The sTeX3 Package Collection *

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

2022-09-01

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 3.2 (last revised 2022-09-01)

i

Contents

Part I
Manual

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

> Boxes like this one explain how some sTeX concept relates to the MMT/OMDOC system, philosophy or language; see [MMT; Koh06] for introductions.

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).

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*:

```latex
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 **??**.

Let's investigate this document in detail to understand the respective parts of the sTeX markup infrastructure:

smodule (*env.*)	```\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	```\importmodule[smglom/calculus]{series}``` ```\importmodule[smglom/arithmetics]{realarith}```

Next, we *import* two modules – `series` from the ᴪTEX archive `smglom/calculus`, and `realarith` from the ᴪTEX 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 ᴪTEX 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`.

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

\symdef	```\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.

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

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

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

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

`\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.

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

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

2.2.1 OMDoc/xhtml Conversion

So, if we run `pdflatex` on our document, then sTEX yields pretty colors and tooltips[1]. But sTEX becomes a lot more powerful if we additionally convert our document to `xhtml` 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 **??**) 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 (⟨*language*⟩∗) Languages to load with the `babel` package.

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

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

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

3.1 How Knowledge is Organized in sTeX

sTeX content is organized on multiple levels:

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

> ↩M→
> —M→
> ⤳T↝
> - ˢTᴇX archives are simultaneously Mᴍᴛ archives, and the same directory structure is consequently used.
> - ˢTᴇX modules correspond to OMDᴏᴄ/Mᴍᴛ *theories*. `\importmodule`s (and similar constructions) induce Mᴍᴛ `include`s and other *theory morphisms*, thus giving rise to a *theory graph* in the OMDᴏᴄ sense [RK13].
> - Symbol declarations induce OMDᴏᴄ/Mᴍᴛ *constants*, with optional (formal) *type* and *definiens* components.
> - Finally, ˢTᴇX expressions are converted to OMDᴏᴄ/Mᴍᴛ terms, which use the abstract syntax (and XML encoding) of OᴘᴇɴMᴀᴛʜ [Bus+04].

3.2 ˢTᴇX 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, ˢTᴇX uses *archives* that determine the global namespaces for symbols and statements and make it possible for ˢTᴇX to find content referenced via such URIs.

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

1. If the ˢTᴇX package is loaded with the option `mathhub=/path/to/mathhub`, then ˢTᴇX 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 ˢTᴇX-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, ˢTᴇX 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, ˢTᴇX will look for a file `~/.stex/mathhub.path`. If this file exists, ˢTᴇX 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 ŞTEX Archives

An ŞTEX 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 ŞTEX 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 ŞTEX 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
ŞTEX archive:

- `/source/mod/` – individual ŞTEX modules, containing symbol declarations, nota-
  tions, and `\begin{sparagraph}[type=symdoc,for=...]` environments for "ency-
  clopaedic" 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 snip-
  pets, definitions, examples etc. in some desirable order

- `/source/tikz/` – tikz images, as individual `.tex`-files

- `/source/PIC/` – image files.

3.2.3 MANIFEST.MF-Files

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

```
id: smglom/calculus
source-base: http://mathhub.info/smglom/calculus
narration-base: http://mathhub.info/smglom/calculus
dependencies: smglom/arithmetics,smglom/sets,smglom/topology,
             smglom/mv,smglom/linear-algebra,smglom/algebra
responsible: Michael.Kohlhase@FAU.de
title: Elementary Calculus
```

```
teaser: Terminology for the mathematical study of change.
description: desc.html
```

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

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

source-base or

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

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

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

dependencies: All archives that this archive depends on. ſTEX ignores this field, but Mмт 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{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[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 $\text{S\!T\!E\!X}$ 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 (*env.*) A new module is declared using the basic syntax

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

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

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

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

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

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

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

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

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

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

> ↩M→ An SТEX 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 SᴛᴇX 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 OMDᴏᴄ/Mᴍᴛ constant in the current mod—M→ ule (=OMDᴏᴄ/Mᴍᴛ 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:

> **Example 7**
> Input:
>
> ```
> 1 \symdef{newbinarysymbol}[hl,args=2]
> 2   {\comp{\text{1.: }}#1\comp{\text{; 2.: }}#2}
> 3 $\newbinarysymbol{a}{b}$
> ```
>
> Output:
>
> | 1.: a; 2.: b |
> | --- |

.

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

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}`.

\textsymdecl In the less mathematical settings where we want a symbol and semantic macro for some concept with a notation *beyond* its mere name, but which should also be available in TeX's text mode, the command `\textsymdecl` is useful. For example, we can declare a symbol `openmath` with the notation `\textsc{OpenMath}` using `\textsymdecl{openmath}[name=OpenMath]{\textsc{OpenMath}}`. The `\openmath` yields OPENMATH both in text and math mode.

Operator Notations

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

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

> **Example 8**
> Input:
>
> ```
> 1    \notation{newbinarysymbol}[ab, 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 Mᴍᴛ 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/Mᴍᴛ constants.
> Correspondingly, the name "type" should be taken with a grain of salt, since OMDoc/Mᴍᴛ– 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 $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$

.

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

Example 16
Input:

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

Output:

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

24

.

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, e.g. for bracket-like notations such as intervals – for those purposes, `\infprec` and `\neginfprec` exist (which are implemented as the maximal and minimal integer values accordingly).g

More precisely, each notation takes

1. One *operator precedence* and

2. one *argument precedence* for each argument.

By default, all precedences are 0, unless the symbol takes no argument, in which case the operator precedence is `\neginfprec` (negative infinity). If we only provide a single number, this is taken as both the operator precedence and all argument precedences.

sTeX decides whether to insert parentheses by comparing operator precedences to a *downward precedence p_d* with initial value `\infprec`. When encountering a semantic macro, sTeX takes the operator precedence p_{op} of the notation used and checks whether $p_{op} > p_d$. If so, sTeX insert parentheses.

When sTeX steps into an argument of a semantic macro, it sets p_d to the respective argument precedence of the notation used.

In the example above:

1. sTeX starts out with $p_d =$ `\infprec`.

2. sTeX encounters `\addition` with $p_{op} = 100$. Since $100 \not> $`\infprec`, it inserts no parentheses.

3. Next, sTeX encounters the two arguments for `\addition`. Both have no specifically provided argument precedence, so sTeX uses $p_d = p_{op} = 100$ for both and recurses.

4. Next, sTeX encounters `\multiplication{b,...}`, whose notation has $p_{op} = 50$.

5. We compare to the current downward precedence p_d set by `\addition`, arriving at $p_{op} = 50 \not> 100 = p_d$, so sTeX again inserts no parentheses.

6. Since the notation of `\multiplication` has no explicitly set argument precedences, sTeX uses the operator precedence for all arguments of `\multiplication`, hence sets $p_d = p_{op} = 50$ and recurses.

7. Next, sTeX encounters the inner `\addition{c,...}` whose notation has $p_{op} = 100$.

8. We compare to the current downward precedence p_d set by `\multiplication`, arriving at $p_{op} = 100 > 50 = p_d$ – which finally prompts sTeX to insert parentheses, and we proceed as before.

3.3.6 Variables

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

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

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

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

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

> **Example 19**
> Input:
>
> ```
>  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\rustexBREAK
> 13 $\fun{\varx}{\varf{\addition{\varx,\varn}}}$
> ```
>
> Output:
>
> > Given a function $f : \mathbb{N} \to \mathbb{N}$, by $f+n$ we mean the function$x \mapsto f(x+n)$

.

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

TODO: bind=forall/exists

3.3.7 Variable Sequences

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

\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:

$a_1 + \ldots + a_n$

·

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:

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

.

3.4   Module Inheritance and Structures

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:

29

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, sTeX would use arbitrary URIs for modules, with no forced relationships between the *logical* namespace of a module and the *physical* location of the file declaring the module – like Mmt does things.
>
> Unfortunately, TeX only provides very restricted access to the file system, so we are forced to generate namespaces systematically in such a way that they reflect the physical location of the associated files, so that sTeX can resolve them accordingly. Largely, users need not concern themselves with namespaces at all, but for completenesses sake, we describe how they are constructed:
>
> - If `\begin{smodule}{Foo}` occurs in a file `/path/to/file/Foo[.⟨lang⟩].tex` which does not belong to an archive, the namespace is `file://path/to/file`.
>
> - If the same statement occurs in a file `/path/to/file/bar[.⟨lang⟩].tex`, the namespace is `file://path/to/file/bar`.
>
> In other words: outside of archives, the namespace corresponds to the file URI

with the filename dropped iff it is equal to the module name, and ignoring the (optional) language suffix.

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

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

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

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

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

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

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

- Finally, `\importmodule{full://uri?Foo}` naturally refers to the module `Foo` in the namespace `full://uri`. Since the file this module is declared in can not be determined directly from the URI, the module must be in memory already, e.g. by being referenced earlier in the same document.

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

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

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 (*env.*) 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 $\circ : U \times U \to U$ ...

.

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:

.

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 due to the obiquity of the symbols involved. Note however, that the metatheory is for all practical purposes a "normal" sTeX module, and the symbols contained "normal" sTeX symbols.

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.

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/Mᴍᴛ 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 Mᴍᴛ and other systems can pick up on it).[1]

EdN:1

Example 36
Input:

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

---

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

Output:

> adding k yields...

˙Note that since the second `\arg` has no explicit argument number, it automatically represents the first not-yet-given argument – i.e. in this case the first one.[2]

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...

.

If we take features like `\inputref` and `\mhinput` (and the `sfragment`-environment, see subsection 7.2.1) seriously, and build large documents modularly from individually compiling documents for sections, chapters and so on, cross-referencing becomes an interesting problem.

Say, we have a document `main.tex`, which `\inputref`s a section `section1.tex`, which references a definition with label `some_definition` in `section2.tex` (subsequently also inputted in `main.tex`). Then the numbering of the definition will depend on the *document context* in which the document fragment `section2.tex` occurs - in `section2.tex` itself (as a standalone document), it might be *Definition 1*, in `main.tex` it might be *Definition 3.1*, and in `section1.tex`, the definition *does not even occur*, so it needs to be referenced by some other text.

What we would want in that instance is an equivalent of `\autoref`, that takes the document context into account to yield something like *Definition 1*, *Definition 3.1* or "*Definition 1 in the section on Foo*" respectively.

The `\sref` command attempts to do precisely that. Unlike plain `\ref`, `\autoref` etc., `\sref` refers to not just a *label*, but instead a pair consisting of a *label* and the *document* in whose context we want to refer to it. Conversely, every *document* (i.e. standalone compilable `.tex`-file) keeps track of the "names" (*Definition 3.1* etc.) for every label as determined in the context of the document, and stores them in a dedicated file `\jobname.sref`. Additionally, every document has a "*reference name*" (e.g. "*the section on Foo*"). This allows us to refer to "label x in document D" to yield "*Definition 1 in the section on Foo*". And of course, sTeX can decide based on the current document

---

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

to either refer to the label by its "full name" or directly as e.g. *Definition 3.1* depending on whether the label occurs in the current document anyway (and link to it accordingly).

For that to work, we need to supply (up to) three pieces of information:

- The *label* of the reference target (e.g. `some_definition`),

- (optionally) the *file*/document containing the reference target (e.g. `section2`). This is not strictly necessary, but allows for additional disambiguation between possibly duplicate labels across files, and

- (optionally) the document context, in which we want to refer to the reference target (e.g. `main`).

Additionally, the document in which we want to reference a label needs a title for external references.

`\sref` `\sref[archive=⟨archive1⟩,file=⟨file⟩]`
`{⟨label⟩}[archive=⟨archive2⟩,in=⟨document-context⟩,title=⟨title⟩]`

This command references ⟨*label*⟩ (declared in ⟨*file*⟩ in ⟨*archive1*⟩). If the object (section, figure, etc.) with that label occurs ultimately in the same document, `\sref` will ignore the second set of optional arguments and simply defer to `\autoref` if that command exists, or `\ref` if the hyperref package is not included.

If the referenced object does *not* occur in the current document however, `\sref` will refer to it by the object's name as it occurs in the file ⟨*document-context*⟩ in ⟨*archive2*⟩.

For example, the reference to the sfragment-environment above will appear as "subsection 7.2.1 (Introduction) in the sTeX3 manual" if you are reading this in the package documentation for `stex-references` directly, but as a linked "subsection 7.2.1" in the full documentation or manual. This is achieved using

`\sref[file=stex-document-structure]{sec:ds:intro}[in=../stex-manual,title={the \sTe`
For a further example, the following:

<div style="text-align:center; color:red">Part III</div>

will say "Part III" (and link accordingly) in the full documentation, and "Part III (Extensions) in the full sTeX3 documentation" everywhere else. This is achieved using

`\sref[file=../stex-doc]{part:extends}[in=../stex-doc,title={the full \sTeX{}3 docum`

`\extref` `\sref[archive=⟨archive1⟩,file=⟨file⟩]`
`{⟨label⟩}{archive=⟨archive2⟩,in=⟨document-context⟩,title=⟨title⟩}`

The `\extref`-command behaves exactly like `\sref`, but takes *required* the document context argument and will always use it for generating the document text, regardless of whether the label occurs in the current document.

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 **??** 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 **??**), type= for customization (see **??**) 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.

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

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:[3]

> **Example 39**
> Input:

---

[3]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[4] 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 ]
```

---

[4]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]

5.2 Proofs

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

Its central component is the `sproof`-environment, whose body consists of:

- *subproofs* via the `subproof`-environment,

- *proof steps* via the `\spfstep`, `\eqstep` `\assumption`, and `\conclude` macros, and

- *comments*, via normal text without special markup.

`sproof`, `subproof` and the various proof step macros take the following optional arguments:

id (⟨*string*⟩) for referencing,

method (⟨*string*⟩) the proof method (e.g. contradiction, induction,...)

term (⟨*token list*⟩) the (ideally semantically-marked up) proposition that is derived/proven by this proof/subproof/proof step.

Additionally, they take one mandatory argument for the document text to be annotated, or (in the case of the environments) as an introductory description of the proof itself. Since the latter often contains the `term` to be derived as text, alternatively to providing it as an optional argument, the mandatory argument can use the `\yield`-macro to mark it up in the text.

The `sproof` and `subproof` environments additionally take two optional arguments:

for the symbol identifier/name corresponding to the `sassertion` to be proven. This too subsumes `\yield` and the `term`-argument.

hide In the pdf, this only shows the mandatory argument text and hides the body of the environment. In the HTML (as served by Mʍᴛ), the bodies of all `proof` and `subproof` environments are *collapsible*, and `hide` collapses the body by default.

```
1  \begin{sassertion}[type=theorem,name=sqrt2irr]
2    \conclusion{\irrational{$\arg{\realroot{2}}$ is \comp{irrational}}}.
3  \end{sassertion}
4
5  \begin{sproof}[for=sqrt2irr,method=contradiction]{By contradiction}
6    \assumption{Assume \yield{\rational{$\arg{\realroot{2}}$ is
7      \comp{rational}}}}
8    \begin{subproof}[method=straightforward]{Then
9      \yield{$\eq{\ratfrac{\intpow{\vara}{2}}{\intpow{\varb}2}}{2}$
10     for some $\inset{\vara,\varb}\PosInt$ with
11     \coprime{$\arg{\vara},\arg{\varb}$ \comp{coprime}}}}
```

---

[2]Of course, SʈEX 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, Mʍᴛ can. TODO: should

```
12          \assumption{By assumption, \yield{there are
13          $\inset{\vara,\varb}\PosInt $ with
14          $\realroot{2}=\ratfrac{\vara}{\varb}$}}
15          \spfstep{wlog, we can assume \coprime{$\arg{\vara},\arg{\varb}$
16          to be \comp{coprime}}}
17              % a comment:
18              If not, reduce the fraction until numerator and denominator
19              are coprime, and let the resulting components be
20              $\vara $ and $\varb $
21          \spfstep{Then \yield{$\eq{\intpow{\ratfrac{\vara}{\varb}}2}2$}}
22          \eqstep{\ratfrac{\intpow{\vara}2}{\intpow{\varb}2}}
23      \end{subproof}
24      \begin{subproof}[term=\divides{2}{\vara},method=straightforward]{
25          Then $\vara $ is even}
26          \spfstep{Multiplying the equation by $\intpow{\varb}2$ yields
27          $\yield{\eq{\intpow{\vara}2}{\inttimes{2}{\intpow{\varb}2}}}$}
28          \spfstep[term=\divides{2}{\intpow{\vara}2}]{Hence
29          $\intpow{\vara}2$ is even}
30          \conclude[term=\divides{2}{\vara}]{Hence $\vara $ is even as well}
31          % another comment:
32          Hint: Think about the prime factorizations of $\vara $ and
33          $\intpow{\vara}2$
34      \end{subproof}
35      \begin{subproof}[term=\divides{2}{\varb},method=straightforward,]{
36          Then $\varb $ is also even}
37          \spfstep{Since $\vara $ is even, we have \yield{some $\varc $
38            such that $\eq{\inttimes{2}{\varc}}{\vara}$}}
39          \spfstep{Plugging into the above, we get
40            \yield{$\eq{\intpow{\inttimes{2}{\vara}}2}
41              {\inttimes{2}{\intpow{\varb}2}}$}}
42          \eqstep{\inttimes{4}{\intpow{\vara}2}}
43          \spfstep{Dividing both sides by $2$ yields
44            \yield{$\eq{\intpow{\varb}2}{\inttimes{2}{\intpow{\vara}2}}$}}
45          \spfstep[term=\divides{2}{\intpow{\varb}2}]{Hence
46            $\intpow{\varb}2$ is even}
47          \conclude[term=\divides{2}{\varb}]{Hence $\varb $ is even}
48          % one more comment:
49          By the same argument as above
50      \end{subproof}
51      \conclude[term=\contradiction]{Contradiction to $\vara,\varb $ being
52      \symname{coprime}.}
53 \end{sproof}
```

which will produce:

Theorem 5.2.1. $\sqrt{2}$ *is irrational.*

Proof: By contradiction

1. Assume $\sqrt{2}$ is rational

2. Then $(\frac{a^2}{b^2}){=}2$ for some $a,b{\in}\mathbb{Z}^+$ with a,b coprime

2.1. By assumption, there are $a,b{\in}\mathbb{Z}^+$ with $\sqrt{2} = \frac{a}{b}$

2.2. wlog, we can assume a,b to be coprime

If not, reduce the fraction until numerator and denominator are coprime, and let the re-

---

> *sulting components be a and b*
>
> 2.3. Then $(\frac{a}{b})^2$=2
>
> $= \frac{a^2}{b^2}$
>
> 3. Then a is even
>
> 3.1. Multiplying the equation by b^2 yields a^2=$2b^2$
>
> 3.2. Hence a^2 is even
>
> ⇒ Hence a is even as well
>
> *Hint: Think about the prime factorizations of a and a^2*
>
> 4. Then b is also even
>
> 4.1. Since a is even, we have some c such that $2c$=a
>
> 4.2. Plugging into the above, we get $(2a)^2$=$2b^2$
>
> $= 4a^2$
>
> 4.3. Dividing both sides by 2 yields b^2=$2a^2$
>
> 4.4. Hence b^2 is even
>
> ⇒ Hence b is even
>
> *By the same argument as above*
>
> ⇒ Contradiction to a, b being coprime.
>
> $\square$

If we mark all subproofs with `hide`, we will obtain the following instead:

> **Theorem 5.2.2.** $\sqrt{2}$ *is irrational.*
>
> **Proof**: By contradiction
>
> 1. Assume $\sqrt{2}$ is rational
>
> 2. Then $(\frac{a^2}{b^2})$=2 for some $a,b \in \mathbb{Z}^+$ with a, b coprime
>
> 3. Then a is even
>
> 4. Then b is also even
>
> ⇒ Contradiction to a, b being coprime.
>
> $\square$

However, the hidden subproofs will still be shown in the HTML, only in an expandable section which is collapsed by default.

The above style of writing proofs is usually called *structured proofs*. They have a huge advantage over the traditional purely prosaic style, in that (as the name suggests) the actual *structure* of the proof is made explicit, which almost always makes it considerably more comprehensible. We, among many others, encourage the general use of structured proofs.

Alas, most proofs are not written in this style, and we would do users a disservice by insisting on this style. For that reason, the `spfblock` environment turns all subproofs and proof step macros into presentationally neutral *inline* annotations, as in the induction step of the following example:

```
1 \begin{sproof}[id=simple-proof,method=induction]
2    {We prove that $\sum_{i=1}^n{2i-1}=n^{2}$ by induction over $n$}
```

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

This yields the following result:

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

For the induction we have to consider three cases:

1. $n = 1$
then we compute $1 = 1^2$

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 $1 + 3 = 2^2 = 4$.

3. $n > 1$
Now, we assume that the assertion is true for a certain $k \geq 1$, i.e. $\sum_{i=1}^k (2i - 1) = k^2$.
We have to show that we can derive the assertion for $n = k+1$ from this assumption,

---

i.e. $\sum_{i=1}^{k+1}(2i-1) = (k+1)^2$.

We obtain $\sum_{i=1}^{k+1} 2i - 1 = \sum_{i=1}^{k} 2i - 1 + 2(k+1) - 1$ by splitting the sum. Thus we have $\sum_{i=1}^{k+1}(2i-1) = k^2 + 2k + 1$ by induction hypothesis. We can simplify the right-hand side to $k+1^2$, which proves the assertion.

$\Rightarrow$ We have considered all the cases, so we have proven the assertion.

$\square$

sproof (*env.*) 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 `\spfstep` macro, 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.

`\yield` See above

`\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 (ideally, a semantically marked-up term).

`\assumption` The `\assumption` macro allows to mark up a (justified) assumption.

`\justarg`

subproof (*env.*) The `subproof` 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.

\sproofend 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.

\sProofEndSymbol 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{}`.

5.3 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 `sassertion`s 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, `\symref`s and `\definiendum`s, respectively.

