\detokenize{#1} _notations }{
2755      \seq_put_right:cx { l_stex_symdecl_\detokenize{#1} _notations } { \detokenize{#2} }
2756    }
2757  }
2758
2759  \cs_new_protected:Nn \__stex_notation_final: {
2760
2761    \stex_execute_in_module:x {
2762      \__stex_notation_restore_notation:nnnnn
2763        {\l_stex_get_symbol_uri_str}
2764        {\l__stex_notation_suffix_str}
2765        {\l__stex_notation_arity_str}
2766        {
2767          \exp_after:wN \exp_after:wN \exp_after:wN
2768          \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2769          { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \STEXInt
2770        }
2771        {\exp_args:No \exp_not:n \l__stex_notation_op_tl }
2772    }
2773
2774    \stex_debug:nn{symbols}{
2775      Notation~\l__stex_notation_suffix_str
2776      ~for~\l_stex_get_symbol_uri_str^^J
2777      Operator~precedence:~\l__stex_notation_opprec_tl^^J
2778      Argument~precedences:~
2779        \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2780      Notation: \cs_meaning:c {
```

156

```
2781        stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2782        \l__stex_notation_suffix_str
2783        _cs
2784      }
2785    }
2786      % HTML annotations
2787    \stex_if_do_html:T {
2788      \stex_annotate_invisible:nnn { notation }
2789      { \l_stex_get_symbol_uri_str } {
2790        \stex_annotate_invisible:nnn { notationfragment }
2791          { \l__stex_notation_suffix_str }{}
2792        \stex_annotate_invisible:nnn { precedence }
2793          { \l__stex_notation_prec_str }{}
2794
2795        \int_zero:N \l_tmpa_int
2796        \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2797        \tl_clear:N \l_tmpa_tl
2798        \int_step_inline:nn { \l__stex_notation_arity_str }{
2799          \int_incr:N \l_tmpa_int
2800          \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2801          \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_rem
2802          \str_if_eq:VnTF \l_tmpb_str a {
2803            \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2804              \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2805              \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2806            } }
2807          }{
2808            \str_if_eq:VnTF \l_tmpb_str B {
2809              \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2810                \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2811                \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2812              } }
2813            }{
2814              \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2815                \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
2816              } }
2817            }
2818          }
2819        }
2820        \stex_annotate_invisible:nnn { notationcomp }{}{
2821          \str_set:Nx \STEXInternalCurrentSymbolStr {\l_stex_get_symbol_uri_str }
2822          $ \exp_args:Nno \use:nn { \use:c {
2823            stex_notation_ \STEXInternalCurrentSymbolStr
2824            \c_hash_str \l__stex_notation_suffix_str _cs
2825          } } { \l_tmpa_tl } $
2826        }
2827        \tl_if_empty:NF \l__stex_notation_op_tl {
2828          \stex_annotate_invisible:nnn { notationopcomp }{}{
2829            $\l__stex_notation_op_tl$
2830          }
2831        }
2832      }
2833    }
2834  }
```

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

```
2835 \keys_define:nn { stex / setnotation } {
2836 %  lang    .tl_set_x:N  = \l__stex_notation_lang_str ,
2837   variant .tl_set_x:N  = \l__stex_notation_variant_str ,
2838   unknown .code:n       = \str_set:Nx
2839        \l__stex_notation_variant_str \l_keys_key_str
2840 }
2841
2842 \cs_new_protected:Nn \_stex_setnotation_args:n {
2843  % \str_clear:N \l__stex_notation_lang_str
2844   \str_clear:N \l__stex_notation_variant_str
2845   \keys_set:nn { stex / setnotation } { #1 }
2846 }
2847
2848 \cs_new_protected:Nn \__stex_notation_setnotation:nn {
2849   \seq_if_exist:cT{l_stex_symdecl_#1_notations}{
2850     \seq_remove_all:cn { l_stex_symdecl_#1 _notations }{ #2 }
2851     \seq_put_left:cn { l_stex_symdecl_#1 _notations }{ #2 }
2852   }
2853 }
2854
2855 \cs_new_protected:Nn \stex_setnotation:n {
2856   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1 _notations }
2857     { \l__stex_notation_variant_str }{
2858       \stex_execute_in_module:x{ \__stex_notation_setnotation:nn {#1}{\l__stex_notation_vari
2859       \stex_debug:nn {notations}{
2860         Setting~default~notation~
2861         {\l__stex_notation_variant_str }~for~
2862         #1 \\
2863         \expandafter\meaning\csname
2864         l_stex_symdecl_#1 _notations\endcsname
2865       }
2866     }{
2867       \msg_error:nnxx{stex}{unknownnotation}{\l__stex_notation_variant_str}{#1}
2868     }
2869 }
2870
2871 \NewDocumentCommand \setnotation {m m} {
2872   \stex_get_symbol:n { #1 }
2873   \_stex_setnotation_args:n { #2 }
2874   \stex_setnotation:n{\l_stex_get_symbol_uri_str}
2875   \stex_smsmode_do:\ignorespacesandpars
2876 }
2877
2878 \cs_new_protected:Nn \stex_copy_notations:nn {
2879   \stex_debug:nn {notations}{
2880     Copying~notations~from~#2~to~#1\\
2881     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
2882   }
2883   \tl_clear:N \l_tmpa_tl
2884   \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2885     \tl_put_right:Nn \l_tmpa_tl { {####### ##1} }
```

158

```
2886        }
2887      \seq_map_inline:cn {l_stex_symdecl_#2_notations}{\begingroup
2888        \stex_debug:nn{Here}{Here:~##1}
2889        \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2890        \edef \l_tmpa_tl {
2891          \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2892          \exp_after:wN\exp_after:wN\exp_after:wN {
2893            \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2894          }
2895        }
2896
2897        \exp_after:wN \def \exp_after:wN \l_tmpa_tl
2898        \exp_after:wN ####\exp_after:wN 1 \exp_after:wN ####\exp_after:wN 2
2899        \exp_after:wN  { \l_tmpa_tl }
2900
2901        \edef \l_tmpa_tl {
2902          \exp_after:wN \exp_not:n \exp_after:wN {
2903            \l_tmpa_tl {######## 1}{######## 2}
2904          }
2905        }
2906
2907        \stex_debug:nn{Here}{Here:~\expandafter\detokenize\expandafter{\l_tmpa_tl}}
2908
2909        \stex_execute_in_module:x {
2910          \__stex_notation_restore_notation:nnnnn
2911            {#1}{##1}
2912            { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }
2913            { \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl} }
2914            {
2915              \cs_if_exist:cT{stex_op_notation_ #2\c_hash_str ##1 _cs}{
2916                \exp_args:NNo\exp_args:No\exp_not:n{\csname stex_op_notation_ #2\c_hash_str ##1
2917              }
2918            }
2919        }\endgroup
2920      }
2921 }
2922
2923 \NewDocumentCommand \copynotation {m m} {
2924   \stex_get_symbol:n { #1 }
2925   \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2926   \stex_get_symbol:n { #2 }
2927   \exp_args:Noo
2928   \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2929   \stex_smsmode_do:\ignorespacesandpars
2930 }
2931
```

(*End definition for* \setnotation. *This function is documented on page* *19.*)

\symdef

```
2932 \keys_define:nn { stex / symdef } {
2933   name     .str_set_x:N = \l_stex_symdecl_name_str ,
2934   local    .bool_set:N  = \l_stex_symdecl_local_bool ,
2935   args     .str_set_x:N = \l_stex_symdecl_args_str ,
```

159

```
2936    type    .tl_set:N   = \l_stex_symdecl_type_tl ,
2937    def     .tl_set:N   = \l_stex_symdecl_definiens_tl ,
2938    reorder .str_set_x:N = \l_stex_symdecl_reorder_str ,
2939    op      .tl_set:N   = \l__stex_notation_op_tl ,
2940 % lang     .str_set_x:N = \l__stex_notation_lang_str ,
2941    variant .str_set_x:N = \l__stex_notation_variant_str ,
2942    prec    .str_set_x:N = \l__stex_notation_prec_str ,
2943    assoc   .choices:nn  =
2944        {bin,binl,binr,pre,conj,pwconj}
2945        {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
2946    unknown .code:n      = \str_set:Nx
2947        \l__stex_notation_variant_str \l_keys_key_str
2948 }
2949
2950 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2951    \str_clear:N \l_stex_symdecl_name_str
2952    \str_clear:N \l_stex_symdecl_args_str
2953    \str_clear:N \l_stex_symdecl_assoctype_str
2954    \str_clear:N \l_stex_symdecl_reorder_str
2955    \bool_set_false:N \l_stex_symdecl_local_bool
2956    \tl_clear:N \l_stex_symdecl_type_tl
2957    \tl_clear:N \l_stex_symdecl_definiens_tl
2958 % \str_clear:N \l__stex_notation_lang_str
2959    \str_clear:N \l__stex_notation_variant_str
2960    \str_clear:N \l__stex_notation_prec_str
2961    \tl_clear:N \l__stex_notation_op_tl
2962
2963    \keys_set:nn { stex / symdef } { #1 }
2964 }
2965
2966 \NewDocumentCommand \symdef { m O{} } {
2967    \__stex_notation_symdef_args:n { #2 }
2968    \bool_set_true:N \l_stex_symdecl_make_macro_bool
2969    \stex_symdecl_do:n { #1 }
2970    \tl_set:Nn \l_stex_notation_after_do_tl {
2971        \__stex_notation_final:
2972        \stex_smsmode_do:\ignorespacesandpars
2973    }
2974    \str_set:Nx \l_stex_get_symbol_uri_str {
2975        \l_stex_current_module_str ? \l_stex_symdecl_name_str
2976    }
2977    \exp_args:Nx \stex_notation_do:nnnnn
2978        { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } }
2979        { \prop_item:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } }
2980        { \l__stex_notation_variant_str }
2981        { \l__stex_notation_prec_str}
2982 }
2983 \stex_deactivate_macro:Nn \symdef {module~environments}
```

(*End definition for* \symdef. *This function is documented on page 79.*)

29.3   Variables

```
2984 ⟨@@=stex_variables⟩
```

```
2986 \keys_define:nn { stex / vardef } {
2987   name    .str_set_x:N  = \l__stex_variables_name_str ,
2988   args    .str_set_x:N  = \l__stex_variables_args_str ,
2989   type    .tl_set:N     = \l__stex_variables_type_tl ,
2990   def     .tl_set:N     = \l__stex_variables_def_tl ,
2991   op      .tl_set:N     = \l__stex_variables_op_tl ,
2992   prec    .str_set_x:N  = \l__stex_variables_prec_str ,
2993   reorder .str_set_x:N  = \l__stex_variables_reorder_str ,
2994   assoc   .choices:nn   =
2995     {bin,binl,binr,pre,conj,pwconj}
2996     {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
2997   bind    .choices:nn   =
2998     {forall,exists}
2999     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
3000 }
3001
3002 \cs_new_protected:Nn \__stex_variables_args:n {
3003   \str_clear:N \l__stex_variables_name_str
3004   \str_clear:N \l__stex_variables_args_str
3005   \str_clear:N \l__stex_variables_prec_str
3006   \str_clear:N \l__stex_variables_assoctype_str
3007   \str_clear:N \l__stex_variables_reorder_str
3008   \str_clear:N \l__stex_variables_bind_str
3009   \tl_clear:N \l__stex_variables_type_tl
3010   \tl_clear:N \l__stex_variables_def_tl
3011   \tl_clear:N \l__stex_variables_op_tl
3012
3013   \keys_set:nn { stex / vardef } { #1 }
3014 }
3015
3016 \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{}} {
3017   \__stex_variables_args:n {#2}
3018   \str_if_empty:NT \l__stex_variables_name_str {
3019     \str_set:Nx \l__stex_variables_name_str { #1 }
3020   }
3021   \prop_clear:N \l_tmpa_prop
3022   \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
3023
3024   \int_zero:N \l_tmpb_int
3025   \bool_set_true:N \l_tmpa_bool
3026   \str_map_inline:Nn \l__stex_variables_args_str {
3027     \token_case_meaning:NnF ##1 {
3028       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
3029       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
3030       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
3031       {\tl_to_str:n a} {
3032         \bool_set_false:N \l_tmpa_bool
3033         \int_incr:N \l_tmpb_int
3034       }
3035       {\tl_to_str:n B} {
3036         \bool_set_false:N \l_tmpa_bool
3037         \int_incr:N \l_tmpb_int
3038       }
```

161

```
3039        }{
3040          \msg_error:nnxx{stex}{error/wrongargs}{
3041            variable~\l__stex_variables_name_str
3042          }{##1}
3043        }
3044      }
3045      \bool_if:NTF \l_tmpa_bool {
3046        % possibly numeric
3047        \str_if_empty:NTF \l__stex_variables_args_str {
3048          \prop_put:Nnn \l_tmpa_prop { args } {}
3049          \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
3050        }{
3051          \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
3052          \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
3053          \str_clear:N \l_tmpa_str
3054          \int_step_inline:nn \l_tmpa_int {
3055            \str_put_right:Nn \l_tmpa_str i
3056          }
3057          \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
3058          \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
3059        }
3060      } {
3061        \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
3062        \prop_put:Nnx \l_tmpa_prop { arity }
3063          { \str_count:N \l__stex_variables_args_str }
3064      }
3065      \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
3066      \tl_set:cx { #1 }{ \stex_invoke_variable:n { \l__stex_variables_name_str } }
3067
3068      \prop_set_eq:cN { l_stex_variable_\l__stex_variables_name_str _prop} \l_tmpa_prop
3069
3070      \tl_if_empty:NF \l__stex_variables_op_tl {
3071        \cs_set:cpx {
3072          stex_var_op_notation_ \l__stex_variables_name_str _cs
3073        } { \exp_not:N\comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } } }
3074      }
3075
3076      \tl_set:Nn \l_stex_notation_after_do_tl {
3077        \exp_args:Nne \use:nn {
3078          \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str _cs }
3079            \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
3080        } {{
3081          \exp_after:wN \exp_after:wN \exp_after:wN
3082          \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
3083          { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \STEXInter
3084        }}
3085        \stex_if_do_html:T {
3086          \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
3087            \stex_annotate_invisible:nnn { precedence }
3088              { \l__stex_variables_prec_str }{}
3089            \tl_if_empty:NF \l__stex_variables_type_tl {\stex_annotate_invisible:nnn{type}{}{$\l
3090            \stex_annotate_invisible:nnn{args}{ \l__stex_variables_args_str }{}
3091            \stex_annotate_invisible:nnn{macroname}{#1}{}
3092            \tl_if_empty:NF \l__stex_variables_def_tl {
```

162

```
3093            \stex_annotate_invisible:nnn{definiens}{}
3094              {$\l__stex_variables_def_tl$}
3095          }
3096          \str_if_empty:NF \l__stex_variables_assoctype_str {
3097            \stex_annotate_invisible:nnn{assoctype}{\l__stex_variables_assoctype_str}{}
3098          }
3099          \str_if_empty:NF \l__stex_variables_reorder_str {
3100            \stex_annotate_invisible:nnn{reorderargs}{\l__stex_variables_reorder_str}{}
3101          }
3102          \int_zero:N \l_tmpa_int
3103          \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
3104          \tl_clear:N \l_tmpa_tl
3105          \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
3106            \int_incr:N \l_tmpa_int
3107            \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
3108            \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables
3109            \str_if_eq:VnTF \l_tmpb_str a {
3110              \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3111                \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3112                \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3113              } }
3114            }{
3115              \str_if_eq:VnTF \l_tmpb_str B {
3116                \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3117                  \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3118                  \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3119                } }
3120              }{
3121                \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3122                  \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
3123                } }
3124              }
3125            }
3126          }
3127          \stex_annotate_invisible:nnn { notationcomp }{}{
3128            \str_set:Nx \STEXInternalCurrentSymbolStr {var://\l__stex_variables_name_str }
3129            $ \exp_args:Nno \use:nn { \use:c {
3130              stex_var_notation_\l__stex_variables_name_str _cs
3131            } } { \l_tmpa_tl } $
3132          }
3133          \tl_if_empty:NF \l__stex_variables_op_tl {
3134            \stex_annotate_invisible:nnn { notationopcomp }{}{
3135              $\l__stex_variables_op_tl$
3136            }
3137          }
3138        }
3139        \str_if_empty:NF \l__stex_variables_bind_str {
3140          \stex_annotate_invisible:nnn {bindtype}{\l__stex_variables_bind_str,\l__stex_variabl
3141        }
3142      }\ignorespacesandpars
3143    }
3144
3145    \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
3146 }
```

163

```
3147
3148  \cs_new:Nn \_stex_reset:N {
3149    \tl_if_exist:NTF #1 {
3150      \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
3151    }{
3152      \let \exp_not:N #1 \exp_not:N \undefined
3153    }
3154  }
3155
3156  \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
3157    \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
3158    \exp_args:Nnx \use:nn {
3159      % TODO
3160      \stex_annotate_invisible:nnn {vardecl}{\clist_use:Nn\l__stex_variables_names,}{
3161        #2
3162      }
3163    }{
3164      \_stex_reset:N \varnot
3165      \_stex_reset:N \vartype
3166      \_stex_reset:N \vardefi
3167    }
3168  }
3169
3170  \NewDocumentCommand \vardef { s } {
3171    \IfBooleanTF#1 {
3172      \__stex_variables_do_complex:nn
3173    }{
3174      \__stex_variables_do_simple:nnn
3175    }
3176  }
3177
3178  \NewDocumentCommand \svar { O{} m }{
3179    \tl_if_empty:nTF {#1}{
3180      \str_set:Nn \l_tmpa_str { #2 }
3181    }{
3182      \str_set:Nn \l_tmpa_str { #1 }
3183    }
3184    \_stex_term_omv:nn {
3185      var://\l_tmpa_str
3186    }{
3187      \exp_args:Nnx \use:nn {
3188        \def\comp{\_varcomp}
3189        \str_set:Nx \STEXInternalCurrentSymbolStr { var://\l_tmpa_str }
3190        \comp{ #2 }
3191      }{
3192        \_stex_reset:N \comp
3193        \_stex_reset:N \STEXInternalCurrentSymbolStr
3194      }
3195    }
3196  }
3197
3198
3199
3200  \keys_define:nn { stex / varseq } {
```

```
3201   name     .str_set_x:N  = \l__stex_variables_name_str ,
3202   args     .int_set:N    = \l__stex_variables_args_int ,
3203   type     .tl_set:N     = \l__stex_variables_type_tl  ,
3204   mid      .tl_set:N     = \l__stex_variables_mid_tl    ,
3205   bind     .choices:nn   =
3206       {forall,exists}
3207       {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
3208 }
3209
3210 \cs_new_protected:Nn \__stex_variables_seq_args:n {
3211   \str_clear:N \l__stex_variables_name_str
3212   \int_set:Nn \l__stex_variables_args_int 1
3213   \tl_clear:N \l__stex_variables_type_tl
3214   \str_clear:N \l__stex_variables_bind_str
3215
3216   \keys_set:nn { stex / varseq } { #1 }
3217 }
3218
3219 \NewDocumentCommand \varseq {m O{} m m m}{
3220   \__stex_variables_seq_args:n { #2 }
3221   \str_if_empty:NT \l__stex_variables_name_str {
3222     \str_set:Nx \l__stex_variables_name_str { #1 }
3223   }
3224   \prop_clear:N \l_tmpa_prop
3225   \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
3226
3227   \seq_set_from_clist:Nn \l_tmpa_seq {#3}
3228   \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
3229     \msg_error:nnxx{stex}{error/seqlength}
3230       {\int_use:N \l__stex_variables_args_int}
3231       {\seq_count:N \l_tmpa_seq}
3232   }
3233   \seq_set_from_clist:Nn \l_tmpb_seq {#4}
3234   \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
3235     \msg_error:nnxx{stex}{error/seqlength}
3236       {\int_use:N \l__stex_variables_args_int}
3237       {\seq_count:N \l_tmpb_seq}
3238   }
3239   \prop_put:Nnn \l_tmpa_prop {starts} {#3}
3240   \prop_put:Nnn \l_tmpa_prop {ends} {#4}
3241
3242   \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
3243     \cs_set:Npn  {\int_use:N \l__stex_variables_args_int} { #5 }
3244
3245   \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3246   \int_step_inline:nn \l__stex_variables_args_int {
3247     \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
3248   }
3249   \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
3250   \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
3251   \tl_if_empty:NF \l__stex_variables_mid_tl {
3252     \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
3253     \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
3254   }
```

```
3255    \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3256    \int_step_inline:nn \l__stex_variables_args_int {
3257      \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
3258    }
3259    \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
3260    \tl_put_right:No \l_tmpa_tl \l_tmpb_tl

3261

3262
3263    \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl

3264
3265    \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}

3266
3267    \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}

3268
3269    \int_step_inline:nn \l__stex_variables_args_int {
3270      \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
3271        \STEXInternalTermMathArgiii{i##1}{0}{\exp_not:n{####}##1}
3272      }}
3273    }

3274
3275    \tl_set:Nx \l_tmpa_tl {
3276      \STEXInternalTermMathOMAiiii { varseq://\l__stex_variables_name_str}{}{0}{
3277        \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
3278      }
3279    }

3280
3281    \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \STEXInternalSymbolAfterInvokationTL} }

3282
3283    \exp_args:Nno \use:nn {
3284    \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
3285      \cs_set:Npn {\int_use:N \l__stex_variables_args_int}}{\l_tmpa_tl}

3286
3287    \stex_debug:nn{sequences}{New~Sequence:~
3288      \expandafter\meaning\csname stex_varseq_\l__stex_variables_name_str _cs\endcsname\\~\\
3289      \prop_to_keyval:N \l_tmpa_prop
3290    }
3291    \stex_if_do_html:T{\stex_annotate_invisible:nnn{varseq}{\l__stex_variables_name_str}{
3292      \tl_if_empty:NF \l__stex_variables_type_tl {
3293        \stex_annotate:nnn {type}{}{$\l__stex_variables_type_tl$}
3294      }
3295      \stex_annotate:nnn {args}{\int_use:N \l__stex_variables_args_int}{}
3296      \str_if_empty:NF \l__stex_variables_bind_str {
3297        \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{}
3298      }
3299      \stex_annotate:nnn{startindex}{}{$#3$}
3300      \stex_annotate:nnn{endindex}{}{$#4$}

3301
3302      \tl_clear:N \l_tmpa_tl
3303      \int_step_inline:nn \l__stex_variables_args_int {
3304        \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3305          \stex_annotate:nnn{argmarker}{##1}{}
3306        } }
3307      }
3308      \stex_annotate_invisible:nnn { notationcomp }{}{
```

166

```
3309      \str_set:Nx \STEXInternalCurrentSymbolStr {varseq://\l__stex_variables_name_str }
3310      $ \exp_args:Nno \use:nn { \use:c {
3311        stex_varseq_\l__stex_variables_name_str _cs
3312      } } { \l_tmpa_tl } $
3313    }
3314    \stex_annotate_invisible:nnn { notationopcomp }{}{
3315      $ \prop_item:Nn \l_tmpa_prop { notation } $
3316    }
3317
3318  }}
3319
3320    \prop_set_eq:cN {stex_varseq_\l__stex_variables_name_str _prop}\l_tmpa_prop
3321    \ignorespacesandpars
3322 }
3323
3324 ⟨/package⟩
```

Chapter 30

SₜₑX -Terms Implementation

```
3325  ⟨∗package⟩
3326
3327  %%%%%%%%%%%%    terms.dtx    %%%%%%%%%%%%
3328
3329  ⟨@@=stex_terms⟩
```

Warnings and error messages

```
3330  \msg_new:nnn{stex}{error/nonotation}{
3331    Symbol~#1~invoked,~but~has~no~notation#2!
3332  }
3333  \msg_new:nnn{stex}{error/notationarg}{
3334    Error~in~parsing~notation~#1
3335  }
3336  \msg_new:nnn{stex}{error/noop}{
3337    Symbol~#1~has~no~operator~notation~for~notation~#2
3338  }
3339  \msg_new:nnn{stex}{error/notallowed}{
3340    Symbol~invokation~#1~not~allowed~in~notation~component~of~#2
3341  }
3342  \msg_new:nnn{stex}{error/doubleargument}{
3343    Argument~#1~of~symbol~#2~already~assigned
3344  }
3345  \msg_new:nnn{stex}{error/overarity}{
3346    Argument~#1~invalid~for~symbol~#2~with~arity~#3
3347  }
3348
```

30.1   Symbol Invocations

`\stex_invoke_symbol:n`   Invokes a semantic macro

```
3349
3350
3351  \bool_new:N \l_stex_allow_semantic_bool
3352  \bool_set_true:N \l_stex_allow_semantic_bool
3353
```

```
3354  \cs_new_protected:Nn \stex_invoke_symbol:n {
3355    \ifvmode\indent\fi
3356    \bool_if:NTF \l_stex_allow_semantic_bool {
3357      \str_if_eq:eeF {
3358        \prop_item:cn {
3359          l_stex_symdecl_#1_prop
3360        }{ deprecate }
3361      }{}{
3362        \msg_warning:nnxx{stex}{warning/deprecated}{
3363          Symbol~#1
3364        }{
3365          \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
3366        }
3367      }
3368      \if_mode_math:
3369        \exp_after:wN \__stex_terms_invoke_math:n
3370      \else:
3371        \exp_after:wN \__stex_terms_invoke_text:n
3372      \fi: { #1 }
3373    }{
3374      \msg_error:nnxx{stex}{error/notallowed}{#1}{\STEXInternalCurrentSymbolStr}
3375    }
3376  }
3377
3378  \cs_new_protected:Nn \__stex_terms_invoke_text:n {
3379    \peek_charcode_remove:NTF ! {
3380      \__stex_terms_invoke_op_custom:nn {#1}
3381    }{
3382      \__stex_terms_invoke_custom:nn {#1}
3383    }
3384  }
3385
3386  \cs_new_protected:Nn \__stex_terms_invoke_math:n {
3387    \peek_charcode_remove:NTF ! {
3388      % operator
3389      \peek_charcode_remove:NTF * {
3390        % custom op
3391        \__stex_terms_invoke_op_custom:nn {#1}
3392      }{
3393        % op notation
3394        \peek_charcode:NTF [ {
3395          \__stex_terms_invoke_op_notation:nw {#1}
3396        }{
3397          \__stex_terms_invoke_op_notation:nw {#1}[]
3398        }
3399      }
3400    }{
3401      \peek_charcode_remove:NTF * {
3402        \__stex_terms_invoke_custom:nn {#1}
3403        % custom
3404      }{
3405        % normal
3406        \peek_charcode:NTF [ {
3407          \__stex_terms_invoke_notation:nw {#1}
```

169

```
3408          }{
3409            \__stex_terms_invoke_notation:nw {#1}[]
3410          }
3411      }
3412    }
3413  }
3414
3415
3416  \cs_new_protected:Nn \__stex_terms_invoke_op_custom:nn {
3417    \exp_args:Nnx \use:nn {
3418      \def\comp{\_comp}
3419      \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3420      \bool_set_false:N \l_stex_allow_semantic_bool
3421      \_stex_term_oms:nnn {#1}{#1 \c_hash_str CUSTOM-}{
3422        \comp{ #2 }
3423      }
3424    }{
3425      \_stex_reset:N \comp
3426      \_stex_reset:N \STEXInternalCurrentSymbolStr
3427      \bool_set_true:N \l_stex_allow_semantic_bool
3428    }
3429  }
3430
3431  \keys_define:nn { stex / terms } {
3432  % lang     .tl_set_x:N = \l_stex_notation_lang_str ,
3433    variant .tl_set_x:N = \l_stex_notation_variant_str ,
3434    unknown .code:n     = \str_set:Nx
3435        \l_stex_notation_variant_str \l_keys_key_str
3436  }
3437
3438  \cs_new_protected:Nn \__stex_terms_args:n {
3439  % \str_clear:N \l_stex_notation_lang_str
3440    \str_clear:N \l_stex_notation_variant_str
3441
3442    \keys_set:nn { stex / terms } { #1 }
3443  }
3444
3445  \cs_new_protected:Nn \stex_find_notation:nn {
3446    \__stex_terms_args:n { #2 }
3447    \seq_if_empty:cTF {
3448      l_stex_symdecl_ #1 _notations
3449    } {
3450      \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
3451    } {
3452      \str_if_empty:NTF \l_stex_notation_variant_str {
3453        \seq_get_left:cN {l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
3454      }{
3455        \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
3456          \l_stex_notation_variant_str
3457        }{
3458        % \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3459        }{
3460          \msg_error:nnxx{stex}{error/nonotation}{#1}{
3461            ~\l_stex_notation_variant_str
```

170