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

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

For each of the four macros, there exists an additional macro that takes the full URI of the relevant symbol currently being highlighted as a second argument. That allows us to e.g. use pdf tooltips and links. For example, this document uses[5]

```
1 \protected\def\symrefemph@uri#1#2{
2   \pdftooltip{
3     \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 6

Cross References

If we take features like `\inputref` and `\mhinput` (and the `sfragment`-environment, see subsection 7.2.1) seriously, and build large documents modularly from individually compiling documents for sections, chapters and so on, cross-referencing becomes an interesting problem.

Say, we have a document `main.tex`, which `\inputref`s a section `section1.tex`, which references a definition with label `some_definition` in `section2.tex` (subsequently also inputted in `main.tex`). Then the numbering of the definition will depend on the *document context* in which the document fragment `section2.tex` occurs - in `section2.tex` itself (as a standalone document), it might be *Definition 1*, in `main.tex` it might be *Definition 3.1*, and in `section1.tex`, the definition *does not even occur*, so it needs to be referenced by some other text.

What we would want in that instance is an equivalent of `\autoref`, that takes the document context into account to yield something like *Definition 1*, *Definition 3.1* or "*Definition 1 in the section on Foo*" respectively.

The `\sref` command attempts to do precisely that. Unlike plain `\ref`, `\autoref` etc., `\sref` refers to not just a *label*, but instead a pair consisting of a *label* and the *document* in whose context we want to refer to it. Conversely, every *document* (i.e. standalone compilable `.tex`-file) keeps track of the "names" (*Definition 3.1* etc.) for every label as determined in the context of the document, and stores them in a dedicated file `\jobname.sref`. Additionally, every document has a "*reference name*" (e.g. "*the section on Foo*"). This allows us to refer to "label x in document D" to yield "*Definition 1 in the section on Foo*". And of course, sTEX can decide based on the current document to either refer to the label by its "full name" or directly as e.g. *Definition 3.1* depending on whether the label occurs in the current document anyway (and link to it accordingly).

For that to work, we need to supply (up to) three pieces of information:

- The *label* of the reference target (e.g. `some_definition`),

- (optionally) the *file*/document containing the reference target (e.g. `section2`). This is not strictly necessary, but allows for additional disambiguation between possibly duplicate labels across files, and

- (optionally) the document context, in which we want to refer to the reference target (e.g. `main`).

53

Additionally, the document in which we want to reference a label needs a title for external references.

\sref `\sref[archive=⟨`*archive1*`⟩,file=⟨`*file*`⟩]`
`{⟨`*label*`⟩}[archive=⟨`*archive2*`⟩,in=⟨`*document-context*`⟩,title=⟨`*title*`⟩]`

This command references ⟨*label*⟩ (declared in ⟨*file*⟩ in ⟨*archive1*⟩). If the object (section, figure, etc.) with that label occurs ultimately in the same document, `\sref` will ignore the second set of optional arguments and simply defer to `\autoref` if that command exists, or `\ref` if the hyperref package is not included.

If the referenced object does *not* occur in the current document however, `\sref` will refer to it by the object's name as it occurs in the file ⟨*document-context*⟩ in ⟨*archive2*⟩.

For example, the reference to the sfragment-environment above will appear as "subsection 7.2.1 (Introduction) in the SITEX3 manual" if you are reading this in the package documentation for `stex-references` directly, but as a linked "subsection 7.2.1" in the full documentation or manual. This is achieved using

`\sref[file=stex-document-structure]{sec:ds:intro}[in=../stex-manual,title={the \sTe`

For a further example, the following:

<p style="text-align:center; color:red">Part III</p>

will say "Part III" (and link accordingly) in the full documentation, and "Part III (Extensions) in the full SITEX3 documentation" everywhere else. This is achieved using

`\sref[file=../stex-doc]{part:extends}[in=../stex-doc,title={the full \sTeX{}3 docu`

\extref `\sref[archive=⟨`*archive1*`⟩,file=⟨`*file*`⟩]`
`{⟨`*label*`⟩}{archive=⟨`*archive2*`⟩,in=⟨`*document-context*`⟩,title=⟨`*title*`⟩}`

The `\extref`-command behaves exactly like `\sref`, but takes *required* the document context argument and will always use it for generating the document text, regardless of whether the label occurs in the current document.

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 |

\mhtizkinput 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, \cmhtizkinput 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

7.2.1 Introduction

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.

7.2.2 Package Options

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 |

7.2.3 Document Fragments

sfragment (*env.*) 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 section. If the title contains semantic macros, 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}
```

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 (*env.*) 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. The `frontmatter` environment also suppresses numbering as is traditional for prefaces.

`\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.

\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.

7.2.4 Ending Documents Prematurely

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

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

   Text fragments and modules can be made more re-usable by the use of global variables. For instance, the admin section of a course can be made course-independent (and therefore re-usable) by using variables (actually token registers) courseAcronym and courseTitle instead of the text itself. The variables can then be set in the sTeX preamble of the course notes file.

7.2.5 Global Document Variables

To make document fragments more reusable, we sometimes want to make the content depend on the context. We use **document variables** for that.

\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

7.3.1 Introduction

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.

   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 sTeX 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).

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.

7.3.2 Package Options

The notesslides class takes a variety of class options:

slides notes The options `slides` and `notes` switch between slides mode and notes mode (see subsection 7.3.3).

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

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

7.3.3 Notes and Slides

frame (*env.*) Slides are represented with the `frame` environment just like in the beamer class, see [Tanb] for details.

note (*env.*) The notesslides class adds the `note` environment for encapsulating the course note fragments.

> ⚠ 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  ...
```

59

```
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 ...
```

\ifnotes 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.

> ⚠ 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}`.

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

7.3.4 Customizing Header and Footer Lines

The `notesslides` package and class comes with a simple default theme named `sTeX` that provided by the `beamterthemesTeX`. It is assumed as the default theme for sTeX-based notes and slides. The result in `notes` mode (which is like the `slides` version except that the slide hight is variable) is

The footer line can be customized. In particular the logos.

\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 notesslides \source stores the author's name as the copyright holder. By default it is the author's name as defined in the \author macro in the preamble. \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.

7.3.5 Frame Images

Sometimes, we want to integrate slides as images after all – e.g. because we already have a PowerPoint presentation, to which we want to add 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: ⚠

7.3.6 Excursions

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

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

It generates a paragraph that references the excursion whose source is in the file `../fragments/founif.en.tex` and automatically books the file for the `\printexcursions` command that is used here to put it into the appendix. We will look at the mechanics now.

`\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.

`\excursiongroup` 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

7.4.1 Introduction

The `problem` package supplies an infrastructure that allows specify problem. Problems are text fragments that come with auxiliary functions: hints, notes, and solutions[4]. 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.

7.4.2 Problems and Solutions

`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.

`problem` (*env.*) 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.

> **Example 40**
> Input:

---

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

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

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

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

7.4.3 Markup for Added-Value Services

The `problem` package is all about specifying the meaning of the various moving parts of practice/exam problems. The motivation for the additional markup is that we can base added-value services from these, for instance auto-grading and immediate feedback.

The simplest example of this are multiple-choice problems, where the `problem` package allows to annotate answer options with the intended values and possibly feedback that can be delivered to the users in an interactive setting. In this section we will give some infrastructure for these, we expect that this will grow over time.

Multiple Choice Blocks

mcb *(env.)* 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.

What we see when this is formatted to PDF depends on the context. In solutions mode (we start the solutions in the code fragment below) 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
Correct!

☐ function
Wrong! *that is for C and C++*

☐ fun
Wrong! *that is for Standard ML*

☐ public static void
Wrong! *that is for Java*

.

In "exam mode" where disable solutions (here via `\stopsolutions`)

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

☐ function

☐ fun

☐ public static void

˙we get the questions without solutions (that is what the students see during the exam/quiz).

Filling-In Concrete Solutions

The next simplest situation, where we can implement auto-grading is the case where we have fill-in-the-blanks

66

The `\fillinsol` macro takes[6] an a singlle argument, which contains a concrete solution (i.e. a number, a string, . . .), which generates a fill-in-box in test mode:

Example 43

Input:

```
1   \stopsolutions
2   \begin{sproblem}[id=elefants.fillin,title=Fitting Elefants]
3     How many Elefants can you fit into a Volkswagen beetle? \fillinsol{4}
4   \end{sproblem}
```

Output:

Problem 7.4.4 (Fitting Elefants)

How many Elefants can you fit into a Volkswagen beetle?

˙and the actual solution in solutions mode:

Example 44

Input:

```
1   \begin{sproblem}[id=elefants.fillin,title=Fitting Elefants]
2     How many Elefants can you fit into a Volkswagen beetle? \fillinsol{4}
3   \end{sproblem}
```

Output:

Problem 7.4.5 (Fitting Elefants)
How many Elefants can you fit into a Volkswagen beetle?   4!

Obviously, the argument of `\fillinsol` can be used for˙auto-grading. For concrete data like numbers, this is immediate, for more complex data like strings "soft comparisons" might be in order. [7]

EdN:7

7.4.4   Including Problems

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

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.

---

[7]EDNOTE: For the moment we only assume a single concrete value as correct. In the future we will almost certainly want to extend the functionality to multiple answer classes that allow different feedback like im MCQ. This still needs a bit of design. Also we want to make the formatting of the answer in solutions/test mode configurable.

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

7.5.1 Introduction

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

7.5.2 Package Options

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

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

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

7.5.3 Assignments

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

7.5.4 Including Assignments

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

7.5.5 Typesetting Exams

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

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

reqpts

```
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)

2022-09-01

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	7.4.4	7.4.5	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:8   8

8EDNOTE: MK: The first three "problems" come from the stex examples above, how do we get rid of
this?

69

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: ⋆` LaTeX3 conditionals for LaTeXML.
`\latexml_if:TF ⋆`

`\stex_if_do_html_p: ⋆` Whether to currently produce any HTML annotations (can be false in some advanced
`\stex_if_do_html:TF ⋆` 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` (*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`
`\stex_filestack_pop:` Push and pop (repsectively) a file path to the file stack, to keep track of the current file. Are called in hooks `file/before` and `file/after`, respectively.

9.1.2 MathHub Archives

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

1. The `mathhub` package option, or

2. the `\mathhub`-macro, if it has been defined before the `\usepackage{stex}`-statement, or

3. the `MATHHUB` system variable, or

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

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

`\l_stex_current_repository_prop`

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

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

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

narr: the narration namespace (for document references),

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

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

`\stex_set_current_repository:n`

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

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

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

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

9.1.3 Using Content in Archives

\mhpath ⋆   \mhpath{⟨*archive-ID*⟩}{⟨*filename*⟩}

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

\inputref   \inputref[⟨*archive-ID*⟩]{⟨*filename*⟩}
\mhinput

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

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

Chapter 10

sTEX-References

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

10.1   Macros and Environments

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

\l_stex_current_docns_str   Stores its result in \l_stex_current_docns_str

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

\l_stex_current_docurl_str   Stores its result in \l_stex_current_docurl_str

10.1.1   Setting Reference Targets

\stex_ref_new_doc_target:n   \stex_ref_new_doc_target:n{⟨id⟩}

Sets a new reference target with id ⟨id⟩.

\stex_ref_new_sym_target:n   \stex_ref_new_sym_target:n{⟨uri⟩}

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

10.1.2 Using References

\sref `\sref[⟨opt-args⟩]{⟨id⟩}`

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

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

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

- A `\definiendum` or `\definame` for ⟨*symbol*⟩,

- The `sassertion`, `sexample` or `sparagraph` with `for=`⟨*symbol*⟩ that generated ⟨*symbol*⟩ in the first place, or

- A `\sparagraph` with `type=symdoc` and `for=`⟨*symbol*⟩.

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

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

Chapter 11

sTEX-Modules

This sub package contains code related to Modules

11.1 Macros and Environments

The content of a module with uri $\langle <URI> \rangle$ is stored in four macros. All modifications of these macros are global:

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

name The *name* of the module,

ns the *namespace* in field `ns`,

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

lang the module's *language*,

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

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

meta the metatheory of the module.

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

`\c_stex_module_<URI>_constants`

The names of all constants declared in the module

`\c_stex_module_<URI>_constants`

The full URIs of all modules imported in this module

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

`\l_stex_all_modules_seq` Stores full URIs for all modules currently in scope.

`\stex_if_in_module_p:` ⋆
`\stex_if_in_module:`*TF* ⋆   Conditional for whether we are currently in a module

`\stex_if_module_exists_p:n` ⋆
`\stex_if_module_exists:n`*TF* ⋆

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

`\stex_add_to_current_module:n`
`\STEXexport`

Adds the provided tokens to the `_code` control sequence of the current module.
`\stex_add_to_current_module:n` is used internally, `\STEXexport` is intended for users and additionally executes the provided code immediately.

`\stex_add_constant_to_current_module:n`

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

`\stex_add_import_to_current_module:n`

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

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

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

\stex_modules_current_namespace:

Computes the current namespace as follows:

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

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

11.1.1 The `smodule` environment

module (*env.*) `\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   stores the result in these four variables.
\l_stex_import_path_str
\l_stex_import_ns_str

\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}.

85

\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 ⟨*URI*⟩⟨*fragment*⟩⟨*precedence*⟩⟨*body*⟩
\STEXInternalTermMathOMAiiii
\STEXInternalTermMathOMBiiii

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*⟩.

\STEXInternalTermMathAssocArgiiiii `\stex_term_arg:nnn⟨int⟩⟨prec⟩⟨notation⟩⟨type⟩⟨body⟩`

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

\infprec
\neginfprec
Maximal and minimal notation precedences.

\dobrackets `\dobrackets {⟨body⟩}`

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

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

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

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

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

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

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

`\comp{⟨args⟩}`

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

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

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

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

\ellipses TODO

Chapter 15

sTEX-Structural Features

Code related to structural features

15.1 Macros and Environments

15.1.1 Structures

`mathstructure` (*env.*) TODO

Chapter 16

sTeX-Statements

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

16.1 Macros and Environments

symboldoc (*env.*) `\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

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}{2022/08/08}{3.2.0}{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}{2022/08/08}{3.2.0}{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*~This~is~sTeX~version~3.2.0~*^^J}
57
58  %\RequirePackage{morewrites}
59  %\RequirePackage{amsmath}
60
```

Package options:
```
61  \keys_define:nn { stex } {
62    debug      .clist_set:N  = \c_stex_debug_clist ,
63    lang       .clist_set:N  = \c_stex_languages_clist ,
64    mathhub    .tl_set_x:N   = \mathhub ,
65    usesms     .bool_set:N   = \c_stex_persist_mode_bool ,
66    writesms   .bool_set:N   = \c_stex_persist_write_mode_bool ,
67    image      .bool_set:N   = \c_tikzinput_image_bool,
68    unknown    .code:n       = {}
69  }
70  \ProcessKeysOptions { stex }
```

\stex The sTEXlogo:
\sTeX
```
71  \RequirePackage{stex-logo} % externalized for backwards-compatibility reasons
```

(*End definition for* \stex *and* \sTeX. *These functions are documented on page* 71.)

24.3 Messages and logging

⟨@@=stex_log⟩

Warnings and error messages

```
73 \msg_new:nnn{stex}{error/unknownlanguage}{
74   Unknown~language:~#1
75 }
76 \msg_new:nnn{stex}{warning/nomathhub}{
77   MATHHUB~system~variable~not~found~and~no~
78   \detokenize{\mathhub}-value~set!
79 }
80 \msg_new:nnn{stex}{error/deactivated-macro}{
81   The~\detokenize{#1}~command~is~only~allowed~in~#2!
82 }
```

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

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

(*End definition for* \stex_debug:nn. *This function is documented on page 71.*)

Redirecting messages:

```
98 \clist_if_in:NnTF \c_stex_debug_clist {all} {
99     \msg_redirect_module:nnn{ stex }{ none }{ term }
100 }{
101   \clist_map_inline:Nn \c_stex_debug_clist {
102     \msg_redirect_name:nnn{ stex }{ debug / #1 }{ term }
103   }
104 }
105
106 \stex_debug:nn{log}{debug~mode~on}
```

24.4   HTML Annotations

⟨@@=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.

```
108 \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`

```
109 \cs_new_protected:Nn \_stex_html_checkempty:n {
110   \tl_set:Nn \l_stex_html_arg_tl { #1 }
111   \tl_if_empty:NT \l_stex_html_arg_tl {
112     \tl_set_eq:NN \l_stex_html_arg_tl \c_stex_html_emptyarg_tl
113   }
114 }
```

(*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

```
115 \bool_new:N \_stex_html_do_output_bool
116 \bool_set_true:N \_stex_html_do_output_bool
117
118 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
119   \bool_if:nTF \_stex_html_do_output_bool
120     \prg_return_true: \prg_return_false:
121 }
```

(*End definition for* `\stex_if_do_html:TF`. *This function is documented on page* *71*.)

`\stex_suppress_html:n`

Whether to (locally) produce HTML output

```
122 \cs_new_protected:Nn \stex_suppress_html:n {
123   \exp_args:Nne \use:nn {
124     \bool_set_false:N \_stex_html_do_output_bool
125     #1
126   }{
127     \stex_if_do_html:T {
128       \bool_set_true:N \_stex_html_do_output_bool
129     }
130   }
131 }
```

(*End definition for* `\stex_suppress_html:n`. *This function is documented on page* *71*.)

`stex\annotate tenv:enn`
`\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.

```
132 \ifcsname if@rustex\endcsname\else
133   \expandafter\newif\csname if@rustex\endcsname
134   \@rustexfalse
135 \fi
136 \ifcsname if@latexml\endcsname\else
137   \expandafter\newif\csname if@latexml\endcsname
138   \@latexmlfalse
139 \fi
140 \tl_if_exist:NF\stex@backend{
141   \if@rustex
142     \def\stex@backend{rustex}
143   \else
144     \if@latexml
145       \def\stex@backend{latexml}
146     \else
```

```
147        \cs_if_exist:NTF\HCode{
148            \def\stex@backend{tex4ht}
149        }{
150            \def\stex@backend{pdflatex}
151        }
152      \fi
153    \fi
154 }
155 \input{stex-backend-\stex@backend.cfg}
156
157 \newif\ifstexhtml
158 \stex_html_backend:TF\stexhtmltrue\stexhtmlfalse
159
```

(*End definition for* \stex_annotate:nnn *,* \stex_annotate_invisible:n *, and* \stex_annotate_invisible:nnn *.
These functions are documented on page 72.*)

24.5 Babel Languages

```
160 ⟨@@=stex_language⟩
```

\c_stex_languages_prop

\c_stex_language_abbrevs_prop

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

```
161 \exp_args:NNx \prop_const_from_keyval:Nn \c_stex_languages_prop { \tl_to_str:n {
162    en = english ,
163    de = ngerman ,
164    ar = arabic ,
165    bg = bulgarian ,
166    ru = russian ,
167    fi = finnish ,
168    ro = romanian ,
169    tr = turkish ,
170    fr = french
171 }}
172
173 \exp_args:NNx \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop { \tl_to_str:n {
174    english   = en ,
175    ngerman   = de ,
176    arabic    = ar ,
177    bulgarian = bg ,
178    russian   = ru ,
179    finnish   = fi ,
180    romanian  = ro ,
181    turkish   = tr ,
182    french    = fr
183 }}
184 % todo: chinese simplified (zhs)
185 %       chinese traditional (zht)
```

(*End definition for* \c_stex_languages_prop *and* \c_stex_language_abbrevs_prop*. These variables are
documented on page 72.*)

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

```
186 \cs_new_protected:Nn \stex_set_language:Nn {
187    \str_set:Nx \l_tmpa_str {#2}
188    \prop_get:NoNT \c_stex_languages_prop \l_tmpa_str #1 {
```

```
189     \ifx\@onlypreamble\@notprerr
190        \ltx@ifpackageloaded{babel}{
191           \exp_args:No \selectlanguage #1
192        }{}
193     \else
194        \exp_args:No \str_if_eq:nnTF #1 {turkish} {
195           \RequirePackage[#1,shorthands=:!]{babel}
196        }{
197           \RequirePackage[#1]{babel}
198        }
199     \fi
200   }
201 }
202
203 \clist_if_empty:NF \c_stex_languages_clist {
204   \bool_set_false:N \l_tmpa_bool
205   \clist_clear:N \l_tmpa_clist
206   \clist_map_inline:Nn \c_stex_languages_clist {
207     \str_set:Nx \l_tmpa_str {#1}
208     \str_if_eq:nnT {#1}{tr}{
209        \bool_set_true:N \l_tmpa_bool
210     }
211     \prop_get:NoNTF \c_stex_languages_prop \l_tmpa_str \l_tmpa_str {
212        \clist_put_right:No \l_tmpa_clist \l_tmpa_str
213     } {
214        \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
215     }
216   }
217   \stex_debug:nn{lang} {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
218   \bool_if:NTF \l_tmpa_bool {
219     \RequirePackage[\clist_use:Nn \l_tmpa_clist,,shorthands=:!]{babel}
220   }{
221     \RequirePackage[\clist_use:Nn \l_tmpa_clist,]{babel}
222   }
223 }
224
225 \AtBeginDocument{
226   \stex_html_backend:T {
227     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
228     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
229     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
230     \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
231     \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
232        \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
233        \stex_debug:nn{basics} {Language~\l_tmpa_str~
234           inferred~from~file~name}
235        \stex_annotate_invisible:nnn{language}{ \l_tmpa_str }{}
236     }
237   }
238 }
239
```

24.6 Persistence

```
240 ⟨@@=stex_persist⟩
241 \bool_if:NTF \c_stex_persist_mode_bool {
242   \def \stex_persist:n #1 {}
243   \def \stex_persist:x #1 {}
244 }{
245   \bool_if:NTF \c_stex_persist_write_mode_bool {
246   \iow_new:N \c__stex_persist_iow
247   \iow_open:Nn \c__stex_persist_iow{\jobname.sms}
248   \AtEndDocument{
249     \iow_close:N \c__stex_persist_iow
250   }
251   \cs_new_protected:Nn \stex_persist:n {
252     \tl_set:Nn \l_tmpa_tl { #1 }
253     \regex_replace_all:nnN { \cP\# } { \cO\# } \l_tmpa_tl
254     \regex_replace_all:nnN { \  } { \~ } \l_tmpa_tl
255     \exp_args:NNo \iow_now:Nn \c__stex_persist_iow \l_tmpa_tl
256   }
257   \cs_generate_variant:Nn \stex_persist:n {x}
258 }{
259   \def \stex_persist:n #1 {}
260   \def \stex_persist:x #1 {}
261 }
262 }
```

24.7 Auxiliary Methods

\stex_deactivate_macro:Nn

```
263 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
264   \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
265   \def#1{
266     \msg_error:nnnn{stex}{error/deactivated-macro}{\detokenize{#1}}{#2}
267   }
268 }
```

(*End definition for* \stex_deactivate_macro:Nn. *This function is documented on page 72.*)

\stex_reactivate_macro:N

```
269 \cs_new_protected:Nn \stex_reactivate_macro:N {
270   \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname
271 }
```

(*End definition for* \stex_reactivate_macro:N. *This function is documented on page 72.*)

\ignorespacesandpars

```
272 \protected\def\ignorespacesandpars{
273   \begingroup\catcode13=10\relax
274   \@ifnextchar\par{
275     \endgroup\expandafter\ignorespacesandpars\@gobble
276   }{
277     \endgroup
278   }
279 }
```

```
280
281  \cs_new_protected:Nn \stex_copy_control_sequence:NNN {
282    \tl_set:Nx \_tmp_args_tl {\cs_argument_spec:N #2}
283    \exp_args:NNo \tl_remove_all:Nn \_tmp_args_tl \c_hash_str
284    \int_set:Nn \l_tmpa_int {\tl_count:N \_tmp_args_tl}
285
286    \tl_clear:N \_tmp_args_tl
287    \int_step_inline:nn \l_tmpa_int {
288      \tl_put_right:Nx \_tmp_args_tl {{\exp_not:n{####}\exp_not:n{##1}}}
289    }
290
291    \tl_set:Nn #3 {\cs_generate_from_arg_count:NNnn #1 \cs_set:Npn}
292    \tl_put_right:Nx #3 { {\int_use:N \l_tmpa_int}{
293        \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
294        \exp_after:wN\exp_after:wN\exp_after:wN {
295          \exp_after:wN #2 \_tmp_args_tl
296        }
297  }}
298  }
299  \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {cNN}
300  \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {NcN}
301  \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {ccN}
302
303  \cs_new_protected:Nn \stex_copy_control_sequence_ii:NNN {
304    \tl_set:Nx \_tmp_args_tl {\cs_argument_spec:N #2}
305    \exp_args:NNo \tl_remove_all:Nn \_tmp_args_tl \c_hash_str
306    \int_set:Nn \l_tmpa_int {\tl_count:N \_tmp_args_tl}
307
308    \tl_clear:N \_tmp_args_tl
309    \int_step_inline:nn \l_tmpa_int {
310      \tl_put_right:Nx \_tmp_args_tl {{\exp_not:n{########}\exp_not:n{##1}}}
311    }
312
313    \edef \_tmp_args_tl {
314      \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
315      \exp_after:wN\exp_after:wN\exp_after:wN {
316        \exp_after:wN #2 \_tmp_args_tl
317      }
318    }
319
320    \exp_after:wN \def \exp_after:wN \_tmp_args_tl
321    \exp_after:wN ##\exp_after:wN 1 \exp_after:wN ##\exp_after:wN 2
322    \exp_after:wN  { \_tmp_args_tl }
323
324    \edef \_tmp_args_tl {
325      \exp_after:wN \exp_not:n \exp_after:wN {
326        \_tmp_args_tl {####1}{####2}
327      }
328    }
329
330    \tl_set:Nn #3 {\cs_generate_from_arg_count:NNnn #1 \cs_set:Npn}
331    \tl_put_right:Nx #3 { {\int_use:N \l_tmpa_int}{
332      \exp_after:wN\exp_not:n\exp_after:wN{\_tmp_args_tl}
333  }}
```

107

```
334  }
335
336  \cs_generate_variant:Nn \stex_copy_control_sequence_ii:NNN {cNN}
337  \cs_generate_variant:Nn \stex_copy_control_sequence_ii:NNN {NcN}
338  \cs_generate_variant:Nn \stex_copy_control_sequence_ii:NNN {ccN}
```

(*End definition for* \ignorespacesandpars. *This function is documented on page 72.*)

\MMTrule
```
339  \NewDocumentCommand \MMTrule {m m}{
340    \seq_set_split:Nnn \l_tmpa_seq , {#2}
341    \int_zero:N \l_tmpa_int
342    \stex_annotate_invisible:nnn{mmtrule}{scala://#1}{
343      \seq_if_empty:NF \l_tmpa_seq {
344        $\seq_map_inline:Nn \l_tmpa_seq {
345          \int_incr:N \l_tmpa_int
346          \stex_annotate:nnn{arg}{i\int_use:N \l_tmpa_int}{##1}
347        }$
348      }
349    }
350  }
351
352  \NewDocumentCommand \MMTinclude {m}{
353    \stex_annotate_invisible:nnn{import}{#1}{}
354  }
355
356  \tl_new:N \g_stex_document_title
357  \cs_new_protected:Npn \STEXtitle #1 {
358    \tl_if_empty:NT \g_stex_document_title {
359      \tl_gset:Nn \g_stex_document_title { #1 }
360    }
361  }
362  \cs_new_protected:Nn \stex_document_title:n {
363    \tl_if_empty:NT \g_stex_document_title {
364      \tl_gset:Nn \g_stex_document_title { #1 }
365      \stex_annotate_invisible:n{\noindent
366        \stex_annotate:nnn{doctitle}{}{ #1 }
367      \par}
368    }
369  }
370  \AtBeginDocument {
371    \let \STEXtitle \stex_document_title:n
372    \tl_if_empty:NF \g_stex_document_title {
373      \stex_annotate_invisible:n{\noindent
374        \stex_annotate:nnn{doctitle}{}{ \g_stex_document_title }
375      \par}
376    }
377    \let\_stex_maketitle:\maketitle
378    \def\maketitle{
379      \tl_if_empty:NF \@title {
380        \exp_args:No \stex_document_title:n \@title
381      }
382      \_stex_maketitle:
383    }
```

```
384 }
385
386 \cs_new_protected:Nn \stex_par: {
387   \mode_if_vertical:F{
388     \if@minipage\else\if@nobreak\else\par\fi\fi
389   }
390 }
391
392 ⟨/package⟩
```

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

Chapter 25

sTEX -MathHub Implementation

```
393 ⟨∗package⟩
394
395 %%%%%%%%%%%%    mathhub.dtx    %%%%%%%%%%%%
396
397 ⟨@@=stex_path⟩
```

Warnings and error messages

```
398 \msg_new:nnn{stex}{error/norepository}{
399   No~archive~#1~found~in~#2
400 }
401 \msg_new:nnn{stex}{error/notinarchive}{
402   Not~currently~in~an~archive,~but~\detokenize{#1}~
403   needs~one!
404 }
405 \msg_new:nnn{stex}{error/nofile}{
406   \detokenize{#1}~could~not~find~file~#2
407 }
408 \msg_new:nnn{stex}{error/twofiles}{
409   \detokenize{#1}~found~two~candidates~for~#2
410 }
```

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

```
411 \cs_new_protected:Nn \stex_path_from_string:Nn {
412   \stex_debug:nn{files}{#2}
413   \str_set:Nx \l_tmpa_str { #2 }
414   \str_if_empty:NTF \l_tmpa_str {
415     \seq_clear:N #1
416   }{
417     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
418     \sys_if_platform_windows:T{
```

```
419     \seq_clear:N \l_tmpa_tl
420     \seq_map_inline:Nn #1 {
421       \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
422       \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
423     }
424     \seq_set_eq:NN #1 \l_tmpa_tl
425   }
426   \stex_path_canonicalize:N #1
427   }
428   \stex_debug:nn{files}{Yields: \stex_path_to_string:N#1}
429 }
430
```

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

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

```
431 \cs_new_protected:Nn \stex_path_to_string:NN {
432   \exp_args:NNe \str_set:Nn #2 { \seq_use:Nn #1 / }
433 }
434
435 \cs_new:Nn \stex_path_to_string:N {
436   \seq_use:Nn #1 /
437 }
```

(*End definition for* `\stex_path_to_string:NN` *and* `\stex_path_to_string:N`*. These functions are documented on page 73.*)

`\c__stex_path_dot_str`
`\c__stex_path_up_str`

. and .., respectively.

```
438 \str_const:Nn \c__stex_path_dot_str {.}
439 \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.

```
440 \cs_new_protected:Nn \stex_path_canonicalize:N {
441   \seq_if_empty:NF #1 {
442     \seq_clear:N \l_tmpa_seq
443     \seq_get_left:NN #1 \l_tmpa_tl
444     \str_if_empty:NT \l_tmpa_tl {
445       \seq_put_right:Nn \l_tmpa_seq {}
446     }
447     \seq_map_inline:Nn #1 {
448       \str_set:Nn \l_tmpa_tl { ##1 }
449       \str_if_eq:NNF \l_tmpa_tl \c__stex_path_dot_str {
450         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
451           \seq_if_empty:NTF \l_tmpa_seq {
452             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
453               \c__stex_path_up_str
454             }
455           }{
456             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
457             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
458               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
459                 \c__stex_path_up_str
```

```
460                              }
461                          }{
462                              \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
463                          }
464                      }
465                  }{
466                      \str_if_empty:NF \l_tmpa_tl {
467                          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
468                      }
469                  }
470              }
471          }
472          \seq_gset_eq:NN #1 \l_tmpa_seq
473      }
474  }
```

(End definition for \stex_path_canonicalize:N. This function is documented on page 73.)

\stex_path_if_absolute_p:N
\stex_path_if_absolute:N*TF*

```
475  \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
476      \seq_if_empty:NTF #1 {
477          \prg_return_false:
478      }{
479          \seq_get_left:NN #1 \l_tmpa_tl
480          \sys_if_platform_windows:TF{
481              \str_if_in:NnTF \l_tmpa_tl {:}{
482                  \prg_return_true:
483              }{
484                  \prg_return_false:
485              }
486          }{
487              \str_if_empty:NTF \l_tmpa_tl {
488                  \prg_return_true:
489              }{
490                  \prg_return_false:
491              }
492          }
493      }
494  }
```

(End definition for \stex_path_if_absolute:NTF. This function is documented on page 73.)

25.2 PWD and kpsewhich

\stex_kpsewhich:n

```
495  \str_new:N\l_stex_kpsewhich_return_str
496  \cs_new_protected:Nn \stex_kpsewhich:n {\begingroup
497      \catcode`\ =12
498      \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
499      \tl_gset_eq:NN \l_tmpa_tl \l_tmpa_tl
500      \endgroup
501      \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
502      \tl_trim_spaces:N \l_stex_kpsewhich_return_str
503  }
```

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

We determine the PWD

\c_stex_pwd_seq
\c_stex_pwd_str

```
504 \sys_if_platform_windows:TF{
505   \begingroup\escapechar=-1\catcode`\\=12
506   \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
507   \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
508   \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_
509 }{
510   \stex_kpsewhich:n{-var-value~PWD}
511 }
512
513 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
514 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
515 \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* *73.*)

25.3   File Hooks and Tracking

```
516 ⟨@@=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

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

(*End definition for* \g__stex_files_stack.)

\c_stex_mainfile_seq
\c_stex_mainfile_str

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

(*End definition for* \c_stex_mainfile_seq *and* \c_stex_mainfile_str. *These variables are documented on page* *73.*)

\g_stex_currentfile_seq

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

(*End definition for* \g_stex_currentfile_seq. *This variable is documented on page* *74.*)

\stex_filestack_push:n

```
522 \cs_new_protected:Nn \stex_filestack_push:n {
523   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
524   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
525     \stex_path_from_string:Nn\g_stex_currentfile_seq{
526       \c_stex_pwd_str/#1
527     }
```

```
528     }
529     \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
530     \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
531     \stex_get_document_uri:
532 }
```

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

```
533 \cs_new_protected:Nn \stex_filestack_pop: {
534     \seq_if_empty:NF\g__stex_files_stack{
535         \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
536     }
537     \seq_if_empty:NTF\g__stex_files_stack{
538         \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
539     }{
540         \seq_get:NN\g__stex_files_stack\l_tmpa_seq
541         \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
542     }
543     \stex_get_document_uri:
544 }
```

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

Hooks for the current file:

```
545 \AddToHook{file/before}{
546     \tl_if_empty:NTF\CurrentFilePath{
547         \stex_filestack_push:n{\CurrentFile}
548     }{
549         \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
550     }
551 }
552 \AddToHook{file/after}{
553     \stex_filestack_pop:
554 }
```

25.4   MathHub Repositories

```
555 ⟨@@=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.

```
556 \str_if_empty:NTF\mathhub{
557     \sys_if_platform_windows:TF{
558         \begingroup\escapechar=-1\catcode'\\=12
559         \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str MATHHUB\c_percent_str}
560         \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
561         \exp_args:NNx\str_if_eq:onT\l_stex_kpsewhich_return_str{\c_percent_str MATHHUB\c_percent
562         \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_ste
563     }{
564         \stex_kpsewhich:n{-var-value~MATHHUB}
565     }
566     \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
567
```

```
568     \str_if_empty:NT \c_stex_mathhub_str {
569       \sys_if_platform_windows:TF{
570         \begingroup\escapechar=-1\catcode'\\=12
571         \exp_args:Nx\stex_kpsewhich:n{-var-value~HOME}
572         \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
573         \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_s
574       }{
575         \stex_kpsewhich:n{-var-value~HOME}
576       }
577       \ior_open:NnT \g_tmpa_ior{\l_stex_kpsewhich_return_str / .stex / mathhub.path}{
578         \begingroup\escapechar=-1\catcode'\\=12
579         \ior_str_get:NN \g_tmpa_ior \l_tmpa_str
580         \sys_if_platform_windows:T{
581           \exp_args:NNx\str_replace_all:Nnn\l_tmpa_str{\c_backslash_str}/
582         }
583         \str_gset_eq:NN \c_stex_mathhub_str\l_tmpa_str
584         \endgroup
585         \ior_close:N \g_tmpa_ior
586       }
587     }
588     \str_if_empty:NTF\c_stex_mathhub_str{
589       \msg_warning:nn{stex}{warning/nomathhub}
590     }{
591       \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
592       \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
593     }
594   }{
595     \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
596     \stex_path_if_absolute:NF \c_stex_mathhub_seq {
597       \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
598         \c_stex_pwd_str/\mathhub
599       }
600     }
601     \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
602     \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
603   }
```

(*End definition for* `\mathhub`, `\c_stex_mathhub_seq`, *and* `\c_stex_mathhub_str`. *These variables are documented on page* *74.*)

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

```
604   \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
605     \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
606       \str_set:Nx \l_tmpa_str { #1 }
607       \prop_new:c { c_stex_mathhub_#1_manifest_prop }
608       \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
609       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
610       \__stex_mathhub_find_manifest:N \l_tmpa_seq
611       \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
612         \msg_error:nnxx{stex}{error/norepository}{#1}{
613           \stex_path_to_string:N \c_stex_mathhub_str
614         }
615         \input{Fatal~Error!}
```

```
616        } {
617          \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
618        }
619      }
620    }
```

(*End definition for* __stex_mathhub_do_manifest:n.)

\l__stex_mathhub_manifest_file_seq

```
621  \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:

```
622  \cs_new_protected:Nn \__stex_mathhub_find_manifest:N {
623    \seq_set_eq:NN\l_tmpa_seq #1
624    \bool_set_true:N\l_tmpa_bool
625    \bool_while_do:Nn \l_tmpa_bool {
626      \seq_if_empty:NTF \l_tmpa_seq {
627        \bool_set_false:N\l_tmpa_bool
628      }{
629        \file_if_exist:nTF{
630          \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
631        }{
632          \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
633          \bool_set_false:N\l_tmpa_bool
634        }{
635          \file_if_exist:nTF{
636            \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
637          }{
638            \seq_put_right:Nn\l_tmpa_seq{META-INF}
639            \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
640            \bool_set_false:N\l_tmpa_bool
641          }{
642            \file_if_exist:nTF{
643              \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
644            }{
645              \seq_put_right:Nn\l_tmpa_seq{meta-inf}
646              \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
647              \bool_set_false:N\l_tmpa_bool
648            }{
649              \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
650            }
651          }
652        }
653      }
654    }
655    \seq_set_eq:NN\l__stex_mathhub_manifest_file_seq\l_tmpa_seq
656  }
```

(*End definition for* __stex_mathhub_find_manifest:N.)

\c__stex_mathhub_manifest_ior   File variable used for MANIFEST-files

```
657  \ior_new:N \c__stex_mathhub_manifest_ior
```

116

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

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

```
658 \cs_new_protected:Nn \__stex_mathhub_parse_manifest:n {
659   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
660   \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
661   \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
662     \str_set:Nn \l_tmpa_str {##1}
663     \exp_args:NNoo \seq_set_split:Nnn
664        \l_tmpb_seq \c_colon_str \l_tmpa_str
665     \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
666       \exp_args:NNe \str_set:Nn \l_tmpb_tl {
667         \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
668       }
669       \exp_args:No \str_case:nnTF \l_tmpa_tl {
670         {id} {
671           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
672             { id } \l_tmpb_tl
673         }
674         {narration-base} {
675           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
676             { narr } \l_tmpb_tl
677         }
678         {url-base} {
679           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
680             { docurl } \l_tmpb_tl
681         }
682         {source-base} {
683           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
684             { ns } \l_tmpb_tl
685         }
686         {ns} {
687           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
688             { ns } \l_tmpb_tl
689         }
690         {dependencies} {
691           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
692             { deps } \l_tmpb_tl
693         }
694       }{}{}
695     }{}
696   }
697   \ior_close:N \c__stex_mathhub_manifest_ior
698   \stex_persist:x {
699     \prop_set_from_keyval:cn{ c_stex_mathhub_#1_manifest_prop }{
700       \exp_after:wN \prop_to_keyval:N \csname c_stex_mathhub_#1_manifest_prop\endcsname
701     }
702   }
703 }
```

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

`\stex_set_current_repository:n`

```
704 \cs_new_protected:Nn \stex_set_current_repository:n {
```

117

```
705    \stex_require_repository:n { #1 }
706    \prop_set_eq:Nc \l_stex_current_repository_prop {
707      c_stex_mathhub_#1_manifest_prop
708    }
709  }
```

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

```
710  \cs_new_protected:Nn \stex_require_repository:n {
711    \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
712      \stex_debug:nn{mathhub}{Opening~archive:~#1}
713      \__stex_mathhub_do_manifest:n { #1 }
714    }
715  }
```

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

`\l_stex_current_repository_prop` Current MathHub repository

```
716  %\prop_new:N \l_stex_current_repository_prop
717  \bool_if:NF \c_stex_persist_mode_bool {
718    \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
719    \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
720      \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
721    } {
722      \__stex_mathhub_parse_manifest:n { main }
723      \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
724        \l_tmpa_str
725      \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str _manifest_prop }
726        \c_stex_mathhub_main_manifest_prop
727      \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
728      \stex_debug:nn{mathhub}{Current~repository:~
729        \prop_item:Nn \l_stex_current_repository_prop {id}
730    }
731  }
732  }
```

(*End definition for* `\l_stex_current_repository_prop`. *This variable is documented on page 74.*)

`\stex_in_repository:nn` Executes the code in the second argument in the context of the repository whose ID is provided as the first argument.

```
733  \cs_new_protected:Nn \stex_in_repository:nn {
734    \str_set:Nx \l_tmpa_str { #1 }
735    \cs_set:Npn \l_tmpa_cs ##1 { #2 }
736    \str_if_empty:NTF \l_tmpa_str {
737      \prop_if_exist:NTF \l_stex_current_repository_prop {
738        \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
739        \exp_args:Ne \l_tmpa_cs{
740          \prop_item:Nn \l_stex_current_repository_prop { id }
741        }
742      }{
743        \l_tmpa_cs{}
744      }
745    }{
```

```
746    \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}
747    \stex_require_repository:n \l_tmpa_str
748    \str_set:Nx \l_tmpa_str { #1 }
749    \exp_args:Nne \use:nn {
750      \stex_set_current_repository:n \l_tmpa_str
751      \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
752    }{
753      \stex_debug:nn{mathhub}{switching~back~to:~
754        \prop_if_exist:NTF \l_stex_current_repository_prop {
755          \prop_item:Nn \l_stex_current_repository_prop { id }:~
756          \meaning\l_stex_current_repository_prop
757        }{
758          no~repository
759        }
760      }
761      \prop_if_exist:NTF \l_stex_current_repository_prop {
762       \stex_set_current_repository:n {
763         \prop_item:Nn \l_stex_current_repository_prop { id }
764       }
765      }{
766        \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
767      }
768    }
769   }
770 }
```

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

25.5   Using Content in Archives

```
771 \def \mhpath #1 #2 {
772   \exp_args:Ne \tl_if_empty:nTF{#1}{
773     \c_stex_mathhub_str /
774       \prop_item:Nn \l_stex_current_repository_prop { id }
775       / source / #2
776   }{
777     \c_stex_mathhub_str / #1 / source / #2
778   }
779 }
```

(End definition for `\mhpath`*. This function is documented on page 75.)*

```
780 \newif \ifinputref \inputreffalse
781
782 \cs_new_protected:Nn \__stex_mathhub_mhinput:nn {
783   \stex_in_repository:nn {#1} {
784     \ifinputref
785       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
786     \else
787       \inputreftrue
788       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
```

119

```
789        \inputreffalse
790      \fi
791    }
792 }
793 \NewDocumentCommand \mhinput { O{} m}{
794   \__stex_mathhub_mhinput:nn{ #1 }{ #2 }
795 }
796
797 \cs_new_protected:Nn \__stex_mathhub_inputref:nn {
798   \stex_in_repository:nn {#1} {
799     \stex_html_backend:TF {
800       \str_clear:N \l_tmpa_str
801       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
802         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
803       }
804
805       \tl_if_empty:nTF{ ##1 }{
806         \IfFileExists{#2}{
807           \stex_annotate_invisible:nnn{inputref}{
808             \l_tmpa_str / #2
809           }{}
810         }{
811           \input{#2}
812         }
813       }{
814         \IfFileExists{ \c_stex_mathhub_str / ##1 / source / #2 }{
815           \stex_annotate_invisible:nnn{inputref}{
816             \l_tmpa_str / #2
817           }{}
818         }{
819           \input{ \c_stex_mathhub_str / ##1 / source / #2 }
820         }
821       }
822
823     }{
824       \begingroup
825         \inputreftrue
826         \tl_if_empty:nTF{ ##1 }{
827           \input{#2}
828         }{
829           \input{ \c_stex_mathhub_str / ##1 / source / #2 }
830         }
831       \endgroup
832     }
833   }
834 }
835 \NewDocumentCommand \inputref { O{} m}{
836   \__stex_mathhub_inputref:nn{ #1 }{ #2 }
837 }
```

(*End definition for* \inputref *and* \mhinput*. These functions are documented on page 75.*)

\addmhbibresource

```
838 \cs_new_protected:Nn \__stex_mathhub_mhbibresource:nn {
```

```
839      \stex_in_repository:nn {#1} {
840        \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
841      }
842  }
843  \newcommand\addmhbibresource[2][]{
844    \__stex_mathhub_mhbibresource:nn{ #1 }{ #2 }
845  }
```

(*End definition for* \addmhbibresource. *This function is documented on page 75.*)

\libinput

```
846  \cs_new_protected:Npn \libinput #1 {
847    \prop_if_exist:NF \l_stex_current_repository_prop {
848      \msg_error:nnn{stex}{error/notinarchive}\libinput
849    }
850    \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
851      \msg_error:nnn{stex}{error/notinarchive}\libinput
852    }
853    \seq_clear:N \l__stex_mathhub_libinput_files_seq
854    \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
855    \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
856
857    \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
858      \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #1.tex}
859      \IfFileExists{ \l_tmpa_str }{
860        \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
861      }{}
862      \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
863      \seq_put_right:No \l_tmpa_seq \l_tmpa_str
864    }
865
866    \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}
867    \IfFileExists{ \l_tmpa_str }{
868      \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
869    }{}
870
871    \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
872      \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
873    }{
874      \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
875        \input{ ##1 }
876      }
877    }
878  }
```

(*End definition for* \libinput. *This function is documented on page 75.*)

\libusepackage

```
879  \NewDocumentCommand \libusepackage {O{} m} {
880    \prop_if_exist:NF \l_stex_current_repository_prop {
881      \msg_error:nnn{stex}{error/notinarchive}\libusepackage
882    }
883    \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
884      \msg_error:nnn{stex}{error/notinarchive}\libusepackage
885    }
```

121

```
886  \seq_clear:N \l__stex_mathhub_libinput_files_seq
887  \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
888  \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
889
890  \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
891    \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #2}
892    \IfFileExists{ \l_tmpa_str.sty }{
893      \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
894    }{}
895    \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
896    \seq_put_right:No \l_tmpa_seq \l_tmpa_str
897  }
898
899  \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2}
900  \IfFileExists{ \l_tmpa_str.sty }{
901    \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
902  }{}
903
904  \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
905    \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
906  }{
907    \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
908      \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
909        \usepackage[#1]{ ##1 }
910      }
911    }{
912      \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
913    }
914  }
915 }
```

(*End definition for* \libusepackage. *This function is documented on page 75.*)

```
916
917 \AddToHook{begindocument}{
918 \ltx@ifpackageloaded{graphicx}{
919    \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
920    \providecommand\mhgraphics[2][]{%
921      \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
922      \includegraphics[#1]{\mhpath\Gin@mhrepos{#2}}}
923    \providecommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
924  }{}
```

(*End definition for* \mhgraphics *and* \cmhgraphics. *These functions are documented on page 75.*)

```
925 \ltx@ifpackageloaded{listings}{
926    \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
927    \newcommand\lstinputmhlisting[2][]{%
928      \def\lst@mhrepos{}\setkeys{lst}{#1}%
929      \lstinputlisting[#1]{\mhpath\lst@mhrepos{#2}}}
930    \newcommand\clstinputmhlisting[2][]{\begin{center}\lstinputmhlisting[#1]{#2}\end{center}
931  }{}
932 }
```

934 ⟨/package⟩

(*End definition for* \lstinputmhlisting *and* \clstinputmhlisting*. These functions are documented on page* 75*.*)

Chapter 26

SₜₑX
-References Implementation

```
935  ⟨∗package⟩
936
937  %%%%%%%%%%%%   stex-references.dtx   %%%%%%%%%%%%
938
939  ⟨@@=stex_refs⟩
```

Warnings and error messages
```
940  \msg_new:nnn{stex}{error/extrefmissing}{
941    Missing~in~or~cite~value~for~\detokenize{\extref}!
942  }
943  \msg_new:nnn{stex}{warning/smsmissing}{
944    .sref~file~#1~doesn't~exist!
945  }
946  \msg_new:nnn{stex}{warning/smslabelmissing}{
947    No~label~#2~in~.sref~file~#1!
948  }
```

References are stored in the file `\jobname.sref`, to enable cross-referencing external documents.
```
949  \iow_new:N \c__stex_refs_refs_iow
950  \AtBeginDocument{
951    \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
952  }
953  \AtEndDocument{
954    \iow_close:N \c__stex_refs_refs_iow
955  }
```

26.1   Document URIs and URLs

\l_stex_current_docns_str