```
3462              }
3463            }
3464          }
3465        }
3466      }
3467
3468      \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3469        \exp_args:Nnx \use:nn {
3470          \def\comp{\_comp}
3471          \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3472          \stex_find_notation:nn { #1 }{ #2 }
3473          \bool_set_false:N \l_stex_allow_semantic_bool
3474          \cs_if_exist:cTF {
3475            stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3476          }{
3477            \_stex_term_oms:nnn { #1 }{
3478              #1 \c_hash_str \l_stex_notation_variant_str
3479            }{
3480              \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3481            }
3482          }{
3483            \int_compare:nNnTF {\prop_item:cn {l_stex_symdecl_#1_prop}{arity}} = 0{
3484              \cs_if_exist:cTF {
3485                stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3486              }{
3487                \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
3488                  \_stex_reset:N \comp
3489                  \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
3490                  \_stex_reset:N \STEXInternalCurrentSymbolStr
3491                  \bool_set_true:N \l_stex_allow_semantic_bool
3492                }
3493                \def\comp{\_comp}
3494                \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3495                \bool_set_false:N \l_stex_allow_semantic_bool
3496                \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3497              }{
3498                \msg_error:nnxx{stex}{error/nonotation}{#1}{
3499                  ~\l_stex_notation_variant_str
3500                }
3501              }
3502            }{
3503              \msg_error:nnxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3504            }
3505          }
3506        }{
3507          \_stex_reset:N \comp
3508          \_stex_reset:N \STEXInternalCurrentSymbolStr
3509          \bool_set_true:N \l_stex_allow_semantic_bool
3510        }
3511      }
3512
3513      \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {
3514        \stex_find_notation:nn { #1 }{ #2 }
3515        \cs_if_exist:cTF {
```

```
3516    stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3517  }{
3518    \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
3519      \_stex_reset:N \comp
3520      \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
3521      \_stex_reset:N \STEXInternalCurrentSymbolStr
3522      \bool_set_true:N \l_stex_allow_semantic_bool
3523    }
3524    \def\comp{\_comp}
3525    \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3526    \bool_set_false:N \l_stex_allow_semantic_bool
3527    \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3528  }{
3529    \msg_error:nnxx{stex}{error/nonotation}{#1}{
3530      ~\l_stex_notation_variant_str
3531    }
3532  }
3533 }
3534
3535 \prop_new:N \l__stex_terms_custom_args_prop
3536
3537 \cs_new_protected:Nn\__stex_terms_custom_comp:n{\bool_set_false:N \l_stex_allow_semantic_boo
3538
3539 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3540   \exp_args:Nnx \use:nn {
3541     \def\comp{\__stex_terms_custom_comp:n}
3542     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3543     \prop_clear:N \l__stex_terms_custom_args_prop
3544     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3545     \prop_get:cnN {
3546       l_stex_symdecl_#1 _prop
3547     }{ args } \l_tmpa_str
3548     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3549     \tl_set:Nn \arg { \__stex_terms_arg: }
3550     \str_if_empty:NTF \l_tmpa_str {
3551       \_stex_term_oms:nnn {#1}{#1\c_hash_str CUSTOM-}{\ignorespaces#2}
3552     }{
3553       \str_if_in:NnTF \l_tmpa_str b {
3554         \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
3555       }{
3556         \str_if_in:NnTF \l_tmpa_str B {
3557           \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
3558         }{
3559           \_stex_term_oma:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
3560         }
3561       }
3562     }
3563     % TODO check that all arguments exist
3564   }{
3565     \_stex_reset:N \STEXInternalCurrentSymbolStr
3566     \_stex_reset:N \arg
3567     \_stex_reset:N \comp
3568     \_stex_reset:N \l__stex_terms_custom_args_prop
3569     %\bool_set_true:N \l_stex_allow_semantic_bool
```

172

```
3570      }
3571  }
3572
3573  \NewDocumentCommand \__stex_terms_arg: { s O{} m}{
3574    \tl_if_empty:nTF {#2}{
3575      \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
3576      \bool_set_true:N \l_tmpa_bool
3577      \bool_do_while:Nn \l_tmpa_bool {
3578        \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int
3579          \int_incr:N \l_tmpa_int
3580        }{
3581          \bool_set_false:N \l_tmpa_bool
3582        }
3583      }
3584    }{
3585      \int_set:Nn \l_tmpa_int { #2 }
3586    }
3587    \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
3588    \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
3589      \msg_error:nnxxx{stex}{error/overarity}
3590        {\int_use:N \l_tmpa_int}
3591        {\STEXInternalCurrentSymbolStr}
3592        {\str_count:N \l_tmpa_str}
3593    }
3594    \str_set:Nx \l_tmpa_str {\str_item:Nn \l_tmpa_str \l_tmpa_int}
3595    \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
3596      \bool_lazy_any:nF {
3597        {\str_if_eq_p:Vn \l_tmpa_str {a}}
3598        {\str_if_eq_p:Vn \l_tmpa_str {B}}
3599      }{
3600        \msg_error:nnxx{stex}{error/doubleargument}
3601          {\int_use:N \l_tmpa_int}
3602          {\STEXInternalCurrentSymbolStr}
3603      }
3604    }
3605    \exp_args:NNx \prop_put:Nnn \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {\ign
3606    \bool_set_true:N \l_stex_allow_semantic_bool
3607    \IfBooleanTF#1{
3608      \stex_annotate_invisible:n { %TODO
3609        \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{\ignorespaces#3}
3610      }
3611    }{ %TODO
3612      \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{\ignorespaces#3}
3613    }
3614    \bool_set_false:N \l_stex_allow_semantic_bool
3615  }
3616
3617
3618  \cs_new_protected:Nn \_stex_term_arg:nn {
3619    \bool_set_true:N \l_stex_allow_semantic_bool
3620    \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3621    \bool_set_false:N \l_stex_allow_semantic_bool
3622  }
3623
```

173

```
3624  \cs_new_protected:Npn \STEXInternalTermMathArgiii #1#2#3 {
3625    \exp_args:Nnx \use:nn
3626      { \int_set:Nn \l__stex_terms_downprec { #2 }
3627          \_stex_term_arg:nn { #1 }{ #3 }
3628      }
3629      { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3630  }
```

(*End definition for* \stex_invoke_symbol:n. *This function is documented on page* *80.*)

```
3631  \cs_new_protected:Npn \STEXInternalTermMathAssocArgiiii #1#2#3#4 {
3632    \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3633    \tl_set:Nn \l_tmpb_tl {\STEXInternalTermMathArgiii{#1}{#2}}
3634    \tl_if_empty:nTF { #3 }{
3635      \STEXInternalTermMathArgiii{#1}{#2}{}
3636    }{
3637      \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 }}{
3638        \expandafter\if\expandafter\relax\noexpand#3
3639          \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_maybe_sequence:Nn#3{#1}}
3640        \else
3641          \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_simple:nn{#1}{#3}}
3642        \fi
3643        \l_tmpa_tl
3644      }{
3645        \__stex_terms_math_assoc_arg_simple:nn{#1}{#3}
3646      }
3647    }
3648  }
3649
3650  \cs_new_protected:Nn \__stex_terms_math_assoc_arg_maybe_sequence:Nn {
3651    \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
3652    \str_if_empty:NTF \l_tmpa_str {
3653      \exp_args:Nx \cs_if_eq:NNTF {
3654        \tl_head:N #1
3655      } \stex_invoke_sequence:n {
3656        \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
3657        \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
3658        \tl_set:Nx \l_tmpa_tl {\prop_item:cn {stex_varseq_\l_tmpa_str _prop}{notation}}
3659        \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
3660        \tl_set:Nx \l_tmpa_tl {{\exp_not:N \exp_not:n{
3661          \exp_not:n{\exp_args:Nnx \use:nn} {
3662            \exp_not:n {
3663              \def\comp{\_varcomp}
3664              \str_set:Nn \STEXInternalCurrentSymbolStr
3665            } {varseq://\l_tmpa_str}
3666            \exp_not:n{ ##1 }
3667          }{
3668            \exp_not:n {
3669              \_stex_reset:N \comp
3670              \_stex_reset:N \STEXInternalCurrentSymbolStr
3671            }
3672          }
3673        }}}
```

174

```
3674        \exp_args:Nno \use:nn {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
3675        \seq_reverse:N \l_tmpa_seq
3676        \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
3677        \seq_map_inline:Nn \l_tmpa_seq {
3678          \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3679            \exp_args:Nno
3680            \l_tmpa_cs { ##1 } \l_tmpa_tl
3681          }
3682        }
3683        \tl_set:Nx \l_tmpa_tl {
3684          \_stex_term_omv:nn {varseq://\l_tmpa_str}{
3685            \exp_args:No \exp_not:n \l_tmpa_tl
3686          }
3687        }
3688        \exp_args:No\l_tmpb_tl\l_tmpa_tl
3689      }{
3690        \__stex_terms_math_assoc_arg_simple:nn{#2} { #1 }
3691      }
3692    } {
3693      \__stex_terms_math_assoc_arg_simple:nn{#2} { #1 }
3694    }
3695
3696  }
3697
3698  \cs_new_protected:Nn \__stex_terms_math_assoc_arg_simple:nn {
3699    \clist_set:Nn \l_tmpa_clist{ #2 }
3700    \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3701      \tl_set:Nn \l_tmpa_tl { \_stex_term_arg:nn{A#1}{ #2 } }
3702    }{
3703      \clist_reverse:N \l_tmpa_clist
3704      \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3705      \tl_set:Nx \l_tmpa_tl { \_stex_term_arg:nn{A#1}{
3706        \exp_args:No \exp_not:n \l_tmpa_tl
3707      }}
3708      \clist_map_inline:Nn \l_tmpa_clist {
3709        \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3710          \exp_args:Nno
3711          \l_tmpa_cs { \_stex_term_arg:nn{A#1}{##1} } \l_tmpa_tl
3712        }
3713      }
3714    }
3715    \exp_args:No\l_tmpb_tl\l_tmpa_tl
3716  }
```

(*End definition for* `\STEXInternalTermMathAssocArgiiii`*. This function is documented on page 81.*)

30.2 Terms

Precedences:

```
3717  \tl_const:Nx \infprec {\int_use:N \c_max_int}
3718  \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
```

```
3719 \int_new:N \l__stex_terms_downprec
3720 \int_set_eq:NN \l__stex_terms_downprec \infprec
```

(*End definition for* \infprec, \neginfprec, *and* \l__stex_terms_downprec. *These variables are documented on page 81.*)

Bracketing:

```
3721 \tl_set:Nn \l__stex_terms_left_bracket_str (
3722 \tl_set:Nn \l__stex_terms_right_bracket_str )
```

(*End definition for* \l__stex_terms_left_bracket_str *and* \l__stex_terms_right_bracket_str.)

Compares precedences and insert brackets accordingly

```
3723 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
3724   \bool_if:NTF \l__stex_terms_brackets_done_bool {
3725     \bool_set_false:N \l__stex_terms_brackets_done_bool
3726     #2
3727   } {
3728     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
3729       \bool_if:NTF \l_stex_inparray_bool { #2 }{
3730         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
3731         \dobrackets { #2 }
3732       }
3733     }{ #2 }
3734   }
3735 }
```

(*End definition for* __stex_terms_maybe_brackets:nn.)

```
3736 \bool_new:N \l__stex_terms_brackets_done_bool
3737 %\RequirePackage{scalerel}
3738 \cs_new_protected:Npn \dobrackets #1 {
3739   %\ThisStyle{\if D\m@switch
3740   %   \exp_args:Nnx \use:nn
3741   %     { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
3742   %     { \exp_not:N\right\l__stex_terms_right_bracket_str }
3743   %  \else
3744     \exp_args:Nnx \use:nn
3745     {
3746       \bool_set_true:N \l__stex_terms_brackets_done_bool
3747       \int_set:Nn \l__stex_terms_downprec \infprec
3748       \l__stex_terms_left_bracket_str
3749       #1
3750     }
3751     {
3752       \bool_set_false:N \l__stex_terms_brackets_done_bool
3753       \l__stex_terms_right_bracket_str
3754       \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3755     }
3756   %\fi}
3757 }
```

(*End definition for* \dobrackets. *This function is documented on page 81.*)

```
3758 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
3759   \exp_args:Nnx \use:nn
3760   {
3761     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
3762     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
3763     #3
3764   }
3765   {
3766     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
3767       {\l__stex_terms_left_bracket_str}
3768     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
3769       {\l__stex_terms_right_bracket_str}
3770   }
3771 }
```

(*End definition for* \withbrackets. *This function is documented on page 81.*)

```
3772 \cs_new_protected:Npn \STEXinvisible #1 {
3773   \stex_annotate_invisible:n { #1 }
3774 }
```

(*End definition for* \STEXinvisible. *This function is documented on page 81.*)

OMDoc terms:

```
3775 \cs_new_protected:Nn \_stex_term_oms:nnn {
3776   \stex_annotate:nnn{ OMID }{ #2 }{
3777     #3
3778   }
3779 }
3780
3781 \cs_new_protected:Npn \STEXInternalTermMathOMSiiii #1#2#3#4 {
3782   \__stex_terms_maybe_brackets:nn { #3 }{
3783     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3784   }
3785 }
```

(*End definition for* \STEXInternalTermMathOMSiiii. *This function is documented on page 80.*)

```
3786 \cs_new_protected:Nn \_stex_term_omv:nn {
3787   \stex_annotate:nnn{ OMV }{ #1 }{
3788     #2
3789   }
3790 }
```

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

```
3791 \cs_new_protected:Nn \_stex_term_oma:nnn {
3792   \stex_annotate:nnn{ OMA }{ #2 }{
3793     #3
3794   }
```

177

```
3795 }
3796
3797 \cs_new_protected:Npn \STEXInternalTermMathOMAiiii #1#2#3#4 {
3798   \__stex_terms_maybe_brackets:nn { #3 }{
3799     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3800   }
3801 }
```

(*End definition for* \STEXInternalTermMathOMAiiii. *This function is documented on page* 80.)

```
3802 \cs_new_protected:Nn \_stex_term_ombind:nnn {
3803   \stex_annotate:nnn{ OMBIND }{ #2 }{
3804     #3
3805   }
3806 }
3807
3808 \cs_new_protected:Npn \STEXInternalTermMathOMBiiii #1#2#3#4 {
3809   \__stex_terms_maybe_brackets:nn { #3 }{
3810     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3811   }
3812 }
```

(*End definition for* \STEXInternalTermMathOMBiiii. *This function is documented on page* 80.)

\symref
\symname

```
3813 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3814
3815 \keys_define:nn { stex / symname } {
3816   pre     .tl_set_x:N  = \l__stex_terms_pre_tl ,
3817   post    .tl_set_x:N  = \l__stex_terms_post_tl ,
3818   root    .tl_set_x:N  = \l__stex_terms_root_tl
3819 }
3820
3821 \cs_new_protected:Nn \stex_symname_args:n {
3822   \tl_clear:N \l__stex_terms_post_tl
3823   \tl_clear:N \l__stex_terms_pre_tl
3824   \tl_clear:N \l__stex_terms_root_str
3825   \keys_set:nn { stex / symname } { #1 }
3826 }
3827
3828 \NewDocumentCommand \symref { m m }{
3829   \let\compemph_uri_prev:\compemph@uri
3830   \let\compemph@uri\symrefemph@uri
3831   \STEXsymbol{#1}!{ #2 }
3832   \let\compemph@uri\compemph_uri_prev:
3833 }
3834
3835 \NewDocumentCommand \synonym { O{} m m}{
3836   \stex_symname_args:n { #1 }
3837   \let\compemph_uri_prev:\compemph@uri
3838   \let\compemph@uri\symrefemph@uri
3839   % TODO
3840   \STEXsymbol{#2}!{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
3841   \let\compemph@uri\compemph_uri_prev:
```

```
3842 }
3843
3844 \NewDocumentCommand \symname { O{} m }{
3845   \stex_symname_args:n { #1 }
3846   \stex_get_symbol:n { #2 }
3847   \str_set:Nx \l_tmpa_str {
3848     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3849   }
3850   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3851
3852   \let\compemph_uri_prev:\compemph@uri
3853   \let\compemph@uri\symrefemph@uri
3854   \exp_args:NNx \use:nn
3855   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmmode*\fi{
3856     \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
3857   } }
3858   \let\compemph@uri\compemph_uri_prev:
3859 }
3860
3861 \NewDocumentCommand \Symname { O{} m }{
3862   \stex_symname_args:n { #1 }
3863   \stex_get_symbol:n { #2 }
3864   \str_set:Nx \l_tmpa_str {
3865     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3866   }
3867   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3868   \let\compemph_uri_prev:\compemph@uri
3869   \let\compemph@uri\symrefemph@uri
3870   \exp_args:NNx \use:nn
3871   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmmode*\fi{
3872     \exp_after:wN \stex_capitalize:n \l_tmpa_str
3873       \l__stex_terms_post_tl
3874   } }
3875   \let\compemph@uri\compemph_uri_prev:
3876 }
```

(*End definition for* \symref *and* \symname. *These functions are documented on page* *80.*)

30.3   Notation Components

```
3877 ⟨@@=stex_notationcomps⟩
```

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

```
3878 \cs_new_protected:Npn \_comp #1 {
3879   \str_if_empty:NF \STEXInternalCurrentSymbolStr {
3880     \stex_html_backend:TF {
3881       \stex_annotate:nnn { comp }{ \STEXInternalCurrentSymbolStr }{ #1 }
3882     }{
3883       \exp_args:Nnx \compemph@uri { #1 } { \STEXInternalCurrentSymbolStr }
3884     }
3885   }
3886 }
3887
3888 \cs_new_protected:Npn \_varcomp #1 {
```

179

```
3889    \str_if_empty:NF \STEXInternalCurrentSymbolStr {
3890      \stex_html_backend:TF {
3891        \stex_annotate:nnn { varcomp }{ \STEXInternalCurrentSymbolStr }{ #1 }
3892      }{
3893        \exp_args:Nnx \varemph@uri { #1 } { \STEXInternalCurrentSymbolStr }
3894      }
3895    }
3896  }
3897
3898  \def\comp{\_comp}
3899
3900  \cs_new_protected:Npn \compemph@uri #1 #2 {
3901      \compemph{ #1 }
3902  }
3903
3904
3905  \cs_new_protected:Npn \compemph #1 {
3906      #1
3907  }
3908
3909  \cs_new_protected:Npn \defemph@uri #1 #2 {
3910      \defemph{#1}
3911  }
3912
3913  \cs_new_protected:Npn \defemph #1 {
3914      \textbf{#1}
3915  }
3916
3917  \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3918      \symrefemph{#1}
3919  }
3920
3921  \cs_new_protected:Npn \symrefemph #1 {
3922      \emph{#1}
3923  }
3924
3925  \cs_new_protected:Npn \varemph@uri #1 #2 {
3926      \varemph{#1}
3927  }
3928
3929  \cs_new_protected:Npn \varemph #1 {
3930      #1
3931  }
```

(*End definition for* `\comp` *and others. These functions are documented on page 81.*)

`\ellipses`

```
3932  \NewDocumentCommand \ellipses {} { \ldots }
```

(*End definition for* `\ellipses`*. This function is documented on page 81.*)

`\parray`
`\prmatrix`
`\parrayline`
`\parraylineh`
`\parraycell`

```
3933  \bool_new:N \l_stex_inparray_bool
3934  \bool_set_false:N \l_stex_inparray_bool
3935  \NewDocumentCommand \parray { m m } {
```

```
3936     \begingroup
3937     \bool_set_true:N \l_stex_inparray_bool
3938     \begin{array}{#1}
3939        #2
3940     \end{array}
3941     \endgroup
3942   }
3943
3944   \NewDocumentCommand \prmatrix { m } {
3945     \begingroup
3946     \bool_set_true:N \l_stex_inparray_bool
3947     \begin{matrix}
3948        #1
3949     \end{matrix}
3950     \endgroup
3951   }
3952
3953   \def \maybephline {
3954     \bool_if:NT \l_stex_inparray_bool {\hline}
3955   }
3956
3957   \def \parrayline #1 #2 {
3958     #1 #2 \bool_if:NT \l_stex_inparray_bool {\\}
3959   }
3960
3961   \def \pmrow #1 { \parrayline{}{ #1 } }
3962
3963   \def \parraylineh #1 #2 {
3964     #1 #2 \bool_if:NT \l_stex_inparray_bool {\\\hline}
3965   }
3966
3967   \def \parraycell #1 {
3968     #1 \bool_if:NT \l_stex_inparray_bool {&}
3969   }
```

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

30.4   Variables

```
3970   ⟨@@=stex_variables⟩
```

\stex_invoke_variable:n   Invokes a variable

```
3971   \cs_new_protected:Nn \stex_invoke_variable:n {
3972     \if_mode_math:
3973       \exp_after:wN \__stex_variables_invoke_math:n
3974     \else:
3975       \exp_after:wN \__stex_variables_invoke_text:n
3976     \fi: {#1}
3977   }
3978
3979   \cs_new_protected:Nn \__stex_variables_invoke_text:n {
3980     \peek_charcode_remove:NTF ! {
3981       \__stex_variables_invoke_op_custom:nn {#1}
3982     }{
```

```
3983        \__stex_variables_invoke_custom:nn {#1}
3984    }
3985 }
3986
3987
3988 \cs_new_protected:Nn \__stex_variables_invoke_math:n {
3989    \peek_charcode_remove:NTF ! {
3990      \peek_charcode_remove:NTF ! {
3991        \peek_charcode:NTF [ {
3992          % TODO throw error
3993        }{
3994          \__stex_variables_invoke_op_custom:nn
3995        }
3996      }{
3997        \__stex_variables_invoke_op:n { #1 }
3998      }
3999    }{
4000      \peek_charcode_remove:NTF * {
4001        \__stex_variables_invoke_custom:nn { #1 }
4002      }{
4003        \__stex_variables_invoke_math_ii:n { #1 }
4004      }
4005    }
4006 }
4007
4008 \cs_new_protected:Nn \__stex_variables_invoke_op_custom:nn {
4009    \exp_args:Nnx \use:nn {
4010      \def\comp{\_varcomp}
4011      \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4012      \bool_set_false:N \l_stex_allow_semantic_bool
4013      \_stex_term_omv:nn {var://#1}{
4014        \comp{ #2 }
4015      }
4016    }{
4017      \_stex_reset:N \comp
4018      \_stex_reset:N \STEXInternalCurrentSymbolStr
4019      \bool_set_true:N \l_stex_allow_semantic_bool
4020    }
4021 }
4022
4023 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
4024    \cs_if_exist:cTF {
4025      stex_var_op_notation_ #1 _cs
4026    }{
4027      \exp_args:Nnx \use:nn {
4028        \def\comp{\_varcomp}
4029        \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4030        \_stex_term_omv:nn { var://#1 }{
4031          \use:c{stex_var_op_notation_ #1 _cs }
4032        }
4033      }{
4034        \_stex_reset:N \comp
4035        \_stex_reset:N \STEXInternalCurrentSymbolStr
4036      }
```

```
4037    }{
4038      \int_compare:nNnTF {\prop_item:cn {l_stex_variable_#1_prop}{arity}} = 0{
4039        \__stex_variables_invoke_math_ii:n {#1}
4040      }{
4041        \msg_error:nnxx{stex}{error/noop}{variable~#1}{}
4042      }
4043    }
4044 }
4045
4046 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n  #1 {
4047    \cs_if_exist:cTF {
4048      stex_var_notation_#1_cs
4049    }{
4050      \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4051        \_stex_reset:N \comp
4052        \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
4053        \_stex_reset:N \STEXInternalCurrentSymbolStr
4054        \bool_set_true:N \l_stex_allow_semantic_bool
4055      }
4056      \def\comp{\_varcomp}
4057      \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4058      \bool_set_false:N \l_stex_allow_semantic_bool
4059      \use:c{stex_var_notation_#1_cs}
4060    }{
4061      \msg_error:nnxx{stex}{error/nonotation}{variable~#1}{s}
4062    }
4063 }
4064
4065 \cs_new_protected:Nn \__stex_variables_invoke_custom:nn {
4066    \exp_args:Nnx \use:nn {
4067      \def\comp{\_varcomp}
4068      \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4069      \prop_clear:N \l__stex_terms_custom_args_prop
4070      \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
4071      \prop_get:cnN {
4072        l_stex_variable_#1 _prop
4073      }{ args } \l_tmpa_str
4074      \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
4075      \tl_set:Nn \arg { \__stex_terms_arg: }
4076      \str_if_empty:NTF \l_tmpa_str {
4077        \_stex_term_omv:nn {var://#1}{\ignorespaces#2}
4078      }{
4079        \str_if_in:NnTF \l_tmpa_str b {
4080          \_stex_term_ombind:nnn {var://#1}{}{\ignorespaces#2}
4081        }{
4082          \str_if_in:NnTF \l_tmpa_str B {
4083            \_stex_term_ombind:nnn {var://#1}{}{\ignorespaces#2}
4084          }{
4085            \_stex_term_oma:nnn {var://#1}{}{\ignorespaces#2}
4086          }
4087        }
4088      }
4089      % TODO check that all arguments exist
4090    }{
```

```
4091        \_stex_reset:N \STEXInternalCurrentSymbolStr
4092        \_stex_reset:N \arg
4093        \_stex_reset:N \comp
4094        \_stex_reset:N \l__stex_terms_custom_args_prop
4095        %\bool_set_true:N \l_stex_allow_semantic_bool
4096    }
4097 }
```

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

30.5   Sequences

```
4098 ⟨@@=stex_sequences⟩
4099
4100 \cs_new_protected:Nn \stex_invoke_sequence:n {
4101    \peek_charcode_remove:NTF ! {
4102        \_stex_term_omv:nn {varseq://#1}{
4103            \exp_args:Nnx \use:nn {
4104                \def\comp{\_varcomp}
4105                \str_set:Nn \STEXInternalCurrentSymbolStr {varseq://#1}
4106                \prop_item:cn{stex_varseq_#1_prop}{notation}
4107            }{
4108                \_stex_reset:N \comp
4109                \_stex_reset:N \STEXInternalCurrentSymbolStr
4110            }
4111        }
4112    }{
4113        \bool_set_false:N \l_stex_allow_semantic_bool
4114        \def\comp{\_varcomp}
4115        \str_set:Nn \STEXInternalCurrentSymbolStr {varseq://#1}
4116        \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4117            \_stex_reset:N \comp
4118            \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
4119            \_stex_reset:N \STEXInternalCurrentSymbolStr
4120            \bool_set_true:N \l_stex_allow_semantic_bool
4121        }
4122        \use:c { stex_varseq_#1_cs }
4123    }
4124 }
```

```
4125 ⟨/package⟩
```

Chapter 31

sTeX
-Structural Features
Implementation

```
4126 ⟨∗package⟩
4127
4128 %%%%%%%%%%%%   features.dtx   %%%%%%%%%%%%
4129
```

Warnings and error messages

```
4130 \msg_new:nnn{stex}{error/copymodule/notallowed}{
4131   Symbol~#1~can~not~be~assigned~in~copymodule~#2
4132 }
4133 \msg_new:nnn{stex}{error/interpretmodule/nodefiniens}{
4134   Symbol~#1~not~assigned~in~interpretmodule~#2
4135 }
4136
4137 \msg_new:nnn{stex}{error/unknownstructure}{
4138   No~structure~#1~found!
4139 }
4140
4141 \msg_new:nnn{stex}{error/unknownfield}{
4142   No~field~#1~in~instance~#2~found!\\#3
4143 }
4144
4145 \msg_new:nnn{stex}{error/keyval}{
4146   Invalid~key=value~pair:#1
4147 }
4148 \msg_new:nnn{stex}{error/instantiate/missing}{
4149   Assignments~missing~in~instantiate:~#1
4150 }
4151 \msg_new:nnn{stex}{error/incompatible}{
4152   Incompatible~signature:~#1~(#2)~and~#3~(#4)
4153 }
4154
```

31.1 Imups with modification

```
4155 ⟨@@=stex_copymodule⟩
4156 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
4157   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4158     \tl_set:Nn \l_tmpa_tl { #1 }
4159     \__stex_copymodule_get_symbol_from_cs:
4160   }{
4161     % argument is a string
4162     % is it a command name?
4163     \cs_if_exist:cTF { #1 }{
4164       \cs_set_eq:Nc \l_tmpa_tl { #1 }
4165       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4166       \str_if_empty:NTF \l_tmpa_str {
4167         \exp_args:Nx \cs_if_eq:NNTF {
4168           \tl_head:N \l_tmpa_tl
4169         } \stex_invoke_symbol:n {
4170           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
4171         }{
4172           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4173         }
4174       } {
4175         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4176       }
4177     }{
4178       % argument is not a command name
4179       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4180       % \l_stex_all_symbols_seq
4181     }
4182   }
4183 }
4184
4185 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
4186   \str_set:Nn \l_tmpa_str { #1 }
4187   \bool_set_false:N \l_tmpa_bool
4188   \bool_if:NF \l_tmpa_bool {
4189     \tl_set:Nn \l_tmpa_tl {
4190       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
4191     }
4192   \str_set:Nn \l_tmpa_str { #1 }
4193   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4194   \seq_map_inline:Nn #2 {
4195     \str_set:Nn \l_tmpb_str { ##1 }
4196     \str_if_eq:eeT { \l_tmpa_str } {
4197       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4198     } {
4199       \seq_map_break:n {
4200         \tl_set:Nn \l_tmpa_tl {
4201           \str_set:Nn \l_stex_get_symbol_uri_str {
4202             ##1
4203           }
4204         }
4205       }
4206     }
```