```
956  \str_new:N \l_stex_current_docns_str
```

(*End definition for* \l_stex_current_docns_str. *This variable is documented on page 76.*)

\stex_get_document_uri:

```
957  \cs_new_protected:Nn \stex_get_document_uri: {
```

```
958    \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
959    \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
960    \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
961    \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
962    \seq_put_right:No \l_tmpa_seq \l_tmpb_str
963
964    \str_clear:N \l_tmpa_str
965    \prop_if_exist:NT \l_stex_current_repository_prop {
966      \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
967        \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
968      }
969    }
970
971    \str_if_empty:NTF \l_tmpa_str {
972      \str_set:Nx \l_stex_current_docns_str {
973        file:/\stex_path_to_string:N \l_tmpa_seq
974      }
975    }{
976      \bool_set_true:N \l_tmpa_bool
977      \bool_while_do:Nn \l_tmpa_bool {
978        \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
979        \exp_args:No \str_case:nnTF { \l_tmpb_str } {
980          {source} { \bool_set_false:N \l_tmpa_bool }
981        }{}{
982          \seq_if_empty:NT \l_tmpa_seq {
983            \bool_set_false:N \l_tmpa_bool
984          }
985        }
986      }
987
988      \seq_if_empty:NTF \l_tmpa_seq {
989        \str_gset_eq:NN \l_stex_current_docns_str \l_tmpa_str
990      }{
991        \str_gset:Nx \l_stex_current_docns_str {
992          \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
993        }
994      }
995    }
996    %\stex_get_document_url:
997 }
```

(End definition for \stex_get_document_uri:. This function is documented on page 76.)

\l_stex_current_docurl_str

```
998 \str_new:N \l_stex_current_docurl_str
```

(End definition for \l_stex_current_docurl_str. This variable is documented on page 76.)

\stex_get_document_url:

```
999  \cs_new_protected:Nn \stex_get_document_url: {
1000    \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1001    \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1002    \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1003    \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1004    \seq_put_right:No \l_tmpa_seq \l_tmpb_str
```

```
1005
1006    \str_clear:N \l_tmpa_str
1007    \prop_if_exist:NT \l_stex_current_repository_prop {
1008      \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
1009        \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
1010          \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
1011        }
1012      }
1013    }
1014
1015    \str_if_empty:NTF \l_tmpa_str {
1016      \str_set:Nx \l_stex_current_docurl_str {
1017        file:/\stex_path_to_string:N \l_tmpa_seq
1018      }
1019    }{
1020      \bool_set_true:N \l_tmpa_bool
1021      \bool_while_do:Nn \l_tmpa_bool {
1022        \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1023        \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1024          {source} { \bool_set_false:N \l_tmpa_bool }
1025        }{}{
1026          \seq_if_empty:NT \l_tmpa_seq {
1027            \bool_set_false:N \l_tmpa_bool
1028          }
1029        }
1030      }
1031
1032      \seq_if_empty:NTF \l_tmpa_seq {
1033        \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
1034      }{
1035        \str_set:Nx \l_stex_current_docurl_str {
1036          \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
1037        }
1038      }
1039    }
1040  }
```

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

26.2   Setting Reference Targets

```
1041  \str_const:Nn \c__stex_refs_url_str{URL}
1042  \str_const:Nn \c__stex_refs_ref_str{REF}
1043  \str_new:N \l__stex_refs_curr_label_str
1044  % @currentlabel -> number
1045  % @currentlabelname -> title
1046  % @currentHref -> name.number <- id of some kind
1047  % @currentcounter <- name/id
1048  % \#autorefname <- "Section"
1049  % \theH# -> \arabic{section}
1050  % \the#  -> number
1051  % \hyper@makecurrent{#}
1052  \int_new:N \l__stex_refs_unnamed_counter_int
```

Restoring references from `.sref`-files

\STEXInternalSrefRestoreTarget

```
1053 \cs_new_protected:Npn \STEXInternalSrefRestoreTarget #1#2#3#4#5 {}
```

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

\stex_ref_new_doc_target:n

```
1054 \seq_new:N \g_stex_ref_files_seq
1055
1056 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
1057   %\stex_get_document_uri:
1058   \str_clear:N \l__stex_refs_curr_label_str
1059   \str_set:Nx \l_tmpa_str { #1 }
1060   \str_if_empty:NT \l_tmpa_str {
1061     \int_gincr:N \l__stex_refs_unnamed_counter_int
1062     \str_set:Nx \l_tmpa_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
1063   }
1064   \str_set:Nx \l__stex_refs_curr_label_str {
1065     \l_stex_current_docns_str?\l_tmpa_str
1066   }
1067
1068   \exp_args:Noo \STEXInternalAuxAddDocRef\l_stex_current_docns_str\l_tmpa_str
1069
1070   %\seq_if_exist:cF{g__stex_refs_labels_\l_tmpa_str _seq}{
1071   %  \seq_new:c {g__stex_refs_labels_\l_tmpa_str _seq}
1072   %}
1073   %\seq_if_in:coF{g__stex_refs_labels_\l_tmpa_str _seq}\l__stex_refs_curr_label_str {
1074   %  \seq_gput_right:co{g__stex_refs_labels_\l_tmpa_str _seq}\l__stex_refs_curr_label_str
1075   %}
1076
1077
1078   \stex_if_smsmode:TF {
1079     %\stex_get_document_url:
1080     %\str_gset_eq:cN {sref_url_\l__stex_refs_curr_label_str _str}\l_stex_current_docurl_str
1081     %\str_gset_eq:cN {sref_\l__stex_refs_curr_label_str _type}\c__stex_refs_url_str
1082   }{
1083     \iow_now:Nx \c__stex_refs_refs_iow {
1084       \STEXInternalSrefRestoreTarget
1085         {\l_stex_current_docns_str}
1086         {\l_tmpa_str}
1087         {\@currentcounter}
1088         {\@currentlabel}
1089         {\tl_if_exist:NT\@currentlabelname{\exp_args:No\unexpanded\@currentlabelname}}
1090     }
1091     %\iow_now:Nx \c__stex_refs_refs_iow {
1092     %  {\l_stex_current_docns_str?\l_tmpa_str}~=~{{\use:c{\@currentcounter autorefname}~\@cu
1093     \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
1094     \immediate\write\@auxout{\STEXInternalAuxAddDocRef{\l_stex_current_docns_str}{\l_tmpa_st
1095     %\str_gset:cx {sref_\l__stex_refs_curr_label_str _type}\c__stex_refs_ref_str
1096   }
1097 }
1098 \NewDocumentCommand \slabel {m} {\stex_ref_new_doc_target:n {#1}}
```

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

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

```
1099 \cs_new_protected:Npn \STEXInternalAuxAddDocRef #1 #2 {
1100   \exp_args:NNx \seq_if_in:NnTF \g_stex_ref_files_seq {\detokenize{#1}} {
1101     \exp_args:Nnx \seq_if_in:cnF{g_stex_ref_ #1 _seq}{\detokenize{#2}}{
1102       \exp_args:Nnx \seq_gput_left:cn{g_stex_ref_ #1 _seq}{\detokenize{#2}}
1103     }
1104   }{
1105     \exp_args:NNx \seq_gput_right:Nn \g_stex_ref_files_seq {\detokenize{#1}}
1106     %\seq_if_exist:cF{g_stex_ref_ #1 _seq}{
1107       \seq_new:c{g_stex_ref_ #1 _seq} % <- seq_new throws errors??
1108     %}
1109     \exp_args:Nnx \seq_gput_left:cn{g_stex_ref_ #1 _seq}{\detokenize{#2}}
1110   }
1111
1112   %\str_set:Nn \l_tmpa_str {#1?#2}
1113   %\str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
1114   %\seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
1115   %  \seq_new:c {g__stex_refs_labels_#2_seq}
1116   %}
1117   %\seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
1118   %  \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
1119   %}
1120 }
```

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

```
1121 \AtEndDocument{
1122   \def\STEXInternalAuxAddDocRef#1 #2 {}{}
1123 }
```

`\stex_ref_new_sym_target:n`

```
1124 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
1125
1126 %  \stex_if_smsmode:TF {
1127 %    \str_if_exist:cF{sref_sym_#1_type}{
1128 %      \stex_get_document_url:
1129 %      \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str
1130 %      \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
1131 %    }
1132 %  }{
1133 %    \str_if_empty:NF \l__stex_refs_curr_label_str {
1134 %      \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
1135 %      \immediate\write\@auxout{
1136 %        \exp_not:N\expandafter\def\exp_not:N\csname \exp_not:N\detokenize{sref_sym_#1_label
1137 %          \l__stex_refs_curr_label_str
1138 %        }
1139 %      }
1140 %    }
1141 %  }
1142 }
```

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

26.3   Using References

`\sref`   Optional arguments:

```
1143
1144 \keys_define:nn { stex / sref / 1 } {
1145   archive   .str_set_x:N  = \l__stex_refs_repo_str,
1146   file      .str_set_x:N  = \l__stex_refs_file_str,
1147   % TODO get rid of this
1148   fallback  .code:n = {},
1149   pre       .code:n = {},
1150   post      .code:n = {}
1151 }
1152 \cs_new_protected:Nn \__stex_refs_args_i:n {
1153   \str_clear:N \l__stex_refs_repo_str
1154   \str_clear:N \l__stex_refs_file_str
1155   \keys_set:nn { stex / sref / 1 } { #1 }
1156 }
1157 \keys_define:nn { stex / sref / 2 } {
1158   in        .str_set_x:N  = \l__stex_refs_in_str,
1159   archive   .str_set_x:N  = \l__stex_refs_repob_str,
1160   title     .tl_set:N  = \l__stex_refs_title_tl
1161 }
1162 \cs_new_protected:Nn \__stex_refs_args_ii:n {
1163   \str_clear:N \l__stex_refs_in_str
1164   \tl_clear:N \l__stex_refs_title_tl
1165   \str_clear:N \l__stex_refs_repob_str
1166   \keys_set:nn { stex / sref / 2 } { #1 }
1167 }
```

The actual macro:

```
1168 \NewDocumentCommand \sref { O{} m O{}}{
1169   \__stex_refs_args_i:n{#1}
1170   \__stex_refs_args_ii:n{#3}
1171   \str_clear:N \l__stex_refs_uri_str
1172   \__stex_refs_find_uri:n{#2}
1173   \__stex_refs_do_sref:n{#2}
1174 }
1175 \NewDocumentCommand \extref { O{} m m}{
1176   \__stex_refs_args_i:n{#1}
1177   \__stex_refs_args_ii:n{#3}
1178   \str_if_empty:NT \l__stex_refs_in_str {
1179     \msg_error:nn{stex}{error/extrefmissing}
1180   }
1181   \str_clear:N \l__stex_refs_uri_str
1182   \__stex_refs_find_uri:n{#2}
1183   \__stex_refs_do_sref_in:n{#2}
1184 }
1185
1186 \cs_new_protected:Nn \__stex_refs_find_uri:n {
1187   \stex_debug:nn{sref}{File: \l__stex_refs_file_str^^JRepo:\l__stex_refs_repo_str}
1188   \str_if_empty:NTF \l__stex_refs_file_str {
1189     \seq_if_exist:cT{g_stex_ref_\l_stex_current_docns_str _seq}{
1190       \seq_map_inline:cn{g_stex_ref_\l_stex_current_docns_str _seq}{
1191         \str_if_eq:nnT{#1}{##1}{
```

```
1192            \str_set_eq:NN \l__stex_refs_uri_str \l_stex_current_docns_str
1193            \seq_map_break:
1194          }
1195        }
1196      }
1197      \str_if_empty:NF \l__stex_refs_uri_str {
1198        \seq_map_inline:Nn \g_stex_ref_files_seq {
1199          \seq_map_inline:cn{g_stex_ref_##1_seq}{
1200            \str_if_eq:nnT{#1}{####1}{
1201              \str_set:Nn \l__stex_refs_uri_str {##1}
1202              \seq_map_break:n{\seq_map_break:}
1203            }
1204          }
1205        }
1206      }
1207    }{
1208      \str_if_empty:NTF \l__stex_refs_repo_str {
1209        \prop_if_exist:NTF \l_stex_current_repository_prop {
1210          \prop_get:NnN \l_stex_current_repository_prop { ns } \l__stex_refs_uri_str
1211          \str_set:Nx \l__stex_refs_uri_str {\l__stex_refs_uri_str / \l__stex_refs_file_str}
1212          \stex_path_from_string:Nn \l_tmpb_seq \l__stex_refs_uri_str
1213          \str_set:Nx \l__stex_refs_uri_str {\stex_path_to_string:N \l_tmpb_seq}
1214        }{
1215          \stex_path_from_string:Nn \l_tmpb_seq {
1216            \stex_path_to_string:N \g_stex_currentfile_seq/ .. / \l__stex_refs_file_str
1217          }
1218          \str_set:Nx \l__stex_refs_uri_str {file:/\stex_path_to_string:N \l_tmpb_seq}
1219        }
1220      }{
1221        \stex_require_repository:n \l__stex_refs_repo_str
1222        \prop_get:cnN { c_stex_mathhub_\l__stex_refs_repo_str _manifest_prop } { ns } \l__stex
1223        \str_set:Nx \l__stex_refs_uri_str {\l__stex_refs_uri_str / \l__stex_refs_file_str}
1224        \stex_path_from_string:Nn \l_tmpb_seq \l__stex_refs_uri_str
1225        \str_set:Nx \l__stex_refs_uri_str {\stex_path_to_string:N \l_tmpb_seq}
1226      }
1227    }
1228 }
1229
1230 \cs_new_protected:Nn \__stex_refs_do_autoref:n{
1231   \cs_if_exist:cTF{autoref}{
1232      \exp_args:Nx\autoref{sref_#1}
1233   }{
1234      \exp_args:Nx\ref{sref_#1}
1235   }
1236 }
1237
1238 \cs_new_protected:Nn \__stex_refs_do_sref:n {
1239   \str_if_empty:NTF \l__stex_refs_uri_str {
1240     \str_if_empty:NTF \l__stex_refs_in_str {
1241       \__stex_refs_do_autoref:n{#1}
1242     }{
1243       \__stex_refs_do_sref_in:n{#1}
1244     }
1245   }{
```

```
1246    \exp_args:NNo \seq_if_in:NnTF \g_stex_ref_files_seq \l__stex_refs_uri_str {
1247        \exp_args:Nnx \seq_if_in:cnTF{g_stex_ref_\l__stex_refs_uri_str _seq}{\detokenize{#1}}{
1248          \__stex_refs_do_autoref:n{\l__stex_refs_uri_str?#1}
1249        }{
1250          \str_if_empty:NTF \l__stex_refs_in_str {
1251            \__stex_refs_do_autoref:n{#1}
1252          }{
1253            \__stex_refs_do_sref_in:n{#1}
1254          }
1255        }
1256      }{
1257        \str_if_empty:NTF \l__stex_refs_in_str {
1258          \__stex_refs_do_autoref:n{#1}
1259        }{
1260          \__stex_refs_do_sref_in:n{#1}
1261        }
1262      }
1263    }
1264 }
1265
1266 \cs_new_protected:Nn \__stex_refs_restore_target:nnnnn {
1267    \str_if_empty:NTF \l__stex_refs_uri_str {
1268      \exp_args:No \str_if_eq:nnT \l__stex_refs_id_str {#2}{
1269        \tl_set:Nn \l__stex_refs_return_tl {
1270          \use:c{#3autorefname}~#4\tl_if_empty:nF{#5}{~(#5)}~in~
1271          \tl_if_empty:nTF\l__stex_refs_title_tl{
1272            ???
1273          }\l__stex_refs_title_tl
1274        }
1275      }
1276    }{
1277      \stex_debug:nn{sref}{\l__stex_refs_uri_str{}~ == ~ #1 ~ ?}
1278      \exp_args:No \str_if_eq:nnT \l__stex_refs_uri_str {#1}{
1279        \stex_debug:nn{sref}{\l__stex_refs_id_str~ == ~ #2 ~ ?}
1280        \exp_args:No \str_if_eq:nnT \l__stex_refs_id_str {#2}{
1281          \stex_debug:nn{sref}{success!}
1282          \tl_set:Nn \l__stex_refs_return_tl {
1283            \use:c{#3autorefname}~#4\tl_if_empty:nF{#5}{~(#5)}~in~
1284            \tl_if_empty:nTF\l__stex_refs_title_tl{
1285              ???
1286            }\l__stex_refs_title_tl
1287          }
1288          \endinput
1289        }
1290      }
1291    }
1292 }
1293
1294 \cs_new_protected:Nn \__stex_refs_do_sref_in:n {
1295    \stex_debug:nn{sref}{In: \l__stex_refs_in_str^^JRepo:\l__stex_refs_repo_str}
1296    \stex_debug:nn{sref}{URI: \l__stex_refs_uri_str?#1}
1297    %\msg_warning:nnn{stex}{warning/smsmissing}{<filename>}
1298    \begingroup\catcode13=9\relax\catcode10=9\relax
1299      \str_if_empty:NTF \l__stex_refs_repob_str {
```

```
1300        \prop_if_exist:NTF \l_stex_current_repository_prop {
1301          \str_set:Nx \l_tmpa_str {
1302            \c_stex_mathhub_str /
1303            \prop_item:Nn \l_stex_current_repository_prop { id }
1304            / source / \l__stex_refs_in_str .sref
1305          }
1306        }{
1307          \str_set:Nx \l_tmpa_str {
1308            \stex_path_to_string:N \g_stex_currentfile_seq/ .. / \l__stex_refs_in_str . sref
1309          }
1310        }
1311      }{
1312        \str_set:Nx \l_tmpa_str {
1313          \c_stex_mathhub_str / \l__stex_refs_repob_str
1314          / source / \l__stex_refs_in_str . sref
1315        }
1316      }
1317      \stex_path_from_string:Nn \l_tmpb_seq \l_tmpa_str
1318      \stex_path_to_string:NN \l_tmpb_seq \l_tmpa_str
1319      \stex_debug:nn{sref}{File: \l_tmpa_str}
1320      \exp_args:No \IfFileExists \l_tmpa_str {
1321        \tl_clear:N \l__stex_refs_return_tl
1322        \str_set:Nn \l__stex_refs_id_str {#1}
1323        \let\STEXInternalSrefRestoreTarget\__stex_refs_restore_target:nnnnn
1324        \use:c{@ @ input}{\l_tmpa_str}
1325        \exp_args:No \tl_if_empty:nTF \l__stex_refs_return_tl {
1326          \exp_args:Nnno \msg_warning:nnnn{stex}{warning/smslabelmissing}\l_tmpa_str{#1}
1327          \__stex_refs_do_autoref:n{
1328            \str_if_empty:NF\l__stex_refs_uri_str{\l__stex_refs_uri_str?}#1
1329          }
1330        }{
1331          \l__stex_refs_return_tl
1332        }
1333      }{
1334        \exp_args:Nnno \msg_warning:nnn{stex}{warning/smsmissing}\l_tmpa_str
1335        \__stex_refs_do_autoref:n{
1336          \str_if_empty:NF\l__stex_refs_uri_str{\l__stex_refs_uri_str?}#1
1337        }
1338      }
1339    \endgroup
1340 }
1341
1342 % \__stex_refs_args:n { #1 }
1343 % \str_if_empty:NTF \l__stex_refs_indocument_str {
1344 %   \str_set:Nx \l_tmpa_str { #2 }
1345 %   \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
1346 %   \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
1347 %     \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
1348 %       \seq_get_left:cNF {g__stex_refs_labels_\l_tmpa_str _seq} \l_tmpa_str {
1349 %         \str_clear:N \l_tmpa_str
1350 %       }
1351 %     }{
1352 %       \str_clear:N \l_tmpa_str
1353 %     }
```

```
1354  %    }{
1355  %      \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1356  %      \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1357  %     \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
1358  %      \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
1359  %        \str_set_eq:NN \l_tmpc_str \l_tmpa_str
1360  %        \str_clear:N \l_tmpa_str
1361  %        \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str _seq} {
1362  %          \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
1363  %            \str_range:nnn { ##1 }{ -\l_tmpa_int}{ -1 }
1364  %          }{
1365  %            \seq_map_break:n {
1366  %              \str_set:Nn \l_tmpa_str { ##1 }
1367  %            }
1368  %          }
1369  %        }
1370  %      }{
1371  %        \str_clear:N \l_tmpa_str
1372  %      }
1373  %    }
1374  %   \str_if_empty:NTF \l_tmpa_str {
1375  %     \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_li
1376  %   }{
1377  %     \str_if_eq:cNTF {sref_\l_tmpa_str _type} \c__stex_refs_ref_str {
1378  %       \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1379  %         \cs_if_exist:cTF{autoref}{
1380  %           \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1381  %         }{
1382  %           \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1383  %         }
1384  %       }{
1385  %         \ltx@ifpackageloaded{hyperref}{
1386  %           \hyperref[sref_\l_tmpa_str]\l__stex_refs_linktext_tl
1387  %         }{
1388  %           \l__stex_refs_linktext_tl
1389  %         }
1390  %       }
1391  %     }{
1392  %       \ltx@ifpackageloaded{hyperref}{
1393  %         \href{\use:c{sref_url_\l_tmpa_str _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_
1394  %       }{
1395  %         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_ref
1396  %       }
1397  %     }
1398  %   }
1399  % }{
1400      % TODO
1401  % }
1402  %}
```

(*End definition for* \sref. *This function is documented on page* *77*.)

\srefsym

```
1403  \NewDocumentCommand \srefsym { O{} m}{
```

133

```
1404    \stex_get_symbol:n { #2 }
1405    \__stex_refs_sym_aux:nn{#1}{\l_stex_get_symbol_uri_str}
1406 }
1407
1408 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1409
1410 %  \str_if_exist:cTF {sref_sym_#2 _label_str }{
1411 %     \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1412 %  }{
1413 %     \__stex_refs_args:n { #1 }
1414 %     \str_if_empty:NTF \l__stex_refs_indocument_str {
1415 %        \tl_if_exist:cTF{sref_sym_#2 _type}{
1416 %           % doc uri in \l_tmpb_str
1417 %           \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1418 %           \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1419 %              % reference
1420 %              \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1421 %                 \cs_if_exist:cTF{autoref}{
1422 %                    \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1423 %                 }{
1424 %                    \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1425 %                 }
1426 %              }{
1427 %                 \ltx@ifpackageloaded{hyperref}{
1428 %                    \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1429 %                 }{
1430 %                    \l__stex_refs_linktext_tl
1431 %                 }
1432 %              }
1433 %           }{
1434 %              % URL
1435 %              \ltx@ifpackageloaded{hyperref}{
1436 %                 \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl
1437 %              }{
1438 %                 \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_r
1439 %              }
1440 %           }
1441 %        }{
1442 %           \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_
1443 %        }
1444 %     }{
1445 %        % TODO
1446 %     }
1447 %  }
1448 }
```

(*End definition for* \srefsym. *This function is documented on page 77.*)

```
1449 \cs_new_protected:Npn \srefsymuri #1 #2 { % TODO
1450    #2%\__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1451 }
```

(*End definition for* \srefsymuri. *This function is documented on page 77.*)

```
1452 ⟨/package⟩
```

Chapter 27

sTeX -Modules Implementation

```
1453 ⟨*package⟩
1454
1455 %%%%%%%%%%%%%%   modules.dtx   %%%%%%%%%%%%%
1456
1457 ⟨@@=stex_modules⟩
```

Warnings and error messages
```
1458 \msg_new:nnn{stex}{error/unknownmodule}{
1459   No~module~#1~found
1460 }
1461 \msg_new:nnn{stex}{error/syntax}{
1462   Syntax~error:~#1
1463 }
1464 \msg_new:nnn{stex}{error/siglanguage}{
1465   Module~#1~declares~signature~#2,~but~does~not~
1466   declare~its~language
1467 }
1468 \msg_new:nnn{stex}{warning/deprecated}{
1469   #1~is~deprecated;~please~use~#2~instead!
1470 }
1471
1472 \msg_new:nnn{stex}{error/conflictingmodules}{
1473   Conflicting~imports~for~module~#1
1474 }
```

\l_stex_current_module_str The current module:
```
1475 \str_new:N \l_stex_current_module_str
```

(End definition for \l_stex_current_module_str. This variable is documented on page 79.)

\l_stex_all_modules_seq Stores all available modules
```
1476 \seq_new:N \l_stex_all_modules_seq
```

(End definition for \l_stex_all_modules_seq. This variable is documented on page 79.)

```
1477 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1478   \str_if_empty:NTF \l_stex_current_module_str
1479     \prg_return_false: \prg_return_true:
1480 }
```

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

```
1481 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1482   \prop_if_exist:cTF { c_stex_module_#1_prop }
1483     \prg_return_true: \prg_return_false:
1484 }
```

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

Only allowed within modules:

```
1485 \cs_new_protected:Nn \stex_execute_in_module:n { \stex_if_in_module:T {
1486   \stex_add_to_current_module:n { #1 }
1487   \stex_do_up_to_module:n { #1 }
1488 }}
1489 \cs_generate_variant:Nn \stex_execute_in_module:n {x}
1490
1491 \cs_new_protected:Nn \stex_add_to_current_module:n {
1492   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1493 }
1494 \cs_generate_variant:Nn \stex_add_to_current_module:n {x}
1495 \cs_new_protected:Npn \STEXexport {
1496   \ExplSyntaxOn
1497   \__stex_modules_export:n
1498 }
1499 \cs_new_protected:Nn \__stex_modules_export:n {
1500   \ignorespacesandpars#1\ExplSyntaxOff
1501   \stex_add_to_current_module:n { \ignorespacesandpars#1}
1502   \stex_smsmode_do:
1503 }
1504 \let \stex_module_export_helper:n \use:n
1505 \stex_deactivate_macro:Nn \STEXexport {module~environments}
```

(*End definition for* `\stex_add_to_current_module:n` *and* `\STEXexport`. *These functions are documented on page* *79*.)

```
1506 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1507   \str_set:Nx \l_tmpa_str { #1 }
1508   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
1509 }
```

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

```
1510 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1511   \str_set:Nx \l_tmpa_str { #1 }
1512   \exp_args:Nno
```

136

```
1513    \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1514      \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1515    }
1516  }
```

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

`\stex_collect_imports:n`

```
1517  \cs_new_protected:Nn \stex_collect_imports:n {
1518    \seq_clear:N \l_stex_collect_imports_seq
1519    \__stex_modules_collect_imports:n {#1}
1520  }
1521  \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1522    \seq_map_inline:cn {c_stex_module_#1_imports} {
1523      \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1524        \__stex_modules_collect_imports:n { ##1 }
1525      }
1526    }
1527    \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1528      \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1529    }
1530  }
```

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

`\stex_do_up_to_module:n`

```
1531  \int_new:N \l__stex_modules_group_depth_int
1532  \cs_new_protected:Nn \stex_do_up_to_module:n {
1533    \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1534      #1
1535    }{
1536      #1
1537      \expandafter \tl_gset:Nn
1538      \csname l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1539      \expandafter\expandafter\expandafter\endcsname
1540      \expandafter\expandafter\expandafter { \csname
1541        l__stex_modules_aftergroup_\l_stex_current_module_str _tl\endcsname #1 }
1542      \aftergroup\__stex_modules_aftergroup_do:
1543    }
1544  }
1545  \cs_generate_variant:Nn \stex_do_up_to_module:n {x}
1546  \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1547    \stex_debug:nn{aftergroup}{\cs_meaning:c{
1548      l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1549    }}
1550    \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1551      \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1552      \tl_gclear:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1553    }{
1554      \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1555      \aftergroup\__stex_modules_aftergroup_do:
1556    }
1557  }
1558  \cs_new_protected:Nn \_stex_reset_up_to_module:n {
1559    \expandafter\let\csname l__stex_modules_aftergroup_#1_tl\endcsname\undefined
```

```
1560 }
```

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

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

```
1561
```

(*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.

```
1562 \str_new:N \l_stex_module_ns_str
1563 \str_new:N \l_stex_module_subpath_str
1564 \cs_new_protected:Nn \__stex_modules_compute_namespace:nN {
1565   \seq_set_eq:NN \l_tmpa_seq #2
1566   % split off file extension
1567   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str % <- filename
1568   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1569   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str % <- filename without suffixes
1570   \seq_put_right:No \l_tmpa_seq \l_tmpb_str % <- file path including name without suffixes
1571
1572   \bool_set_true:N \l_tmpa_bool
1573   \bool_while_do:Nn \l_tmpa_bool {
1574     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1575     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1576       {source} { \bool_set_false:N \l_tmpa_bool }
1577     }{}{
1578       \seq_if_empty:NT \l_tmpa_seq {
1579         \bool_set_false:N \l_tmpa_bool
1580       }
1581     }
1582   }
1583
1584   \stex_path_to_string:NN \l_tmpa_seq \l_stex_module_subpath_str
1585   % \l_tmpa_seq <- sub-path relative to archive
1586   \str_if_empty:NTF \l_stex_module_subpath_str {
1587     \str_set:Nx \l_stex_module_ns_str {#1}
1588   }{
1589     \str_set:Nx \l_stex_module_ns_str {
1590       #1/\l_stex_module_subpath_str
1591     }
1592   }
1593 }
1594
1595 \cs_new_protected:Nn \stex_modules_current_namespace: {
1596   \str_clear:N \l_stex_module_subpath_str
1597   \prop_if_exist:NTF \l_stex_current_repository_prop {
1598     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1599     \__stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1600   }{
1601     % split off file extension
1602     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1603     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
```

138

```
1604     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1605     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1606     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1607     \str_set:Nx \l_stex_module_ns_str {
1608       file:/\stex_path_to_string:N \l_tmpa_seq
1609     }
1610   }
1611 }
```

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

27.1   The `smodule` environment

`smodule` arguments:

```
1612 \keys_define:nn { stex / module } {
1613   title         .tl_set:N    = \smoduletitle ,
1614   type          .str_set_x:N = \smoduletype ,
1615   id            .str_set_x:N = \smoduleid ,
1616   deprecate     .str_set_x:N = \l_stex_module_deprecate_str ,
1617   ns            .str_set_x:N = \l_stex_module_ns_str ,
1618   lang          .str_set_x:N = \l_stex_module_lang_str ,
1619   sig           .str_set_x:N = \l_stex_module_sig_str ,
1620   creators      .str_set_x:N = \l_stex_module_creators_str ,
1621   contributors  .str_set_x:N = \l_stex_module_contributors_str ,
1622   meta          .str_set_x:N = \l_stex_module_meta_str ,
1623   srccite       .str_set_x:N = \l_stex_module_srccite_str
1624 }
1625
1626 \cs_new_protected:Nn \__stex_modules_args:n {
1627   \str_clear:N \smoduletitle
1628   \str_clear:N \smoduletype
1629   \str_clear:N \smoduleid
1630   \str_clear:N \l_stex_module_ns_str
1631   \str_clear:N \l_stex_module_deprecate_str
1632   \str_clear:N \l_stex_module_lang_str
1633   \str_clear:N \l_stex_module_sig_str
1634   \str_clear:N \l_stex_module_creators_str
1635   \str_clear:N \l_stex_module_contributors_str
1636   \str_clear:N \l_stex_module_meta_str
1637   \str_clear:N \l_stex_module_srccite_str
1638   \keys_set:nn { stex / module } { #1 }
1639 }
1640
1641 % module parameters here? In the body?
1642
```

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

```
1643 \cs_new_protected:Nn \stex_module_setup:nn {
1644   \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1645   \str_set:Nx \l_stex_module_name_str { #2 }
1646   \__stex_modules_args:n { #1 }
```

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

```
1647   \stex_if_in_module:TF {
1648     % Nested module
1649     \prop_get:cnN {c_stex_module_\l_stex_current_module_str _prop}
1650       { ns } \l_stex_module_ns_str
1651     \str_set:Nx \l_stex_module_name_str {
1652       \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
1653         { name } / \l_stex_module_name_str
1654     }
1655     \str_if_empty:NT \l_stex_module_lang_str {
1656       \str_set:Nx \l_stex_module_lang_str {
1657         \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
1658           { lang }
1659       }
1660     }
1661   }{
1662     % not nested:
1663     \str_if_empty:NT \l_stex_module_ns_str {
1664       \stex_modules_current_namespace:
1665       \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1666         / {\l_stex_module_ns_str}
1667       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1668       \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1669         \str_set:Nx \l_stex_module_ns_str {
1670           \stex_path_to_string:N \l_tmpa_seq
1671         }
1672       }
1673     }
1674   }
```

Next, we determine the language of the module:

```
1675   \str_if_empty:NT \l_stex_module_lang_str {
1676     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1677     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1678     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1679     \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
1680       \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
1681         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
1682       }
1683     }
1684     \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1685     \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
1686       \seq_pop_right:NN \l_tmpa_seq \l_stex_module_lang_str
1687       \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1688         inferred~from~file~name}
1689     }
1690   }
1691
1692   \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
1693     \exp_args:NNo \stex_set_language:Nn \l_tmpa_str \l_stex_module_lang_str
1694   }}
```

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

```
1695    \str_if_empty:NTF \l_stex_module_sig_str {
1696      \exp_args:Nnx \prop_gset_from_keyval:cn {
1697        c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1698      } {
1699        name      = \l_stex_module_name_str ,
1700        ns        = \l_stex_module_ns_str ,
1701        file      = \exp_not:o { \g_stex_currentfile_seq } ,
1702        lang      = \l_stex_module_lang_str ,
1703        sig       = \l_stex_module_sig_str ,
1704        deprecate = \l_stex_module_deprecate_str ,
1705        meta      = \l_stex_module_meta_str
1706      }
1707      \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1708      \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1709      \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _copymodules}
1710      \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1711      \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
```

We load the metatheory:

```
1712      \str_if_empty:NT \l_stex_module_meta_str {
1713        \str_set:Nx \l_stex_module_meta_str {
1714          \c_stex_metatheory_ns_str ? Metatheory
1715        }
1716      }
1717      \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1718        \bool_set_true:N \l_stex_in_meta_bool
1719        \exp_args:Nx \stex_add_to_current_module:n {
1720          \bool_set_true:N \l_stex_in_meta_bool
1721          \stex_activate_module:n {\l_stex_module_meta_str}
1722          \bool_set_false:N \l_stex_in_meta_bool
1723        }
1724        \stex_activate_module:n {\l_stex_module_meta_str}
1725        \bool_set_false:N \l_stex_in_meta_bool
1726      }
1727    }{
1728      \str_if_empty:NT \l_stex_module_lang_str {
1729        \msg_error:nnxx{stex}{error/siglanguage}{
1730          \l_stex_module_ns_str?\l_stex_module_name_str
1731        }{\l_stex_module_sig_str}
1732      }
1733      \stex_debug:nn{modules}{Signature~\l_stex_module_sig_str~for~\l_stex_module_ns_str?\l_st
1734      \stex_if_module_exists:nTF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1735        \stex_debug:nn{modules}{(already exists)}
1736      }{
1737        \stex_debug:nn{modules}{(needs loading)}
1738        \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1739        \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1740        \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1741        \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1742        \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1743        \str_set:Nx \l_tmpa_str {
1744          \stex_path_to_string:N \l_tmpa_seq /
```

```
1745          \l_tmpa_str . \l_stex_module_sig_str .tex
1746        }
1747        \IfFileExists \l_tmpa_str {
1748          \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1749            \str_clear:N \l_stex_current_module_str
1750            \seq_clear:N \l_stex_all_modules_seq
1751            \stex_debug:nn{modules}{Loading~signature}
1752          }
1753        }{
1754          \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1755        }
1756      }
1757      \stex_if_smsmode:F {
1758        \stex_activate_module:n {
1759          \l_stex_module_ns_str ? \l_stex_module_name_str
1760        }
1761      }
1762      \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1763    }
1764    \str_if_empty:NF \l_stex_module_deprecate_str {
1765      \msg_warning:nnxx{stex}{warning/deprecated}{
1766        Module~\l_stex_current_module_str
1767      }{
1768        \l_stex_module_deprecate_str
1769      }
1770    }
1771    \seq_put_right:Nx \l_stex_all_modules_seq {
1772      \l_stex_module_ns_str ? \l_stex_module_name_str
1773    }
1774    \tl_clear:c{l__stex_modules_aftergroup_\l_stex_module_ns_str ? \l_stex_module_name_str _tl
1775 }
```

(*End definition for* `\stex_module_setup:nn`*. This function is documented on page 80.*)

smodule (*env.*) The module environment.

__stex_modules_begin_module: implements \begin{smodule}

```
1776 \cs_new_protected:Nn \__stex_modules_begin_module: {
1777   \stex_reactivate_macro:N \STEXexport
1778   \stex_reactivate_macro:N \importmodule
1779   \stex_reactivate_macro:N \symdecl
1780   \stex_reactivate_macro:N \notation
1781   \stex_reactivate_macro:N \symdef
1782
1783   \stex_debug:nn{modules}{
1784     New~module:\\
1785     Namespace:~\l_stex_module_ns_str\\
1786     Name:~\l_stex_module_name_str\\
1787     Language:~\l_stex_module_lang_str\\
1788     Signature:~\l_stex_module_sig_str\\
1789     Metatheory:~\l_stex_module_meta_str\\
1790     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1791   }
1792
```

```
1793    \stex_if_do_html:T{
1794      \begin{stex_annotate_env} {theory} {
1795        \l_stex_module_ns_str ? \l_stex_module_name_str
1796      }
1797
1798      \stex_annotate_invisible:nnn{header}{} {
1799        \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1800        \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1801        \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1802          \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1803        }
1804        \str_if_empty:NF \smoduletype {
1805          \stex_annotate:nnn{type}{\smoduletype}{}
1806        }
1807      }
1808    }
1809    % TODO: Inherit metatheory for nested modules?
1810 }
1811 \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}

```
1812 \cs_new_protected:Nn \__stex_modules_end_module: {
1813    \stex_debug:nn{modules}{Closing~module~\prop_item:cn {c_stex_module_\l_stex_current_module
1814    \_stex_reset_up_to_module:n \l_stex_current_module_str
1815    \stex_if_smsmode:T {
1816      \stex_persist:x {
1817        \prop_set_from_keyval:cn{c_stex_module_\l_stex_current_module_str _prop}{
1818          \exp_after:wN \prop_to_keyval:N \csname c_stex_module_\l_stex_current_module_str _pr
1819        }
1820        \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _constants}{
1821          \seq_use:cn{c_stex_module_\l_stex_current_module_str _constants},
1822        }
1823        \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _imports}{
1824          \seq_use:cn{c_stex_module_\l_stex_current_module_str _imports},
1825        }
1826        \tl_set:cn {c_stex_module_\l_stex_current_module_str _code}
1827      }
1828      \exp_after:wN \let \exp_after:wN \l_tmpa_tl \csname c_stex_module_\l_stex_current_module
1829      \exp_after:wN \stex_persist:n \exp_after:wN { \exp_after:wN { \l_tmpa_tl } }
1830    }
1831 }
```

(*End definition for* __stex_modules_end_module:.)

The core environment

```
1832 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1833 \NewDocumentEnvironment { smodule } { O{} m } {
1834    \stex_module_setup:nn{#1}{#2}
1835    %\par
1836    \stex_if_smsmode:F{
1837      \tl_if_empty:NF \smoduletitle {
1838        \exp_args:No \stex_document_title:n \smoduletitle
1839      }
```

```
1840       \tl_clear:N \l_tmpa_tl
1841       \clist_map_inline:Nn \smoduletype {
1842         \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1843           \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1844         }
1845       }
1846       \tl_if_empty:NTF \l_tmpa_tl {
1847         \__stex_modules_smodule_start:
1848       }{
1849         \l_tmpa_tl
1850       }
1851     }
1852     \__stex_modules_begin_module:
1853     \str_if_empty:NF \smoduleid {
1854       \stex_ref_new_doc_target:n \smoduleid
1855     }
1856     \stex_smsmode_do:
1857   } {
1858     \__stex_modules_end_module:
1859     \stex_if_smsmode:F {
1860       \end{stex_annotate_env}
1861       \clist_set:No \l_tmpa_clist \smoduletype
1862       \tl_clear:N \l_tmpa_tl
1863       \clist_map_inline:Nn \l_tmpa_clist {
1864         \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1865           \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1866         }
1867       }
1868       \tl_if_empty:NTF \l_tmpa_tl {
1869         \__stex_modules_smodule_end:
1870       }{
1871         \l_tmpa_tl
1872       }
1873     }
1874 }
```

\stexpatchmodule

```
1875 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1876 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1877
1878 \newcommand\stexpatchmodule[3][] {
1879     \str_set:Nx \l_tmpa_str{ #1 }
1880     \str_if_empty:NTF \l_tmpa_str {
1881       \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1882       \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1883     }{
1884       \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1885       \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1886     }
1887 }
```

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

27.2 Invoking modules

```
1888 \NewDocumentCommand \STEXModule { m } {
1889   \exp_args:NNx \str_set:Nn \l_tmpa_str { # 1 }
1890   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1891   \tl_set:Nn \l_tmpa_tl {
1892     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1893   }
1894   \seq_map_inline:Nn \l_stex_all_modules_seq {
1895     \str_set:Nn \l_tmpb_str { ##1 }
1896     \str_if_eq:eeT { \l_tmpa_str } {
1897       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1898     } {
1899       \seq_map_break:n {
1900         \tl_set:Nn \l_tmpa_tl {
1901           \stex_invoke_module:n { ##1 }
1902         }
1903       }
1904     }
1905   }
1906   \l_tmpa_tl
1907 }
1908
1909 \cs_new_protected:Nn \stex_invoke_module:n {
1910   \stex_debug:nn{modules}{Invoking~module~#1}
1911   \peek_charcode_remove:NTF ! {
1912     \__stex_modules_invoke_uri:nN { #1 }
1913   } {
1914     \peek_charcode_remove:NTF ? {
1915       \__stex_modules_invoke_symbol:nn { #1 }
1916     } {
1917       \msg_error:nnx{stex}{error/syntax}{
1918         ?~or~!~expected~after~
1919         \c_backslash_str STEXModule{#1}
1920       }
1921     }
1922   }
1923 }
1924
1925 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1926   \str_set:Nn #2 { #1 }
1927 }
1928
1929 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1930   \stex_invoke_symbol:n{#1?#2}
1931 }
```

(*End definition for* \STEXModule *and* \stex_invoke_module:n. *These functions are documented on page* *80*.)

```
1932 \bool_new:N \l_stex_in_meta_bool
1933 \bool_set_false:N \l_stex_in_meta_bool
```

145

```
1934 \cs_new_protected:Nn \stex_activate_module:n {
1935   \stex_debug:nn{modules}{Activating~module~#1}
1936   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1937     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1938     \use:c{ c_stex_module_#1_code }
1939   }
1940 }
```

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

mmtinterface (*env.*)

```
1941 \NewDocumentEnvironment { mmtinterface } { O{} m m } {
1942   \begin{smodule}[#1]{#3}
1943     \str_set:Nx \l_stex_module_mmtfor_str {#2}
1944     \MMTinclude{#2}
1945     \stex_reactivate_macro:N \mmtdecl
1946     \stex_reactivate_macro:N \mmtdef
1947 }{
1948   \end{smodule}
1949 }
```

```
1950 ⟨/package⟩
```

146

Chapter 28

sTeX -Module Inheritance Implementation

```
1951  ⟨*package⟩
1952
1953  %%%%%%%%%%%%   inheritance.dtx   %%%%%%%%%%%%
1954
```

28.1   SMS Mode

```
1955  ⟨@@=stex_smsmode⟩
```

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

```
1956  \tl_new:N \g_stex_smsmode_allowedmacros_tl
1957  \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1958  \seq_new:N \g_stex_smsmode_allowedenvs_seq
1959
1960  \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1961    \makeatletter
1962    \makeatother
1963    \ExplSyntaxOn
1964    \ExplSyntaxOff
1965    \rustexBREAK
1966  }
1967
1968  \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1969    \symdef
1970    \importmodule
1971    \notation
1972    \symdecl
1973    \STEXexport
1974    \inlineass
1975    \inlinedef
1976    \inlineex
1977    \endinput
1978    \setnotation
```

```
1979    \copynotation
1980    \assign
1981    \renamedecl
1982    \donotcopy
1983    \instantiate
1984    \textsymdecl
1985 }
1986
1987 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1988    \tl_to_str:n {
1989       smodule,
1990       copymodule,
1991       interpretmodule,
1992       realization,
1993       sdefinition,
1994       sexample,
1995       sassertion,
1996       sparagraph,
1997       mathstructure
1998    }
1999 }
```

(*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 82.*)

\stex_if_smsmode_p:
\stex_if_smsmode:*TF*

```
2000 \bool_new:N \g__stex_smsmode_bool
2001 \bool_set_false:N \g__stex_smsmode_bool
2002 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
2003    \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
2004 }
```

(*End definition for* \stex_if_smsmode:TF*. This function is documented on page 82.*)

__stex_smsmode_in_smsmode:nn

```
2005 \cs_new_protected:Nn \__stex_smsmode_in_smsmode:nn { \stex_suppress_html:n {
2006    \vbox_set:Nn \l_tmpa_box {
2007       \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
2008       \bool_gset_true:N \g__stex_smsmode_bool
2009       #2
2010       \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
2011    }
2012    \box_clear:N \l_tmpa_box
2013 } }
```

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

\stex_file_in_smsmode:nn

```
2014 \quark_new:N \q__stex_smsmode_break
2015
2016 \NewDocumentCommand \__stex_smsmode_importmodule: { O{} m} {
2017    \seq_gput_right:Nn \l__stex_smsmode_importmodules_seq {{#1}{#2}}
2018    \stex_smsmode_do:
2019 }
2020
```

148

```
2021  \cs_new_protected:Nn \__stex_smsmode_module:nn {
2022    \__stex_modules_args:n{#1}
2023    \stex_if_in_module:F {
2024      \str_if_empty:NF \l_stex_module_sig_str {
2025        \stex_modules_current_namespace:
2026        \str_set:Nx \l_stex_module_name_str { #2 }
2027        \stex_if_module_exists:nF{\l_stex_module_ns_str?\l_stex_module_name_str}{
2028          \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
2029          \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
2030          \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
2031          \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
2032          \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
2033          \str_set:Nx \l_tmpa_str {
2034            \stex_path_to_string:N \l_tmpa_seq /
2035            \l_tmpa_str . \l_stex_module_sig_str .tex
2036          }
2037          \IfFileExists \l_tmpa_str {
2038            \exp_args:NNx \seq_gput_right:Nn \l__stex_smsmode_sigmodules_seq \l_tmpa_str
2039          }{
2040            \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
2041          }
2042        }
2043      }
2044    }
2045  }
2046
2047  \prg_new_conditional:Nnn \__stex_smsmode_check_import_pair:nn {T,F,TF} {
2048    %\stex_debug:nn{import-pair}{\detokenize{{#1}~{#2}}}
2049    \tl_if_empty:nTF{#1}{
2050      \prop_if_exist:NTF \l_stex_current_repository_prop
2051        {
2052          %\stex_debug:nn{import-pair}{in repository \prop_item:Nn \l_stex_current_repository_
2053          \prg_return_true:
2054        } {
2055          \seq_set_split:Nnn \l_tmpa_seq ? {#2}
2056          \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
2057          \tl_if_empty:NT \l_tmpa_tl {
2058            \seq_pop_left:NN \l_tmpa_seq \l_tmpa_tl
2059          }
2060          %\stex_debug:nn{import-pair}{\seq_use:Nn \l_tmpa_seq,~of~length~\seq_count:N \l_tmpa
2061          \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1
2062            \prg_return_true: \prg_return_false:
2063        }
2064    }\prg_return_true:
2065  }
2066
2067  \cs_new_protected:Nn \stex_file_in_smsmode:nn {
2068    \stex_filestack_push:n{#1}
2069    \seq_gclear:N \l__stex_smsmode_importmodules_seq
2070    \seq_gclear:N \l__stex_smsmode_sigmodules_seq
2071    % ----- new --------------------------
2072    \__stex_smsmode_in_smsmode:nn{#1}{
2073      \let\importmodule\__stex_smsmode_importmodule:
2074      \let\stex_module_setup:nn\__stex_smsmode_module:nn
```