```
4207          }
4208        \l_tmpa_tl
4209      }
4210  }
4211
4212  \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
4213    \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4214      { \tl_tail:N \l_tmpa_tl }
4215    \tl_if_single:NTF \l_tmpa_tl {
4216      \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
4217        \exp_after:wN \str_set:Nn \exp_after:wN
4218          \l_stex_get_symbol_uri_str \l_tmpa_tl
4219        \__stex_copymodule_get_symbol_check:n { #1 }
4220      }{
4221        % TODO
4222        % tail is not a single group
4223      }
4224    }{
4225      % TODO
4226      % tail is not a single group
4227    }
4228  }
4229
4230  \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
4231    \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
4232      \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
4233        :~\seq_use:Nn #1 {,~}
4234      }
4235    }
4236  }
4237
4238  \cs_new_protected:Nn \stex_copymodule_start:nnnn {
4239    % import module
4240    \stex_import_module_uri:nn { #1 } { #2 }
4241    \str_set:Nx \l_stex_current_copymodule_name_str {#3}
4242    \stex_import_require_module:nnnn
4243      { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4244      { \l_stex_import_path_str } { \l_stex_import_name_str }
4245
4246    \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
4247    \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
4248
4249    % fields
4250    \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
4251    \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4252      \seq_map_inline:cn {c_stex_module_##1_constants}{
4253        \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
4254          ##1 ? ####1
4255        }
4256      }
4257    }
4258
4259    % setup prop
4260    \seq_clear:N \l_tmpa_seq
```

```
4261    \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
4262      name     = \l_stex_current_copymodule_name_str ,
4263      module   = \l_stex_current_module_str ,
4264      from     = \l_stex_import_ns_str ?\l_stex_import_name_str ,
4265      includes = \l_tmpa_seq %,
4266    % fields   = \l_tmpa_seq
4267    }
4268    \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
4269      as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
4270    \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_se
4271    \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,

4273    \stex_if_do_html:T {
4274      \begin{stex_annotate_env} {#4} {
4275        \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4276      }
4277      \stex_annotate_invisible:nnn{domain}{\l_stex_import_ns_str ?\l_stex_import_name_str}{}
4278    }
4279  }

4281 \cs_new_protected:Nn \stex_copymodule_end:n {
4282   % apply to every field
4283   \def \l_tmpa_cs ##1 ##2 {#1}

4285   \tl_clear:N \__stex_copymodule_module_tl
4286   \tl_clear:N \__stex_copymodule_exec_tl

4288   %\prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
4289   \seq_clear:N \__stex_copymodule_fields_seq

4291   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4292     \seq_map_inline:cn {c_stex_module_##1_constants}{

4294       \tl_clear:N \__stex_copymodule_curr_symbol_tl % <- wrap in current symbol html
4295       \l_tmpa_cs{##1}{####1}

4297       \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
4298         \str_set_eq:Nc \__stex_copymodule_curr_name_str {l__stex_copymodule_copymodule_##1?#
4299         \stex_if_do_html:T {
4300           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4301             \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?###
4302           }
4303         }
4304       }{
4305         \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str /
4306       }

4308       \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
4309       \prop_put:Nnx \l_tmpa_prop { name } \__stex_copymodule_curr_name_str
4310       \prop_put:Nnx \l_tmpa_prop { module } \l_stex_current_module_str

4312       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
4313         \stex_if_do_html:T {
4314           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
```

```
4315              $\stex_annotate_invisible:nnn{definiens}{}{\exp_after:wN \exp_not:N\csname l__st
4316           }
4317        }
4318        \prop_put:Nnn \l_tmpa_prop { defined } { true }
4319      }
4320
4321      \stex_add_constant_to_current_module:n \__stex_copymodule_curr_name_str
4322      \tl_put_right:Nx \__stex_copymodule_module_tl {
4323        \seq_clear:c {l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_n
4324        \prop_set_from_keyval:cn {
4325          l_stex_symdecl_\l_stex_current_module_str ? \__stex_copymodule_curr_name_str _prop
4326        }{
4327          \prop_to_keyval:N \l_tmpa_prop
4328        }
4329      }
4330
4331      \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
4332        \stex_if_do_html:T {
4333          \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4334            \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1
4335          }
4336        }
4337        \tl_put_right:Nx \__stex_copymodule_module_tl {
4338          \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
4339            \stex_invoke_symbol:n {
4340              \l_stex_current_module_str ? \__stex_copymodule_curr_name_str
4341            }
4342          }
4343        }
4344      }
4345
4346      \seq_put_right:Nx \__stex_copymodule_fields_seq {\l_stex_current_module_str ? \__stex_
4347
4348      \tl_put_right:Nx \__stex_copymodule_exec_tl {
4349        \stex_copy_notations:nn {\l_stex_current_module_str ? \__stex_copymodule_curr_name_s
4350      }
4351
4352      \tl_put_right:Nx \__stex_copymodule_exec_tl {
4353        \stex_if_do_html:TF{
4354          \stex_annotate_invisible:nnn{assignment} {##1?####1} { \exp_after:wN \exp_not:n \e
4355        }{
4356          \exp_after:wN \exp_not:n \exp_after:wN {\__stex_copymodule_curr_symbol_tl}
4357        }
4358      }
4359    }
4360  }
4361
4362
4363    \prop_put:Nno \l_stex_current_copymodule_prop {fields} \__stex_copymodule_fields_seq
4364    \tl_put_left:Nx \__stex_copymodule_module_tl {
4365      \prop_set_from_keyval:cn {
4366        l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _pro
4367      }{
4368        \prop_to_keyval:N \l_stex_current_copymodule_prop
```

```
4369        }
4370      }
4371
4372      \seq_gput_right:cx{c_stex_module_\l_stex_current_module_str _copymodules}{
4373        \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4374      }
4375
4376      \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4377      \stex_debug:nn{copymodule}{result:\meaning \__stex_copymodule_module_tl}
4378      \stex_debug:nn{copymodule}{output:\meaning \__stex_copymodule_exec_tl}
4379
4380      \__stex_copymodule_exec_tl
4381      \stex_if_do_html:T {
4382        \end{stex_annotate_env}
4383      }
4384  }
4385
4386  \NewDocumentEnvironment {copymodule} { O{} m m}{
4387      \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ copymodule }
4388      \stex_deactivate_macro:Nn \symdecl {module~environments}
4389      \stex_deactivate_macro:Nn \symdef {module~environments}
4390      \stex_deactivate_macro:Nn \notation {module~environments}
4391      \stex_reactivate_macro:N \assign
4392      \stex_reactivate_macro:N \renamedecl
4393      \stex_reactivate_macro:N \donotcopy
4394      \stex_smsmode_do:
4395  }{
4396      \stex_copymodule_end:n {}
4397  }
4398
4399  \NewDocumentEnvironment {interpretmodule} { O{} m m}{
4400      \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ interpretmodule }
4401      \stex_deactivate_macro:Nn \symdecl {module~environments}
4402      \stex_deactivate_macro:Nn \symdef {module~environments}
4403      \stex_deactivate_macro:Nn \notation {module~environments}
4404      \stex_reactivate_macro:N \assign
4405      \stex_reactivate_macro:N \renamedecl
4406      \stex_reactivate_macro:N \donotcopy
4407      \stex_smsmode_do:
4408  }{
4409      \stex_copymodule_end:n {
4410        \tl_if_exist:cF {
4411          l__stex_copymodule_copymodule_##1?##2_def_tl
4412        }{
4413          \str_if_eq:eeF {
4414            \prop_item:cn{
4415              l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
4416          }{ true }{
4417            \msg_error:nnxx{stex}{error/interpretmodule/nodefiniens}{
4418              ##1?##2
4419            }{\l_stex_current_copymodule_name_str}
4420          }
4421        }
4422      }
```

190

```
4423 }
4424
4425 \iffalse \begin{stex_annotate_env} \fi
4426 \NewDocumentEnvironment {realization} { O{} m}{
4427   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #2 }{ realize }
4428   \stex_deactivate_macro:Nn \symdecl {module~environments}
4429   \stex_deactivate_macro:Nn \symdef {module~environments}
4430   \stex_deactivate_macro:Nn \notation {module~environments}
4431   \stex_reactivate_macro:N \donotcopy
4432   \stex_reactivate_macro:N \assign
4433   \stex_smsmode_do:
4434 }{
4435   \stex_import_module_uri:nn { #1 } { #2 }
4436   \tl_clear:N \__stex_copymodule_exec_tl
4437   \tl_set:Nx \__stex_copymodule_module_tl {
4438     \stex_import_require_module:nnnn
4439       { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4440       { \l_stex_import_path_str } { \l_stex_import_name_str }
4441   }
4442
4443   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4444     \seq_map_inline:cn {c_stex_module_##1_constants}{
4445       \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str / #
4446       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
4447         \stex_if_do_html:T {
4448           \tl_put_right:Nx \__stex_copymodule_exec_tl {
4449             \stex_annotate_invisible:nnn{assignment} {##1?####1} {
4450               $\stex_annotate_invisible:nnn{definiens}{}{\exp_after:wN \exp_not:N\csname l__
4451             }
4452           }
4453         }
4454         \tl_put_right:Nx \__stex_copymodule_module_tl {
4455           \prop_put:cnn {l_stex_symdecl_##1?####1_prop}{ defined }{ true }
4456         }
4457       }
4458   }}
4459
4460   \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4461
4462   \__stex_copymodule_exec_tl
4463   \stex_if_do_html:T {\end{stex_annotate_env}}
4464 }
4465
4466 \NewDocumentCommand \donotcopy { m }{
4467   \str_clear:N \l_stex_import_name_str
4468   \str_set:Nn \l_tmpa_str { #1 }
4469   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4470   \seq_map_inline:Nn \l_stex_all_modules_seq {
4471     \str_set:Nn \l_tmpb_str { ##1 }
4472     \str_if_eq:eeT { \l_tmpa_str } {
4473       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4474     } {
4475       \seq_map_break:n {
4476         \stex_if_do_html:T {
```

191

```
4477              \stex_if_smsmode:F {
4478                \stex_annotate_invisible:nnn{donotcopy}{##1}{
4479                  \stex_annotate:nnn{domain}{##1}{}
4480                }
4481              }
4482            }
4483            \str_set_eq:NN \l_stex_import_name_str \l_tmpb_str
4484          }
4485        }
4486        \seq_map_inline:cn {c_stex_module_##1_copymodules}{
4487          \str_set:Nn \l_tmpb_str { ####1 }
4488          \str_if_eq:eeT { \l_tmpa_str } {
4489            \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4490          } {
4491            \seq_map_break:n {\seq_map_break:n {
4492              \stex_if_do_html:T {
4493                \stex_if_smsmode:F {
4494                  \stex_annotate_invisible:nnn{donotcopy}{####1}{
4495                    \stex_annotate:nnn{domain}{
4496                      \prop_item:cn {l_stex_copymodule_ ####1 _prop}{module}
4497                    }{}
4498                  }
4499                }
4500              }
4501              \str_set:Nx \l_stex_import_name_str {
4502                \prop_item:cn {l_stex_copymodule_ ####1 _prop}{module}
4503              }
4504            }}
4505          }
4506        }
4507      }
4508      \str_if_empty:NTF \l_stex_import_name_str {
4509        % TODO throw error
4510      }{
4511        \stex_collect_imports:n {\l_stex_import_name_str }
4512        \seq_map_inline:Nn \l_stex_collect_imports_seq {
4513          \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
4514          \seq_map_inline:cn {c_stex_module_##1_constants}{
4515            \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? ####1 }
4516            \bool_lazy_any:nT {
4517              { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_name_str}}
4518              { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_macroname_str}}
4519              { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_def_tl}}
4520            }{
4521              % TODO throw error
4522            }
4523          }
4524        }
4525        \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
4526        \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_name_str }
4527        \prop_put:Nno \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
4528      }
4529      \stex_smsmode_do:
4530    }
```

```
4531
4532 \NewDocumentCommand \assign { m m }{
4533   \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
4534   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
4535   \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _def_tl}{#2}
4536   \stex_smsmode_do:
4537 }
4538
4539 \keys_define:nn { stex / renamedecl } {
4540   name        .str_set_x:N  = \l_stex_renamedecl_name_str
4541 }
4542 \cs_new_protected:Nn \__stex_copymodule_renamedecl_args:n {
4543   \str_clear:N \l_stex_renamedecl_name_str
4544   \keys_set:nn { stex / renamedecl } { #1 }
4545 }
4546
4547 \NewDocumentCommand \renamedecl { O{} m m}{
4548   \__stex_copymodule_renamedecl_args:n { #1 }
4549   \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
4550   \stex_debug:nn{renamedecl}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
4551   \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
4552   \str_if_empty:NTF \l_stex_renamedecl_name_str {
4553     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4554       \l_stex_get_symbol_uri_str
4555     } }
4556   } {
4557     \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex
4558     \stex_debug:nn{renamedecl}{@~\l_stex_current_module_str ? \l_stex_renamedecl_name_str}
4559     \prop_set_eq:cc {l_stex_symdecl_
4560       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4561       _prop
4562     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
4563     \seq_set_eq:cc {l_stex_symdecl_
4564       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4565       _notations
4566     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
4567     \prop_put:cnx {l_stex_symdecl_
4568       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4569       _prop
4570     }{ name }{ \l_stex_renamedecl_name_str }
4571     \prop_put:cnx {l_stex_symdecl_
4572       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4573       _prop
4574     }{ module }{ \l_stex_current_module_str }
4575     \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
4576       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4577     }
4578     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4579       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4580     } }
4581   }
4582   \stex_smsmode_do:
4583 }
4584
```

```
4585 \stex_deactivate_macro:Nn \assign {copymodules}
4586 \stex_deactivate_macro:Nn \renamedecl {copymodules}
4587 \stex_deactivate_macro:Nn \donotcopy {copymodules}
4588
4589
```

31.2   The feature environment

```
4590 ⟨@@=stex_features⟩
4591
4592 \NewDocumentEnvironment{structural_feature_module}{ m m m }{
4593   \stex_if_in_module:F {
4594     \msg_set:nnn{stex}{error/nomodule}{
4595       Structural~Feature~has~to~occur~in~a~module:\\
4596       Feature~#2~of~type~#1\\
4597       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
4598     }
4599     \msg_error:nn{stex}{error/nomodule}
4600   }
4601
4602   \str_set_eq:NN \l_stex_feature_parent_str \l_stex_current_module_str
4603
4604   \stex_module_setup:nn{meta=NONE}{#2 - #1}
4605
4606   \stex_if_do_html:T {
4607     \begin{stex_annotate_env}{ feature:#1 }{\l_stex_feature_parent_str ? #2 - #1}
4608       \stex_annotate_invisible:nnn{header}{}{ #3 }
4609   }
4610 }{
4611   \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
4612   \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
4613   \stex_debug:nn{features}{
4614     Feature: \l_stex_last_feature_str
4615   }
4616   \stex_if_do_html:T {
4617     \end{stex_annotate_env}
4618   }
4619 }
```

31.3   Structure

```
4620 ⟨@@=stex_structures⟩
4621 \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
4622   \prop_if_exist:cF {c_stex_module_\l_stex_current_module_str _structures}{
4623     \prop_new:c {c_stex_module_\l_stex_current_module_str _structures}
4624   }
4625   \prop_gput:cxx{c_stex_module_\l_stex_current_module_str _structures}
4626     {#1}{#2}
4627 }
4628
```

194

```
4629 \keys_define:nn { stex / features / structure } {
4630   name          .str_set_x:N  = \l__stex_structures_name_str ,
4631 }
4632
4633 \cs_new_protected:Nn \__stex_structures_structure_args:n {
4634   \str_clear:N \l__stex_structures_name_str
4635   \keys_set:nn { stex / features / structure } { #1 }
4636 }
4637
4638 \NewDocumentEnvironment{mathstructure}{m O{}}{
4639   \__stex_structures_structure_args:n { #2 }
4640   \str_if_empty:NT \l__stex_structures_name_str {
4641     \str_set:Nx \l__stex_structures_name_str { #1 }
4642   }
4643   \stex_suppress_html:n {
4644     \bool_set_true:N \l_stex_symdecl_make_macro_bool
4645     \exp_args:Nx \stex_symdecl_do:nn {
4646       name = \l__stex_structures_name_str ,
4647       def  = {\STEXsymbol{module-type}{
4648         \STEXInternalTermMathOMSiiii {
4649           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4650             { ns } ?
4651             \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4652               { name } / \l__stex_structures_name_str - structure
4653         }{}{0}{}
4654       }}
4655     }{ #1 }
4656   }
4657   \exp_args:Nnnx
4658   \begin{structural_feature_module}{ structure }
4659     { \l__stex_structures_name_str }{}
4660   \stex_smsmode_do:
4661 }{
4662   \end{structural_feature_module}
4663   \_stex_reset_up_to_module:n \l_stex_last_feature_str
4664   \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
4665   \seq_clear:N \l_tmpa_seq
4666   \seq_map_inline:Nn \l_stex_collect_imports_seq {
4667     \seq_map_inline:cn{c_stex_module_##1_constants}{
4668       \seq_put_right:Nn \l_tmpa_seq { ##1 ? ####1 }
4669     }
4670   }
4671   \exp_args:Nnno
4672   \prop_gput:cnn {c_stex_module_ \l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
4673   \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
4674   \stex_add_structure_to_current_module:nn
4675     \l__stex_structures_name_str
4676     \l_stex_last_feature_str
4677
4678   \stex_execute_in_module:x {
4679     \tl_set:cn { #1 }{
4680       \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structure
4681     }
4682   }
```

```
4683 }
4684
4685 \cs_new:Nn \stex_invoke_structure:nn {
4686   \stex_invoke_symbol:n { #1?#2 }
4687 }
4688
4689 \cs_new_protected:Nn \stex_get_structure:n {
4690   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4691     \tl_set:Nn \l_tmpa_tl { #1 }
4692     \__stex_structures_get_from_cs:
4693   }{
4694     \cs_if_exist:cTF { #1 }{
4695       \cs_set_eq:Nc \l_tmpa_cs { #1 }
4696       \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
4697       \str_if_empty:NTF \l_tmpa_str {
4698         \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs} \stex_invoke_structure:nn {
4699           \__stex_structures_get_from_cs:
4700         }{
4701           \__stex_structures_get_from_string:n { #1 }
4702         }
4703       }{
4704         \__stex_structures_get_from_string:n { #1 }
4705       }
4706     }{
4707       \__stex_structures_get_from_string:n { #1 }
4708     }
4709   }
4710 }
4711
4712 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
4713   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4714     { \tl_tail:N \l_tmpa_tl }
4715   \str_set:Nx \l_tmpa_str {
4716     \exp_after:wN \use_i:nn \l_tmpa_tl
4717   }
4718   \str_set:Nx \l_tmpb_str {
4719     \exp_after:wN \use_ii:nn \l_tmpa_tl
4720   }
4721   \str_set:Nx \l_stex_get_structure_str {
4722     \l_tmpa_str ? \l_tmpb_str
4723   }
4724   \str_set:Nx \l_stex_get_structure_module_str {
4725     \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
4726   }
4727 }
4728
4729 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
4730   \tl_set:Nn \l_tmpa_tl {
4731     \msg_error:nnn{stex}{error/unknownstructure}{#1}
4732   }
4733   \str_set:Nn \l_tmpa_str { #1 }
4734   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4735
4736   \seq_map_inline:Nn \l_stex_all_modules_seq {
```

```
4737        \prop_if_exist:cT {c_stex_module_##1_structures} {
4738          \prop_map_inline:cn {c_stex_module_##1_structures} {
4739            \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
4740              \prop_map_break:n{\seq_map_break:n{
4741                \tl_set:Nn \l_tmpa_tl {
4742                  \str_set:Nn \l_stex_get_structure_str {##1?####1}
4743                  \str_set:Nn \l_stex_get_structure_module_str {####2}
4744                }
4745            }}
4746          }
4747        }
4748      }
4749    }
4750    \l_tmpa_tl
4751  }
```

```
4752
4753  \keys_define:nn { stex / instantiate } {
4754    name        .str_set_x:N  = \l__stex_structures_name_str
4755  }
4756  \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
4757    \str_clear:N \l__stex_structures_name_str
4758    \keys_set:nn { stex / instantiate } { #1 }
4759  }
4760
4761  \NewDocumentCommand \instantiate {m O{} m m O{}}{
4762    \begingroup
4763      \stex_get_structure:n {#3}
4764      \__stex_structures_instantiate_args:n { #2 }
4765      \str_if_empty:NT \l__stex_structures_name_str {
4766        \str_set:Nn \l__stex_structures_name_str { #1 }
4767      }
4768      \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4769      \seq_clear:N \l__stex_structures_fields_seq
4770      \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4771      \seq_map_inline:Nn \l_stex_collect_imports_seq {
4772        \seq_map_inline:cn {c_stex_module_##1_constants}{
4773          \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4774        }
4775      }
4776
4777      \tl_if_empty:nF{#5}{
4778        \seq_set_split:Nnn \l_tmpa_seq , {#5}
4779        \prop_clear:N \l_tmpa_prop
4780        \seq_map_inline:Nn \l_tmpa_seq {
4781          \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4782          \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4783            \msg_error:nnn{stex}{error/keyval}{##1}
4784          }
4785          \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
4786          \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4787          \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
4788          \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}
```

```
4789          \exp_args:Nxx \str_if_eq:nnF
4790            {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4791            {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
4792            \msg_error:nnxxxx{stex}{error/incompatible}
4793              {\l__stex_structures_dom_str}
4794              {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4795              {\l_stex_get_symbol_uri_str}
4796              {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
4797          }
4798          \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str
4799        }
4800      }
4801
4802      \seq_map_inline:Nn \l__stex_structures_fields_seq {
4803        \str_set:Nx \l_tmpa_str {field:\l__stex_structures_name_str . \prop_item:cn {l_stex_sy
4804        \stex_debug:nn{instantiate}{Field~\l_tmpa_str :~##1}
4805
4806        \stex_add_constant_to_current_module:n {\l_tmpa_str}
4807        \stex_execute_in_module:x {
4808          \prop_set_from_keyval:cn { l_stex_symdecl_ \l_stex_current_module_str?\l_tmpa_str _p
4809            name  = \l_tmpa_str ,
4810            args  = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4811            arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4812            assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4813          }
4814          \seq_clear:c {l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _notations}
4815        }
4816
4817        \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4818          \stex_find_notation:nn{##1}{}
4819          \stex_execute_in_module:x {
4820            \seq_put_right:cn {l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _notation
4821          }
4822
4823          \stex_copy_control_sequence_ii:ccN
4824            {stex_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_notation_
4825            {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4826            \l_tmpa_tl
4827          \exp_args:No \stex_execute_in_module:n \l_tmpa_tl
4828
4829
4830          \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
4831            \tl_set_eq:Nc \l_tmpa_cs {stex_op_notation_##1\c_hash_str \l_stex_notation_variant
4832            \stex_execute_in_module:x {
4833              \tl_set:cn
4834              {stex_op_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_nota
4835              { \exp_args:No \exp_not:n \l_tmpa_cs}
4836            }
4837          }
4838
4839        }
4840
4841        \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\l_stex_cur
4842      }
```

198

```
4843
4844      \stex_execute_in_module:x {
4845        \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
4846          domain = \l_stex_get_structure_module_str ,
4847          \prop_to_keyval:N \l_tmpa_prop
4848        }
4849        \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
4850      }
4851      \stex_debug:nn{instantiate}{
4852        Instance~\l_stex_current_module_str?\l__stex_structures_name_str \\
4853        \prop_to_keyval:N \l_tmpa_prop
4854      }
4855      \exp_args:Nxx \stex_symdecl_do:nn {
4856        type={\STEXsymbol{module-type}{
4857          \STEXInternalTermMathOMSiiii {
4858            \l_stex_get_structure_module_str
4859          }{}{0}{}
4860        }}
4861      }{\l__stex_structures_name_str}
4862 %     {
4863        \str_set:Nx \l_stex_get_symbol_uri_str {\l_stex_current_module_str?\l__stex_structures
4864        \tl_set:Nn \l_stex_notation_after_do_tl {\__stex_notation_final:}
4865        \stex_notation_do:nnnnn{}{0}{}{}{\comp{#4}}
4866 %   }
4867      %\exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}
4868    \endgroup
4869    \stex_smsmode_do:\ignorespacesandpars
4870 }
4871
4872 \cs_new_protected:Nn \stex_symbol_or_var:n {
4873    \cs_if_exist:cTF{#1}{
4874      \cs_set_eq:Nc \l_tmpa_tl { # 1 }
4875      \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4876      \str_if_empty:NTF \l_tmpa_str {
4877        \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
4878          \stex_invoke_variable:n {
4879            \bool_set_true:N \l_stex_symbol_or_var_bool
4880            \bool_set_false:N \l_stex_instance_or_symbol_bool
4881            \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
4882            \tl_set:Nx \l_tmpa_tl {\exp_after:wN \use:n \l_tmpa_tl}
4883            \str_set:Nx \l_stex_get_symbol_uri_str {
4884              \exp_after:wN \use:n \l_tmpa_tl
4885            }
4886          }{ % TODO \stex_invoke_varinstance:n
4887            \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl } \stex_invoke_varinstance:n {
4888              \bool_set_true:N \l_stex_symbol_or_var_bool
4889              \bool_set_true:N \l_stex_instance_or_symbol_bool
4890              \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
4891              \tl_set:Nx \l_tmpa_tl {\exp_after:wN \use:n \l_tmpa_tl}
4892              \str_set:Nx \l_stex_get_symbol_uri_str {
4893                \exp_after:wN \use:n \l_tmpa_tl
4894              }
4895            }{
4896              \bool_set_false:N \l_stex_symbol_or_var_bool
```

199

```
4897              \stex_get_symbol:n{#1}
4898            }
4899          }
4900      }{
4901        \__stex_structures_symbolorvar_from_string:n{ #1 }
4902      }
4903    }{
4904      \__stex_structures_symbolorvar_from_string:n{ #1 }
4905    }
4906  }
4907
4908  \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
4909    \prop_if_exist:cTF {l_stex_variable_#1 _prop}{
4910      \bool_set_true:N \l_stex_symbol_or_var_bool
4911      \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
4912    }{
4913      \bool_set_false:N \l_stex_symbol_or_var_bool
4914      \stex_get_symbol:n{#1}
4915    }
4916  }
4917
4918  \keys_define:nn { stex / varinstantiate } {
4919    name         .str_set_x:N  = \l__stex_structures_name_str,
4920    bind         .choices:nn   =
4921        {forall,exists}
4922        {\str_set:Nx \l__stex_structures_bind_str {\l_keys_choice_tl}}
4923
4924  }
4925  \cs_new_protected:Nn \__stex_structures_varinstantiate_args:n {
4926    \str_clear:N \l__stex_structures_name_str
4927    \str_clear:N \l__stex_structures_bind_str
4928    \keys_set:nn { stex / varinstantiate } { #1 }
4929  }
4930
4931  \NewDocumentCommand \varinstantiate {m O{} m m O{}}{
4932    \begingroup
4933      \stex_get_structure:n {#3}
4934      \__stex_structures_varinstantiate_args:n { #2 }
4935      \str_if_empty:NT \l__stex_structures_name_str {
4936        \str_set:Nn \l__stex_structures_name_str { #1 }
4937      }
4938      \stex_if_do_html:TF{
4939        \stex_annotate:nnn{varinstance}{\l__stex_structures_name_str}
4940      }{\use:n}
4941      {
4942        \stex_if_do_html:T{
4943          \stex_annotate_invisible:nnn{domain}{\l_stex_get_structure_module_str}{}
4944        }
4945        \seq_clear:N \l__stex_structures_fields_seq
4946        \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4947        \seq_map_inline:Nn \l_stex_collect_imports_seq {
4948          \seq_map_inline:cn {c_stex_module_##1_constants}{
4949            \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4950          }
```

```
4951          }
4952        \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4953        \prop_clear:N \l_tmpa_prop
4954        \tl_if_empty:nF {#5} {
4955          \seq_set_split:Nnn \l_tmpa_seq , {#5}
4956          \seq_map_inline:Nn \l_tmpa_seq {
4957            \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4958            \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4959              \msg_error:nnn{stex}{error/keyval}{##1}
4960            }
4961            \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_stru
4962            \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4963            \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol
4964            \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
4965            \stex_if_do_html:T{
4966              \stex_annotate:nnn{assign}{\l__stex_structures_dom_str,
4967              \bool_if:NTF\l_stex_symbol_or_var_bool{var://}{}\l_stex_get_symbol_uri_str}{}
4968            }
4969            \bool_if:NTF \l_stex_symbol_or_var_bool {
4970              \exp_args:Nxx \str_if_eq:nnF
4971                {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4972                {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}{
4973                \msg_error:nnxxxx{stex}{error/incompatible}
4974                  {\l__stex_structures_dom_str}
4975                  {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4976                  {\l_stex_get_symbol_uri_str}
4977                  {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}
4978              }
4979              \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n
4980            }{
4981              \exp_args:Nxx \str_if_eq:nnF
4982                {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4983                {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
4984                \msg_error:nnxxxx{stex}{error/incompatible}
4985                  {\l__stex_structures_dom_str}
4986                  {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4987                  {\l_stex_get_symbol_uri_str}
4988                  {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
4989              }
4990              \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n {
4991            }
4992          }
4993        }
4994        \tl_gclear:N \g__stex_structures_aftergroup_tl
4995        \seq_map_inline:Nn \l__stex_structures_fields_seq {
4996          \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdec
4997          \stex_debug:nn{varinstantiate}{Field~\l_tmpa_str :~##1}
4998          \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4999            \stex_find_notation:nn{##1}{}
5000            \cs_gset_eq:cc{g__stex_structures_tmpa_\l_tmpa_str _cs}
5001              {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5002            \stex_debug:nn{varinstantiate}{Notation:~\cs_meaning:c{g__stex_structures_tmpa_\l_
5003            \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
5004              \cs_gset_eq:cc {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
```

201