149

```
2075      \let\__stex_modules_begin_module:\relax
2076      \let\__stex_modules_end_module:\relax
2077      \seq_clear:N \g_stex_smsmode_allowedenvs_seq
2078      \exp_args:NNx \seq_put_right:Nn \g_stex_smsmode_allowedenvs_seq {\tl_to_str:n{smodule}}
2079      \tl_clear:N \g_stex_smsmode_allowedmacros_tl
2080      \tl_clear:N \g_stex_smsmode_allowedmacros_escape_tl
2081      \tl_put_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\importmodule}
2082      \everyeof{\q__stex_smsmode_break\noexpand}
2083      \expandafter\expandafter\expandafter
2084      \stex_smsmode_do:
2085      \csname @ @ input\endcsname "#1"\relax
2086
2087      \seq_map_inline:Nn \l__stex_smsmode_sigmodules_seq {
2088        \stex_filestack_push:n{##1}
2089        \expandafter\expandafter\expandafter
2090        \stex_smsmode_do:
2091        \csname @ @ input\endcsname "##1"\relax
2092        \stex_filestack_pop:
2093      }
2094    }
2095    % ----- new ---------------------------
2096    \__stex_smsmode_in_smsmode:nn{#1} {
2097      #2
2098      % ----- new ---------------------------
2099      \begingroup
2100      %\stex_debug:nn{smsmode}{Here:~\seq_use:Nn\l__stex_smsmode_importmodules_seq, }
2101      \seq_map_inline:Nn \l__stex_smsmode_importmodules_seq {
2102        \__stex_smsmode_check_import_pair:nnT ##1 { \begingroup
2103          \stex_import_module_uri:nn ##1
2104          \stex_import_require_module:nnnn
2105            \l_stex_import_ns_str
2106            \l_stex_import_archive_str
2107            \l_stex_import_path_str
2108            \l_stex_import_name_str \endgroup
2109        }
2110      }
2111      \endgroup
2112      \stex_debug:nn{smsmode}{Actually~loading~file~#1}
2113      % ----- new ---------------------------
2114      \everyeof{\q__stex_smsmode_break\noexpand}
2115      \expandafter\expandafter\expandafter
2116      \stex_smsmode_do:
2117      \csname @ @ input\endcsname "#1"\relax
2118    }
2119    \stex_filestack_pop:
2120  }
```

(*End definition for* `\stex_file_in_smsmode:nn`*. This function is documented on page 83.*)

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

```
2121 \cs_new_protected:Npn \stex_smsmode_do: {
2122   \stex_if_smsmode:T {
2123     \__stex_smsmode_do:w
```

```
2124     }
2125 }
2126 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
2127     \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 }}{
2128         \expandafter\if\expandafter\relax\noexpand#1
2129             \expandafter\__stex_smsmode_do_aux:N\expandafter#1
2130         \else\expandafter\__stex_smsmode_do:w\fi
2131     }{
2132         \__stex_smsmode_do:w %#1
2133     }
2134 }
2135 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
2136     \cs_if_eq:NNF #1 \q__stex_smsmode_break {
2137         \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
2138             #1\__stex_smsmode_do:w
2139         }{
2140             \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
2141                 #1
2142             }{
2143                 \cs_if_eq:NNTF \begin #1 {
2144                     \__stex_smsmode_check_begin:n
2145                 }{
2146                     \cs_if_eq:NNTF \end #1 {
2147                         \__stex_smsmode_check_end:n
2148                     }{
2149                         \__stex_smsmode_do:w
2150                     }
2151                 }
2152             }
2153         }
2154     }
2155 }
2156
2157 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
2158     \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
2159         \begin{#1}
2160     }{
2161         \__stex_smsmode_do:w
2162     }
2163 }
2164 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
2165     \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
2166         \end{#1}\__stex_smsmode_do:w
2167     }{
2168         \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
2169     }
2170 }
```

(*End definition for* `\stex_smsmode_do:`. *This function is documented on page 83.*)

28.2   Inheritance

```
2171 ⟨@@=stex_importmodule⟩
```

151

```
2172 \cs_new_protected:Nn \stex_import_module_uri:nn {
2173   \str_set:Nx \l_stex_import_archive_str { #1 }
2174   \str_set:Nn \l_stex_import_path_str { #2 }
2175
2176   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
2177   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
2178   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
2179
2180   \stex_modules_current_namespace:
2181   \bool_lazy_all:nTF {
2182     {\str_if_empty_p:N \l_stex_import_archive_str}
2183     {\str_if_empty_p:N \l_stex_import_path_str}
2184     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
2185   }{
2186     \str_set_eq:NN \l_stex_import_path_str \l_stex_module_subpath_str
2187     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
2188   }{
2189     \str_if_empty:NT \l_stex_import_archive_str {
2190       \prop_if_exist:NT \l_stex_current_repository_prop {
2191         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
2192       }
2193     }
2194     \str_if_empty:NTF \l_stex_import_archive_str {
2195       \str_if_empty:NF \l_stex_import_path_str {
2196         \stex_path_from_string:Nn \l_tmpb_seq {
2197           \l_stex_module_ns_str  / .. / \l_stex_import_path_str
2198         }
2199         \str_set:Nx \l_stex_import_ns_str {\stex_path_to_string:N \l_tmpb_seq}
2200         \str_replace_once:Nnn \l_stex_import_ns_str {file:/} {file://}
2201       }
2202     }{
2203       \stex_require_repository:n \l_stex_import_archive_str
2204       \prop_get:cnN { c_stex_mathhub_\l_stex_import_archive_str _manifest_prop } { ns }
2205         \l_stex_import_ns_str
2206       \str_if_empty:NF \l_stex_import_path_str {
2207         \str_set:Nx \l_stex_import_ns_str {
2208           \l_stex_import_ns_str / \l_stex_import_path_str
2209         }
2210       }
2211     }
2212   }
2213 }
```

(*End definition for* \stex_import_module_uri:nn. *This function is documented on page 84.*)

Store the return values of \stex_import_module_uri:nn.

```
2214 \str_new:N \l_stex_import_name_str
2215 \str_new:N \l_stex_import_archive_str
2216 \str_new:N \l_stex_import_path_str
2217 \str_new:N \l_stex_import_ns_str
```

(*End definition for* \l_stex_import_name_str *and others. These variables are documented on page 84.*)

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

```
2218 \cs_new_protected:Nn \stex_import_require_module:nnnn {
2219   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
2220
2221     \stex_debug:nn{requiremodule}{Here:\\~~1:~#1\\~~2:~#2\\~~3:~#3\\~~4:~#4}
2222
2223     \exp_args:NNxx \seq_set_split:Nnn \l_tmpa_seq {\tl_to_str:n{/}} {#4}
2224     \seq_get_left:NN \l_tmpa_seq \l_tmpc_str
2225
2226     %\stex_debug:nn{requiremodule}{Top~module:\l_tmpc_str}
2227
2228     % archive
2229     \str_set:Nx \l_tmpa_str { #2 }
2230     \str_if_empty:NTF \l_tmpa_str {
2231       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
2232       \seq_put_right:Nn \l_tmpa_seq {..}
2233     } {
2234       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
2235       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
2236       \seq_put_right:Nn \l_tmpa_seq { source }
2237     }
2238
2239     % path
2240     \str_set:Nx \l_tmpb_str { #3 }
2241     \str_if_empty:NTF \l_tmpb_str {
2242       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / \l_tmpc_str }
2243
2244       \ltx@ifpackageloaded{babel} {
2245         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
2246           { \languagename } \l_tmpb_str {
2247             \msg_error:nnx{stex}{error/unknownlanguage}{\languagename}
2248           }
2249       } {
2250         \str_clear:N \l_tmpb_str
2251       }
2252
2253       \stex_debug:nn{modules}{Checking~a1~\l_tmpa_str.\l_tmpb_str.tex}
2254       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2255         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2256       }{
2257         \stex_debug:nn{modules}{Checking~a2~\l_tmpa_str.tex}
2258         \IfFileExists{ \l_tmpa_str.tex }{
2259           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2260         }{
2261           % try english as default
2262           \stex_debug:nn{modules}{Checking~a3~\l_tmpa_str.en.tex}
2263           \IfFileExists{ \l_tmpa_str.en.tex }{
2264             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2265           }{
2266             \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2267           }
2268         }
2269       }
2270
```

```
2271        } {
2272          \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
2273          \seq_concat:NNN \l_tmpb_seq \l_tmpa_seq \l_tmpb_seq
2274
2275          \ltx@ifpackageloaded{babel} {
2276            \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
2277                { \languagename } \l_tmpb_str {
2278                  \msg_error:nnx{stex}{error/unknownlanguage}{\languagename}
2279                }
2280          } {
2281            \str_clear:N \l_tmpb_str
2282          }
2283
2284          \stex_path_canonicalize:N \l_tmpb_seq
2285          \stex_path_to_string:NN \l_tmpb_seq \l_tmpa_str
2286
2287          \stex_debug:nn{modules}{Checking~b1~\l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex}
2288          \IfFileExists{ \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }{
2289            \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.te
2290          }{
2291            \stex_debug:nn{modules}{Checking~b2~\l_tmpa_str/\l_tmpc_str.tex}
2292            \IfFileExists{ \l_tmpa_str/\l_tmpc_str.tex }{
2293              \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.tex }
2294            }{
2295              % try english as default
2296              \stex_debug:nn{modules}{Checking~b3~\l_tmpa_str/\l_tmpc_str.en.tex}
2297              \IfFileExists{ \l_tmpa_str/\l_tmpc_str.en.tex }{
2298                \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.en.tex }
2299              }{
2300                \stex_debug:nn{modules}{Checking~b4~\l_tmpa_str.\l_tmpb_str.tex}
2301                \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2302                  \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2303                }{
2304                  \stex_debug:nn{modules}{Checking~b4~\l_tmpa_str.tex}
2305                  \IfFileExists{ \l_tmpa_str.tex }{
2306                    \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2307                  }{
2308                    % try english as default
2309                    \stex_debug:nn{modules}{Checking~b5~\l_tmpa_str.en.tex}
2310                    \IfFileExists{ \l_tmpa_str.en.tex }{
2311                      \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2312                    }{
2313                      \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2314                    }
2315                  }
2316                }
2317              }
2318            }
2319          }
2320        }
2321
2322        \str_if_eq:eeF{\g__stex_importmodule_file_str}{\seq_use:Nn \g_stex_currentfile_seq /}{
2323          \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
2324            \seq_clear:N \l_stex_all_modules_seq
```

154

```
2325          \str_clear:N \l_stex_current_module_str
2326          \str_set:Nx \l_tmpb_str { #2 }
2327          \str_if_empty:NF \l_tmpb_str {
2328              \stex_set_current_repository:n { #2 }
2329          }
2330          \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
2331        }
2332
2333        \stex_if_module_exists:nF { #1 ? #4 } {
2334          \msg_error:nnx{stex}{error/unknownmodule}{
2335              #1?#4~(in~file~\g__stex_importmodule_file_str)
2336          }
2337        }
2338      }
2339
2340    }
2341    \stex_activate_module:n { #1 ? #4 }
2342 }
```

(End definition for \stex_import_require_module:nnnn*. This function is documented on page 84.)*

\importmodule

```
2343 \NewDocumentCommand \importmodule { O{} m } {
2344    \stex_import_module_uri:nn { #1 } { #2 }
2345    \stex_debug:nn{modules}{Importing~module:~
2346      \l_stex_import_ns_str ? \l_stex_import_name_str
2347    }
2348    \stex_import_require_module:nnnn
2349    { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2350    { \l_stex_import_path_str } { \l_stex_import_name_str }
2351    \stex_if_smsmode:F {
2352      \stex_annotate_invisible:nnn
2353        {import} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
2354    }
2355    \exp_args:Nx \stex_add_to_current_module:n {
2356      \stex_import_require_module:nnnn
2357      { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2358      { \l_stex_import_path_str } { \l_stex_import_name_str }
2359    }
2360    \exp_args:Nx \stex_add_import_to_current_module:n {
2361      \l_stex_import_ns_str ? \l_stex_import_name_str
2362    }
2363    \stex_smsmode_do:
2364    \ignorespacesandpars
2365 }
2366 \stex_deactivate_macro:Nn \importmodule {module~environments}
```

(End definition for \importmodule*. This function is documented on page 83.)*

\usemodule

```
2367 \NewDocumentCommand \usemodule { O{} m } {
2368    \stex_if_smsmode:F {
2369      \stex_import_module_uri:nn { #1 } { #2 }
2370      \stex_import_require_module:nnnn
2371      { \l_stex_import_ns_str } { \l_stex_import_archive_str }
```

```
2372       { \l_stex_import_path_str } { \l_stex_import_name_str }
2373       \stex_annotate_invisible:nnn
2374         {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
2375     }
2376     \stex_smsmode_do:
2377     \ignorespacesandpars
2378 }
```

(*End definition for* \usemodule. *This function is documented on page 83.*)

```
2379 \cs_new_protected:Nn \stex_csl_to_imports:Nn {
2380   \tl_if_empty:nF{#2}{
2381     \clist_set:Nn \l_tmpa_clist {#2}
2382     \clist_map_inline:Nn \l_tmpa_clist {
2383       \tl_if_head_eq_charcode:nNTF {##1}[{
2384         #1 ##1
2385       }{
2386         #1{##1}
2387       }
2388     }
2389   }
2390 }
2391 \cs_generate_variant:Nn \stex_csl_to_imports:Nn {No}
2392
2393
2394 ⟨/package⟩
```

Chapter 29

sTEX
-Symbols Implementation

```
2395 ⟨*package⟩
2396
2397 %%%%%%%%%%%%    symbols.dtx    %%%%%%%%%%%%
2398
```

Warnings and error messages
```
2399 \msg_new:nnn{stex}{error/wrongargs}{
2400   args~value~in~symbol~declaration~for~#1~
2401   needs~to~be~i,~a,~b~or~B,~but~#2~given
2402 }
2403 \msg_new:nnn{stex}{error/unknownsymbol}{
2404   No~symbol~#1~found!
2405 }
2406 \msg_new:nnn{stex}{error/seqlength}{
2407   Expected~#1~arguments;~got~#2!
2408 }
2409 \msg_new:nnn{stex}{error/unknownnotation}{
2410   Unknown~notation~#1~for~#2!
2411 }
```

29.1   Symbol Declarations

```
2412 ⟨@@=stex_symdecl⟩
```

\stex_all_symbols:n   Map over all available symbols
```
2413 \cs_new_protected:Nn \stex_all_symbols:n {
2414   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
2415   \seq_map_inline:Nn \l_stex_all_modules_seq {
2416     \seq_map_inline:cn{c_stex_module_##1_constants}{
2417       \__stex_symdecl_all_symbols_cs{##1?####1}
2418     }
2419   }
2420 }
```

(*End definition for* \stex_all_symbols:n. *This function is documented on page 86.*)

```
2421 \NewDocumentCommand \STEXsymbol { m } {
2422   \stex_get_symbol:n { #1 }
2423   \exp_args:No
2424   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
2425 }
```

(*End definition for* \STEXsymbol. *This function is documented on page* *87.*)

symdecl arguments:

```
2426 \keys_define:nn { stex / symdecl } {
2427   name       .str_set_x:N  = \l_stex_symdecl_name_str ,
2428   local      .bool_set:N   = \l_stex_symdecl_local_bool ,
2429   args       .str_set_x:N  = \l_stex_symdecl_args_str ,
2430   type       .tl_set:N     = \l_stex_symdecl_type_tl ,
2431   deprecate  .str_set_x:N  = \l_stex_symdecl_deprecate_str ,
2432   align      .str_set:N    = \l_stex_symdecl_align_str , % TODO(?)
2433   gfc        .str_set:N    = \l_stex_symdecl_gfc_str , % TODO(?)
2434   def        .tl_set:N     = \l_stex_symdecl_definiens_tl ,
2435   reorder    .str_set_x:N  = \l_stex_symdecl_reorder_str ,
2436   argnames   .clist_set:N  = \l_stex_symdecl_argnames_clist ,
2437   assoc      .choices:nn   =
2438       {bin,binl,binr,pre,conj,pwconj}
2439       {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}
2440 }
2441
2442 \bool_new:N \l_stex_symdecl_make_macro_bool
2443
2444 \cs_new_protected:Nn \__stex_symdecl_args:n {
2445   \str_clear:N \l_stex_symdecl_name_str
2446   \str_clear:N \l_stex_symdecl_args_str
2447   \str_clear:N \l_stex_symdecl_deprecate_str
2448   \str_clear:N \l_stex_symdecl_reorder_str
2449   \str_clear:N \l_stex_symdecl_assoctype_str
2450   \bool_set_false:N \l_stex_symdecl_local_bool
2451   \tl_clear:N \l_stex_symdecl_type_tl
2452   \tl_clear:N \l_stex_symdecl_definiens_tl
2453   \clist_clear:N \l_stex_symdecl_argnames_clist
2454
2455   \keys_set:nn { stex / symdecl } { #1 }
2456 }
```

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

```
2457
2458 \NewDocumentCommand \symdecl { s m O{}} {
2459   \__stex_symdecl_args:n { #3 }
2460   \IfBooleanTF #1 {
2461     \bool_set_false:N \l_stex_symdecl_make_macro_bool
2462   } {
2463     \bool_set_true:N \l_stex_symdecl_make_macro_bool
2464   }
2465   \stex_symdecl_do:n { #2 }
2466   \stex_smsmode_do:
```

```
2467 }
2468
2469 \cs_new_protected:Nn \stex_symdecl_do:nn {
2470   \__stex_symdecl_args:n{#1}
2471   \bool_set_false:N \l_stex_symdecl_make_macro_bool
2472   \stex_symdecl_do:n{#2}
2473 }
2474
2475 \stex_deactivate_macro:Nn \symdecl {module~environments}
```

(*End definition for* \symdecl. *This function is documented on page* 85.)

\stex_symdecl_do:n

```
2476 \cs_new_protected:Nn \stex_symdecl_do:n {
2477   \stex_if_in_module:F {
2478     % TODO throw error? some default namespace?
2479   }
2480
2481   \str_if_empty:NT \l_stex_symdecl_name_str {
2482     \str_set:Nx \l_stex_symdecl_name_str { #1 }
2483   }
2484
2485   \prop_if_exist:cT { l_stex_symdecl_
2486       \l_stex_current_module_str ?
2487       \l_stex_symdecl_name_str
2488     _prop
2489   }{
2490     % TODO throw error (beware of circular dependencies)
2491   }
2492
2493   \prop_clear:N \l_tmpa_prop
2494   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
2495   \seq_clear:N \l_tmpa_seq
2496   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
2497   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
2498
2499   \str_if_empty:NT \l_stex_symdecl_deprecate_str {
2500     \str_if_empty:NF \l_stex_module_deprecate_str {
2501       \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
2502     }
2503   }
2504   \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
2505
2506   \exp_args:No \stex_add_constant_to_current_module:n {
2507     \l_stex_symdecl_name_str
2508   }
2509
2510   % arity/args
2511   \int_zero:N \l_tmpb_int
2512
2513   \bool_set_true:N \l_tmpa_bool
2514   \str_map_inline:Nn \l_stex_symdecl_args_str {
2515     \token_case_meaning:NnF ##1 {
2516       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
```

159

```
2517        {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2518        {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2519        {\tl_to_str:n a} {
2520          \bool_set_false:N \l_tmpa_bool
2521          \int_incr:N \l_tmpb_int
2522        }
2523        {\tl_to_str:n B} {
2524          \bool_set_false:N \l_tmpa_bool
2525          \int_incr:N \l_tmpb_int
2526        }
2527      }{
2528        \msg_error:nnxx{stex}{error/wrongargs}{
2529          \l_stex_current_module_str ?
2530          \l_stex_symdecl_name_str
2531        }{##1}
2532      }
2533    }
2534
2535    \bool_if:NTF \l_tmpa_bool {
2536      % possibly numeric
2537      \str_if_empty:NTF \l_stex_symdecl_args_str {
2538        \prop_put:Nnn \l_tmpa_prop { args } {}
2539        \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2540      }{
2541        \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
2542        \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2543        \str_clear:N \l_tmpa_str
2544        \int_step_inline:nn \l_tmpa_int {
2545          \str_put_right:Nn \l_tmpa_str i
2546        }
2547        \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
2548      }
2549    } {
2550      \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
2551      \prop_put:Nnx \l_tmpa_prop { arity }
2552        { \str_count:N \l_stex_symdecl_args_str }
2553    }
2554    \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2555
2556    \tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
2557      \prop_put:Nnx \l_tmpa_prop { defined }{ false }
2558    }{
2559      \prop_put:Nnx \l_tmpa_prop { defined }{ true }
2560    }
2561
2562    % argnames
2563
2564    \clist_clear:N \l_tmpa_clist
2565    \int_step_inline:nn {\prop_item:Nn \l_tmpa_prop {arity}} {
2566      \clist_if_empty:NTF \l_stex_symdecl_argnames_clist {
2567        \clist_put_right:Nn \l_tmpa_clist {##1}
2568      }{
2569        \clist_pop:NN \l_stex_symdecl_argnames_clist \l_tmpa_tl
2570        \exp_args:NNx \clist_put_right:Nn \l_tmpa_clist {\c_dollar_str\l_tmpa_tl}
```

160

```
2571        }
2572      }
2573      \prop_put:Nnx \l_tmpa_prop {argnames} {\clist_use:Nn \l_tmpa_clist ,}
2574
2575      % semantic macro
2576
2577      \bool_if:NT \l_stex_symdecl_make_macro_bool {
2578        \exp_args:Nx \stex_do_up_to_module:n {
2579          \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2580            \l_stex_current_module_str ? \l_stex_symdecl_name_str
2581          }}
2582        }
2583      }
2584
2585      \stex_debug:nn{symbols}{New~symbol:~
2586        \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2587        Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2588        Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
2589        Definiens:~\exp_not:o {\l_stex_symdecl_definiens_tl}
2590      }
2591
2592      % circular dependencies require this:
2593      \stex_if_do_html:T {
2594        \stex_annotate_invisible:nnn {symdecl} {
2595          \l_stex_current_module_str ? \l_stex_symdecl_name_str
2596        } {
2597          \tl_if_empty:NF \l_stex_symdecl_type_tl {
2598            \stex_annotate_invisible:nnn{type}{}{$\l_stex_symdecl_type_tl$}
2599          }
2600          \stex_annotate_invisible:nnn{args}{\prop_item:Nn \l_tmpa_prop { args }}{}
2601          \stex_annotate_invisible:nnn{macroname}{#1}{}
2602          \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2603            \stex_annotate_invisible:nnn{definiens}{}
2604              {$\l_stex_symdecl_definiens_tl$}
2605          }
2606          \str_if_empty:NF \l_stex_symdecl_assoctype_str {
2607            \stex_annotate_invisible:nnn{assoctype}{\l_stex_symdecl_assoctype_str}{}
2608          }
2609          \str_if_empty:NF \l_stex_symdecl_reorder_str {
2610            \stex_annotate_invisible:nnn{reorderargs}{\l_stex_symdecl_reorder_str}{}
2611          }
2612        }
2613      }
2614      \prop_if_exist:cF {
2615        l_stex_symdecl_
2616        \l_stex_current_module_str ? \l_stex_symdecl_name_str
2617        _prop
2618      } {
2619        \bool_if:NTF \l_stex_symdecl_local_bool \stex_do_up_to_module:x \stex_execute_in_module:
2620          \__stex_symdecl_restore_symbol:nnnnnnnn
2621            {\l_stex_symdecl_name_str}
2622            { \prop_item:Nn \l_tmpa_prop {args} }
2623            { \prop_item:Nn \l_tmpa_prop {arity} }
2624            { \prop_item:Nn \l_tmpa_prop {assocs} }
```