```
5005                {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5006                \stex_debug:nn{varinstantiate}{Operator~Notation:~\cs_meaning:c{g__stex_struct
5007              }
5008            }
5009
5010          \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
5011            \prop_set_from_keyval:cn { l_stex_variable_ \l_tmpa_str _prop}{
5012              name  = \l_tmpa_str ,
5013              args  = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
5014              arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
5015              assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
5016            }
5017            \cs_set_eq:cc {stex_var_notation_\l_tmpa_str _cs}
5018              {g__stex_structures_tmpa_\l_tmpa_str _cs}
5019            \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str _cs}
5020              {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
5021          }
5022          \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\stex_inv
5023        }
5024      \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
5025        \prop_set_from_keyval:cn {l_stex_varinstance_\l__stex_structures_name_str _prop }{
5026          domain = \l_stex_get_structure_module_str ,
5027          \prop_to_keyval:N \l_tmpa_prop
5028        }
5029        \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
5030        \tl_set:cn {l_stex_varinstance_\l__stex_structures_name_str _op_tl}{
5031          \exp_args:Nnx \exp_not:N \use:nn {
5032            \str_set:Nn \exp_not:N \STEXInternalCurrentSymbolStr {var://\l__stex_structures_
5033            \_stex_term_omv:nn {var://\l__stex_structures_name_str}{
5034              \exp_not:n{
5035                \_varcomp{#4}
5036              }
5037            }
5038          }{
5039            \exp_not:n{\_stex_reset:N \STEXInternalCurrentSymbolStr}
5040          }
5041        }
5042        }
5043      }
5044    \stex_debug:nn{varinstantiate}{\expandafter\detokenize\expandafter{\g__stex_structures_a
5045    \aftergroup\g__stex_structures_aftergroup_tl
5046  \endgroup
5047  \stex_smsmode_do:\ignorespacesandpars
5048 }
5049
5050 \cs_new_protected:Nn \stex_invoke_instance:n {
5051   \peek_charcode_remove:NTF ! {
5052     \stex_invoke_symbol:n{#1}
5053   }{
5054     \_stex_invoke_instance:nn {#1}
5055   }
5056 }
5057
5058
```

```
5059 \cs_new_protected:Nn \stex_invoke_varinstance:n {
5060   \peek_charcode_remove:NTF ! {
5061     \exp_args:Nnx \use:nn {
5062       \def\comp{\_varcomp}
5063       \use:c{l_stex_varinstance_#1_op_tl}
5064     }{
5065       \_stex_reset:N \comp
5066     }
5067   }{
5068     \_stex_invoke_varinstance:nn {#1}
5069   }
5070 }
5071
5072 \cs_new_protected:Nn \_stex_invoke_instance:nn {
5073   \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
5074     \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
5075   }{
5076     \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
5077     \msg_error:nnxxxx{stex}{error/unknownfield}{#2}{#1}{
5078       \prop_to_keyval:N \l_tmpa_prop
5079     }
5080   }
5081 }
5082
5083 \cs_new_protected:Nn \_stex_invoke_varinstance:nn {
5084   \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
5085     \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
5086     \l_tmpa_tl
5087   }{
5088     \msg_error:nnnnn{stex}{error/unknownfield}{#2}{#1}{}
5089   }
5090 }
```

(*End definition for* \instantiate. *This function is documented on page* *33.*)

\stex_invoke_structure:nnn

```
5091 % #1: URI of the instance
5092 % #2: URI of the instantiated module
5093 \cs_new_protected:Nn \stex_invoke_structure:nnn {
5094   \tl_if_empty:nTF{ #3 }{
5095     \prop_set_eq:Nc \l__stex_structures_structure_prop {
5096       c_stex_feature_ #2 _prop
5097     }
5098     \tl_clear:N \l_tmpa_tl
5099     \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
5100     \seq_map_inline:Nn \l_tmpa_seq {
5101       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
5102       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
5103       \cs_if_exist:cT {
5104         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
5105       }{
5106         \tl_if_empty:NF \l_tmpa_tl {
5107           \tl_put_right:Nn \l_tmpa_tl {,}
5108         }
```

```
5109        \tl_put_right:Nx \l_tmpa_tl {
5110          \stex_invoke_symbol:n {#1/\l_tmpa_str}!
5111        }
5112      }
5113    }
5114    \exp_args:No \mathstruct \l_tmpa_tl
5115  }{
5116    \stex_invoke_symbol:n{#1/#3}
5117  }
5118 }
```

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

```
5119 ⟨/package⟩
```

Chapter 32

STEX
-Statements Implementation

5120 ⟨∗package⟩
5121
5122 %%%%%%%%%%%%   features.dtx   %%%%%%%%%%%%
5123
5124 ⟨@@=stex_statements⟩

Warnings and error messages

5125

\titleemph

5126 \def\titleemph#1{\textbf{#1}}

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

32.1   Definitions

definiendum

5127 \keys_define:nn {stex / definiendum }{
5128   pre   .tl_set:N   = \l__stex_statements_definiendum_pre_tl,
5129   post   .tl_set:N   = \l__stex_statements_definiendum_post_tl,
5130   root   .str_set_x:N = \l__stex_statements_definiendum_root_str,
5131   gfa   .str_set_x:N = \l__stex_statements_definiendum_gfa_str
5132 }
5133 \cs_new_protected:Nn __stex_statements_definiendum_args:n {
5134   \str_clear:N \l__stex_statements_definiendum_root_str
5135   \tl_clear:N \l__stex_statements_definiendum_post_tl
5136   \str_clear:N \l__stex_statements_definiendum_gfa_str
5137   \keys_set:nn { stex / definiendum }{ #1 }
5138 }
5139 \NewDocumentCommand \definiendum { O{} m m} {
5140   __stex_statements_definiendum_args:n { #1 }
5141   \stex_get_symbol:n { #2 }
5142   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5143   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
5144   \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {

```
5145          \tl_set:Nn \l_tmpa_tl { #3 }
5146        } {
5147          \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
5148          \tl_set:Nn \l_tmpa_tl {
5149            \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
5150          }
5151        }
5152      } {
5153        \tl_set:Nn \l_tmpa_tl { #3 }
5154      }
5155
5156      % TODO root
5157      \stex_html_backend:TF {
5158        \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
5159      } {
5160        \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
5161      }
5162    }
5163    \stex_deactivate_macro:Nn \definiendum {definition~environments}
```

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

definame

```
5164
5165    \NewDocumentCommand \definame { O{} m } {
5166      \__stex_statements_definiendum_args:n { #1 }
5167      % TODO: root
5168      \stex_get_symbol:n { #2 }
5169      \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5170      \str_set:Nx \l_tmpa_str {
5171        \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5172      }
5173      \str_replace_all:Nnn \l_tmpa_str {-} {~}
5174      \stex_html_backend:TF {
5175        \stex_if_do_html:T {
5176          \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5177            \l_tmpa_str\l__stex_statements_definiendum_post_tl
5178          }
5179        }
5180      } {
5181        \exp_args:Nnx \defemph@uri {
5182          \l_tmpa_str\l__stex_statements_definiendum_post_tl
5183        } { \l_stex_get_symbol_uri_str }
5184      }
5185    }
5186    \stex_deactivate_macro:Nn \definame {definition~environments}
5187
5188    \NewDocumentCommand \Definame { O{} m } {
5189      \__stex_statements_definiendum_args:n { #1 }
5190      \stex_get_symbol:n { #2 }
5191      \str_set:Nx \l_tmpa_str {
5192        \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5193      }
5194      \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
```

```
5195    \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5196    \stex_html_backend:TF {
5197      \stex_if_do_html:T {
5198        \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5199          \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5200        }
5201      }
5202    } {
5203      \exp_args:Nnx \defemph@uri {
5204        \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5205      } { \l_stex_get_symbol_uri_str }
5206    }
5207 }
5208 \stex_deactivate_macro:Nn \Definame {definition~environments}
5209
5210 \NewDocumentCommand \premise { m }{
5211    \noindent\stex_annotate:nnn{ premise }{}{\ignorespaces #1 }
5212 }
5213 \NewDocumentCommand \conclusion { m }{
5214    \noindent\stex_annotate:nnn{ conclusion }{}{\ignorespaces #1 }
5215 }
5216 \NewDocumentCommand \definiens { O{} m }{
5217    \str_clear:N \l_stex_get_symbol_uri_str
5218    \tl_if_empty:nF {#1} {
5219      \stex_get_symbol:n { #1 }
5220    }
5221    \str_if_empty:NT \l_stex_get_symbol_uri_str {
5222      \int_compare:nNnTF {\clist_count:N \l__stex_statements_sdefinition_for_clist} = 1 {
5223        \str_set:Nx \l_stex_get_symbol_uri_str {\clist_item:Nn \l__stex_statements_sdefinition
5224      }{
5225        % TODO throw error
5226      }
5227    }
5228    \str_if_eq:eeT {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{module}}
5229      {\l_stex_current_module_str}{
5230        \str_if_eq:eeF {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defin
5231        {true}{
5232          \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5233          \exp_args:Nx \stex_add_to_current_module:n {
5234            \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5235          }
5236        }
5237    }
5238    \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
5239 }
5240
5241 \NewDocumentCommand \varbindforall {m}{
5242    \stex_symbol_or_var:n {#1}
5243    \bool_if:NTF\l_stex_symbol_or_var_bool{
5244      \stex_if_do_html:T {
5245        \stex_annotate_invisible:nnn {bindtype}{forall,\l_stex_get_symbol_uri_str}{}
5246      }
5247    }{
5248      % todo throw error
```

```
5249      }
5250 }
5251
5252 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
5253 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
5254 \stex_deactivate_macro:Nn \definiens {definition~environments}
5255 \stex_deactivate_macro:Nn \varbindforall {definition~or~assertion~environments}
5256
```

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

sdefinition

```
5257
5258 \keys_define:nn {stex / sdefinition }{
5259   type    .str_set_x:N  = \sdefinitiontype,
5260   id      .str_set_x:N  = \sdefinitionid,
5261   name    .str_set_x:N  = \sdefinitionname,
5262   for     .clist_set:N   = \l__stex_statements_sdefinition_for_clist ,
5263   title   .tl_set:N      = \sdefinitiontitle
5264 }
5265 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
5266   \str_clear:N \sdefinitiontype
5267   \str_clear:N \sdefinitionid
5268   \str_clear:N \sdefinitionname
5269   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5270   \tl_clear:N \sdefinitiontitle
5271   \keys_set:nn { stex / sdefinition }{ #1 }
5272 }
5273
5274 \NewDocumentEnvironment{sdefinition}{O{}}{
5275   \__stex_statements_sdefinition_args:n{ #1 }
5276   \stex_reactivate_macro:N \definiendum
5277   \stex_reactivate_macro:N \definame
5278   \stex_reactivate_macro:N \Definame
5279   \stex_reactivate_macro:N \premise
5280   \stex_reactivate_macro:N \definiens
5281   \stex_reactivate_macro:N \varbindforall
5282   \stex_if_smsmode:F{
5283     \seq_clear:N \l_tmpb_seq
5284     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5285       \tl_if_empty:nF{ ##1 }{
5286         \stex_get_symbol:n { ##1 }
5287         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5288           \l_stex_get_symbol_uri_str
5289         }
5290       }
5291     }
5292     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5293     \exp_args:Nnnx
5294     \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpb_seq {,}}
5295     \str_if_empty:NF \sdefinitiontype {
5296       \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5297     }
5298     \str_if_empty:NF \sdefinitionname {
```

208

```
5299        \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5300      }
5301      \clist_set:No \l_tmpa_clist \sdefinitiontype
5302      \tl_clear:N \l_tmpa_tl
5303      \clist_map_inline:Nn \l_tmpa_clist {
5304        \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
5305          \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
5306        }
5307      }
5308      \tl_if_empty:NTF \l_tmpa_tl {
5309        \__stex_statements_sdefinition_start:
5310      }{
5311        \l_tmpa_tl
5312      }
5313    }
5314    \stex_ref_new_doc_target:n \sdefinitionid
5315    \stex_smsmode_do:
5316  }{
5317    \stex_suppress_html:n {
5318      \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5319    }
5320    \stex_if_smsmode:F {
5321      \clist_set:No \l_tmpa_clist \sdefinitiontype
5322      \tl_clear:N \l_tmpa_tl
5323      \clist_map_inline:Nn \l_tmpa_clist {
5324        \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
5325          \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
5326        }
5327      }
5328      \tl_if_empty:NTF \l_tmpa_tl {
5329        \__stex_statements_sdefinition_end:
5330      }{
5331        \l_tmpa_tl
5332      }
5333      \end{stex_annotate_env}
5334    }
5335  }
```

```
5336  \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
5337    \stex_par:\noindent\titleemph{Definition\tl_if_empty:NF \sdefinitiontitle {
5338      ~(\sdefinitiontitle)
5339    }~}
5340  }
5341  \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\stex_par:\medskip}
5342
5343  \newcommand\stexpatchdefinition[3][] {
5344      \str_set:Nx \l_tmpa_str{ #1 }
5345      \str_if_empty:NTF \l_tmpa_str {
5346        \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
5347        \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
5348      }{
5349        \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2
5350        \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
```

(End definition for \stexpatchdefinition. *This function is documented on page 48.)*

\inlinedef   inline:

```
5353 \keys_define:nn {stex / inlinedef }{
5354   type     .str_set_x:N  = \sdefinitiontype,
5355   id       .str_set_x:N  = \sdefinitionid,
5356   for      .clist_set:N   = \l__stex_statements_sdefinition_for_clist ,
5357   name     .str_set_x:N  = \sdefinitionname
5358 }
5359 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
5360   \str_clear:N \sdefinitiontype
5361   \str_clear:N \sdefinitionid
5362   \str_clear:N \sdefinitionname
5363   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5364   \keys_set:nn { stex / inlinedef }{ #1 }
5365 }
5366 \NewDocumentCommand \inlinedef { O{} m } {
5367   \begingroup
5368   \__stex_statements_inlinedef_args:n{ #1 }
5369   \stex_reactivate_macro:N \definiendum
5370   \stex_reactivate_macro:N \definame
5371   \stex_reactivate_macro:N \Definame
5372   \stex_reactivate_macro:N \premise
5373   \stex_reactivate_macro:N \definiens
5374   \stex_reactivate_macro:N \varbindforall
5375   \stex_ref_new_doc_target:n \sdefinitionid
5376   \stex_if_smsmode:TF{\stex_suppress_html:n {
5377     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5378   }}{
5379     \seq_clear:N \l_tmpb_seq
5380     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5381       \tl_if_empty:nF{ ##1 }{
5382         \stex_get_symbol:n { ##1 }
5383         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5384           \l_stex_get_symbol_uri_str
5385         }
5386       }
5387     }
5388     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5389     \exp_args:Nnx
5390     \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpb_seq {,}}{
5391       \str_if_empty:NF \sdefinitiontype {
5392         \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5393       }
5394       #2
5395       \str_if_empty:NF \sdefinitionname {
5396         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sdefinitionname}}
5397         \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5398       }
5399     }
5400   }
```

```
5401     \endgroup
5402     \stex_smsmode_do:
5403 }
```

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

32.2   Assertions

```
5404
5405 \keys_define:nn {stex / sassertion }{
5406   type    .str_set_x:N  = \sassertiontype,
5407   id      .str_set_x:N  = \sassertionid,
5408   title   .tl_set:N     = \sassertiontitle ,
5409   for     .clist_set:N  = \l__stex_statements_sassertion_for_clist ,
5410   name    .str_set_x:N  = \sassertionname
5411 }
5412 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
5413   \str_clear:N \sassertiontype
5414   \str_clear:N \sassertionid
5415   \str_clear:N \sassertionname
5416   \clist_clear:N \l__stex_statements_sassertion_for_clist
5417   \tl_clear:N \sassertiontitle
5418   \keys_set:nn { stex / sassertion }{ #1 }
5419 }
5420
5421 %\tl_new:N \g__stex_statements_aftergroup_tl
5422
5423 \NewDocumentEnvironment{sassertion}{O{}}{
5424   \__stex_statements_sassertion_args:n{ #1 }
5425   \stex_reactivate_macro:N \premise
5426   \stex_reactivate_macro:N \conclusion
5427   \stex_reactivate_macro:N \varbindforall
5428   \stex_if_smsmode:F {
5429     \seq_clear:N \l_tmpb_seq
5430     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5431       \tl_if_empty:nF{ ##1 }{
5432         \stex_get_symbol:n { ##1 }
5433         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5434           \l_stex_get_symbol_uri_str
5435         }
5436       }
5437     }
5438     \exp_args:Nnnx
5439     \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
5440     \str_if_empty:NF \sassertiontype {
5441       \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
5442     }
5443     \str_if_empty:NF \sassertionname {
5444       \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
5445     }
5446     \clist_set:No \l_tmpa_clist \sassertiontype
5447     \tl_clear:N \l_tmpa_tl
```

```
5448        \clist_map_inline:Nn \l_tmpa_clist {
5449          \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
5450            \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}}
5451        }
5452      }
5453      \tl_if_empty:NTF \l_tmpa_tl {
5454        \__stex_statements_sassertion_start:
5455      }{
5456        \l_tmpa_tl
5457      }
5458    }
5459    \str_if_empty:NTF \sassertionid {
5460      \str_if_empty:NF \sassertionname {
5461        \stex_ref_new_doc_target:n {}
5462      }
5463    } {
5464      \stex_ref_new_doc_target:n \sassertionid
5465    }
5466    \stex_smsmode_do:
5467  }{
5468    \str_if_empty:NF \sassertionname {
5469      \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5470      \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5471    }
5472    \stex_if_smsmode:F {
5473      \clist_set:No \l_tmpa_clist \sassertiontype
5474      \tl_clear:N \l_tmpa_tl
5475      \clist_map_inline:Nn \l_tmpa_clist {
5476        \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
5477          \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}}
5478      }
5479    }
5480    \tl_if_empty:NTF \l_tmpa_tl {
5481      \__stex_statements_sassertion_end:
5482    }{
5483      \l_tmpa_tl
5484    }
5485    \end{stex_annotate_env}
5486  }
5487 }
```

\stexpatchassertion

```
5488
5489 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
5490   \stex_par:\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {
5491     (\sassertiontitle)
5492   }~}
5493 }
5494 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\stex_par:\medskip}
5495
5496 \newcommand\stexpatchassertion[3][] {
5497     \str_set:Nx \l_tmpa_str{ #1 }
5498     \str_if_empty:NTF \l_tmpa_str {
5499         \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
```

```
5500        \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
5501      }{
5502        \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
5503        \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
5504      }
5505 }
```

(*End definition for* \stexpatchassertion. *This function is documented on page 48.*)

\inlineass   inline:

```
5506 \keys_define:nn {stex / inlineass }{
5507    type      .str_set_x:N  = \sassertiontype,
5508    id        .str_set_x:N  = \sassertionid,
5509    for       .clist_set:N   = \l__stex_statements_sassertion_for_clist ,
5510    name      .str_set_x:N  = \sassertionname
5511 }
5512 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
5513    \str_clear:N \sassertiontype
5514    \str_clear:N \sassertionid
5515    \str_clear:N \sassertionname
5516    \clist_clear:N \l__stex_statements_sassertion_for_clist
5517    \keys_set:nn { stex / inlineass }{ #1 }
5518 }
5519 \NewDocumentCommand \inlineass { O{} m } {
5520    \begingroup
5521    \stex_reactivate_macro:N \premise
5522    \stex_reactivate_macro:N \conclusion
5523    \stex_reactivate_macro:N \varbindforall
5524    \__stex_statements_inlineass_args:n{ #1 }
5525    \str_if_empty:NTF \sassertionid {
5526      \str_if_empty:NF \sassertionname {
5527        \stex_ref_new_doc_target:n {}
5528      }
5529    } {
5530      \stex_ref_new_doc_target:n \sassertionid
5531    }
5532
5533    \stex_if_smsmode:TF{
5534      \str_if_empty:NF \sassertionname {
5535        \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5536        \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5537      }
5538    }{
5539      \seq_clear:N \l_tmpb_seq
5540      \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5541        \tl_if_empty:nF{ ##1 }{
5542          \stex_get_symbol:n { ##1 }
5543          \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5544            \l_stex_get_symbol_uri_str
5545          }
5546        }
5547      }
5548      \exp_args:Nnx
5549      \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpb_seq {,}}{
```

```
5550        \str_if_empty:NF \sassertiontype {
5551          \stex_annotate_invisible:nnn{typestrings}{\sassertiontype}{}
5552        }
5553        #2
5554        \str_if_empty:NF \sassertionname {
5555          \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5556          \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5557          \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
5558        }
5559      }
5560    }
5561    \endgroup
5562    \stex_smsmode_do:
5563 }
```

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

32.3   Examples

sexample

```
5564
5565 \keys_define:nn {stex / sexample }{
5566   type    .str_set_x:N  = \exampletype,
5567   id      .str_set_x:N  = \sexampleid,
5568   title   .tl_set:N     = \sexampletitle,
5569   name    .str_set_x:N  = \sexamplename ,
5570   for     .clist_set:N  = \l__stex_statements_sexample_for_clist,
5571 }
5572 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
5573   \str_clear:N \exampletype
5574   \str_clear:N \sexampleid
5575   \str_clear:N \sexamplename
5576   \tl_clear:N \sexampletitle
5577   \clist_clear:N \l__stex_statements_sexample_for_clist
5578   \keys_set:nn { stex / sexample }{ #1 }
5579 }
5580
5581 \NewDocumentEnvironment{sexample}{O{}}{
5582   \__stex_statements_sexample_args:n{ #1 }
5583   \stex_reactivate_macro:N \premise
5584   \stex_reactivate_macro:N \conclusion
5585   \stex_if_smsmode:F {
5586     \seq_clear:N \l_tmpb_seq
5587     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5588       \tl_if_empty:nF{ ##1 }{
5589         \stex_get_symbol:n { ##1 }
5590         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5591           \l_stex_get_symbol_uri_str
5592         }
5593       }
5594     }
5595     \exp_args:Nnnx
5596     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpb_seq {,}}
```

```
5597        \str_if_empty:NF \sexampletype {
5598          \stex_annotate_invisible:nnn{typestrings}{\sexampletype}{}
5599        }
5600        \str_if_empty:NF \sexamplename {
5601          \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}
5602        }
5603        \clist_set:No \l_tmpa_clist \sexampletype
5604        \tl_clear:N \l_tmpa_tl
5605        \clist_map_inline:Nn \l_tmpa_clist {
5606          \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
5607            \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
5608          }
5609        }
5610        \tl_if_empty:NTF \l_tmpa_tl {
5611          \__stex_statements_sexample_start:
5612        }{
5613          \l_tmpa_tl
5614        }
5615      }
5616      \str_if_empty:NF \sexampleid {
5617        \stex_ref_new_doc_target:n \sexampleid
5618      }
5619      \stex_smsmode_do:
5620    }{
5621      \str_if_empty:NF \sexamplename {
5622        \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5623      }
5624      \stex_if_smsmode:F {
5625        \clist_set:No \l_tmpa_clist \sexampletype
5626        \tl_clear:N \l_tmpa_tl
5627        \clist_map_inline:Nn \l_tmpa_clist {
5628          \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
5629            \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
5630          }
5631        }
5632        \tl_if_empty:NTF \l_tmpa_tl {
5633          \__stex_statements_sexample_end:
5634        }{
5635          \l_tmpa_tl
5636        }
5637        \end{stex_annotate_env}
5638      }
5639    }
```

\stexpatchexample

```
5640
5641  \cs_new_protected:Nn \__stex_statements_sexample_start: {
5642    \stex_par:\noindent\titleemph{Example~\tl_if_empty:NF \sexampletitle {
5643      (\sexampletitle)
5644    }~}
5645  }
5646  \cs_new_protected:Nn \__stex_statements_sexample_end: {\stex_par:\medskip}
5647
5648  \newcommand\stexpatchexample[3][] {
```

```
5649        \str_set:Nx \l_tmpa_str{ #1 }
5650        \str_if_empty:NTF \l_tmpa_str {
5651          \tl_set:Nn \__stex_statements_sexample_start: { #2 }
5652          \tl_set:Nn \__stex_statements_sexample_end: { #3 }
5653        }{
5654          \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
5655          \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
5656        }
5657  }
```

(*End definition for* \stexpatchexample. *This function is documented on page 48.*)

\inlineex inline:

```
5658  \keys_define:nn {stex / inlineex }{
5659    type     .str_set_x:N  = \sexampletype,
5660    id       .str_set_x:N  = \sexampleid,
5661    for      .clist_set:N   = \l__stex_statements_sexample_for_clist ,
5662    name     .str_set_x:N  = \sexamplename
5663  }
5664  \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
5665    \str_clear:N \sexampletype
5666    \str_clear:N \sexampleid
5667    \str_clear:N \sexamplename
5668    \clist_clear:N \l__stex_statements_sexample_for_clist
5669    \keys_set:nn { stex / inlineex }{ #1 }
5670  }
5671  \NewDocumentCommand \inlineex { O{} m } {
5672    \begingroup
5673    \stex_reactivate_macro:N \premise
5674    \stex_reactivate_macro:N \conclusion
5675    \__stex_statements_inlineex_args:n{ #1 }
5676    \str_if_empty:NF \sexampleid {
5677      \stex_ref_new_doc_target:n \sexampleid
5678    }
5679    \stex_if_smsmode:TF{
5680      \str_if_empty:NF \sexamplename {
5681        \stex_suppress_html:n{\stex_symdecl_do:nn{}{\examplename}}
5682      }
5683    }{
5684      \seq_clear:N \l_tmpb_seq
5685      \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5686        \tl_if_empty:nF{ ##1 }{
5687          \stex_get_symbol:n { ##1 }
5688          \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5689            \l_stex_get_symbol_uri_str
5690          }
5691        }
5692      }
5693      \exp_args:Nnx
5694      \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpb_seq {,}}{
5695        \str_if_empty:NF \sexampletype {
5696          \stex_annotate_invisible:nnn{typestrings}{\sexampletype}{}
5697        }
5698        #2
```

```
5699        \str_if_empty:NF \sexamplename {
5700            \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5701            \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}
5702        }
5703      }
5704    }
5705    \endgroup
5706    \stex_smsmode_do:
5707 }
```

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

32.4 Logical Paragraphs

```
5708 \keys_define:nn { stex / sparagraph} {
5709    id      .str_set_x:N    = \sparagraphid ,
5710    title   .tl_set:N       = \l_stex_sparagraph_title_tl ,
5711    type    .str_set_x:N    = \sparagraphtype ,
5712    for     .clist_set:N    = \l__stex_statements_sparagraph_for_clist ,
5713    from    .tl_set:N       = \sparagraphfrom ,
5714    to      .tl_set:N       = \sparagraphto ,
5715    start   .tl_set:N       = \l_stex_sparagraph_start_tl ,
5716    name    .str_set:N      = \sparagraphname ,
5717    imports .tl_set:N       = \l__stex_statements_sparagraph_imports_tl
5718 }
5719
5720 \cs_new_protected:Nn \stex_sparagraph_args:n {
5721    \tl_clear:N \l_stex_sparagraph_title_tl
5722    \tl_clear:N \sparagraphfrom
5723    \tl_clear:N \sparagraphto
5724    \tl_clear:N \l_stex_sparagraph_start_tl
5725    \tl_clear:N \l__stex_statements_sparagraph_imports_tl
5726    \str_clear:N \sparagraphid
5727    \str_clear:N \sparagraphtype
5728    \clist_clear:N \l__stex_statements_sparagraph_for_clist
5729    \str_clear:N \sparagraphname
5730    \keys_set:nn { stex / sparagraph }{ #1 }
5731 }
5732 \newif\if@in@omtext\@in@omtextfalse
5733
5734 \NewDocumentEnvironment {sparagraph} { O{} } {
5735    \stex_sparagraph_args:n { #1 }
5736    \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5737        \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
5738    }{
5739        \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
5740    }
5741    \@in@omtexttrue
5742    \stex_if_smsmode:F {
5743        \seq_clear:N \l_tmpb_seq
5744        \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5745            \tl_if_empty:nF{ ##1 }{
```

```
5746        \stex_get_symbol:n { ##1 }
5747        \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5748          \l_stex_get_symbol_uri_str
5749        }
5750      }
5751    }
5752    \exp_args:Nnnx
5753    \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}
5754    \str_if_empty:NF \sparagraphtype {
5755      \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
5756    }
5757    \str_if_empty:NF \sparagraphfrom {
5758      \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5759    }
5760    \str_if_empty:NF \sparagraphto {
5761      \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5762    }
5763    \str_if_empty:NF \sparagraphname {
5764      \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
5765    }
5766    \clist_set:No \l_tmpa_clist \sparagraphtype
5767    \tl_clear:N \l_tmpa_tl
5768    \clist_map_inline:Nn \sparagraphtype {
5769      \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
5770        \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
5771      }
5772    }
5773    \stex_csl_to_imports:No \usemodule \l__stex_statements_sparagraph_imports_tl
5774    \tl_if_empty:NTF \l_tmpa_tl {
5775      \__stex_statements_sparagraph_start:
5776    }{
5777      \l_tmpa_tl
5778    }
5779  }
5780  \clist_set:No \l_tmpa_clist \sparagraphtype
5781  \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}
5782  {
5783    \stex_reactivate_macro:N \definiendum
5784    \stex_reactivate_macro:N \definame
5785    \stex_reactivate_macro:N \Definame
5786    \stex_reactivate_macro:N \premise
5787    \stex_reactivate_macro:N \definiens
5788  }
5789  \str_if_empty:NTF \sparagraphid {
5790    \str_if_empty:NTF \sparagraphname {
5791      \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5792        \stex_ref_new_doc_target:n {}
5793      }
5794    } {
5795      \stex_ref_new_doc_target:n {}
5796    }
5797  } {
5798    \stex_ref_new_doc_target:n \sparagraphid
5799  }
```

```
5800      \exp_args:NNx
5801      \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5802        \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5803          \tl_if_empty:nF{ ##1 }{
5804            \stex_get_symbol:n { ##1 }
5805            \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5806          }
5807        }
5808      }
5809      \stex_smsmode_do:
5810      \ignorespacesandpars
5811  }{
5812      \str_if_empty:NF \sparagraphname {
5813        \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5814        \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5815      }
5816      \stex_if_smsmode:F {
5817        \clist_set:No \l_tmpa_clist \sparagraphtype
5818        \tl_clear:N \l_tmpa_tl
5819        \clist_map_inline:Nn \l_tmpa_clist {
5820          \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
5821            \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
5822          }
5823        }
5824        \tl_if_empty:NTF \l_tmpa_tl {
5825          \__stex_statements_sparagraph_end:
5826        }{
5827          \l_tmpa_tl
5828        }
5829        \end{stex_annotate_env}
5830      }
5831  }
```

\stexpatchparagraph

```
5832
5833  \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
5834    \stex_par:\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5835      \tl_if_empty:NF \l_stex_sparagraph_title_tl {
5836        \titleemph{\l_stex_sparagraph_title_tl}:~
5837      }
5838    }{
5839      \titleemph{\l_stex_sparagraph_start_tl}~
5840    }
5841  }
5842  \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\stex_par:\medskip}
5843
5844  \newcommand\stexpatchparagraph[3][] {
5845      \str_set:Nx \l_tmpa_str{ #1 }
5846      \str_if_empty:NTF \l_tmpa_str {
5847        \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
5848        \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
5849      }{
5850        \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
5851        \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
```

```
5852        }
5853 }
5854
5855 \keys_define:nn { stex / inlinepara} {
5856   id      .str_set_x:N   = \sparagraphid ,
5857   type    .str_set_x:N   = \sparagraphtype ,
5858   for     .clist_set:N    = \l__stex_statements_sparagraph_for_clist ,
5859   from    .tl_set:N       = \sparagraphfrom ,
5860   to      .tl_set:N       = \sparagraphto ,
5861   name    .str_set:N      = \sparagraphname
5862 }
5863 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
5864   \tl_clear:N \sparagraphfrom
5865   \tl_clear:N \sparagraphto
5866   \str_clear:N \sparagraphid
5867   \str_clear:N \sparagraphtype
5868   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5869   \str_clear:N \sparagraphname
5870   \keys_set:nn { stex / inlinepara }{ #1 }
5871 }
5872 \NewDocumentCommand \inlinepara { O{} m } {
5873   \begingroup
5874   \__stex_statements_inlinepara_args:n{ #1 }
5875   \clist_set:No \l_tmpa_clist \sparagraphtype
5876   \str_if_empty:NTF \sparagraphid {
5877     \str_if_empty:NTF \sparagraphname {
5878       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5879         \stex_ref_new_doc_target:n {}
5880       }
5881     } {
5882       \stex_ref_new_doc_target:n {}
5883     }
5884   } {
5885     \stex_ref_new_doc_target:n \sparagraphid
5886   }
5887   \stex_if_smsmode:TF{
5888     \str_if_empty:NF \sparagraphname {
5889       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5890       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5891     }
5892   }{
5893     \seq_clear:N \l_tmpb_seq
5894     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5895       \tl_if_empty:nF{ ##1 }{
5896         \stex_get_symbol:n { ##1 }
5897         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5898           \l_stex_get_symbol_uri_str
5899         }
5900       }
5901     }
5902     \exp_args:Nnx
5903     \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}{
5904       \str_if_empty:NF \sparagraphtype {
5905         \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
```

220

```
5906          }
5907          \str_if_empty:NF \sparagraphfrom {
5908            \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5909          }
5910          \str_if_empty:NF \sparagraphto {
5911            \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5912          }
5913          \str_if_empty:NF \sparagraphname {
5914            \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5915            \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
5916            \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5917          }
5918          \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5919            \clist_map_inline:Nn \l_tmpb_seq {
5920              \stex_ref_new_sym_target:n {##1}
5921            }
5922          }
5923          #2
5924        }
5925      }
5926      \endgroup
5927      \stex_smsmode_do:
5928    }
5929
```

(*End definition for* \stexpatchparagraph. *This function is documented on page* *48.*)

```
5930  ⟨/package⟩
```

Chapter 33

The Implementation

```
5931 ⟨∗package⟩
5932 ⟨@@=stex_sproof⟩
5933
5934 %%%%%%%%%%%%   sproof.dtx   %%%%%%%%%%%%
5935
```

33.1   Proofs

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

```
5936 \keys_define:nn { stex / spf } {
5937   id          .str_set_x:N  = \spfid,
5938   for         .clist_set:N  = \l__stex_sproof_spf_for_clist ,
5939   from        .tl_set:N     = \l__stex_sproof_spf_from_tl ,
5940   proofend    .tl_set:N     = \l__stex_sproof_spf_proofend_tl,
5941   type        .str_set_x:N  = \spftype,
5942   title       .tl_set:N     = \spftitle,
5943   continues   .tl_set:N     = \l__stex_sproof_spf_continues_tl,
5944   functions   .tl_set:N     = \l__stex_sproof_spf_functions_tl,
5945   method      .tl_set:N     = \l__stex_sproof_spf_method_tl
5946 }
5947 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
5948 \str_clear:N \spfid
5949 \tl_clear:N \l__stex_sproof_spf_for_tl
5950 \tl_clear:N \l__stex_sproof_spf_from_tl
5951 \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
5952 \str_clear:N \spftype
5953 \tl_clear:N \spftitle
5954 \tl_clear:N \l__stex_sproof_spf_continues_tl
5955 \tl_clear:N \l__stex_sproof_spf_functions_tl
5956 \tl_clear:N \l__stex_sproof_spf_method_tl
5957   \bool_set_false:N \l__stex_sproof_inc_counter_bool
5958 \keys_set:nn { stex / spf }{ #1 }
5959 }
```

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

```
5960 \str_set:Nn\c__stex_sproof_flow_str{inline}
```

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.

```
5961 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
5962 \cs_new_protected:Npn \sproofnumber {
5963   \int_set:Nn \l_tmpa_int {1}
5964   \bool_while_do:nn {
5965     \int_compare_p:nNn {
5966       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5967     } > 0
5968   }{
5969     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
5970     \int_incr:N \l_tmpa_int
5971   }
5972 }
5973 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
5974   \int_set:Nn \l_tmpa_int {1}
5975   \bool_while_do:nn {
5976     \int_compare_p:nNn {
5977       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5978     } > 0
5979   }{
5980     \int_incr:N \l_tmpa_int
5981   }
5982   \int_compare:nNnF \l_tmpa_int = 1 {
5983     \int_decr:N \l_tmpa_int
5984   }
5985   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
5986     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1
5987   }
5988 }
5989
5990 \cs_new_protected:Npn \__stex_sproof_add_counter: {
5991   \int_set:Nn \l_tmpa_int {1}
5992   \bool_while_do:nn {
5993     \int_compare_p:nNn {
5994       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5995     } > 0
5996   }{
5997     \int_incr:N \l_tmpa_int
5998   }
5999   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 1 }
6000 }
6001
6002 \cs_new_protected:Npn \__stex_sproof_remove_counter: {
6003   \int_set:Nn \l_tmpa_int {1}
6004   \bool_while_do:nn {
```

```
6005        \int_compare_p:nNn {
6006            \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6007        } > 0
6008    }{
6009        \int_incr:N \l_tmpa_int
6010    }
6011    \int_decr:N \l_tmpa_int
6012    \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 0 }
6013 }
```

\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

```
6014 \def\sproof@box{
6015    \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
6016 }
6017 \def\sproofend{
6018    \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
6019        \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
6020    }
6021 }
```

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

spf@*@kw

```
6022 \def\spf@proofsketch@kw{Proof~Sketch}
6023 \def\spf@proof@kw{Proof}
6024 \def\spf@step@kw{Step}
```

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

For the other languages, we set up triggers

```
6025 \AddToHook{begindocument}{
6026    \ltx@ifpackageloaded{babel}{
6027        \makeatletter
6028        \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6029        \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6030            \input{sproof-ngerman.ldf}
6031        }
6032        \clist_if_in:NnT \l_tmpa_clist {finnish}{
6033            \input{sproof-finnish.ldf}
6034        }
6035        \clist_if_in:NnT \l_tmpa_clist {french}{
6036            \input{sproof-french.ldf}
6037        }
6038        \clist_if_in:NnT \l_tmpa_clist {russian}{
6039            \input{sproof-russian.ldf}
6040        }
6041        \makeatother
6042    }{}
6043 }
```

spfsketch

```
6044 \newcommand\spfsketch[2][]{
6045    \begingroup
6046    \let \premise \stex_proof_premise:
```

224

```
6047      \__stex_sproof_spf_args:n{#1}
6048      \stex_if_smsmode:TF {
6049        \str_if_empty:NF \spfid {
6050          \stex_ref_new_doc_target:n \spfid
6051        }
6052      }{
6053        \seq_clear:N \l_tmpa_seq
6054        \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6055          \tl_if_empty:nF{ ##1 }{
6056            \stex_get_symbol:n { ##1 }
6057            \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6058              \l_stex_get_symbol_uri_str
6059            }
6060          }
6061        }
6062        \exp_args:Nnx
6063        \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
6064          \str_if_empty:NF \spftype {
6065            \stex_annotate_invisible:nnn{type}{\spftype}{}
6066          }
6067          \clist_set:No \l_tmpa_clist \spftype
6068          \tl_set:Nn \l_tmpa_tl {
6069            \titleemph{
6070              \tl_if_empty:NTF \spftitle {
6071                \spf@proofsketch@kw
6072              }{
6073                \spftitle
6074              }
6075            }:~
6076          }
6077          \clist_map_inline:Nn \l_tmpa_clist {
6078            \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6079              \tl_clear:N \l_tmpa_tl
6080            }
6081          }
6082          \str_if_empty:NF \spfid {
6083            \stex_ref_new_doc_target:n \spfid
6084          }
6085          \l_tmpa_tl #2 \sproofend
6086        }
6087      }
6088      \endgroup
6089      \stex_smsmode_do:
6090  }
6091
```

(*End definition for* spfsketch. *This function is documented on page 45.*)

spfeq   This is very similar to \spfsketch, but uses a computation array[12][13]

```
6092  \newenvironment{spfeq}[2][]{
6093      \__stex_sproof_spf_args:n{#1}
```

---

[12]EDNOTE: This should really be more like a tabular with an ensuremath in it. or invoke text on the last column

[13]EDNOTE: document above

```
6094    \let \premise \stex_proof_premise:
6095    \stex_if_smsmode:TF {
6096      \str_if_empty:NF \spfid {
6097        \stex_ref_new_doc_target:n \spfid
6098      }
6099    }{
6100      \seq_clear:N \l_tmpa_seq
6101      \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6102        \tl_if_empty:nF{ ##1 }{
6103          \stex_get_symbol:n { ##1 }
6104          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6105            \l_stex_get_symbol_uri_str
6106          }
6107        }
6108      }
6109      \exp_args:Nnnx
6110      \begin{stex_annotate_env}{spfeq}{\seq_use:Nn \l_tmpa_seq {,}}
6111      \str_if_empty:NF \spftype {
6112        \stex_annotate_invisible:nnn{type}{\spftype}{}
6113      }
6114
6115      \clist_set:No \l_tmpa_clist \spftype
6116      \tl_clear:N \l_tmpa_tl
6117      \clist_map_inline:Nn \l_tmpa_clist {
6118        \tl_if_exist:cT {__stex_sproof_spfeq_##1_start:}{
6119          \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_start:}}
6120        }
6121        \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6122          \tl_set:Nn \l_tmpa_tl {\use:n{}}
6123        }
6124      }
6125      \tl_if_empty:NTF \l_tmpa_tl {
6126        \__stex_sproof_spfeq_start:
6127      }{
6128        \l_tmpa_tl
6129      }{~#2}
6130      \str_if_empty:NF \spfid {
6131        \stex_ref_new_doc_target:n \spfid
6132      }
6133      \begin{displaymath}\begin{array}{rcll}
6134    }
6135    \stex_smsmode_do:
6136  }{
6137    \stex_if_smsmode:F {
6138      \end{array}\end{displaymath}
6139      \clist_set:No \l_tmpa_clist \spftype
6140      \tl_clear:N \l_tmpa_tl
6141      \clist_map_inline:Nn \l_tmpa_clist {
6142        \tl_if_exist:cT {__stex_sproof_spfeq_##1_end:}{
6143          \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_end:}}
6144        }
6145      }
6146      \tl_if_empty:NTF \l_tmpa_tl {
6147        \__stex_sproof_spfeq_end:
```

```
6148        }{
6149          \l_tmpa_tl
6150        }
6151      \end{stex_annotate_env}
6152    }
6153 }
6154
6155 \cs_new_protected:Nn \__stex_sproof_spfeq_start: {
6156    \titleemph{
6157      \tl_if_empty:NTF \spftitle {
6158        \spf@proof@kw
6159      }{
6160        \spftitle
6161      }
6162    }:
6163 }
6164 \cs_new_protected:Nn \__stex_sproof_spfeq_end: {\sproofend}
6165
6166 \newcommand\stexpatchspfeq[3][] {
6167      \str_set:Nx \l_tmpa_str{ #1 }
6168      \str_if_empty:NTF \l_tmpa_str {
6169        \tl_set:Nn \__stex_sproof_spfeq_start: { #2 }
6170        \tl_set:Nn \__stex_sproof_spfeq_end: { #3 }
6171      }{
6172        \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_start:\endcsname{ #2 }
6173        \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_end:\endcsname{ #3 }
6174      }
6175 }
6176
```

(*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.

```
6177 \newenvironment{sproof}[2][]{
6178    \let \premise \stex_proof_premise:
6179    \intarray_gzero:N \l__stex_sproof_counter_intarray
6180    \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
6181    \__stex_sproof_spf_args:n{#1}
6182    \stex_if_smsmode:TF {
6183      \str_if_empty:NF \spfid {
6184        \stex_ref_new_doc_target:n \spfid
6185      }
6186    }{
6187      \seq_clear:N \l_tmpa_seq
6188      \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6189        \tl_if_empty:nF{ ##1 }{
6190          \stex_get_symbol:n { ##1 }
6191          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6192            \l_stex_get_symbol_uri_str
6193          }
6194        }
6195      }
```

```
6196    \exp_args:Nnnx
6197    \begin{stex_annotate_env}{sproof}{\seq_use:Nn \l_tmpa_seq {,}}
6198    \str_if_empty:NF \spftype {
6199      \stex_annotate_invisible:nnn{type}{\spftype}{}
6200    }
6201
6202    \clist_set:No \l_tmpa_clist \spftype
6203    \tl_clear:N \l_tmpa_tl
6204    \clist_map_inline:Nn \l_tmpa_clist {
6205      \tl_if_exist:cT {__stex_sproof_sproof_##1_start:}{
6206        \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_start:}}
6207      }
6208      \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6209        \tl_set:Nn \l_tmpa_tl {\use:n{}}
6210      }
6211    }
6212    \tl_if_empty:NTF \l_tmpa_tl {
6213      \__stex_sproof_sproof_start:
6214    }{
6215      \l_tmpa_tl
6216    }{~#2}
6217    \str_if_empty:NF \spfid {
6218      \stex_ref_new_doc_target:n \spfid
6219    }
6220    \begin{description}
6221  }
6222  \stex_smsmode_do:
6223 }{
6224  \stex_if_smsmode:F{
6225    \end{description}
6226    \clist_set:No \l_tmpa_clist \spftype
6227    \tl_clear:N \l_tmpa_tl
6228    \clist_map_inline:Nn \l_tmpa_clist {
6229      \tl_if_exist:cT {__stex_sproof_sproof_##1_end:}{
6230        \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_end:}}
6231      }
6232    }
6233    \tl_if_empty:NTF \l_tmpa_tl {
6234      \__stex_sproof_sproof_end:
6235    }{
6236      \l_tmpa_tl
6237    }
6238    \end{stex_annotate_env}
6239  }
6240 }
6241
6242 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
6243  \par\noindent\titleemph{
6244    \tl_if_empty:NTF \spftype {
6245      \spf@proof@kw
6246    }{
6247      \spftype
6248    }
6249  }:
```

```
6250 }
6251 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
6252
6253 \newcommand\stexpatchproof[3][] {
6254   \str_set:Nx \l_tmpa_str{ #1 }
6255   \str_if_empty:NTF \l_tmpa_str {
6256     \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
6257     \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
6258   }{
6259     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
6260     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
6261   }
6262 }
```

```
6263 \newcommand\spfidea[2][]{
6264   \__stex_sproof_spf_args:n{#1}
6265   \titleemph{
6266     \tl_if_empty:NTF \spftype {Proof~Idea}{
6267       \spftype
6268     }:
6269   }~#2
6270   \sproofend
6271 }
```

(*End definition for* \spfidea. *This function is documented on page* *45.*)

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.

```
6272 \newenvironment{spfstep}[1][]{
6273   \__stex_sproof_spf_args:n{#1}
6274   \stex_if_smsmode:TF {
6275     \str_if_empty:NF \spfid {
6276       \stex_ref_new_doc_target:n \spfid
6277     }
6278   }{
6279     \@in@omtexttrue
6280     \clist_set:No \l_tmpa_clist \spftype
6281     \tl_set:Nn \l_tmpa_tl {
6282       \item[\sproofnumber]
6283       \bool_set_true:N \l__stex_sproof_inc_counter_bool
6284     }
6285     \clist_map_inline:Nn \l_tmpa_clist {
6286       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6287         \tl_clear:N \l_tmpa_tl
6288       }
6289     }
6290     \l_tmpa_tl
6291     \seq_clear:N \l_tmpa_seq
6292     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
```

229

```
6293        \tl_if_empty:nF{ ##1 }{
6294          \stex_get_symbol:n { ##1 }
6295          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6296            \l_stex_get_symbol_uri_str
6297          }
6298        }
6299      }
6300      \exp_args:Nnnx
6301      \begin{stex_annotate_env}{spfstep}{\seq_use:Nn \l_tmpa_seq {,}}
6302      \str_if_empty:NF \spftype {
6303        \stex_annotate_invisible:nnn{type}{\spftype}{}
6304      }
6305      \tl_if_empty:NF \spftitle {
6306        {(\titleemph{\spftitle})\enspace}
6307      }
6308      \str_if_empty:NF \spfid {
6309        \stex_ref_new_doc_target:n \spfid
6310      }
6311    }
6312    \stex_smsmode_do:
6313    \ignorespacesandpars
6314  }{
6315    \bool_if:NT \l__stex_sproof_inc_counter_bool {
6316      \__stex_sproof_inc_counter:
6317    }
6318    \stex_if_smsmode:F {
6319      \end{stex_annotate_env}
6320    }
6321  }
```

spfcomment

```
6322  \newenvironment{spfcomment}[1][]{
6323    \__stex_sproof_spf_args:n{#1}
6324    \clist_set:No \l_tmpa_clist \spftype
6325    \tl_set:Nn \l_tmpa_tl {
6326      \item[\sproofnumber]
6327      \bool_set_true:N \l__stex_sproof_inc_counter_bool
6328    }
6329    \clist_map_inline:Nn \l_tmpa_clist {
6330      \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6331        \tl_clear:N \l_tmpa_tl
6332      }
6333    }
6334    \l_tmpa_tl
6335  }{
6336    \bool_if:NT \l__stex_sproof_inc_counter_bool {
6337      \__stex_sproof_inc_counter:
6338    }
6339  }
```

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.

```
6340  \newenvironment{subproof}[2][]{
6341    \__stex_sproof_spf_args:n{#1}
6342    \stex_if_smsmode:TF{
6343      \str_if_empty:NF \spfid {
6344        \stex_ref_new_doc_target:n \spfid
6345      }
6346    }{
6347      \seq_clear:N \l_tmpa_seq
6348      \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6349        \tl_if_empty:nF{ ##1 }{
6350          \stex_get_symbol:n { ##1 }
6351          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6352            \l_stex_get_symbol_uri_str
6353          }
6354        }
6355      }
6356      \exp_args:Nnnx
6357      \begin{stex_annotate_env}{subproof}{\seq_use:Nn \l_tmpa_seq {,}}
6358      \str_if_empty:NF \spftype {
6359        \stex_annotate_invisible:nnn{type}{\spftype}{}
6360      }
6361
6362      \clist_set:No \l_tmpa_clist \spftype
6363      \tl_set:Nn \l_tmpa_tl {
6364        \item[\sproofnumber]
6365        \bool_set_true:N \l__stex_sproof_inc_counter_bool
6366      }
6367      \clist_map_inline:Nn \l_tmpa_clist {
6368        \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6369          \tl_clear:N \l_tmpa_tl
6370        }
6371      }
6372      \l_tmpa_tl
6373      \tl_if_empty:NF \spftitle {
6374        {(\titleemph{\spftitle})\enspace}
6375      }
6376      {~#2}
6377      \str_if_empty:NF \spfid {
6378        \stex_ref_new_doc_target:n \spfid
6379      }
6380    }
6381    \__stex_sproof_add_counter:
6382    \stex_smsmode_do:
6383  }{
6384    \__stex_sproof_remove_counter:
6385    \bool_if:NT \l__stex_sproof_inc_counter_bool {
6386      \__stex_sproof_inc_counter:
6387    }
6388    \stex_if_smsmode:F{
6389      \end{stex_annotate_env}
6390    }
6391  }
```

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

```
6392  \newenvironment{spfcases}[2][]{
6393    \tl_if_empty:nTF{#1}{
6394      \begin{subproof}[method=by-cases]{#2}
6395    }{
6396      \begin{subproof}[#1,method=by-cases]{#2}
6397    }
6398  }{
6399    \end{subproof}
6400  }
```

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

```
6401  \newenvironment{spfcase}[2][]{
6402    \__stex_sproof_spf_args:n{#1}
6403    \stex_if_smsmode:TF {
6404      \str_if_empty:NF \spfid {
6405        \stex_ref_new_doc_target:n \spfid
6406      }
6407    }{
6408      \seq_clear:N \l_tmpa_seq
6409      \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6410        \tl_if_empty:nF{ ##1 }{
6411          \stex_get_symbol:n { ##1 }
6412          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6413            \l_stex_get_symbol_uri_str
6414          }
6415        }
6416      }
6417      \exp_args:Nnnx
6418      \begin{stex_annotate_env}{spfcase}{\seq_use:Nn \l_tmpa_seq {,}}
6419      \str_if_empty:NF \spftype {
6420        \stex_annotate_invisible:nnn{type}{\spftype}{}
6421      }
6422      \clist_set:No \l_tmpa_clist \spftype
6423      \tl_set:Nn \l_tmpa_tl {
6424        \item[\sproofnumber]
6425        \bool_set_true:N \l__stex_sproof_inc_counter_bool
6426      }
6427      \clist_map_inline:Nn \l_tmpa_clist {
6428        \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6429          \tl_clear:N \l_tmpa_tl
6430        }
6431      }
6432      \l_tmpa_tl
6433      \tl_if_empty:nF{#2}{
6434        \titleemph{#2}:~
6435      }
6436    }
6437    \__stex_sproof_add_counter:
6438    \stex_smsmode_do:
6439  }{
6440    \__stex_sproof_remove_counter:
6441    \bool_if:NT \l__stex_sproof_inc_counter_bool {
6442      \__stex_sproof_inc_counter:
```

```
6443    }
6444    \stex_if_smsmode:F{
6445      \clist_set:No \l_tmpa_clist \spftype
6446      \tl_set:Nn \l_tmpa_tl{\sproofend}
6447      \clist_map_inline:Nn \l_tmpa_clist {
6448        \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6449          \tl_clear:N \l_tmpa_tl
6450        }
6451      }
6452      \l_tmpa_tl
6453      \end{stex_annotate_env}
6454    }
6455 }
```

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

```
6456 \newcommand\spfcasesketch[3][]{
6457    \begin{spfcase}[#1]{#2}#3\end{spfcase}
6458 }
```

33.2 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.

```
6459 \keys_define:nn { stex / just }{
6460    id        .str_set_x:N  = \l__stex_sproof_just_id_str,
6461    method    .tl_set:N     = \l__stex_sproof_just_method_tl,
6462    premises  .tl_set:N     = \l__stex_sproof_just_premises_tl,
6463    args      .tl_set:N     = \l__stex_sproof_just_args_tl
6464 }
```

EdN:14   The next three environments and macros are purely semantic, so we ignore the keyval arguments for now and only display the content.[14]

\spfjust

```
6465 \newcommand\spfjust[1][]{}
```

(*End definition for* \spfjust. *This function is documented on page 46.*)

\premise

```
6466 \newcommand\stex_proof_premise:[2][]{#2}
```

(*End definition for* \premise. *This function is documented on page 46.*)

\justarg   the \justarg macro is purely semantic, so we ignore the keyval arguments for now and only display the content.