161

```
2625          { \prop_item:Nn \l_tmpa_prop {defined} }
2626          {\bool_if:NT \l_stex_symdecl_make_macro_bool {#1} }
2627          {\l_stex_current_module_str}
2628          { \prop_item:Nn \l_tmpa_prop {argnames} }
2629      }
2630    }
2631 }
2632 \cs_new_protected:Nn \__stex_symdecl_restore_symbol:nnnnnnnn {
2633    \prop_clear:N \l_tmpa_prop
2634    \prop_put:Nnn \l_tmpa_prop { module } { #7 }
2635    \prop_put:Nnn \l_tmpa_prop { name } { #1}
2636    \prop_put:Nnn \l_tmpa_prop { args } {#2}
2637    \prop_put:Nnn \l_tmpa_prop { arity } { #3 }
2638    \prop_put:Nnn \l_tmpa_prop { assocs } { #4 }
2639    \prop_put:Nnn \l_tmpa_prop { defined } { #5 }
2640    \prop_put:Nnn \l_tmpa_prop { argnames } { #8 }
2641    \tl_if_empty:nF{#6}{
2642      \tl_set:cx{#6}{\stex_invoke_symbol:n{\detokenize{#7 ? #1}}}
2643    }
2644    \prop_set_eq:cN{l_stex_symdecl_ \detokenize{#7 ? #1} _prop}\l_tmpa_prop
2645    \seq_clear:c{l_stex_symdecl_ \detokenize{#7 ? #1} _notations}
2646 }
```

(*End definition for* \stex_symdecl_do:n. *This function is documented on page 86.*)

\textsymdecl

```
2647
2648 \keys_define:nn { stex / textsymdecl } {
2649    name     .str_set_x:N = \l__stex_symdecl_name_str ,
2650    type     .tl_set:N    = \l__stex_symdecl_type_tl
2651 }
2652
2653 \cs_new_protected:Nn \_stex_textsymdecl_args:n {
2654    \str_clear:N \l__stex_symdecl_name_str
2655    \tl_clear:N \l__stex_symdecl_type_tl
2656    \clist_clear:N \l_stex_symdecl_argnames_clist
2657    \keys_set:nn { stex / textsymdecl } { #1 }
2658 }
2659
2660 \NewDocumentCommand \textsymdecl {m O{} m} {
2661    \_stex_textsymdecl_args:n { #2 }
2662    \str_if_empty:NTF \l__stex_symdecl_name_str {
2663      \__stex_symdecl_args:n{name=#1,#2}
2664    }{
2665      \__stex_symdecl_args:n{#2}
2666    }
2667    \bool_set_true:N \l_stex_symdecl_make_macro_bool
2668    \stex_symdecl_do:n{#1-sym}
2669    \stex_execute_in_module:n{
2670      \cs_set_nopar:cpn{#1name}{
2671        \ifvmode\hbox_unpack:N\c_empty_box\fi
2672        \ifmmode\hbox{#3}\else#3\fi\xspace
2673      }
2674      \cs_set_nopar:cpn{#1}{
```

```
2675        \ifmmode\csname#1-sym\expandafter\endcsname\else
2676        \ifvmode\hbox_unpack:N\c_empty_box\fi
2677        \symref{#1-sym}{#3}\expandafter\xspace
2678        \fi
2679      }
2680    }
2681    \stex_execute_in_module:x{
2682      \__stex_notation_restore_notation:nnnnn
2683      {\l_stex_current_module_str?\tl_if_empty:NTF\l__stex_symdecl_name_str{#1}\l__stex_symdec
2684      {}{0}
2685      {\exp_not:n{\STEXInternalTermMathOMSiiii{\STEXInternalCurrentSymbolStr}{}{\neginfprec}{
2686        \comp{\hbox{#3}}\STEXInternalSymbolAfterInvokationTL
2687      }}}
2688      {}
2689    }
2690    \stex_smsmode_do:
2691 }
```

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

\stex_get_symbol:n

```
2692 \str_new:N \l_stex_get_symbol_uri_str
2693
2694 \cs_new_protected:Nn \stex_get_symbol:n {
2695    \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2696      \tl_set:Nn \l_tmpa_tl { #1 }
2697      \__stex_symdecl_get_symbol_from_cs:
2698    }{
2699      % argument is a string
2700      % is it a command name?
2701      \cs_if_exist:cTF { #1 }{
2702        \cs_set_eq:Nc \l_tmpa_tl { #1 }
2703        \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2704        \str_if_empty:NTF \l_tmpa_str {
2705          \exp_args:Nx \cs_if_eq:NNTF {
2706            \tl_head:N \l_tmpa_tl
2707          } \stex_invoke_symbol:n {
2708            \__stex_symdecl_get_symbol_from_cs:
2709          }{
2710            \__stex_symdecl_get_symbol_from_string:n { #1 }
2711          }
2712        } {
2713          \__stex_symdecl_get_symbol_from_string:n { #1 }
2714        }
2715      }{
2716        % argument is not a command name
2717        \__stex_symdecl_get_symbol_from_string:n { #1 }
2718        % \l_stex_all_symbols_seq
2719      }
2720    }
2721    \str_if_eq:eeF {
2722      \prop_item:cn {
2723        l_stex_symdecl_\l_stex_get_symbol_uri_str _prop
2724      }{ deprecate }
```

```
2725    }{}{
2726      \msg_warning:nnxx{stex}{warning/deprecated}{
2727        Symbol~\l_stex_get_symbol_uri_str
2728      }{
2729        \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{ deprecate }
2730      }
2731    }
2732 }
2733
2734 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2735    \tl_set:Nn \l_tmpa_tl {
2736      \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2737    }
2738    \str_set:Nn \l_tmpa_str { #1 }
2739
2740    %\int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2741
2742    \str_if_in:NnTF \l_tmpa_str ? {
2743      \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
2744      \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
2745      \str_set:Nx \l_tmpb_str {\seq_use:Nn \l_tmpa_seq ?}
2746    }{
2747      \str_clear:N \l_tmpb_str
2748    }
2749    \str_if_empty:NTF \l_tmpb_str {
2750      \seq_map_inline:Nn \l_stex_all_modules_seq {
2751        \seq_map_inline:cn{c_stex_module_##1_constants}{
2752          \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2753            \seq_map_break:n{\seq_map_break:n{
2754              \tl_set:Nn \l_tmpa_tl {
2755                \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2756              }
2757          }}
2758        }
2759      }
2760    }
2761    }{
2762      \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpb_str }
2763      \seq_map_inline:Nn \l_stex_all_modules_seq {
2764        \str_if_eq:eeT{ \l_tmpb_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2765          \seq_map_inline:cn{c_stex_module_##1_constants}{
2766            \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2767              \seq_map_break:n{\seq_map_break:n{
2768                \tl_set:Nn \l_tmpa_tl {
2769                  \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2770                }
2771            }}
2772          }
2773        }
2774      }
2775    }
2776    }
2777
2778    \l_tmpa_tl
```

164

```
2779  }
2780
2781  \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
2782    \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2783      { \tl_tail:N \l_tmpa_tl }
2784    \tl_if_single:NTF \l_tmpa_tl {
2785      \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2786        \exp_after:wN \str_set:Nn \exp_after:wN
2787          \l_stex_get_symbol_uri_str \l_tmpa_tl
2788      }{
2789        % TODO
2790        % tail is not a single group
2791      }
2792    }{
2793      % TODO
2794      % tail is not a single group
2795    }
2796  }
```

(*End definition for* \stex_get_symbol:n. *This function is documented on page 86.*)

29.2   Notations

```
2797  ⟨@@=stex_notation⟩
```

notation arguments:
```
2798  \keys_define:nn { stex / notation } {
2799  %  lang    .tl_set_x:N  = \l__stex_notation_lang_str ,
2800    variant .tl_set_x:N  = \l__stex_notation_variant_str ,
2801    prec    .str_set_x:N = \l__stex_notation_prec_str ,
2802    op      .tl_set:N    = \l__stex_notation_op_tl ,
2803    primary .bool_set:N  = \l__stex_notation_primary_bool ,
2804    primary .default:n   = {true} ,
2805    hints   .str_set_x:N = \l__stex_notation_hints_str,
2806    unknown .code:n      = \str_set:Nx
2807        \l__stex_notation_variant_str \l_keys_key_str
2808  }
2809
2810  \cs_new_protected:Nn \_stex_notation_args:n {
2811  %  \str_clear:N \l__stex_notation_lang_str
2812    \str_clear:N \l__stex_notation_variant_str
2813    \str_clear:N \l__stex_notation_prec_str
2814    \str_clear:N \l__stex_notation_hints_str
2815    \tl_clear:N \l__stex_notation_op_tl
2816    \bool_set_false:N \l__stex_notation_primary_bool
2817
2818    \keys_set:nn { stex / notation } { #1 }
2819  }
```

\notation
```
2820  \NewDocumentCommand \notation { s m O{}} {
2821    \_stex_notation_args:n { #3 }
2822    \tl_clear:N \l_stex_symdecl_definiens_tl
2823    \stex_get_symbol:n { #2 }
```

```
2824    \tl_set:Nn \l_stex_notation_after_do_tl {
2825      \__stex_notation_final:
2826      \IfBooleanTF#1{
2827        \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2828      }{}
2829      \stex_smsmode_do:\ignorespacesandpars
2830    }
2831    \stex_notation_do:nnnnn
2832      { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } }
2833      { \prop_item:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } }
2834      { \l__stex_notation_variant_str }
2835      { \l__stex_notation_prec_str}
2836  }
2837  \stex_deactivate_macro:Nn \notation {module~environments}
```

(*End definition for* `\notation`. *This function is documented on page* *86.*)

`\stex_notation_do:nnnnn`

```
2838  \seq_new:N \l__stex_notation_precedences_seq
2839  \tl_new:N \l__stex_notation_opprec_tl
2840  \int_new:N \l__stex_notation_currarg_int
2841  \tl_new:N \STEXInternalSymbolAfterInvokationTL
2842
2843  \cs_new_protected:Nn \stex_notation_do:nnnnn {
2844    \let\STEXInternalCurrentSymbolStr\relax
2845    \seq_clear:N \l__stex_notation_precedences_seq
2846    \tl_clear:N \l__stex_notation_opprec_tl
2847    \str_set:Nx \l__stex_notation_args_str { #1 }
2848    \str_set:Nx \l__stex_notation_arity_str { #2 }
2849    \str_set:Nx \l__stex_notation_suffix_str { #3 }
2850    \str_set:Nx \l__stex_notation_prec_str { #4 }
2851
2852    % precedences
2853    \str_if_empty:NTF \l__stex_notation_prec_str {
2854      \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2855        \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2856      }{
2857        \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2858      }
2859    } {
2860      \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2861        \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2862        \int_step_inline:nn { \l__stex_notation_arity_str } {
2863          \exp_args:NNo
2864          \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2865        }
2866      }{
2867        \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2868        \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2869          \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2870          \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2871            \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2872              \l_tmpa_seq {\tl_to_str:n{x} } { \l_tmpa_str }
2873            \seq_map_inline:Nn \l_tmpa_seq {
```

```
2874            \seq_put_right:Nn \l__stex_notation_precedences_seq { ##1 }
2875          }
2876        }
2877      }{
2878        \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2879          \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2880        }{
2881          \tl_set:No \l__stex_notation_opprec_tl { 0 }
2882        }
2883      }
2884    }
2885  }
2886
2887  \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2888  \int_step_inline:nn { \l__stex_notation_arity_str } {
2889    \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
2890      \exp_args:NNo
2891      \seq_put_right:No \l__stex_notation_precedences_seq {
2892        \l__stex_notation_opprec_tl
2893      }
2894    }
2895  }
2896  \tl_clear:N \l_stex_notation_dummyargs_tl
2897
2898  \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2899    \exp_args:NNe
2900    \cs_set:Npn \l_stex_notation_macrocode_cs {
2901      \STEXInternalTermMathOMSiiii { \STEXInternalCurrentSymbolStr }
2902        { \l__stex_notation_suffix_str }
2903        { \l__stex_notation_opprec_tl }
2904        { \exp_not:n { #5 } }
2905    }
2906    \l_stex_notation_after_do_tl
2907  }{
2908    \str_if_in:NnTF \l__stex_notation_args_str b {
2909      \exp_args:Nne \use:nn
2910      {
2911      \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2912      \cs_set:Npn \l__stex_notation_arity_str } { {
2913        \STEXInternalTermMathOMBiiii { \STEXInternalCurrentSymbolStr }
2914          { \l__stex_notation_suffix_str }
2915          { \l__stex_notation_opprec_tl }
2916          { \exp_not:n { #5 } }
2917      }}
2918    }{
2919      \str_if_in:NnTF \l__stex_notation_args_str B {
2920        \exp_args:Nne \use:nn
2921        {
2922        \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2923        \cs_set:Npn \l__stex_notation_arity_str } { {
2924          \STEXInternalTermMathOMBiiii { \STEXInternalCurrentSymbolStr }
2925            { \l__stex_notation_suffix_str }
2926            { \l__stex_notation_opprec_tl }
2927            { \exp_not:n { #5 } }
```

```
2928             } }
2929         }{
2930             \exp_args:Nne \use:nn
2931             {
2932             \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2933             \cs_set:Npn \l__stex_notation_arity_str } { {
2934                 \STEXInternalTermMathOMAiiii { \STEXInternalCurrentSymbolStr }
2935                     { \l__stex_notation_suffix_str }
2936                     { \l__stex_notation_opprec_tl }
2937                     { \exp_not:n { #5 } }
2938             } }
2939         }
2940     }
2941
2942     \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2943     \int_zero:N \l__stex_notation_currarg_int
2944     \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2945     \__stex_notation_arguments:
2946   }
2947 }
```

(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

```
2948 \cs_new_protected:Nn \__stex_notation_arguments: {
2949   \int_incr:N \l__stex_notation_currarg_int
2950   \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2951     \l_stex_notation_after_do_tl
2952   }{
2953     \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2954     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2955     \str_if_eq:VnTF \l_tmpa_str a {
2956       \__stex_notation_argument_assoc:nn{a}
2957     }{
2958       \str_if_eq:VnTF \l_tmpa_str B {
2959         \__stex_notation_argument_assoc:nn{B}
2960       }{
2961         \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpb_str
2962         \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2963           { \STEXInternalTermMathArgiii
2964             { \l_tmpa_str\int_use:N \l__stex_notation_currarg_int }
2965             { \l_tmpb_str }
2966             { ####\int_use:N \l__stex_notation_currarg_int }
2967           }
2968         }
2969         \__stex_notation_arguments:
2970       }
2971     }
2972   }
2973 }
```

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

`\__stex_notation_argument_assoc:nn`

```
2974 \cs_new_protected:Nn \__stex_notation_argument_assoc:nn {
2975
2976   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2977     {\l__stex_notation_arity_str}{
2978     #2
2979   }
2980   \int_zero:N \l_tmpa_int
2981   \tl_clear:N \l_tmpa_tl
2982   \str_map_inline:Nn \l__stex_notation_args_str {
2983     \int_incr:N \l_tmpa_int
2984     \tl_put_right:Nx \l_tmpa_tl {
2985       \str_if_eq:nnTF {##1}{a}{ {} }{
2986         \str_if_eq:nnTF {##1}{B}{ {} }{
2987           {\_stex_term_arg:nn{##1\int_use:N \l_tmpa_int}{############## \int_use:N \l_tmpa
2988         }
2989       }
2990     }
2991   }
2992   \exp_after:wN\exp_after:wN\exp_after:wN \def
2993   \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2994   \exp_after:wN\exp_after:wN\exp_after:wN ##
2995   \exp_after:wN\exp_after:wN\exp_after:wN 1
2996   \exp_after:wN\exp_after:wN\exp_after:wN ##
2997   \exp_after:wN\exp_after:wN\exp_after:wN 2
2998   \exp_after:wN\exp_after:wN\exp_after:wN {
2999     \exp_after:wN \exp_after:wN \exp_after:wN
3000     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
3001       \exp_after:wN \l_tmpa_cs \l_tmpa_tl
3002     }
3003   }
3004
3005   \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
3006   \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
3007     \STEXInternalTermMathAssocArgiiiii
3008       { \int_use:N \l__stex_notation_currarg_int }
3009       { \l_tmpa_str }
3010       { ####\int_use:N \l__stex_notation_currarg_int }
3011       { \l_tmpa_cs {####1} {####2} }
3012       {#1}
3013   } }
3014   \__stex_notation_arguments:
3015 }
```

(*End definition for* __stex_notation_argument_assoc:nn.)

__stex_notation_final: Called after processing all notation arguments

```
3016 \cs_new_protected:Nn \__stex_notation_restore_notation:nnnnn {
3017   \cs_generate_from_arg_count:cNnn{stex_notation_\detokenize{#1} \c_hash_str \detokenize{#2}
3018   \cs_set_nopar:Npn {#3}{#4}
3019   \tl_if_empty:nF {#5}{
3020     \tl_set:cn{stex_op_notation_\detokenize{#1} \c_hash_str \detokenize{#2}_cs}{ \comp{ #5 }
3021   }
3022   \seq_if_exist:cT { l_stex_symdecl_\detokenize{#1} _notations }{
3023     \seq_put_right:cx { l_stex_symdecl_\detokenize{#1} _notations } { \detokenize{#2} }
```

169

```
3024     }
3025 }
3026
3027 \cs_new_protected:Nn \__stex_notation_final: {
3028
3029   \stex_execute_in_module:x {
3030     \__stex_notation_restore_notation:nnnnn
3031       {\l_stex_get_symbol_uri_str}
3032       {\l__stex_notation_suffix_str}
3033       {\l__stex_notation_arity_str}
3034       {
3035         \exp_after:wN \exp_after:wN \exp_after:wN
3036         \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
3037         { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \STEXInt
3038       }
3039       {\exp_args:No \exp_not:n \l__stex_notation_op_tl }
3040   }
3041
3042   \stex_debug:nn{symbols}{
3043     Notation~\l__stex_notation_suffix_str
3044     ~for~\l_stex_get_symbol_uri_str^^J
3045     Operator~precedence:~\l__stex_notation_opprec_tl^^J
3046     Argument~precedences:~
3047       \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
3048     Notation: \cs_meaning:c {
3049       stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
3050       \l__stex_notation_suffix_str
3051       _cs
3052     }
3053   }
3054     % HTML annotations
3055   \stex_if_do_html:T {
3056     \stex_annotate_invisible:nnn { notation }
3057     { \l_stex_get_symbol_uri_str } {
3058       \stex_annotate_invisible:nnn { notationfragment }
3059         { \l__stex_notation_suffix_str }{}
3060       \stex_annotate_invisible:nnn { precedence }
3061         { \l__stex_notation_prec_str }{}
3062
3063       \int_zero:N \l_tmpa_int
3064       \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
3065       \tl_clear:N \l_tmpa_tl
3066       \int_step_inline:nn { \l__stex_notation_arity_str }{
3067         \int_incr:N \l_tmpa_int
3068         \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
3069         \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_rem
3070         \str_if_eq:VnTF \l_tmpb_str a {
3071           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3072             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3073             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3074           } }
3075         }{
3076           \str_if_eq:VnTF \l_tmpb_str B {
3077             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
```

```
3078                    \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3079                    \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3080                  } }
3081                }{
3082                  \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3083                    \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
3084                  } }
3085                }
3086              }
3087            }
3088            \stex_annotate_invisible:nnn { notationcomp }{}{
3089              \str_set:Nx \STEXInternalCurrentSymbolStr {\l_stex_get_symbol_uri_str }
3090              $ \exp_args:Nno \use:nn { \use:c {
3091                stex_notation_ \STEXInternalCurrentSymbolStr
3092                \c_hash_str \l__stex_notation_suffix_str _cs
3093              } } { \l_tmpa_tl } $
3094            }
3095            \tl_if_empty:NF \l__stex_notation_op_tl {
3096              \stex_annotate_invisible:nnn { notationopcomp }{}{
3097                $\l__stex_notation_op_tl$
3098              }
3099            }
3100          }
3101        }
3102 }
```

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

\setnotation

```
3103 \keys_define:nn { stex / setnotation } {
3104 %  lang     .tl_set_x:N  = \l__stex_notation_lang_str ,
3105   variant .tl_set_x:N  = \l__stex_notation_variant_str ,
3106   unknown .code:n      = \str_set:Nx
3107       \l__stex_notation_variant_str \l_keys_key_str
3108 }
3109
3110 \cs_new_protected:Nn \_stex_setnotation_args:n {
3111 % \str_clear:N \l__stex_notation_lang_str
3112   \str_clear:N \l__stex_notation_variant_str
3113   \keys_set:nn { stex / setnotation } { #1 }
3114 }
3115
3116 \cs_new_protected:Nn \__stex_notation_setnotation:nn {
3117   \seq_if_exist:cT{l_stex_symdecl_#1_notations}{
3118     \seq_remove_all:cn { l_stex_symdecl_#1 _notations }{ #2 }
3119     \seq_put_left:cn { l_stex_symdecl_#1 _notations }{ #2 }
3120   }
3121 }
3122
3123 \cs_new_protected:Nn \stex_setnotation:n {
3124   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1 _notations }
3125     { \l__stex_notation_variant_str }{
3126       \stex_execute_in_module:x{ \__stex_notation_setnotation:nn {#1}{\l__stex_notation_vari
3127       \stex_debug:nn {notations}{
```

```
3128        Setting~default~notation~
3129        {\l__stex_notation_variant_str }~for~
3130        #1 \\
3131        \expandafter\meaning\csname
3132        l_stex_symdecl_#1 _notations\endcsname
3133      }
3134    }{
3135      \msg_error:nnxx{stex}{unknownnotation}{\l__stex_notation_variant_str}{#1}
3136    }
3137 }
3138
3139 \NewDocumentCommand \setnotation {m m} {
3140   \stex_get_symbol:n { #1 }
3141   \_stex_setnotation_args:n { #2 }
3142   \stex_setnotation:n{\l_stex_get_symbol_uri_str}
3143   \stex_smsmode_do:\ignorespacesandpars
3144 }
3145
3146 \cs_new_protected:Nn \stex_copy_notations:nn {
3147   \stex_debug:nn {notations}{
3148     Copying~notations~from~#2~to~#1\\
3149     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
3150   }
3151   \tl_clear:N \l_tmpa_tl
3152   \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
3153     \tl_put_right:Nn \l_tmpa_tl { {######## ##1} }
3154   }
3155   \seq_map_inline:cn {l_stex_symdecl_#2_notations}{\begingroup
3156     \stex_debug:nn{Here}{Here:~##1}
3157     \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
3158     \edef \l_tmpa_tl {
3159       \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
3160       \exp_after:wN\exp_after:wN\exp_after:wN {
3161         \exp_after:wN \l_tmpa_cs \l_tmpa_tl
3162       }
3163     }
3164
3165     \exp_after:wN \def \exp_after:wN \l_tmpa_tl
3166     \exp_after:wN ####\exp_after:wN 1 \exp_after:wN ####\exp_after:wN 2
3167     \exp_after:wN  { \l_tmpa_tl }
3168
3169     \edef \l_tmpa_tl {
3170       \exp_after:wN \exp_not:n \exp_after:wN {
3171         \l_tmpa_tl {######## 1}{######## 2}
3172       }
3173     }
3174
3175     \stex_debug:nn{Here}{Here:~\expandafter\detokenize\expandafter{\l_tmpa_tl}}
3176
3177     \stex_execute_in_module:x {
3178       \__stex_notation_restore_notation:nnnnn
3179         {#1}{##1}
3180         { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }
3181         { \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl} }
```

```
3182          {
3183            \cs_if_exist:cT{stex_op_notation_ #2\c_hash_str ##1 _cs}{
3184              \exp_args:NNo\exp_args:No\exp_not:n{\csname stex_op_notation_ #2\c_hash_str ##1
3185            }
3186          }
3187      }\endgroup
3188    }
3189 }
3190
3191 \NewDocumentCommand \copynotation {m m} {
3192    \stex_get_symbol:n { #1 }
3193    \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
3194    \stex_get_symbol:n { #2 }
3195    \exp_args:Noo
3196    \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
3197    \stex_smsmode_do:\ignorespacesandpars
3198 }
3199
```