```
6467 \newcommand\justarg[2][]{#2}
6468 ⟨/package⟩
```

(*End definition for* \justarg. *This function is documented on page 46.*)

Some auxiliary code, and clean up to be executed at the end of the package.

---

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

Chapter 34

ST_EX
-Others Implementation

```
6469  ⟨∗package⟩
6470
6471  %%%%%%%%%%%%%   others.dtx   %%%%%%%%%%%%%
6472
6473  ⟨@@=stex_others⟩
```

Warnings and error messages
```
6474  % None
```

\MSC Math subject classifier
```
6475  \NewDocumentCommand \MSC {m} {
6476    % TODO
6477  }
```

(*End definition for* \MSC. *This function is documented on page* **??**.)
Patching tikzinput, if loaded

```
6478  \@ifpackageloaded{tikzinput}{
6479    \RequirePackage{stex-tikzinput}
6480  }{}
6481
6482  \bool_if:NT \c_stex_persist_mode_bool {
6483    \let\__stex_notation_restore_notation_old:nnnnn
6484      \__stex_notation_restore_notation:nnnnn
6485    \def\__stex_notation_restore_notation_new:nnnnn#1#2#3#4#5{
6486      \__stex_notation_restore_notation_old:nnnnn{#1}{#2}{#3}{#4}{#5}
6487      \ExplSyntaxOn
6488    }
6489    \def\__stex_notation_restore_notation:nnnnn{
6490      \ExplSyntaxOff
6491      \catcode'~10
6492      \__stex_notation_restore_notation_new:nnnnn
6493    }
6494    \input{\jobname.sms}
6495    \let\__stex_notation_restore_notation:nnnnn
6496      \__stex_notation_restore_notation_old:nnnnn
6497    \prop_if_exist:NT\c_stex_mathhub_main_manifest_prop{
```

```
6498      \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
6499        \l_tmpa_str
6500      \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str _manifest_prop }
6501        \c_stex_mathhub_main_manifest_prop
6502      \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
6503    }
6504  }
6505  ⟨/package⟩
```

Chapter 35

STEX -Metatheory Implementation

```
6506  ⟨∗package⟩
6507  ⟨@@=stex_modules⟩
6508
6509  %%%%%%%%%%%%   metatheory.dtx   %%%%%%%%%%%%
6510
6511  \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX/meta}
6512  \begingroup
6513  \stex_module_setup:nn{
6514    ns=\c_stex_metatheory_ns_str,
6515    meta=NONE
6516  }{Metatheory}
6517  \stex_reactivate_macro:N \symdecl
6518  \stex_reactivate_macro:N \notation
6519  \stex_reactivate_macro:N \symdef
6520  \ExplSyntaxOff
6521  \csname stex_suppress_html:n\endcsname{
6522    % is-a (a:A, a \in A, a is an A, etc.)
6523    \symdecl{isa}[args=ai]
6524    \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
6525    \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
6526    \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
6527
6528    % bind (\forall, \Pi, \lambda etc.)
6529    \symdecl{bind}[args=Bi,assoc=pre]
6530    \notation{bind}[depfun,prec=nobrackets,op={(\cdot)\;\to\;\cdot}]{\comp( #1 \comp{)\;\to\;}
6531    \notation{bind}[forall]{\comp\forall #1.\;#2}{##1 \comp, ##2}
6532    \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6533
6534    % implicit bind
6535    \symdecl{implicitbind}[args=Bi,assoc=pre]
6536    \notation{implicitbind}[braces,prec=nobrackets,op={\{\cdot\}_I\;\cdot}]{\comp\{ #1 \comp\
6537    \notation{implicitbind}[depfun,prec=nobrackets]{\comp( #1 \comp)\;\to_I\;} #2}{##1 \comp,
6538    \notation{implicitbind}[Pi]{\comp\prod^I_{#1}#2}{##1\comp,##2}
6539
6540    % dummy variable
```

```
6541    \symdecl{dummyvar}
6542    \notation{dummyvar}[underscore]{\comp\_}
6543    \notation{dummyvar}[dot]{\comp\cdot}
6544    \notation{dummyvar}[dash]{\comp{{\rm --}}}
6545
6546    %fromto (function space, Hom-set, implication etc.)
6547    \symdecl{fromto}[args=ai]
6548    \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
6549    \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
6550
6551    % mapto (lambda etc.)
6552    %\symdecl{mapto}[args=Bi]
6553    %\notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
6554    %\notation{mapto}[lambda]{\comp\lambda #1 \comp.\; #2}{#1 \comp, #2}
6555    %\notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.\; #2}{#1 \comp, #2}
6556
6557    % function/operator application
6558    \symdecl{apply}[args=ia]
6559    \notation{apply}[prec=0;0x\infprec,parens]{#1 \comp( #2 \comp)}{##1 \comp, ##2}
6560    \notation{apply}[prec=0;0x\infprec,lambda]{#1 \; #2 }{##1 \; ##2}
6561
6562    % collection of propositions/booleans/truth values
6563    \symdecl{prop}[name=proposition]
6564    \notation{prop}[prop]{\comp{{\rm prop}}}
6565    \notation{prop}[BOOL]{\comp{{\rm BOOL}}}
6566
6567    \symdecl{judgmentholds}[args=1]
6568    \notation{judgmentholds}[vdash,op=\vdash]{\comp\vdash\; #1}
6569
6570    % sequences
6571    \symdecl{seqtype}[args=1]
6572    \notation{seqtype}[kleene]{#1^{\comp\ast}}
6573
6574    \symdecl{seqexpr}[args=a]
6575    \notation{seqexpr}[angle,prec=nobrackets]{\comp\langle #1\comp\rangle}{##1\comp,##2}
6576
6577    \symdef{seqmap}[args=abi,setlike]{\comp\{#3 \comp| #2\comp\in \dobrackets{#1} \comp\}}{##1
6578    \symdef{seqprepend}[args=ia]{#1 \comp{::} #2}{##1 \comp, ##2}
6579    \symdef{seqappend}[args=ai]{#1 \comp{::} #2}{##1 \comp, ##2}
6580    \symdef{seqfoldleft}[args=iabbi]{ \comp{foldl}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\com
6581    \symdef{seqfoldright}[args=iabbi,op=foldr]{ \comp{foldr}\dobrackets{#1,#2}\dobrackets{#3\c
6582    \symdef{seqhead}[args=a]{\comp{head}\dobrackets{#1}}{##1 \comp, ##2}
6583    \symdef{seqtail}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6584    \symdef{seqlast}[args=a]{\comp{last}\dobrackets{#1}}{##1 \comp, ##2}
6585    \symdef{seqinit}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6586
6587    \symdef{sequence-index}[args=2,li,prec=nobrackets]{{#1}_{#2}}
6588    \notation{sequence-index}[ui,prec=nobrackets]{{#1}^{#2}}
6589
6590    \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{,\ellipses}}{##1\comp,##2}
6591    \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{,\ellipses,}#2}{##1\comp,##2}
6592    \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{,\ellipses,}#2\comp{,\ellipses,}#
6593
6594    % nat literals
```

237

```
6595    \symdef{natliteral}{\comp{\mathtt{Ord}}}}

6596
6597    % letin (``let'', local definitions, variable substitution)
6598    \symdecl{letin}[args=bii]
6599    \notation{letin}[let]{\comp{{\rm let}}\;#1\comp{=}#2\;\comp{{\rm in}}\;#3}
6600    \notation{letin}[subst]{#3 \comp[ #1 \comp/ #2 \comp]}
6601    \notation{letin}[frac]{#3 \comp[ \frac{#2}{#1} \comp]}

6602
6603    % structures
6604    \symdecl*{module-type}[args=1]
6605    \notation{module-type}{\comp{\mathtt{MOD}} #1}
6606    \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
6607    \notation{mathstruct}[angle,prec=nobrackets]{\comp\langle #1 \comp\rangle}{##1 \comp, ##2}

6608
6609    % objects
6610    \symdecl{object}
6611    \notation{object}{\comp{\mathtt{OBJECT}}}}

6612
6613 }

6614
6615 % The following are abbreviations in the sTeX corpus that are left over from earlier
6616 % developments. They will eventually be phased out.

6617
6618    \ExplSyntaxOn
6619    \stex_add_to_current_module:n{
6620      \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
6621      \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
6622      \def\livar{\csname sequence-index\endcsname[li]}
6623      \def\uivar{\csname sequence-index\endcsname[ui]}
6624      \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
6625      \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
6626    }
6627 \__stex_modules_end_module:
6628 \endgroup

6629 ⟨/package⟩
```

Chapter 36

Tikzinput Implementation

```
6630 ⟨@@=tikzinput⟩
6631 ⟨∗package⟩
6632
6633 %%%%%%%%%%%%  tikzinput.dtx   %%%%%%%%%%%%
6634
6635 \ProvidesExplPackage{tikzinput}{2022/02/26}{3.0.1}{tikzinput package}
6636 \RequirePackage{l3keys2e}
6637
6638 \keys_define:nn { tikzinput } {
6639   image    .bool_set:N   = \c_tikzinput_image_bool,
6640   image    .default:n    = false ,
6641   unknown    .code:n       = {}
6642 }
6643
6644 \ProcessKeysOptions { tikzinput }
6645
6646 \bool_if:NTF \c_tikzinput_image_bool {
6647   \RequirePackage{graphicx}
6648
6649   \providecommand\usetikzlibrary[]{}
6650   \newcommand\tikzinput[2][]{\includegraphics[#1]{#2}}
6651 }{
6652   \RequirePackage{tikz}
6653   \RequirePackage{standalone}
6654
6655   \newcommand \tikzinput [2] [] {
6656     \setkeys{Gin}{#1}
6657     \ifx \Gin@ewidth \Gin@exclamation
6658       \ifx \Gin@eheight \Gin@exclamation
6659         \input { #2 }
6660       \else
6661         \resizebox{!}{ \Gin@eheight }{
6662           \input { #2 }
6663         }
6664       \fi
6665     \else
6666       \ifx \Gin@eheight \Gin@exclamation
6667         \resizebox{ \Gin@ewidth }{!}{
```

```
6668            \input { #2 }
6669          }
6670        \else
6671          \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
6672            \input { #2 }
6673          }
6674        \fi
6675      \fi
6676    }
6677  }

6678
6679  \newcommand \ctikzinput [2] [] {
6680    \begin{center}
6681      \tikzinput [#1] {#2}
6682    \end{center}
6683  }

6684
6685  \@ifpackageloaded{stex}{
6686    \RequirePackage{stex-tikzinput}
6687  }{}

6688
6689  ⟨/package⟩
6690  ⟨∗stex⟩

6691  \ProvidesExplPackage{stex-tikzinput}{2022/02/26}{3.0.1}{stex-tikzinput}
6692  \RequirePackage{stex}
6693  \RequirePackage{tikzinput}

6694
6695  \newcommand\mhtikzinput[2][]{%
6696    \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
6697    \stex_in_repository:nn\Gin@mhrepos{
6698      \tikzinput[#1]{\mhpath{##1}{#2}}
6699    }
6700  }
6701  \newcommand\cmhtikzinput[2][]{\begin{center}\mhtikzinput[#1]{#2}\end{center}}

6702
6703  \cs_new_protected:Nn \__tikzinput_usetikzlibrary:nn {
6704    \pgfkeys@spdef\pgf@temp{#1}
6705    \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
6706    \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgfut
6707    \expandafter\edef\csname tikz@library@#1atcode\endcsname{\the\catcode`\@}
6708    \expandafter\edef\csname tikz@library@#1barcode\endcsname{\the\catcode`\|}
6709    \expandafter\edef\csname tikz@library@#1dollarcode\endcsname{\the\catcode`\$}
6710    \catcode`\@=11
6711    \catcode`\|=12
6712    \catcode`\$=3
6713    \pgfutil@InputIfFileExists{#2}{}{}
6714    \catcode`\@=\csname tikz@library@#1atcode\endcsname
6715    \catcode`\|=\csname tikz@library@#1barcode\endcsname
6716    \catcode`\$=\csname tikz@library@#1dollarcode\endcsname
6717  }

6718

6719
6720  \newcommand\libusetikzlibrary[1]{
```

240

```
6721    \prop_if_exist:NF \l_stex_current_repository_prop {
6722      \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6723    }
6724    \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
6725      \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6726    }
6727    \seq_clear:N \l__tikzinput_libinput_files_seq
6728    \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
6729    \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str

6731    \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
6732      \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / tikzlibra
6733      \IfFileExists{ \l_tmpa_str }{
6734        \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6735      }{}
6736      \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
6737      \seq_put_right:No \l_tmpa_seq \l_tmpa_str
6738    }

6740    \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / tikzlibrary #1 .code.t
6741    \IfFileExists{ \l_tmpa_str }{
6742      \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6743    }{}

6745    \seq_if_empty:NTF \l__tikzinput_libinput_files_seq {
6746      \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .code.t
6747    }{
6748      \int_compare:nNnTF {\seq_count:N \l__tikzinput_libinput_files_seq} = 1 {
6749        \seq_map_inline:Nn \l__tikzinput_libinput_files_seq {
6750          \__tikzinput_usetikzlibrary:nn{#1}{ ##1 }
6751        }
6752      }{
6753        \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .co
6754      }
6755    }
6756 }

6757 ⟨/stex⟩
```

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

241

Chapter 37

document-structure.sty Implementation

⟨∗package⟩
⟨@@=document_structure⟩
`\ProvidesExplPackage{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure}`
`\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).

```
6762
6763 \keys_define:nn{ document-structure }{
6764   class       .str_set_x:N  = \c_document_structure_class_str,
6765   topsect     .str_set_x:N  = \c_document_structure_topsect_str,,
6766   unknown     .code:n       = {
6767     \PassOptionsToClass{\CurrentOption}{stex}
6768     \PassOptionsToClass{\CurrentOption}{tikzinput}
6769   }
6770 %  showignores .bool_set:N   = \c_document_structure_showignores_bool,
6771 }
6772 \ProcessKeysOptions{ document-structure }
6773 \str_if_empty:NT \c_document_structure_class_str {
6774   \str_set:Nn \c_document_structure_class_str {article}
6775 }
6776 \str_if_empty:NT \c_document_structure_topsect_str {
6777   \str_set:Nn \c_document_structure_topsect_str {section}
6778 }
```

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

```
6779 \RequirePackage{xspace}
6780 \RequirePackage{comment}
6781 \RequirePackage{stex}
6782 \AddToHook{begindocument}{
```

```
6783   \ltx@ifpackageloaded{babel}{
6784       \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6785       \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6786           \makeatletter\input{document-structure-ngerman.ldf}\makeatother
6787       }
6788   }{}
6789 }
```

\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.

```
6790 \int_new:N \l_document_structure_section_level_int
6791 \str_case:VnF \c_document_structure_topsect_str {
6792   {part}{
6793       \int_set:Nn \l_document_structure_section_level_int {0}
6794   }
6795   {chapter}{
6796       \int_set:Nn \l_document_structure_section_level_int {1}
6797   }
6798 }{
6799   \str_case:VnF \c_document_structure_class_str {
6800     {book}{
6801         \int_set:Nn \l_document_structure_section_level_int {0}
6802     }
6803     {report}{
6804         \int_set:Nn \l_document_structure_section_level_int {0}
6805     }
6806   }{
6807       \int_set:Nn \l_document_structure_section_level_int {2}
6808   }
6809 }
```

37.2   Document Structure

The structure of the document is given by the sfragment environment. 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:15 element of class omdoc_currentsectionlevel, wich will be instantiated by CSS later.[15]

```
6810 \def\current@section@level{document}%
6811 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
6812 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

(*End definition for* \currentsectionlevel. *This function is documented on page 53.*)

\skipfragment

```
6813 \cs_new_protected:Npn \skipfragment {
```

---

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

```
6814    \ifcase\l_document_structure_section_level_int
6815    \or\stepcounter{part}
6816    \or\stepcounter{chapter}
6817    \or\stepcounter{section}
6818    \or\stepcounter{subsection}
6819    \or\stepcounter{subsubsection}
6820    \or\stepcounter{paragraph}
6821    \or\stepcounter{subparagraph}
6822    \fi
6823 }
```

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

blindfragment

```
6824 \newcommand\at@begin@blindsfragment[1]{}
6825 \newenvironment{blindfragment}
6826 {
6827    \int_incr:N\l_document_structure_section_level_int
6828    \at@begin@blindsfragment\l_document_structure_section_level_int
6829 }{}
```

\sfragment@nonum   convenience macro: `\sfragment@nonum{⟨level⟩}{⟨title⟩}` makes an unnumbered section-
ing with title ⟨*title*⟩ at level ⟨*level*⟩.

```
6830 \newcommand\sfragment@nonum[2]{
6831    \ifx\hyper@anchor\@undefined\else\phantomsection\fi
6832    \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
6833 }
```

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

\sfragment@num   convenience macro: `\sfragment@nonum{⟨level⟩}{⟨title⟩}` makes numbered sectioning
with title ⟨*title*⟩ at level ⟨*level*⟩. We have to check the `short` key was given in the
`sfragment` 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.

```
6834 \newcommand\sfragment@num[2]{
6835    \tl_if_empty:NTF \l__document_structure_sfragment_short_tl {
6836       \@nameuse{#1}{#2}
6837    }{
6838       \cs_if_exist:NTF\rdfmeta@sectioning{
6839          \@nameuse{rdfmeta@#1@old}[\l__document_structure_sfragment_short_tl]{#2}
6840       }{
6841          \@nameuse{#1}[\l__document_structure_sfragment_short_tl]{#2}
6842       }
6843    }
6844 %\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\sfragment@id
6845 }
```

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

sfragment

```
6846 \keys_define:nn { document-structure / sfragment }{
6847    id              .str_set_x:N = \l__document_structure_sfragment_id_str,
6848    date            .str_set_x:N = \l__document_structure_sfragment_date_str,
```

```
6849    creators      .clist_set:N = \l__document_structure_sfragment_creators_clist,
6850    contributors  .clist_set:N = \l__document_structure_sfragment_contributors_clist,
6851    srccite       .tl_set:N    = \l__document_structure_sfragment_srccite_tl,
6852    type          .tl_set:N    = \l__document_structure_sfragment_type_tl,
6853    short         .tl_set:N    = \l__document_structure_sfragment_short_tl,
6854    display       .tl_set:N    = \l__document_structure_sfragment_display_tl,
6855    intro         .tl_set:N    = \l__document_structure_sfragment_intro_tl,
6856    imports       .tl_set:N    = \l__document_structure_sfragment_imports_tl,
6857    loadmodules   .bool_set:N  = \l__document_structure_sfragment_loadmodules_bool
6858 }
6859 \cs_new_protected:Nn \__document_structure_sfragment_args:n {
6860    \str_clear:N \l__document_structure_sfragment_id_str
6861    \str_clear:N \l__document_structure_sfragment_date_str
6862    \clist_clear:N \l__document_structure_sfragment_creators_clist
6863    \clist_clear:N \l__document_structure_sfragment_contributors_clist
6864    \tl_clear:N \l__document_structure_sfragment_srccite_tl
6865    \tl_clear:N \l__document_structure_sfragment_type_tl
6866    \tl_clear:N \l__document_structure_sfragment_short_tl
6867    \tl_clear:N \l__document_structure_sfragment_display_tl
6868    \tl_clear:N \l__document_structure_sfragment_imports_tl
6869    \tl_clear:N \l__document_structure_sfragment_intro_tl
6870    \bool_set_false:N \l__document_structure_sfragment_loadmodules_bool
6871    \keys_set:nn { document-structure / sfragment } { #1 }
6872 }
```

we define a switch for numbering lines and a hook for the beginning of groups: The
\at@begin@sfragment \at@begin@sfragment macro allows customization. It is run at the beginning of the
sfragment, i.e. after the section heading.

```
6873 \newif\if@mainmatter\@mainmattertrue
6874 \newcommand\at@begin@sfragment[3][]{}
```

Then we define a helper macro that takes care of the sectioning magic. It comes
with its own key/value interface for customization.

```
6875 \keys_define:nn { document-structure / sectioning }{
6876    name    .str_set_x:N  = \l__document_structure_sect_name_str   ,
6877    ref     .str_set_x:N  = \l__document_structure_sect_ref_str    ,
6878    clear   .bool_set:N   = \l__document_structure_sect_clear_bool ,
6879    clear   .default:n    = {true}                  ,
6880    num     .bool_set:N   = \l__document_structure_sect_num_bool   ,
6881    num     .default:n    = {true}
6882 }
6883 \cs_new_protected:Nn \__document_structure_sect_args:n {
6884    \str_clear:N \l__document_structure_sect_name_str
6885    \str_clear:N \l__document_structure_sect_ref_str
6886    \bool_set_false:N \l__document_structure_sect_clear_bool
6887    \bool_set_false:N \l__document_structure_sect_num_bool
6888    \keys_set:nn { document-structure / sectioning } { #1 }
6889 }
6890 \newcommand\omdoc@sectioning[3][]{
6891    \__document_structure_sect_args:n {#1 }
6892    \let\omdoc@sect@name\l__document_structure_sect_name_str
6893    \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
6894    \if@mainmatter% numbering not overridden by frontmatter, etc.
6895      \bool_if:NTF \l__document_structure_sect_num_bool {
```

245

```
6896        \sfragment@num{#2}{#3}
6897      }{
6898        \sfragment@nonum{#2}{#3}
6899      }
6900      \def\current@section@level{\omdoc@sect@name}
6901    \else
6902      \sfragment@nonum{#2}{#3}
6903    \fi
6904 }% 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.

```
6905 \newcommand\sfragment@redefine@addtocontents[1]{%
6906 %\edef\__document_structureimport{#1}%
6907 %\@for\@I:=\__document_structureimport\do{%
6908 %\edef\@path{\csname module@\@I   @path\endcsname}%
6909 %\@ifundefined{tf@toc}\relax%
6910 %    {\protected@write\tf@toc{}{\string\@requiremodules{\@path}}}}
6911 %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
6912 %\def\addcontentsline##1##2##3{%
6913 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}
6914 %\else% hyperref.sty not loaded
6915 %\def\addcontentsline##1##2##3{%
6916 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}{
6917 %\fi
6918 }% hypreref.sty loaded?
```

now the `sfragment` 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 sfragments in the `\sfragment@level` counter.

```
6919 \newenvironment{sfragment}[2][]% keys, title
6920 {
6921    \__document_structure_sfragment_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.

```
6922    \stex_csl_to_imports:No \usemodule \l__document_structure_sfragment_imports_tl
6923
6924    \bool_if:NT \l__document_structure_sfragment_loadmodules_bool {
6925      \sfragment@redefine@addtocontents{
6926        %\@ifundefined{module@id}\used@modules%
6927        %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}
6928      }
6929    }
```

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

```
6930
6931    \stex_document_title:n { #2 }
6932
6933    \int_incr:N\l_document_structure_section_level_int
6934    \ifcase\l_document_structure_section_level_int
6935      \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
6936      \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
```

246

```
6937      \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
6938      \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
6939      \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
6940      \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#
6941      \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{paragr
6942    \fi
6943    \at@begin@sfragment[#1]\l_document_structure_section_level_int{#2}
6944    \str_if_empty:NF \l__document_structure_sfragment_id_str {
6945      \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
6946    }
6947 }% for customization
6948 {}
```

and finally, we localize the sections

```
6949 \newcommand\omdoc@part@kw{Part}
6950 \newcommand\omdoc@chapter@kw{Chapter}
6951 \newcommand\omdoc@section@kw{Section}
6952 \newcommand\omdoc@subsection@kw{Subsection}
6953 \newcommand\omdoc@subsubsection@kw{Subsubsection}
6954 \newcommand\omdoc@paragraph@kw{paragraph}
6955 \newcommand\omdoc@subparagraph@kw{subparagraph}
```

37.3   Front and Backmatter

Index markup is provided by the omtext package [**Kohlhase:smmtf:git**], so in the
document-structure package we only need to supply the corresponding \printindex
command, if it is not already defined

\printindex

```
6956 \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 environ-
ments, we save their behavior (possibly defining it) in orig@*matter macros and make
them undefined (so that we can define the environments).

```
6957 \cs_if_exist:NTF\frontmatter{
6958    \let\__document_structure_orig_frontmatter\frontmatter
6959    \let\frontmatter\relax
6960 }{
6961    \tl_set:Nn\__document_structure_orig_frontmatter{
6962      \clearpage
6963      \@mainmatterfalse
6964      \pagenumbering{roman}
6965    }
6966 }
6967 \cs_if_exist:NTF\backmatter{
6968    \let\__document_structure_orig_backmatter\backmatter
6969    \let\backmatter\relax
6970 }{
6971    \tl_set:Nn\__document_structure_orig_backmatter{
6972      \clearpage
6973      \@mainmatterfalse
```

```
6974        \pagenumbering{roman}
6975      }
6976 }
```

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, oth-
erwise we define it.

```
6977 \newenvironment{frontmatter}{
6978     \__document_structure_orig_frontmatter
6979 }{
6980     \cs_if_exist:NTF\mainmatter{
6981        \mainmatter
6982     }{
6983        \clearpage
6984        \@mainmattertrue
6985        \pagenumbering{arabic}
6986     }
6987 }
```

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

```
6988 \newenvironment{backmatter}{
6989     \__document_structure_orig_backmatter
6990 }{
6991     \cs_if_exist:NTF\mainmatter{
6992        \mainmatter
6993     }{
6994        \clearpage
6995        \@mainmattertrue
6996        \pagenumbering{arabic}
6997     }
6998 }
```

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

```
6999 \@mainmattertrue\pagenumbering{arabic}
```

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

```
7000 \def \c__document_structure_document_str{document}
7001 \newcommand\afterprematurestop{}
7002 \def\prematurestop@endsfragment{
7003     \unless\ifx\@currenvir\c__document_structure_document_str
7004        \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafte
7005        \expandafter\prematurestop@endsfragment
7006     \fi
7007 }
7008 \providecommand\prematurestop{
7009     \message{Stopping~sTeX~processing~prematurely}
7010     \prematurestop@endsfragment
7011     \afterprematurestop
7012     \end{document}
7013 }
```

(*End definition for* `\prematurestop`. *This function is documented on page* *53.*)

37.4 Global Variables

\setSGvar set a global variable

```
7014 \RequirePackage{etoolbox}
7015 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}
```

(*End definition for* \setSGvar*. This function is documented on page 53.*)

\useSGvar use a global variable

```
7016 \newrobustcmd\useSGvar[1]{%
7017   \@ifundefined{sTeX@Gvar@#1}
7018   {\PackageError{document-structure}
7019     {The sTeX Global variable #1 is undefined}
7020     {set it with \protect\setSGvar}}
7021   \@nameuse{sTeX@Gvar@#1}}
```

(*End definition for* \useSGvar*. This function is documented on page 53.*)

\ifSGvar execute something conditionally based on the state of the global variable.

```
7022 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
7023   \@ifundefined{sTeX@Gvar@#1}
7024   {\PackageError{document-structure}
7025     {The sTeX Global variable #1 is undefined}
7026     {set it with \protect\setSGvar}}
7027   {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}
```

(*End definition for* \ifSGvar*. This function is documented on page 53.*)

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.

```
7028 ⟨*cls⟩
7029 ⟨@@=notesslides⟩
7030 \ProvidesExplClass{notesslides}{2022/02/28}{3.1.0}{notesslides Class}
7031 \RequirePackage{l3keys2e}
7032
7033 \keys_define:nn{notesslides / cls}{
7034   class   .str_set_x:N = \c__notesslides_class_str,
7035   notes   .bool_set:N  = \c__notesslides_notes_bool ,
7036   slides  .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
7037   docopt  .str_set_x:N = \c__notesslides_docopt_str,
7038   unknown .code:n      = {
7039     \PassOptionsToPackage{\CurrentOption}{document-structure}
7040     \PassOptionsToClass{\CurrentOption}{beamer}
7041     \PassOptionsToPackage{\CurrentOption}{notesslides}
7042     \PassOptionsToPackage{\CurrentOption}{stex}
7043   }
7044 }
7045 \ProcessKeysOptions{ notesslides / cls }
7046
7047 \str_if_empty:NF \c__notesslides_class_str {
7048   \PassOptionsToPackage{class=\c__notesslides_class_str}{document-structure}
7049 }
7050
7051 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{book}{
7052   \PassOptionsToPackage{defaulttopsect=part}{notesslides}
7053 }
7054 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{report}{
7055   \PassOptionsToPackage{defaulttopsect=part}{notesslides}
7056 }
7057
7058 \RequirePackage{stex}
```

```
7059  \stex_html_backend:T {
7060    \bool_set_true:N\c__notesslides_notes_bool
7061  }
7062
7063  \bool_if:NTF \c__notesslides_notes_bool {
7064    \PassOptionsToPackage{notes=true}{notesslides}
7065    \message{notesslides.cls:~Formatting~course~materials~in~notes~mode}
7066  }{
7067    \PassOptionsToPackage{notes=false}{notesslides}
7068    \message{notesslides.cls:~Formatting~course~materials~in~slides~mode}
7069  }
7070  ⟨/cls⟩
```

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

```
7071  ⟨*package⟩
7072  \ProvidesExplPackage{notesslides}{2022/02/28}{3.1.0}{notesslides Package}
7073  \RequirePackage{l3keys2e}
7074
7075  \keys_define:nn{notesslides / pkg}{
7076    topsect          .str_set_x:N  = \c__notesslides_topsect_str,
7077    defaulttopsect   .str_set_x:N  = \c__notesslides_defaulttopsec_str,
7078    notes            .bool_set:N   = \c__notesslides_notes_bool ,
7079    slides           .code:n       = { \bool_set_false:N \c__notesslides_notes_bool },
7080    sectocframes     .bool_set:N   = \c__notesslides_sectocframes_bool ,
7081    frameimages      .bool_set:N   = \c__notesslides_frameimages_bool ,
7082    fiboxed          .bool_set:N   = \c__notesslides_fiboxed_bool ,
7083    noproblems       .bool_set:N   = \c__notesslides_noproblems_bool,
7084    unknown          .code:n       = {
7085      \PassOptionsToClass{\CurrentOption}{stex}
7086      \PassOptionsToClass{\CurrentOption}{tikzinput}
7087    }
7088  }
7089  \ProcessKeysOptions{ notesslides / pkg }
7090
7091  \RequirePackage{stex}
7092  \stex_html_backend:T {
7093    \bool_set_true:N\c__notesslides_notes_bool
7094  }
7095
7096  \newif\ifnotes
7097  \bool_if:NTF \c__notesslides_notes_bool {
7098    \notestrue
7099  }{
7100    \notesfalse
7101  }
7102
```

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

```
7103  \str_if_empty:NTF \c__notesslides_topsect_str {
7104    \str_set_eq:NN \__notesslidestopsect \c__notesslides_defaulttopsec_str
7105  }{
7106    \str_set_eq:NN \__notesslidestopsect \c__notesslides_topsect_str
7107  }
7108  \PassOptionsToPackage{topsect=\__notesslidestopsect}{document-structure}
```

7109 ⟨/package⟩

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

7110 ⟨∗cls⟩
7111 `\bool_if:NTF \c__notesslides_notes_bool {`
7112 `  \str_if_empty:NT \c__notesslides_class_str {`
7113 `    \str_set:Nn \c__notesslides_class_str {article}`
7114 `  }`
7115 `  \exp_after:wN\LoadClass\exp_after:wN[\c__notesslides_docopt_str]`
7116 `    {\c__notesslides_class_str}`
7117 `}{`
7118 `  \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}`
7119 `  \newcounter{Item}`
7120 `  \newcounter{paragraph}`
7121 `  \newcounter{subparagraph}`
7122 `  \newcounter{Hfootnote}`
7123 `}`
7124 `\RequirePackage{document-structure}`

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

7125 `\RequirePackage{notesslides}`
7126 ⟨/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.

7127 ⟨∗package⟩
7128 `\bool_if:NT \c__notesslides_notes_bool {`
7129 `  \RequirePackage{a4wide}`
7130 `  \RequirePackage{marginnote}`
7131 `  \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}`
7132 `  \RequirePackage{mdframed}`
7133 `  \RequirePackage[noxcolor,noamsthm]{beamerarticle}`
7134 `  \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}`
7135 `}`
7136 `\RequirePackage{stex-tikzinput}`
7137 `\RequirePackage{etoolbox}`
7138 `\RequirePackage{amssymb}`
7139 `\RequirePackage{amsmath}`
7140 `\RequirePackage{comment}`
7141 `\RequirePackage{textcomp}`
7142 `\RequirePackage{url}`
7143 `\RequirePackage{graphicx}`
7144 `\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`.[16]

```
7145  \bool_if:NT \c__notesslides_notes_bool {
7146    \renewcommand\usetheme[2][]{\usepackage[#1]{beamernotestheme#2}}
7147  }
7148
7149
7150  \NewDocumentCommand \libusetheme {O{} m} {
7151    \bool_if:NTF \c__notesslides_notes_bool {
7152      \libusepackage[#1]{beamernotestheme#2}
7153    }{
7154      \libusepackage[#1]{beamertheme#2}
7155    }
7156  }
7157
```

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

```
7158  \newcounter{slide}
7159  \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
7160  \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.

```
7161  \bool_if:NTF \c__notesslides_notes_bool {
7162    \renewenvironment{note}{\ignorespaces}{}
7163  }{
7164    \excludecomment{note}
7165  }
```

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.

```
7166  \bool_if:NT \c__notesslides_notes_bool {
7167    \newlength{\slideframewidth}
7168    \setlength{\slideframewidth}{1.5pt}
```

frame We first define the keys.

```
7169    \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
7170      \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
7171        \bool_set_true:N #1
7172      }{
7173        \bool_set_false:N #1
7174      }
7175    }
7176    \keys_define:nn{notesslides / frame}{
7177      label                .str_set_x:N  = \l__notesslides_frame_label_str,
7178      allowframebreaks     .code:n       = {
7179        \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
7180      },
```

---

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

```
7181    allowdisplaybreaks  .code:n     = {
7182      \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
7183    },
7184    fragile            .code:n     = {
7185      \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
7186    },
7187    shrink             .code:n     = {
7188      \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
7189    },
7190    squeeze            .code:n     = {
7191      \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
7192    },
7193    t                  .code:n     = {
7194      \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
7195    },
7196    unknown   .code:n     = {}
7197  }
7198  \cs_new_protected:Nn \__notesslides_frame_args:n {
7199    \str_clear:N \l__notesslides_frame_label_str
7200    \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
7201    \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
7202    \bool_set_true:N \l__notesslides_frame_fragile_bool
7203    \bool_set_true:N \l__notesslides_frame_shrink_bool
7204    \bool_set_true:N \l__notesslides_frame_squeeze_bool
7205    \bool_set_true:N \l__notesslides_frame_t_bool
7206    \keys_set:nn { notesslides / frame }{ #1 }
7207  }
```

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

```
7208    \renewenvironment{frame}[1][]{
7209      \__notesslides_frame_args:n{#1}
7210      \sffamily
7211      \stepcounter{slide}
7212      \def\@currentlabel{\theslide}
7213      \str_if_empty:NF \l__notesslides_frame_label_str {
7214        \label{\l__notesslides_frame_label_str}
7215      }
```

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

```
7216      \def\itemize@level{outer}
7217      \def\itemize@outer{outer}
7218      \def\itemize@inner{inner}
7219      \renewcommand\newpage{\addtocounter{framenumber}{1}}
7220      %\newcommand\metakeys@show@keys[2]{\marginnote{{\scriptsize ##2}}}
7221      \renewenvironment{itemize}{
7222        \ifx\itemize@level\itemize@outer
7223          \def\itemize@label{$\rhd$}
7224        \fi
7225        \ifx\itemize@level\itemize@inner
7226          \def\itemize@label{$\scriptstyle\rhd$}
7227        \fi
7228        \begin{list}
7229        {\itemize@label}
7230        {\setlength{\labelsep}{.3em}
7231          \setlength{\labelwidth}{.5em}
```

254

```
7232        \setlength{\leftmargin}{1.5em}
7233      }
7234      \edef\itemize@level{\itemize@inner}
7235    }{
7236      \end{list}
7237    }
```

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

```
7238      \stex_html_backend:TF {
7239        \begin{stex_annotate_env}{frame}{}\vbox\bgroup
7240          \mdf@patchamsthm
7241      }{
7242        \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwid
7243      }
7244    }{
7245      \stex_html_backend:TF {
7246        \miko@slidelabel\egroup\end{stex_annotate_env}
7247      }{\medskip\miko@slidelabel\end{mdframed}}
7248    }
```

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

\frametitle

```
7249      \renewcommand{\frametitle}[1]{
7250        \stex_document_title:n { #1 }
7251        {\Large\bf\sf\color{blue}{#1}}\medskip
7252    }
7253  }
```

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

\pause   [17]

```
7254  \bool_if:NT \c__notesslides_notes_bool {
7255    \newcommand\pause{}
7256  }
```

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

nparagraph

```
7257  \bool_if:NTF \c__notesslides_notes_bool {
7258    \newenvironment{nparagraph}[1][]{\begin{sparagraph}[#1]}{\end{sparagraph}}
7259  }{
7260    \excludecomment{nparagraph}
7261  }
```

nfragment

```
7262  \bool_if:NTF \c__notesslides_notes_bool {
7263    \newenvironment{nfragment}[2][]{\begin{sfragment}[#1]{#2}}{\end{sfragment}}
7264  }{
7265    \excludecomment{nfragment}
7266  }
```

---

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

ndefinition

```
7267  \bool_if:NTF \c__notesslides_notes_bool {
7268    \newenvironment{ndefinition}[1][]{\begin{sdefinition}[#1]}{\end{sdefinition}}
7269  }{
7270    \excludecomment{ndefinition}
7271  }
```

nassertion

```
7272  \bool_if:NTF \c__notesslides_notes_bool {
7273    \newenvironment{nassertion}[1][]{\begin{sassertion}[#1]}{\end{sassertion}}
7274  }{
7275    \excludecomment{nassertion}
7276  }
```

nsproof

```
7277  \bool_if:NTF \c__notesslides_notes_bool {
7278    \newenvironment{nproof}[2][]{\begin{sproof}[#1]{#2}}{\end{sproof}}
7279  }{
7280    \excludecomment{nproof}
7281  }
```

nexample

```
7282  \bool_if:NTF \c__notesslides_notes_bool {
7283    \newenvironment{nexample}[1][]{\begin{sexample}[#1]}{\end{sexample}}
7284  }{
7285    \excludecomment{nexample}
7286  }
```

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

```
7287  \def\inputref@preskip{\smallskip}
7288  \def\inputref@postskip{\medskip}
```

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

\inputref*

```
7289  \let\orig@inputref\inputref
7290  \def\inputref{\@ifstar\ninputref\orig@inputref}
7291  \newcommand\ninputref[2][]{
7292    \bool_if:NT \c__notesslides_notes_bool {
7293      \orig@inputref[#1]{#2}
7294    }
7295  }
```

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

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⟩}`.

```
7296 \newlength{\slidelogoheight}
7297
7298 \RequirePackage{graphicx}
7299
7300 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7301 \providecommand\mhgraphics[2][]{
7302   \def\Gin@mhrepos{}\setkeys{Gin}{#1}
7303   \includegraphics[#1]{\mhpath\Gin@mhrepos{#2}}
7304 }
7305
7306
7307
7308 \bool_if:NTF \c__notesslides_notes_bool {
7309   \setlength{\slidelogoheight}{.4cm}
7310 }{
7311   \setlength{\slidelogoheight}{1cm}
7312 }
7313 \ifcsname slidelogo\endcsname\else
7314   \newsavebox{\slidelogo}
7315   \sbox{\slidelogo}{\sTeX}
7316 \fi
7317 \newrobustcmd{\setslidelogo}[2][]{
7318   \tl_if_empty:nTF{#1}{
7319     \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#2}}
7320   }{
7321     \sbox{\slidelogo}{\mhgraphics[height=\slidelogoheight,mhrepos=#1]{#2}}
7322   }
7323 }
```

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

\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.

```
7324 \def\source{Michael Kohlhase}% customize locally
7325 \newrobustcmd{\setsource}[1]{\def\source{#1}}
```

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

\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.

```
7326 \def\copyrightnotice{\footnotesize\copyright :\hspace{.3ex}{\source}}
7327 \newsavebox{\cclogo}
7328 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{stex-cc_somerights}}
7329 \newif\ifcchref\cchreffalse
7330 \AtBeginDocument{
7331   \@ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
7332 }
7333 \def\licensing{
7334   \ifcchref
7335     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
```

```
7336    \else
7337      {\usebox{\cclogo}}
7338    \fi
7339 }
7340 \newrobustcmd{\setlicensing}[2][]{
7341    \def\@url{#1}
7342    \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
7343    \ifx\@url\@empty
7344      \def\licensing{{\usebox{\cclogo}}}
7345    \else
7346      \def\licensing{
7347        \ifcchref
7348        \href{#1}{\usebox{\cclogo}}
7349        \else
7350        {\usebox{\cclogo}}
7351        \fi
7352      }
7353    \fi
7354 }
```

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

\slidelabel   Now, we set up the slide label for the article mode.[18]

```
7355 \newrobustcmd\miko@slidelabel{
7356    \vbox to \slidelogoheight{
7357      \vss\hbox to \slidewidth
7358      {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
7359    }
7360 }
```

(*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.

```
7361 \def\Gin@mhrepos{}
7362 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7363 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
7364 \newrobustcmd\frameimage[2][]{
7365    \stepcounter{slide}
7366    \bool_if:NT \c__notesslides_frameimages_bool {
7367      \def\Gin@ewidth{}\setkeys{Gin}{#1}
7368      \bool_if:NF \c__notesslides_notes_bool { \vfill }
7369      \begin{center}
7370        \bool_if:NTF \c__notesslides_fiboxed_bool {
7371          \fbox{
7372            \ifx\Gin@ewidth\@empty
7373              \ifx\Gin@mhrepos\@empty
7374                \mhgraphics[width=\slidewidth,#1]{#2}
7375              \else
7376                \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
```

---

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

258

```
7377              \fi
7378           \else% Gin@ewidth empty
7379             \ifx\Gin@mhrepos\@empty
7380               \mhgraphics[#1]{#2}
7381             \else
7382               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7383             \fi
7384           \fi% Gin@ewidth empty
7385          }
7386        }{
7387          \ifx\Gin@ewidth\@empty
7388            \ifx\Gin@mhrepos\@empty
7389              \mhgraphics[width=\slidewidth,#1]{#2}
7390            \else
7391              \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7392            \fi
7393            \ifx\Gin@mhrepos\@empty
7394              \mhgraphics[#1]{#2}
7395            \else
7396              \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7397            \fi
7398          \fi% Gin@ewidth empty
7399        }
7400      \end{center}
7401      \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
7402      \bool_if:NF \c__notesslides_notes_bool { \vfill }
7403    }
7404  } % ifmks@sty@frameimages
```

(*End definition for* \frameimage. *This function is documented on page* 56.)

38.5   Colors and Highlighting

We first specify sans serif fonts as the default.

```
7405 \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

```
7406 \AddToHook{begindocument}{
7407   \definecolor{green}{rgb}{0,.5,0}
7408   \definecolor{purple}{cmyk}{.3,1,0,.17}
7409 }
```

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.

```
7410 % \def\STpresent#1{\textcolor{blue}{#1}}
7411 \def\defemph#1{{\textcolor{magenta}{#1}}}
7412 \def\symrefemph#1{{\textcolor{cyan}{#1}}}
7413 \def\compemph#1{{\textcolor{blue}{#1}}}
7414 \def\titleemph#1{{\textcolor{blue}{#1}}}
7415 \def\__omtext_lec#1{(\textcolor{green}{#1})}
```

259

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.

```
7416 \pgfdeclareimage[width=.8em]{miko@small@dbend}{stex-dangerous-bend}
7417 \def\smalltextwarning{
7418    \pgfuseimage{miko@small@dbend}
7419    \xspace
7420 }
7421 \pgfdeclareimage[width=1.2em]{miko@dbend}{stex-dangerous-bend}
7422 \newrobustcmd\textwarning{
7423    \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
7424    \xspace
7425 }
7426 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{stex-dangerous-bend}
7427 \newrobustcmd\bigtextwarning{
7428    \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
7429    \xspace
7430 }
```

(*End definition for* \textwarning. *This function is documented on page 56.*)

```
7431 \newrobustcmd\putgraphicsat[3]{
7432    \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
7433 }
7434 \newrobustcmd\putat[2]{
7435    \begin{picture}(0,0)\put(#1){#2}\end{picture}
7436 }
```

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`.

```
7437 \stex_html_backend:F {
7438    \bool_if:NT \c__notesslides_sectocframes_bool {
7439       \str_if_eq:VnTF \__notesslidestopsect{part}{
7440          \newcounter{chapter}\counterwithin*{section}{chapter}
7441       }{
7442          \str_if_eq:VnT\__notesslidestopsect{chapter}{
7443             \newcounter{chapter}\counterwithin*{section}{chapter}
7444          }
7445       }
7446    }
7447 }
```

\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

```
7448 \def\part@prefix{}
7449 \@ifpackageloaded{document-structure}{}{
7450    \str_case:VnF \__notesslidestopsect {
7451       {part}{
```

```
7452        \int_set:Nn \l_document_structure_section_level_int {0}
7453        \def\thesection{\arabic{chapter}.\arabic{section}}
7454        \def\part@prefix{\arabic{chapter}.}
7455      }
7456    {chapter}{
7457        \int_set:Nn \l_document_structure_section_level_int {1}
7458        \def\thesection{\arabic{chapter}.\arabic{section}}
7459        \def\part@prefix{\arabic{chapter}.}
7460      }
7461  }{
7462      \int_set:Nn \l_document_structure_section_level_int {2}
7463      \def\part@prefix{}
7464    }
7465 }
7466
7467 \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 sfragment environment that choses the LaTeX sectioning macros according to \section@level.

sfragment

```
7468    \renewenvironment{sfragment}[2][]{
7469      \__document_structure_sfragment_args:n { #1 }
7470      \int_incr:N \l_document_structure_section_level_int
7471      \bool_if:NT \c__notesslides_sectocframes_bool {
7472        \stepcounter{slide}
7473        \begin{frame}[noframenumbering]
7474        \vfill\Large\centering
7475        \red{
7476          \ifcase\l_document_structure_section_level_int\or
7477            \stepcounter{part}
7478            \def\__notesslideslabel{{\omdoc@part@kw}~\Roman{part}}
7479            \def\currentsectionlevel{\omdoc@part@kw}
7480          \or
7481            \stepcounter{chapter}
7482            \def\__notesslideslabel{{\omdoc@chapter@kw}~\arabic{chapter}}
7483            \def\currentsectionlevel{\omdoc@chapter@kw}
7484          \or
7485            \stepcounter{section}
7486            \def\__notesslideslabel{\part@prefix\arabic{section}}
7487            \def\currentsectionlevel{\omdoc@section@kw}
7488          \or
7489            \stepcounter{subsection}
7490            \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
7491            \def\currentsectionlevel{\omdoc@subsection@kw}
7492          \or
7493            \stepcounter{subsubsection}
7494            \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
7495            \def\currentsectionlevel{\omdoc@subsubsection@kw}
7496          \or
7497            \stepcounter{paragraph}
7498            \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
7499            \def\currentsectionlevel{\omdoc@paragraph@kw}
```

```
7500        \else
7501          \def\__notesslideslabel{}
7502          \def\currentsectionlevel{\omdoc@paragraph@kw}
7503        \fi% end ifcase
7504        \__notesslideslabel%\sref@label@id\__notesslideslabel
7505        \quad #2%
7506      }%
7507      \vfill%
7508      \end{frame}%
7509    }
7510    \str_if_empty:NF \l__document_structure_sfragment_id_str {
7511      \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
7512    }
7513  }{}
7514 }
```

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

```
7515 \def\inserttheorembodyfont{\normalfont}
7516 %\bool_if:NF \c__notesslides_notes_bool {
7517 %  \defbeamertemplate{theorem begin}{miko}
7518 %  {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
7519 %    \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
7520 %    \inserttheorempunctuation\inserttheorembodyfont\xspace}
7521 %  \defbeamertemplate{theorem end}{miko}{}
```

and we set it as the default one.

```
7522 %  \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.

```
7523 %  \expandafter\def\csname Parent2\endcsname{}
7524 %}
7525
7526 \AddToHook{begindocument}{ % this does not work for some reasone
7527   \setbeamertemplate{theorems}[ams style]
7528 }
7529 \bool_if:NT \c__notesslides_notes_bool {
7530   \renewenvironment{columns}[1][]{%
7531     \par\noindent%
7532     \begin{minipage}%
7533     \slidewidth\centering\leavevmode%
7534   }{%
7535     \end{minipage}\par\noindent%
7536   }%
7537   \newsavebox\columnbox%
7538   \renewenvironment<>{column}[2][]{%
7539     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
7540   }{%
7541     \end{minipage}\end{lrbox}\usebox\columnbox%
7542   }%
7543 }
7544 \bool_if:NTF \c__notesslides_noproblems_bool {
7545   \newenvironment{problems}{}{}
```

```
7546  }{
7547      \excludecomment{problems}
7548  }
```

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.

```
7549  \gdef\printexcursions{}
7550  \newcommand\excursionref[2]{% label, text
7551      \bool_if:NT \c__notesslides_notes_bool {
7552          \begin{sparagraph}[title=Excursion]
7553              #2 \sref[fallback=the appendix]{#1}.
7554          \end{sparagraph}
7555      }
7556  }
7557  \newcommand\activate@excursion[2][]{
7558      \gappto\printexcursions{\inputref[#1]{#2}}
7559  }
7560  \newcommand\excursion[4][]{% repos, label, path, text
7561      \bool_if:NT \c__notesslides_notes_bool {
7562          \activate@excursion[#1]{#3}\excursionref{#2}{#4}
7563      }
7564  }
```

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

\excursiongroup

```
7565  \keys_define:nn{notesslides / excursiongroup }{
7566      id          .str_set_x:N  = \l__notesslides_excursion_id_str,
7567      intro       .tl_set:N     = \l__notesslides_excursion_intro_tl,
7568      mhrepos     .str_set_x:N  = \l__notesslides_excursion_mhrepos_str
7569  }
7570  \cs_new_protected:Nn \__notesslides_excursion_args:n {
7571      \tl_clear:N \l__notesslides_excursion_intro_tl
7572      \str_clear:N \l__notesslides_excursion_id_str
7573      \str_clear:N \l__notesslides_excursion_mhrepos_str
7574      \keys_set:nn {notesslides / excursiongroup }{ #1 }
7575  }
7576  \newcommand\excursiongroup[1][]{
7577      \__notesslides_excursion_args:n{ #1 }
7578      \ifdefempty\printexcursions{}% only if there are excursions
7579      {\begin{note}
7580          \begin{sfragment}[#1]{Excursions}%
7581              \ifdefempty\l__notesslides_excursion_intro_tl{}{
7582                  \inputref[\l__notesslides_excursion_mhrepos_str]{
7583                      \l__notesslides_excursion_intro_tl
7584                  }
7585              }
7586              \printexcursions%
7587          \end{sfragment}
7588      \end{note}}
```

```
7589  }
7590  \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
7591 ⟨/package⟩
```

(*End definition for* \excursiongroup. *This function is documented on page* *57.*)

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.

```
7592  ⟨∗package⟩
7593  ⟨@@=problems⟩
7594  \ProvidesExplPackage{problem}{2022/02/26}{3.0.1}{Semantic Markup for Problems}
7595  \RequirePackage{l3keys2e,stex}
7596
7597  \keys_define:nn { problem / pkg }{
7598    notes      .default:n   = { true },
7599    notes      .bool_set:N  = \c__problems_notes_bool,
7600    gnotes     .default:n   = { true },
7601    gnotes     .bool_set:N  = \c__problems_gnotes_bool,
7602    hints      .default:n   = { true },
7603    hints      .bool_set:N  = \c__problems_hints_bool,
7604    solutions .default:n   = { true },
7605    solutions .bool_set:N  = \c__problems_solutions_bool,
7606    pts        .default:n   = { true },
7607    pts        .bool_set:N  = \c__problems_pts_bool,
7608    min        .default:n   = { true },
7609    min        .bool_set:N  = \c__problems_min_bool,
7610    boxed      .default:n   = { true },
7611    boxed      .bool_set:N  = \c__problems_boxed_bool,
7612    unknown    .code:n      = {}
7613  }
7614  \newif\ifsolutions
7615
7616  \ProcessKeysOptions{ problem / pkg }
7617  \bool_if:NTF \c__problems_solutions_bool {
7618    \solutionstrue
7619  }{
7620    \solutionsfalse
7621  }
```

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

```
7622  \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.

```
7623  \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.

```
7624  \def\prob@problem@kw{Problem}
7625  \def\prob@solution@kw{Solution}
7626  \def\prob@hint@kw{Hint}
7627  \def\prob@note@kw{Note}
7628  \def\prob@gnote@kw{Grading}
7629  \def\prob@pt@kw{pt}
7630  \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
7631  \AddToHook{begindocument}{
7632    \ltx@ifpackageloaded{babel}{
7633      \makeatletter
7634      \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7635      \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7636        \input{problem-ngerman.ldf}
7637      }
7638      \clist_if_in:NnT \l_tmpa_clist {finnish}{
7639        \input{problem-finnish.ldf}
7640      }
7641      \clist_if_in:NnT \l_tmpa_clist {french}{
7642        \input{problem-french.ldf}
7643      }
7644      \clist_if_in:NnT \l_tmpa_clist {russian}{
7645        \input{problem-russian.ldf}
7646      }
7647      \makeatother
7648    }{}
7649  }
```

39.2   Problems and Solutions

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

```
7650  \keys_define:nn{ problem / problem }{
7651    id      .str_set_x:N  = \l__problems_prob_id_str,
7652    pts     .tl_set:N     = \l__problems_prob_pts_tl,
7653    min     .tl_set:N     = \l__problems_prob_min_tl,
7654    title   .tl_set:N     = \l__problems_prob_title_tl,
7655    type    .tl_set:N     = \l__problems_prob_type_tl,
7656    imports .tl_set:N     = \l__problems_prob_imports_tl,
7657    name    .str_set_x:N  = \l__problems_prob_name_str,
7658    refnum  .int_set:N    = \l__problems_prob_refnum_int
```

```
7659 }
7660 \cs_new_protected:Nn \__problems_prob_args:n {
7661   \str_clear:N \l__problems_prob_id_str
7662   \str_clear:N \l__problems_prob_name_str
7663   \tl_clear:N \l__problems_prob_pts_tl
7664   \tl_clear:N \l__problems_prob_min_tl
7665   \tl_clear:N \l__problems_prob_title_tl
7666   \tl_clear:N \l__problems_prob_type_tl
7667   \tl_clear:N \l__problems_prob_imports_tl
7668   \int_zero_new:N \l__problems_prob_refnum_int
7669   \keys_set:nn { problem / problem }{ #1 }
7670   \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
7671     \let\l__problems_prob_refnum_int\undefined
7672   }
7673 }
```

Then we set up a counter for problems.

\numberproblemsin

```
7674 \newcounter{problem}[section]
7675 \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.

```
7676 \newcommand\prob@label[1]{\thesection.#1}
```

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

\prob@number We consolidate the problem number into a reusable internal macro

```
7677 \newcommand\prob@number{
7678   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
7679     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
7680   }{
7681     \int_if_exist:NTF \l__problems_prob_refnum_int {
7682       \prob@label{\int_use:N \l__problems_prob_refnum_int }
7683     }{
7684       \prob@label\theproblem
7685     }
7686   }
7687 }
```

(*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.

```
7688 \newcommand\prob@title[3]{%
7689   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
7690     #2 \l__problems_inclprob_title_tl #3
7691   }{
7692     \tl_if_exist:NTF \l__problems_prob_title_tl {
7693       #2 \l__problems_prob_title_tl #3
7694     }{
7695       #1
```

267

```
7696        }
7697      }
7698   }
```

(*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.

```
7699   \def\prob@heading{
7700     {\prob@problem@kw}\ \prob@number\prob@title{~}{~(}{)}\strut}
7701     %\sref@label@id{\prob@problem@kw~\prob@number}{}
7702   }
```

(*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

```
7703   \newenvironment{sproblem}[1][]{
7704     \__problems_prob_args:n{#1}%\sref@target%
7705     \@in@omtexttrue% we are in a statement (for inline definitions)
7706     \stepcounter{problem}\record@problem
7707     \def\current@section@level{\prob@problem@kw}
7708
7709     \str_if_empty:NT \l__problems_prob_name_str {
7710       \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
7711       \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
7712       \seq_get_left:NN \l_tmpa_seq \l__problems_prob_name_str
7713     }
7714
7715     \stex_if_do_html:T{
7716       \tl_if_empty:NF \l__problems_prob_title_tl {
7717         \exp_args:No \stex_document_title:n \l__problems_prob_title_tl
7718       }
7719     }
7720
7721     \exp_args:Nno\stex_module_setup:nn{type=problem}\l__problems_prob_name_str
7722
7723     \stex_reactivate_macro:N \STEXexport
7724     \stex_reactivate_macro:N \importmodule
7725     \stex_reactivate_macro:N \symdecl
7726     \stex_reactivate_macro:N \notation
7727     \stex_reactivate_macro:N \symdef
7728
7729     \stex_if_do_html:T{
7730       \begin{stex_annotate_env} {problem} {
7731         \l_stex_module_ns_str ? \l_stex_module_name_str
7732       }
7733
7734       \stex_annotate_invisible:nnn{header}{} {
7735         \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
```

```
7736        \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
7737        \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
7738          \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
7739        }
7740      }
7741    }
7742
7743    \stex_csl_to_imports:No \importmodule \l__problems_prob_imports_tl
7744
7745
7746    \tl_if_exist:NTF \l__problems_inclprob_type_tl {
7747      \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
7748    }{
7749      \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
7750    }
7751    \str_if_exist:NTF \l__problems_inclprob_id_str {
7752      \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
7753    }{
7754      \str_set_eq:NN \sproblemid \l__problems_prob_id_str
7755    }
7756
7757
7758    \stex_if_smsmode:F {
7759      \clist_set:No \l_tmpa_clist \sproblemtype
7760      \tl_clear:N \l_tmpa_tl
7761      \clist_map_inline:Nn \l_tmpa_clist {
7762        \tl_if_exist:cT {__problems_sproblem_##1_start:}{
7763          \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_start:}}
7764        }
7765      }
7766      \tl_if_empty:NTF \l_tmpa_tl {
7767        \__problems_sproblem_start:
7768      }{
7769        \l_tmpa_tl
7770      }
7771    }
7772    \stex_ref_new_doc_target:n \sproblemid
7773    \stex_smsmode_do:
7774  }{
7775    \__stex_modules_end_module:
7776    \stex_if_smsmode:F{
7777      \clist_set:No \l_tmpa_clist \sproblemtype
7778      \tl_clear:N \l_tmpa_tl
7779      \clist_map_inline:Nn \l_tmpa_clist {
7780        \tl_if_exist:cT {__problems_sproblem_##1_end:}{
7781          \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_end:}}
7782        }
7783      }
7784      \tl_if_empty:NTF \l_tmpa_tl {
7785        \__problems_sproblem_end:
7786      }{
7787        \l_tmpa_tl
7788      }
7789    }
```

269

```
7790    \stex_if_do_html:T{
7791       \end{stex_annotate_env}
7792    }
7793
7794    \smallskip
7795 }
7796
7797 \seq_put_right:Nx\g_stex_smsmode_allowedenvs_seq{\tl_to_str:n{sproblem}}
7798
7799
7800
7801 \cs_new_protected:Nn \__problems_sproblem_start: {
7802    \par\noindent\textbf\prob@heading\show@pts\show@min\\\ignorespacesandpars
7803 }
7804 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
7805
7806 \newcommand\stexpatchproblem[3][] {
7807    \str_set:Nx \l_tmpa_str{ #1 }
7808    \str_if_empty:NTF \l_tmpa_str {
7809       \tl_set:Nn \__problems_sproblem_start: { #2 }
7810       \tl_set:Nn \__problems_sproblem_end: { #3 }
7811    }{
7812       \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
7813       \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
7814    }
7815 }
7816
7817
7818 \bool_if:NT \c__problems_boxed_bool {
7819    \surroundwithmdframed{problem}
7820 }
```

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

```
7821 \def\record@problem{
7822    \protected@write\@auxout{}
7823    {
7824       \string\@problem{\prob@number}
7825       {
7826          \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7827             \l__problems_inclprob_pts_tl
7828          }{
7829             \l__problems_prob_pts_tl
7830          }
7831       }%
7832       {
7833          \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7834             \l__problems_inclprob_min_tl
7835          }{
7836             \l__problems_prob_min_tl
7837          }
7838       }
7839    }
7840 }
```

(*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).

```
7841 \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.

```
7842 \keys_define:nn { problem / solution }{
7843   id              .str_set_x:N  = \l__problems_solution_id_str ,
7844   for             .tl_set:N     = \l__problems_solution_for_tl ,
7845   height          .dim_set:N    = \l__problems_solution_height_dim ,
7846   creators        .clist_set:N  = \l__problems_solution_creators_clist ,
7847   contributors    .clist_set:N  = \l__problems_solution_contributors_clist ,
7848   srccite         .tl_set:N     = \l__problems_solution_srccite_tl
7849 }
7850 \cs_new_protected:Nn \__problems_solution_args:n {
7851   \str_clear:N \l__problems_solution_id_str
7852   \tl_clear:N \l__problems_solution_for_tl
7853   \tl_clear:N \l__problems_solution_srccite_tl
7854   \clist_clear:N \l__problems_solution_creators_clist
7855   \clist_clear:N \l__problems_solution_contributors_clist
7856   \dim_zero:N \l__problems_solution_height_dim
7857   \keys_set:nn { problem / solution }{ #1 }
7858 }
```

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

```
7859 \newcommand\@startsolution[1][]{
7860   \__problems_solution_args:n { #1 }
7861   \@in@omtexttrue% we are in a statement.
7862   \bool_if:NF \c__problems_boxed_bool { \hrule }
7863   \smallskip\noindent
7864   {\textbf\prob@solution@kw :\enspace}
7865   \begin{small}
7866   \def\current@section@level{\prob@solution@kw}
7867   \ignorespacesandpars
7868 }
```

\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.

```
7869 \box_new:N \l__problems_solution_box
7870 \newenvironment{solution}[1][]{
7871   \stex_html_backend:TF{
7872     \stex_if_do_html:T{
7873       \begin{stex_annotate_env}{solution}{}
7874     }
7875   }{
7876     \setbox\l__problems_solution_box\vbox\bgroup
7877       \par\smallskip\hrule\smallskip
7878       \noindent\textbf{Solution:}~
7879   }
7880 }{
7881   \stex_html_backend:TF{
```

271

```
7882    \stex_if_do_html:T{
7883      \end{stex_annotate_env}
7884    }
7885  }{
7886    \smallskip\hrule
7887    \egroup
7888    \bool_if:NT \c__problems_solutions_bool {
7889      \box\l__problems_solution_box
7890    }
7891  }
7892 }
7893
7894 \newcommand\startsolutions{
7895   \bool_set_true:N \c__problems_solutions_bool
7896 %  \specialcomment{solution}{\@startsolution}{
7897 %    \bool_if:NF \c__problems_boxed_bool {
7898 %      \hrule\medskip
7899 %    }
7900 %    \end{small}%
7901 %  }
7902 %  \bool_if:NT \c__problems_boxed_bool {
7903 %    \surroundwithmdframed{solution}
7904 %  }
7905 }
```

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

```
7906 \newcommand\stopsolutions{\bool_set_false:N \c__problems_solutions_bool}%\excludecomment{sol
```

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

so it only remains to start/stop solutions depending on what option was specified.

```
7907 \ifsolutions
7908   \startsolutions
7909 \else
7910   \stopsolutions
7911 \fi
```

exnote

```
7912 \bool_if:NTF \c__problems_notes_bool {
7913   \newenvironment{exnote}[1][]{
7914     \par\smallskip\hrule\smallskip
7915     \noindent\textbf{\prob@note@kw :~ }\small
7916   }{
7917     \smallskip\hrule
7918   }
7919 }{
7920   \excludecomment{exnote}
7921 }
```

hint

```
7922 \bool_if:NTF \c__problems_notes_bool {
7923   \newenvironment{hint}[1][]{
7924     \par\smallskip\hrule\smallskip
```

```
7925    \noindent\textbf{\prob@hint@kw :~ }\small
7926  }{
7927    \smallskip\hrule
7928  }
7929  \newenvironment{exhint}[1][]{
7930    \par\smallskip\hrule\smallskip
7931    \noindent\textbf{\prob@hint@kw :~ }\small
7932  }{
7933    \smallskip\hrule
7934  }
7935  }{
7936  \excludecomment{hint}
7937  \excludecomment{exhint}
7938  }
```

gnote

```
7939  \bool_if:NTF \c__problems_notes_bool {
7940    \newenvironment{gnote}[1][]{
7941    \par\smallskip\hrule\smallskip
7942    \noindent\textbf{\prob@gnote@kw :~ }\small
7943  }{
7944    \smallskip\hrule
7945  }
7946  }{
7947  \excludecomment{gnote}
7948  }
```

39.3   Multiple Choice Blocks

mcb   [19]

```
7949  \newenvironment{mcb}{
7950    \begin{enumerate}
7951  }{
7952    \end{enumerate}
7953  }
```

we define the keys for the mcc macro

```
7954  \cs_new_protected:Nn \__problems_do_yes_param:Nn {
7955    \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
7956      \bool_set_true:N #1
7957    }{
7958      \bool_set_false:N #1
7959    }
7960  }
7961  \keys_define:nn { problem / mcc }{
7962    id        .str_set_x:N  = \l__problems_mcc_id_str ,
7963    feedback  .tl_set:N     = \l__problems_mcc_feedback_tl ,
7964    T         .default:n    = { false } ,
7965    T         .bool_set:N   = \l__problems_mcc_t_bool ,
7966    F         .default:n    = { false } ,
7967    F         .bool_set:N   = \l__problems_mcc_f_bool ,
```

---

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

```
7968    Ttext      .tl_set:N    = \l__problems_mcc_Ttext_str ,
7969    Ftext      .tl_set:N    = \l__problems_mcc_Ftext_str
7970  }
7971  \cs_new_protected:Nn \l__problems_mcc_args:n {
7972    \str_clear:N \l__problems_mcc_id_str
7973    \tl_clear:N \l__problems_mcc_feedback_tl
7974    \bool_set_false:N \l__problems_mcc_t_bool
7975    \bool_set_false:N \l__problems_mcc_f_bool
7976    \tl_clear:N \l__problems_mcc_Ttext_tl
7977    \tl_clear:N \l__problems_mcc_Ftext_tl
7978    \str_clear:N \l__problems_mcc_id_str
7979    \keys_set:nn { problem / mcc }{ #1 }
7980  }
```

\mcc

```
7981  \def\mccTrueText{\textbf{(true)~}}
7982  \def\mccFalseText{\textbf{(false)~}}
7983  \newcommand\mcc[2][]{
7984    \l__problems_mcc_args:n{ #1 }
7985    \item[$\Box$] #2
7986    \ifsolutions
7987      \\
7988      \bool_if:NT \l__problems_mcc_t_bool {
7989        \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccTrueText\l__problems_mcc_Ttext_tl
7990      }
7991      \bool_if:NT \l__problems_mcc_f_bool {
7992        \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccFalseText\l__problems_mcc_Ftext_tl
7993      }
7994      \tl_if_empty:NF \l__problems_mcc_feedback_tl {
7995        \emph{(\l__problems_mcc_feedback_tl)}
7996      }
7997    \fi
7998  } %solutions
```

(*End definition for* \mcc. *This function is documented on page 59.*)

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.

```
7999
8000  \keys_define:nn{ problem / inclproblem }{
8001    id       .str_set_x:N  = \l__problems_inclprob_id_str,
8002    pts      .tl_set:N     = \l__problems_inclprob_pts_tl,
8003    min      .tl_set:N     = \l__problems_inclprob_min_tl,
8004    title    .tl_set:N     = \l__problems_inclprob_title_tl,
8005    refnum   .int_set:N    = \l__problems_inclprob_refnum_int,
8006    type     .tl_set:N     = \l__problems_inclprob_type_tl,
8007    mhrepos  .str_set_x:N  = \l__problems_inclprob_mhrepos_str
8008  }
8009  \cs_new_protected:Nn \__problems_inclprob_args:n {
8010    \str_clear:N \l__problems_prob_id_str
```

```
8011    \tl_clear:N \l__problems_inclprob_pts_tl
8012    \tl_clear:N \l__problems_inclprob_min_tl
8013    \tl_clear:N \l__problems_inclprob_title_tl
8014    \tl_clear:N \l__problems_inclprob_type_tl
8015    \int_zero_new:N \l__problems_inclprob_refnum_int
8016    \str_clear:N \l__problems_inclprob_mhrepos_str
8017    \keys_set:nn { problem / inclproblem }{ #1 }
8018    \tl_if_empty:NT \l__problems_inclprob_pts_tl {
8019      \let\l__problems_inclprob_pts_tl\undefined
8020    }
8021    \tl_if_empty:NT \l__problems_inclprob_min_tl {
8022      \let\l__problems_inclprob_min_tl\undefined
8023    }
8024    \tl_if_empty:NT \l__problems_inclprob_title_tl {
8025      \let\l__problems_inclprob_title_tl\undefined
8026    }
8027    \tl_if_empty:NT \l__problems_inclprob_type_tl {
8028      \let\l__problems_inclprob_type_tl\undefined
8029    }
8030    \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
8031      \let\l__problems_inclprob_refnum_int\undefined
8032    }
8033  }
8034
8035  \cs_new_protected:Nn \__problems_inclprob_clear: {
8036    \let\l__problems_inclprob_id_str\undefined
8037    \let\l__problems_inclprob_pts_tl\undefined
8038    \let\l__problems_inclprob_min_tl\undefined
8039    \let\l__problems_inclprob_title_tl\undefined
8040    \let\l__problems_inclprob_type_tl\undefined
8041    \let\l__problems_inclprob_refnum_int\undefined
8042    \let\l__problems_inclprob_mhrepos_str\undefined
8043  }
8044  \__problems_inclprob_clear:
8045
8046  \newcommand\includeproblem[2][]{
8047    \__problems_inclprob_args:n{ #1 }
8048    \exp_args:No \stex_in_repository:nn\l__problems_inclprob_mhrepos_str{
8049      \stex_html_backend:TF {
8050        \str_clear:N \l_tmpa_str
8051        \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
8052          \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
8053        }
8054        \stex_annotate_invisible:nnn{includeproblem}{
8055          \l_tmpa_str / #2
8056        }{}
8057      }{
8058        \begingroup
8059          \inputreftrue
8060          \tl_if_empty:nTF{ ##1 }{
8061            \input{#2}
8062          }{
8063            \input{ \c_stex_mathhub_str / ##1 / source / #2 }
8064          }
```

```
8065        \endgroup
8066      }
8067    }
8068    \__problems_inclprob_clear:
8069  }
```

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

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.

```
8070  \AddToHook{enddocument}{
8071    \bool_if:NT \c__problems_pts_bool {
8072      \message{Total:~\arabic{pts}~points}
8073    }
8074    \bool_if:NT \c__problems_min_bool {
8075      \message{Total:~\arabic{min}~minutes}
8076    }
8077  }
```

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

```
8078  \def\pts#1{
8079    \bool_if:NT \c__problems_pts_bool {
8080      \marginpar{#1~\prob@pt@kw}
8081    }
8082  }
8083  \def\min#1{
8084    \bool_if:NT \c__problems_min_bool {
8085      \marginpar{#1~\prob@min@kw}
8086    }
8087  }
```

`\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.

```
8088  \newcounter{pts}
8089  \def\show@pts{
8090    \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
8091      \bool_if:NT \c__problems_pts_bool {
8092        \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
8093        \addtocounter{pts}{\l__problems_inclprob_pts_tl}
8094      }
8095    }{
8096      \tl_if_exist:NT \l__problems_prob_pts_tl {
8097        \bool_if:NT \c__problems_pts_bool {
8098          \tl_if_empty:NT\l__problems_prob_pts_tl{
8099            \tl_set:Nn \l__problems_prob_pts_tl {0}
8100          }
8101          \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
8102          \addtocounter{pts}{\l__problems_prob_pts_tl}
8103        }
8104      }
```

```
8105        }
8106      }
```

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

and now the same for the minutes

`\show@min`

```
8107  \newcounter{min}
8108  \def\show@min{
8109    \tl_if_exist:NTF \l__problems_inclprob_min_tl {
8110      \bool_if:NT \c__problems_min_bool {
8111        \marginpar{\l__problems_inclprob_pts_tl\ min}
8112        \addtocounter{min}{\l__problems_inclprob_min_tl}
8113      }
8114    }{
8115      \tl_if_exist:NT \l__problems_prob_min_tl {
8116        \bool_if:NT \c__problems_min_bool {
8117          \tl_if_empty:NT\l__problems_prob_min_tl{
8118            \tl_set:Nn \l__problems_prob_min_tl {0}
8119          }
8120          \marginpar{\l__problems_prob_min_tl\ min}
8121          \addtocounter{min}{\l__problems_prob_min_tl}
8122        }
8123      }
8124    }
8125  }
8126  ⟨/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.

```
8127  ⟨∗package⟩
8128  \ProvidesExplPackage{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
8129  \RequirePackage{l3keys2e}
8130
8131  \newif\iftest\testfalse
8132  \DeclareOption{test}{\testtrue}
8133  \newif\ifmultiple\multiplefalse
8134  \DeclareOption{multiple}{\multipletrue}
8135  \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
8136  \ProcessOptions
```

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

```
8137  \RequirePackage{keyval}[1997/11/10]
8138  \RequirePackage{problem}
```

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

```
8139  \newcommand\hwexam@assignment@kw{Assignment}
8140  \newcommand\hwexam@given@kw{Given}
8141  \newcommand\hwexam@due@kw{Due}
8142  \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
8143  blank~for~extra~space}
8144  \def\hwexam@minutes@kw{minutes}
8145  \newcommand\correction@probs@kw{prob.}
8146  \newcommand\correction@pts@kw{total}
8147  \newcommand\correction@reached@kw{reached}
8148  \newcommand\correction@sum@kw{Sum}
8149  \newcommand\correction@grade@kw{grade}
8150  \newcommand\correction@forgrading@kw{To~be~used~for~grading,~do~not~write~here}
```

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

For the other languages, we set up triggers

```
8151 \AddToHook{begindocument}{
8152 \ltx@ifpackageloaded{babel}{
8153 \makeatletter
8154 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
8155 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
8156    \input{hwexam-ngerman.ldf}
8157 }
8158 \clist_if_in:NnT \l_tmpa_clist {finnish}{
8159    \input{hwexam-finnish.ldf}
8160 }
8161 \clist_if_in:NnT \l_tmpa_clist {french}{
8162    \input{hwexam-french.ldf}
8163 }
8164 \clist_if_in:NnT \l_tmpa_clist {russian}{
8165    \input{hwexam-russian.ldf}
8166 }
8167 \makeatother
8168 }{}
8169 }
8170
```

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.

```
8171 \newcounter{assignment}
8172 %\numberproblemsin{assignment}
```

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

```
8173 \keys_define:nn { hwexam / assignment } {
8174 id   .str_set_x:N = \l_@@_assign_id_str,
8175 number  .int_set:N = \l_@@_assign_number_int,
8176 title  .tl_set:N  = \l_@@_assign_title_tl,
8177 type  .tl_set:N  = \l_@@_assign_type_tl,
8178 given .tl_set:N  = \l_@@_assign_given_tl,
8179 due .tl_set:N  = \l_@@_assign_due_tl,
8180 loadmodules .code:n  = {
8181 \bool_set_true:N \l_@@_assign_loadmodules_bool
8182 }
8183 }
8184 \cs_new_protected:Nn \_@@_assignment_args:n {
8185 \str_clear:N \l_@@_assign_id_str
8186 \int_set:Nn \l_@@_assign_number_int {-1}
8187 \tl_clear:N \l_@@_assign_title_tl
8188 \tl_clear:N \l_@@_assign_type_tl
8189 \tl_clear:N \l_@@_assign_given_tl
8190 \tl_clear:N \l_@@_assign_due_tl
8191 \bool_set_false:N \l_@@_assign_loadmodules_bool
8192 \keys_set:nn { hwexam / assignment }{ #1 }
8193 }
```

279

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.

```
8194  \newcommand\given@due[2]{
8195  \bool_lazy_all:nF {
8196  {\tl_if_empty_p:V \l_@@_inclassign_given_tl}
8197  {\tl_if_empty_p:V \l_@@_assign_given_tl}
8198  {\tl_if_empty_p:V \l_@@_inclassign_due_tl}
8199  {\tl_if_empty_p:V \l_@@_assign_due_tl}
8200  }{ #1 }
8201
8202  \tl_if_empty:NTF \l_@@_inclassign_given_tl {
8203  \tl_if_empty:NF \l_@@_assign_given_tl {
8204  \hwexam@given@kw\xspace\l_@@_assign_given_tl
8205  }
8206  }{
8207  \hwexam@given@kw\xspace\l_@@_inclassign_given_tl
8208  }
8209
8210  \bool_lazy_or:nnF {
8211  \bool_lazy_and_p:nn {
8212  \tl_if_empty_p:V \l_@@_inclassign_due_tl
8213  }{
8214  \tl_if_empty_p:V \l_@@_assign_due_tl
8215  }
8216  }{
8217  \bool_lazy_and_p:nn {
8218  \tl_if_empty_p:V \l_@@_inclassign_due_tl
8219  }{
8220  \tl_if_empty_p:V \l_@@_assign_due_tl
8221  }
8222  }{ ,~ }
8223
8224  \tl_if_empty:NTF \l_@@_inclassign_due_tl {
8225  \tl_if_empty:NF \l_@@_assign_due_tl {
8226  \hwexam@due@kw\xspace \l_@@_assign_due_tl
8227  }
8228  }{
8229  \hwexam@due@kw\xspace \l_@@_inclassign_due_tl
8230  }
8231
8232  \bool_lazy_all:nF {
8233  { \tl_if_empty_p:V \l_@@_inclassign_given_tl }
8234  { \tl_if_empty_p:V \l_@@_assign_given_tl }
8235  { \tl_if_empty_p:V \l_@@_inclassign_due_tl }
8236  { \tl_if_empty_p:V \l_@@_assign_due_tl }
8237  }{ #2 }
8238  }
```

\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.

```
8239  \newcommand\assignment@title[3]{
8240  \tl_if_empty:NTF \l_@@_inclassign_title_tl {
8241  \tl_if_empty:NTF \l_@@_assign_title_tl {
8242  #1
8243  }{
8244  #2\l_@@_assign_title_tl#3
8245  }
8246  }{
8247  #2\l_@@_inclassign_title_tl#3
8248  }
8249  }
```

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

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

```
8250  \newcommand\assignment@number{
8251  \int_compare:nNnTF \l_@@_inclassign_number_int = {-1} {
8252  \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8253  \arabic{assignment}
8254  } {
8255  \int_use:N \l_@@_assign_number_int
8256  }
8257  }{
8258  \int_use:N \l_@@_inclassign_number_int
8259  }
8260  }
```

(*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.

```
8261  \newenvironment{assignment}[1][]{
8262  \_@@_assignment_args:n { #1 }
8263  %\sref@target
8264  \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8265  \global\stepcounter{assignment}
8266  }{
8267  \global\setcounter{assignment}{\int_use:N\l_@@_assign_number_int}
8268  }
8269  \setcounter{problem}{0}
8270  \renewcommand\prob@label[1]{\assignment@number.##1}
8271  \def\current@section@level{\document@hwexamtype}
8272  %\sref@label@id{\document@hwexamtype \thesection}
8273  \begin{@assignment}
8274  }{
8275  \end{@assignment}
8276  }
```

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

```
8277  \def\ass@title{
8278  {\protect\document@hwexamtype}~\arabic{assignment}
8279  \assignment@title{}{\;(}{)\;} -- \given@due{}{}
8280  }
8281  \ifmultiple
8282  \newenvironment{@assignment}{
8283  \bool_if:NTF \l_@@_assign_loadmodules_bool {
8284  \begin{sfragment}[loadmodules]{\ass@title}
8285  }{
8286  \begin{sfragment}{\ass@title}
8287  }
8288  }{
8289  \end{sfragment}
8290  }
```

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

```
8291  \else
8292  \newenvironment{@assignment}{
8293  \begin{center}\bf
8294  \Large\@title\strut\\
8295  \document@hwexamtype~\arabic{assignment}\assignment@title{\;}{:\;}{\\}
8296  \large\given@due{--\;}{\;--}
8297  \end{center}
8298  }{}
8299  \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.

```
8300  \keys_define:nn { hwexam / inclassignment } {
8301  %id   .str_set_x:N = \l_@@_assign_id_str,
8302  number   .int_set:N  = \l_@@_inclassign_number_int,
8303  title  .tl_set:N  = \l_@@_inclassign_title_tl,
8304  type  .tl_set:N  = \l_@@_inclassign_type_tl,
8305  given .tl_set:N  = \l_@@_inclassign_given_tl,
8306  due .tl_set:N  = \l_@@_inclassign_due_tl,
8307  mhrepos   .str_set_x:N = \l_@@_inclassign_mhrepos_str
8308  }
8309  \cs_new_protected:Nn \_@@_inclassignment_args:n {
8310  \int_set:Nn \l_@@_inclassign_number_int {-1}
8311  \tl_clear:N \l_@@_inclassign_title_tl
8312  \tl_clear:N \l_@@_inclassign_type_tl
8313  \tl_clear:N \l_@@_inclassign_given_tl
8314  \tl_clear:N \l_@@_inclassign_due_tl
8315  \str_clear:N \l_@@_inclassign_mhrepos_str
8316  \keys_set:nn { hwexam / inclassignment }{ #1 }
8317  }
8318  \_@@_inclassignment_args:n {}
8319
8320  \newcommand\inputassignment[2][]{
```

```
8321  \_@@_inclassignment_args:n { #1 }
8322  \str_if_empty:NTF \l_@@_inclassign_mhrepos_str {
8323  \input{#2}
8324  }{
8325  \stex_in_repository:nn{\l_@@_inclassign_mhrepos_str}{
8326  \input{\mhpath{\l_@@_inclassign_mhrepos_str}{#2}}}
8327  }
8328  }
8329  \_@@_inclassignment_args:n {}
8330  }
8331  \newcommand\includeassignment[2][]{
8332  \newpage
8333  \inputassignment[#1]{#2}
8334  }
```

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

40.4 Typesetting Exams

\quizheading

```
8335  \ExplSyntaxOff
8336  \newcommand\quizheading[1]{%
8337  \def\@tas{#1}%
8338  \large\noindent NAME: \hspace{8cm}  MAILBOX:\\[2ex]%
8339  \ifx\@tas\@empty\else%
8340  \noindent TA:~\@for\@I:=\@tas\do{{\Large$\Box$}\@I\hspace*{1em}}\\[2ex]%
8341  \fi%
8342  }
8343  \ExplSyntaxOn
```

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

\testheading

```
8344
8345  \def\hwexamheader{\input{hwexam-default.header}}
8346
8347  \def\hwexamminutes{
8348  \tl_if_empty:NTF \testheading@duration {
8349  {\testheading@min}~\hwexam@minutes@kw
8350  }{
8351  \testheading@duration
8352  }
8353  }
8354
8355  \keys_define:nn { hwexam / testheading } {
8356  min  .tl_set:N  = \testheading@min,
8357  duration .tl_set:N  = \testheading@duration,
8358  reqpts .tl_set:N  = \testheading@reqpts,
8359  tools .tl_set:N  = \testheading@tools
8360  }
8361  \cs_new_protected:Nn \_@@_testheading_args:n {
8362  \tl_clear:N \testheading@min
8363  \tl_clear:N \testheading@duration
```

```
8364    \tl_clear:N \testheading@reqpts
8365    \tl_clear:N \testheading@tools
8366    \keys_set:nn { hwexam / testheading }{ #1 }
8367    }
8368    \newenvironment{testheading}[1][]{
8369    \_@@_testheading_args:n{ #1 }
8370    \newcount\check@time\check@time=\testheading@min
8371    \advance\check@time by -\theassignment@totalmin
8372    \newif\if@bonuspoints
8373    \tl_if_empty:NTF \testheading@reqpts {
8374    \@bonuspointsfalse
8375    }{
8376    \newcount\bonus@pts
8377    \bonus@pts=\theassignment@totalpts
8378    \advance\bonus@pts by -\testheading@reqpts
8379    \edef\bonus@pts{\the\bonus@pts}
8380    \@bonuspointstrue
8381    }
8382    \edef\check@time{\the\check@time}
8383
8384    \makeatletter\hwexamheader\makeatother
8385    }{
8386    \newpage
8387    }
```

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

\testspace

```
8388    \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}
```

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

\testnewpage

```
8389    \newcommand\testnewpage{\iftest\newpage\fi}
```

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

\testemptypage

```
8390    \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.

```
8391    ⟨@@=problems⟩
8392    \renewcommand\@problem[3]{
8393    \stepcounter{assignment@probs}
8394    \def\__problemspts{#2}
8395    \ifx\__problemspts\@empty\else
8396    \addtocounter{assignment@totalpts}{#2}
8397    \fi
8398    \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\f
8399    \xdef\correction@probs{\correction@probs & #1}%
8400    \xdef\correction@pts{\correction@pts & #2}
8401    \xdef\correction@reached{\correction@reached &}
```

284

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

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

```
8404 \newcounter{assignment@probs}
8405 \newcounter{assignment@totalpts}
8406 \newcounter{assignment@totalmin}
8407 \def\correction@probs{\correction@probs@kw}
8408 \def\correction@pts{\correction@pts@kw}
8409 \def\correction@reached{\correction@reached@kw}
8410 \stepcounter{assignment@probs}
8411 \newcommand\correction@table{
8412 \resizebox{\textwidth}{!}{%
8413 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
8414 &\multicolumn{\theassignment@probs}{c||}%|
8415 {\footnotesize\correction@forgrading@kw} &\\\hline
8416 \correction@probs & \correction@sum@kw & \correction@grade@kw\\\hline
8417 \correction@pts &\theassignment@totalpts & \\\hline
8418 \correction@reached & & \\[.7cm]\hline
8419 \end{tabular}}}
8420 ⟨/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}
```

Chapter 41

References

EdN:20

[20]

[Bus+04]   Stephen Buswell et al. *The Open Math Standard, Version 2.0.* Tech. rep. The OpenMath Society, 2004. URL: http://www.openmath.org/standard/om20.

[CR99]   David Carlisle and Sebastian Rathz. *The graphicxl package.* Part of the TeX distribution. The Comprehensive TeX Archive Network. 1999. URL: https://www.tug.org/texlive/devsrc/Master/texmf-dist/doc/latex/graphics/graphicx.pdf.

[DCM03]   The DCMI Usage Board. *DCMI Metadata Terms.* DCMI Recommendation. Dublin Core Metadata Initiative, 2003. URL: http://dublincore.org/documents/dcmi-terms/.

[Koh06]   Michael Kohlhase. *OMDoc – An open markup format for mathematical documents [Version 1.2].* LNAI 4180. Springer Verlag, Aug. 2006. URL: http://omdoc.org/pubs/omdoc1.2.pdf.

[LMH]   *LMH Scripts.* URL: https://github.com/sLaTeX/lmhtools.

[MMT]   *MMT – Language and System for the Uniform Representation of Knowledge.* Project web site. URL: https://uniformal.github.io/ (visited on 01/15/2019).

[MRK18]   Dennis Müller, Florian Rabe, and Michael Kohlhase. "Theories as Types". In: *9th International Joint Conference on Automated Reasoning.* Ed. by Didier Galmiche, Stephan Schulz, and Roberto Sebastiani. Springer Verlag, 2018. URL: https://kwarc.info/kohlhase/papers/ijcar18-records.pdf.

[Rab15]   Florian Rabe. "The Future of Logic: Foundation-Independence". In: *Logica Universalis* 10.1 (2015). 10.1007/s11787-015-0132-x; Winner of the Contest "The Future of Logic" at the World Congress on Universal Logic, pp. 1–20.

[RK13]   Florian Rabe and Michael Kohlhase. "A Scalable Module System". In: *Information & Computation* 0.230 (2013), pp. 1–54. URL: https://kwarc.info/frabe/Research/mmt.pdf.

[RT]   *sLaTeX/RusTeX.* URL: https://github.com/sLaTeX/RusTeX (visited on 04/22/2022).

---

[20]EdNote: we need an un-numbered version sfragment*

[SIa]   *sLaTeX/sTeX-IDE*. URL: https://github.com/slatex/sTeX-IDE (visited on 04/22/2022).

[SIb]   *sLaTeX/stexls-vscode-plugin*. URL: https://github.com/slatex/stexls-vscode-plugin (visited on 04/22/2022).

[SLS]   *sLaTeX/stexls*. URL: https://github.com/slatex/stexls (visited on 04/22/2022).

[ST]   *sTeX – An Infrastructure for Semantic Preloading of LaTeX Documents*. URL: https://ctan.org/pkg/stex (visited on 04/22/2022).

[sTeX]   *sTeX: A semantic Extension of TeX/LaTeX*. URL: https://github.com/sLaTeX/sTeX (visited on 05/11/2020).

[Tana]   Till Tantau. *beamer – A LaTeX class for producing presentations and slides*. URL: http://ctan.org/pkg/beamer (visited on 01/07/2014).

[Tanb]   Till Tantau. *User Guide to the Beamer Class*. URL: http://ctan.org/macros/latex/contrib/beamer/doc/beameruserguide.pdf.

[TL]   *TeX Live*. URL: http://www.tug.org/texlive/ (visited on 12/11/2012).