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

```
3200 \keys_define:nn { stex / symdef } {
3201    name     .str_set_x:N = \l_stex_symdecl_name_str ,
3202    local    .bool_set:N  = \l_stex_symdecl_local_bool ,
3203    args     .str_set_x:N = \l_stex_symdecl_args_str ,
3204    type     .tl_set:N    = \l_stex_symdecl_type_tl ,
3205    def      .tl_set:N    = \l_stex_symdecl_definiens_tl ,
3206    reorder  .str_set_x:N = \l_stex_symdecl_reorder_str ,
3207    op       .tl_set:N    = \l__stex_notation_op_tl ,
3208 % lang     .str_set_x:N = \l__stex_notation_lang_str ,
3209    variant  .str_set_x:N = \l__stex_notation_variant_str ,
3210    prec     .str_set_x:N = \l__stex_notation_prec_str ,
3211    argnames    .clist_set:N  = \l_stex_symdecl_argnames_clist ,
3212    assoc    .choices:nn =
3213        {bin,binl,binr,pre,conj,pwconj}
3214        {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
3215    unknown .code:n      = \str_set:Nx
3216        \l__stex_notation_variant_str \l_keys_key_str
3217 }
3218
3219 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
3220    \str_clear:N \l_stex_symdecl_name_str
3221    \str_clear:N \l_stex_symdecl_args_str
3222    \str_clear:N \l_stex_symdecl_assoctype_str
3223    \str_clear:N \l_stex_symdecl_reorder_str
3224    \bool_set_false:N \l_stex_symdecl_local_bool
3225    \tl_clear:N \l_stex_symdecl_type_tl
3226    \tl_clear:N \l_stex_symdecl_definiens_tl
3227    \clist_clear:N \l_stex_symdecl_argnames_clist
3228 % \str_clear:N \l__stex_notation_lang_str
3229    \str_clear:N \l__stex_notation_variant_str
3230    \str_clear:N \l__stex_notation_prec_str
3231    \tl_clear:N \l__stex_notation_op_tl
```

```
3232
3233    \keys_set:nn { stex / symdef } { #1 }
3234 }
3235
3236 \NewDocumentCommand \symdef { m O{} } {
3237    \__stex_notation_symdef_args:n { #2 }
3238    \bool_set_true:N \l_stex_symdecl_make_macro_bool
3239    \stex_symdecl_do:n { #1 }
3240    \tl_set:Nn \l_stex_notation_after_do_tl {
3241       \__stex_notation_final:
3242       \stex_smsmode_do:\ignorespacesandpars
3243    }
3244    \str_set:Nx \l_stex_get_symbol_uri_str {
3245       \l_stex_current_module_str ? \l_stex_symdecl_name_str
3246    }
3247    \exp_args:Nx \stex_notation_do:nnnnn
3248       { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } } }
3249       { \prop_item:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } } }
3250       { \l__stex_notation_variant_str }
3251       { \l__stex_notation_prec_str}
3252 }
3253 \stex_deactivate_macro:Nn \symdef {module~environments}
3254
3255 \keys_define:nn { stex / mmtdef } {
3256    name     .str_set_x:N = \l_stex_symdecl_name_str ,
3257    args     .str_set_x:N = \l_stex_symdecl_args_str ,
3258    reorder .str_set_x:N = \l_stex_symdecl_reorder_str ,
3259    op       .tl_set:N    = \l__stex_notation_op_tl ,
3260  % lang     .str_set_x:N = \l__stex_notation_lang_str ,
3261    variant .str_set_x:N = \l__stex_notation_variant_str ,
3262    prec     .str_set_x:N = \l__stex_notation_prec_str ,
3263    argnames     .clist_set:N  = \l_stex_symdecl_argnames_clist ,
3264    assoc    .choices:nn =
3265       {bin,binl,binr,pre,conj,pwconj}
3266       {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
3267    unknown .code:n       = \str_set:Nx
3268       \l__stex_notation_variant_str \l_keys_key_str
3269 }
3270 \cs_new_protected:Nn \_stex_mmtdef_args:n {
3271    \str_clear:N \l_stex_symdecl_name_str
3272    \str_clear:N \l_stex_symdecl_args_str
3273    \str_clear:N \l_stex_symdecl_assoctype_str
3274    \str_clear:N \l_stex_symdecl_reorder_str
3275    \clist_clear:N \l_stex_symdecl_argnames_clist
3276  % \str_clear:N \l__stex_notation_lang_str
3277    \str_clear:N \l__stex_notation_variant_str
3278    \str_clear:N \l__stex_notation_prec_str
3279    \tl_clear:N \l__stex_notation_op_tl
3280
3281    \keys_set:nn { stex / mmtdef } { #1 }
3282 }
3283
3284 \NewDocumentCommand \mmtdef {m O{} }{
3285    \_stex_mmtdef_args:n{ #2 }
```

```
3286    \bool_set_true:N \l_stex_symdecl_make_macro_bool
3287    \str_if_empty:NT \l_stex_symdecl_name_str {
3288      \str_set:Nx \l_stex_symdecl_name_str { #1 }
3289    }
3290    %\tl_set:Nx \l_stex_symdecl_definiens_tl {
3291    %   \stex_annotate:nnn{ OMID }{
3292    %     \l_stex_module_mmtfor_str?\l_stex_symdecl_name_str
3293    %   }{}
3294    %}
3295    \stex_symdecl_do:n { #1 }
3296    \MMTrule{rules.stex.mmt.kwarc.info?SubstitutionRule}{
3297      \stex_annotate:nnn{ OMID }{
3298        \l_stex_current_module_str ? \l_stex_symdecl_name_str
3299      }{},
3300      \stex_annotate:nnn{ OMID }{
3301        \l_stex_module_mmtfor_str?\l_stex_symdecl_name_str
3302      }{}
3303    }
3304    \tl_set:Nn \l_stex_notation_after_do_tl {
3305      \__stex_notation_final:
3306      \stex_smsmode_do:\ignorespacesandpars
3307    }
3308    \str_set:Nx \l_stex_get_symbol_uri_str {
3309      \l_stex_current_module_str ? \l_stex_symdecl_name_str
3310    }
3311    \exp_args:Nx \stex_notation_do:nnnnn
3312      { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } }
3313      { \prop_item:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } }
3314      { \l__stex_notation_variant_str }
3315      { \l__stex_notation_prec_str}
3316 }
```

(*End definition for* `\symdef`. *This function is documented on page 86.*)

29.3   Variables

```
3317  ⟨@@=stex_variables⟩
3318
3319  \keys_define:nn { stex / vardef } {
3320    name     .str_set_x:N  = \l__stex_variables_name_str ,
3321    args     .str_set_x:N  = \l__stex_variables_args_str ,
3322    type     .tl_set:N     = \l__stex_variables_type_tl ,
3323    def      .tl_set:N     = \l__stex_variables_def_tl ,
3324    op       .tl_set:N     = \l__stex_variables_op_tl ,
3325    prec     .str_set_x:N  = \l__stex_variables_prec_str ,
3326    reorder  .str_set_x:N  = \l__stex_variables_reorder_str ,
3327    argnames     .clist_set:N  = \l__stex_variables_argnames_clist ,
3328    assoc    .choices:nn  =
3329        {bin,binl,binr,pre,conj,pwconj}
3330        {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
3331    bind     .choices:nn  =
3332        {forall,exists}
3333        {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
3334 }
```

```
3335
3336 \cs_new_protected:Nn \__stex_variables_args:n {
3337   \str_clear:N \l__stex_variables_name_str
3338   \str_clear:N \l__stex_variables_args_str
3339   \str_clear:N \l__stex_variables_prec_str
3340   \str_clear:N \l__stex_variables_assoctype_str
3341   \str_clear:N \l__stex_variables_reorder_str
3342   \str_clear:N \l__stex_variables_bind_str
3343   \tl_clear:N \l__stex_variables_type_tl
3344   \tl_clear:N \l__stex_variables_def_tl
3345   \tl_clear:N \l__stex_variables_op_tl
3346   \clist_clear:N \l__stex_variables_argnames_clist
3347
3348   \keys_set:nn { stex / vardef } { #1 }
3349 }
3350
3351 \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{}} {
3352   \__stex_variables_args:n {#2}
3353   \str_if_empty:NT \l__stex_variables_name_str {
3354     \str_set:Nx \l__stex_variables_name_str { #1 }
3355   }
3356   \prop_clear:N \l_tmpa_prop
3357   \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
3358
3359   \int_zero:N \l_tmpb_int
3360   \bool_set_true:N \l_tmpa_bool
3361   \str_map_inline:Nn \l__stex_variables_args_str {
3362     \token_case_meaning:NnF ##1 {
3363       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
3364       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
3365       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
3366       {\tl_to_str:n a} {
3367         \bool_set_false:N \l_tmpa_bool
3368         \int_incr:N \l_tmpb_int
3369       }
3370       {\tl_to_str:n B} {
3371         \bool_set_false:N \l_tmpa_bool
3372         \int_incr:N \l_tmpb_int
3373       }
3374     }{
3375       \msg_error:nnxx{stex}{error/wrongargs}{
3376         variable~\l__stex_variables_name_str
3377       }{##1}
3378     }
3379   }
3380   \bool_if:NTF \l_tmpa_bool {
3381     % possibly numeric
3382     \str_if_empty:NTF \l__stex_variables_args_str {
3383       \prop_put:Nnn \l_tmpa_prop { args } {}
3384       \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
3385     }{
3386       \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
3387       \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
3388       \str_clear:N \l_tmpa_str
```

```
3389        \int_step_inline:nn \l_tmpa_int {
3390          \str_put_right:Nn \l_tmpa_str i
3391        }
3392        \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
3393        \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
3394      }
3395    } {
3396      \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
3397      \prop_put:Nnx \l_tmpa_prop { arity }
3398        { \str_count:N \l__stex_variables_args_str }
3399    }
3400    \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
3401    \tl_set:cx { #1 }{ \stex_invoke_variable:n { \l__stex_variables_name_str } }
3402
3403    % argnames
3404
3405    \clist_clear:N \l_tmpa_clist
3406    \int_step_inline:nn {\prop_item:Nn \l_tmpa_prop {arity}} {
3407      \clist_if_empty:NTF \l__stex_variables_argnames_clist {
3408        \clist_put_right:Nn \l_tmpa_clist {##1}
3409      }{
3410        \clist_pop:NN \l__stex_variables_argnames_clist \l_tmpa_tl
3411        \exp_args:NNx \clist_put_right:Nn \l_tmpa_clist {\c_dollar_str\l_tmpa_tl}
3412      }
3413    }
3414    \prop_put:Nnx \l_tmpa_prop {argnames} {\clist_use:Nn \l_tmpa_clist ,}
3415
3416
3417    \prop_set_eq:cN { l_stex_symdecl_var://\l__stex_variables_name_str _prop} \l_tmpa_prop
3418
3419    \tl_if_empty:NF \l__stex_variables_op_tl {
3420      \cs_set:cpx {
3421        stex_var_op_notation_ \l__stex_variables_name_str _cs
3422      } { \exp_not:N\comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } } }
3423    }
3424
3425    \tl_set:Nn \l_stex_notation_after_do_tl {
3426      \exp_args:Nne \use:nn {
3427        \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str _cs }
3428          \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
3429      } {{
3430        \exp_after:wN \exp_after:wN \exp_after:wN
3431        \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
3432        { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \STEXInter
3433      }}
3434      \stex_if_do_html:T {
3435        \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
3436          \stex_annotate_invisible:nnn { precedence }
3437            { \l__stex_variables_prec_str }{}
3438          \tl_if_empty:NF \l__stex_variables_type_tl {\stex_annotate_invisible:nnn{type}{}{$\l
3439          \stex_annotate_invisible:nnn{args}{ \l__stex_variables_args_str }{}
3440          \stex_annotate_invisible:nnn{macroname}{#1}{}
3441          \tl_if_empty:NF \l__stex_variables_def_tl {
3442            \stex_annotate_invisible:nnn{definiens}{}
```

177

```
3443                {$\l__stex_variables_def_tl$}
3444              }
3445            \str_if_empty:NF \l__stex_variables_assoctype_str {
3446              \stex_annotate_invisible:nnn{assoctype}{\l__stex_variables_assoctype_str}{}
3447            }
3448            \str_if_empty:NF \l__stex_variables_reorder_str {
3449              \stex_annotate_invisible:nnn{reorderargs}{\l__stex_variables_reorder_str}{}
3450            }
3451            \int_zero:N \l_tmpa_int
3452            \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
3453            \tl_clear:N \l_tmpa_tl
3454            \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
3455              \int_incr:N \l_tmpa_int
3456              \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
3457              \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables
3458              \str_if_eq:VnTF \l_tmpb_str a {
3459                \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3460                  \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3461                  \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3462                } }
3463              }{
3464                \str_if_eq:VnTF \l_tmpb_str B {
3465                  \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3466                    \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3467                    \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3468                  } }
3469                }{
3470                  \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3471                    \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
3472                  } }
3473                }
3474              }
3475            }
3476            \stex_annotate_invisible:nnn { notationcomp }{}{
3477              \str_set:Nx \STEXInternalCurrentSymbolStr {var://\l__stex_variables_name_str }
3478              $ \exp_args:Nno \use:nn { \use:c {
3479                stex_var_notation_\l__stex_variables_name_str _cs
3480              } } { \l_tmpa_tl } $
3481            }
3482            \tl_if_empty:NF \l__stex_variables_op_tl {
3483              \stex_annotate_invisible:nnn { notationopcomp }{}{
3484                $\l__stex_variables_op_tl$
3485              }
3486            }
3487          }
3488          \str_if_empty:NF \l__stex_variables_bind_str {
3489            \stex_annotate_invisible:nnn {bindtype}{\l__stex_variables_bind_str,\l__stex_variabl
3490          }
3491      }\ignorespacesandpars
3492    }
3493
3494    \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
3495  }
3496
```

178

```
3497  \cs_new:Nn \_stex_reset:N {
3498    \tl_if_exist:NTF #1 {
3499      \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
3500    }{
3501      \let \exp_not:N #1 \exp_not:N \undefined
3502    }
3503  }
3504
3505  \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
3506    \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
3507    \exp_args:Nnx \use:nn {
3508      % TODO
3509      \stex_annotate_invisible:nnn {vardecl}{\clist_use:Nn\l__stex_variables_names,}{
3510        #2
3511      }
3512    }{
3513      \_stex_reset:N \varnot
3514      \_stex_reset:N \vartype
3515      \_stex_reset:N \vardefi
3516    }
3517  }
3518
3519  \NewDocumentCommand \vardef { s } {
3520    \IfBooleanTF#1 {
3521      \__stex_variables_do_complex:nn
3522    }{
3523      \__stex_variables_do_simple:nnn
3524    }
3525  }
3526
3527  \NewDocumentCommand \svar { O{} m }{
3528    \tl_if_empty:nTF {#1}{
3529      \str_set:Nn \l_tmpa_str { #2 }
3530    }{
3531      \str_set:Nn \l_tmpa_str { #1 }
3532    }
3533    \_stex_term_omv:nn {
3534      var://\l_tmpa_str
3535    }{
3536      \exp_args:Nnx \use:nn {
3537        \def\comp{\_varcomp}
3538        \str_set:Nx \STEXInternalCurrentSymbolStr { var://\l_tmpa_str }
3539        \comp{ #2 }
3540      }{
3541        \_stex_reset:N \comp
3542        \_stex_reset:N \STEXInternalCurrentSymbolStr
3543      }
3544    }
3545  }
3546
3547
3548
3549  \keys_define:nn { stex / varseq } {
3550    name      .str_set_x:N  = \l__stex_variables_name_str ,
```

179

```
3551    args    .int_set:N    = \l__stex_variables_args_int ,
3552    type    .tl_set:N     = \l__stex_variables_type_tl  ,
3553    mid     .tl_set:N     = \l__stex_variables_mid_tl    ,
3554    bind    .choices:nn   =
3555        {forall,exists}
3556        {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
3557 }
3558
3559 \cs_new_protected:Nn \__stex_variables_seq_args:n {
3560    \str_clear:N \l__stex_variables_name_str
3561    \int_set:Nn \l__stex_variables_args_int 1
3562    \tl_clear:N \l__stex_variables_type_tl
3563    \str_clear:N \l__stex_variables_bind_str
3564
3565    \keys_set:nn { stex / varseq } { #1 }
3566 }
3567
3568 \NewDocumentCommand \varseq {m O{} m m m}{
3569    \__stex_variables_seq_args:n { #2 }
3570    \str_if_empty:NT \l__stex_variables_name_str {
3571       \str_set:Nx \l__stex_variables_name_str { #1 }
3572    }
3573    \prop_clear:N \l_tmpa_prop
3574    \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
3575
3576    \seq_set_from_clist:Nn \l_tmpa_seq {#3}
3577    \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
3578       \msg_error:nnxx{stex}{error/seqlength}
3579          {\int_use:N \l__stex_variables_args_int}
3580          {\seq_count:N \l_tmpa_seq}
3581    }
3582    \seq_set_from_clist:Nn \l_tmpb_seq {#4}
3583    \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
3584       \msg_error:nnxx{stex}{error/seqlength}
3585          {\int_use:N \l__stex_variables_args_int}
3586          {\seq_count:N \l_tmpb_seq}
3587    }
3588    \prop_put:Nnn \l_tmpa_prop {starts} {#3}
3589    \prop_put:Nnn \l_tmpa_prop {ends} {#4}
3590
3591    \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
3592       \cs_set:Npn  {\int_use:N \l__stex_variables_args_int} { #5 }
3593
3594    % argnames
3595
3596    \clist_clear:N \l_tmpa_clist
3597    \int_step_inline:nn {\l__stex_variables_args_int} {
3598       \clist_put_right:Nn \l_tmpa_clist {##1}
3599    }
3600    \prop_put:Nnx \l_tmpa_prop {argnames} {\clist_use:Nn \l_tmpa_clist ,}
3601
3602
3603
3604
```

```
3605    \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3606    \int_step_inline:nn \l__stex_variables_args_int {
3607      \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
3608    }
3609    \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
3610    \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
3611    \tl_if_empty:NF \l__stex_variables_mid_tl {
3612      \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
3613      \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
3614    }
3615    \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3616    \int_step_inline:nn \l__stex_variables_args_int {
3617      \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
3618    }
3619    \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
3620    \tl_put_right:No \l_tmpa_tl \l_tmpb_tl


3622
3623    \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl

3624
3625    \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}

3626
3627    \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}

3628
3629    \int_step_inline:nn \l__stex_variables_args_int {
3630      \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
3631        \STEXInternalTermMathArgiii{i##1}{0}{\exp_not:n{####}##1}
3632      }}
3633    }

3634
3635    \tl_set:Nx \l_tmpa_tl {
3636      \STEXInternalTermMathOMAiiii { varseq://\l__stex_variables_name_str}{}{0}{
3637        \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
3638      }
3639    }

3640
3641    \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \STEXInternalSymbolAfterInvokationTL} }

3642
3643    \exp_args:Nno \use:nn {
3644    \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
3645      \cs_set:Npn {\int_use:N \l__stex_variables_args_int}}{\l_tmpa_tl}

3646
3647    \stex_debug:nn{sequences}{New~Sequence:~
3648      \expandafter\meaning\csname stex_varseq_\l__stex_variables_name_str _cs\endcsname\\~\\
3649      \prop_to_keyval:N \l_tmpa_prop
3650    }
3651    \prop_set_eq:cN {l_stex_symdecl_varseq://\l__stex_variables_name_str _prop}\l_tmpa_prop

3652
3653    \stex_if_do_html:T{\stex_annotate_invisible:nnn{varseq}{\l__stex_variables_name_str}{
3654      \tl_if_empty:NF \l__stex_variables_type_tl {
3655        \stex_annotate:nnn {type}{}{$\l__stex_variables_type_tl$}
3656      }
3657      \stex_annotate:nnn {args}{\int_use:N \l__stex_variables_args_int}{}
3658      \str_if_empty:NF \l__stex_variables_bind_str {
```

```
3659        \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{}
3660      }
3661    \stex_annotate:nnn{startindex}{}{$#3$}
3662    \stex_annotate:nnn{endindex}{}{$#4$}
3663
3664    \tl_clear:N \l_tmpa_tl
3665    \int_step_inline:nn \l__stex_variables_args_int {
3666      \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3667        \stex_annotate:nnn{argmarker}{##1}{}
3668      } }
3669    }
3670    \stex_annotate_invisible:nnn { notationcomp }{}{
3671      \str_set:Nx \STEXInternalCurrentSymbolStr {varseq://\l__stex_variables_name_str }
3672      $ \exp_args:Nno \use:nn { \use:c {
3673        stex_varseq_\l__stex_variables_name_str _cs
3674      } } { \l_tmpa_tl } $
3675    }
3676    \stex_annotate_invisible:nnn { notationopcomp }{}{
3677      $ \prop_item:Nn \l_tmpa_prop { notation } $
3678    }
3679
3680  }}
3681
3682  \ignorespacesandpars
3683 }
3684
3685
3686 \keys_define:nn { stex / mmtdecl } {
3687   name        .str_set_x:N  = \l_stex_symdecl_name_str ,
3688   args        .str_set_x:N  = \l_stex_symdecl_args_str ,
3689   deprecate   .str_set_x:N  = \l_stex_symdecl_deprecate_str ,
3690   reorder     .str_set_x:N  = \l_stex_symdecl_reorder_str ,
3691   argnames    .clist_set:N  = \l_stex_symdecl_argnames_clist ,
3692   assoc       .choices:nn   =
3693      {bin,binl,binr,pre,conj,pwconj}
3694      {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}
3695 }
3696
3697 \cs_new_protected:Nn \_stex_mmtdecl_args:n {
3698   \str_clear:N \l_stex_symdecl_name_str
3699   \str_clear:N \l_stex_symdecl_args_str
3700   \str_clear:N \l_stex_symdecl_deprecate_str
3701   \str_clear:N \l_stex_symdecl_reorder_str
3702   \str_clear:N \l_stex_symdecl_assoctype_str
3703   \bool_set_false:N \l_stex_symdecl_local_bool
3704   \clist_clear:N \l_stex_symdecl_argnames_clist
3705
3706   \keys_set:nn { stex / symdecl } { #1 }
3707 }
3708
3709 \NewDocumentCommand \mmtdecl { s m O{}} {
3710   \_stex_mmtdecl_args:n{#3}
3711   \IfBooleanTF #1 {
3712     \bool_set_false:N \l_stex_symdecl_make_macro_bool
```

```
3713    } {
3714      \bool_set_true:N \l_stex_symdecl_make_macro_bool
3715    }
3716    \str_if_empty:NT \l_stex_symdecl_name_str {
3717      \str_set:Nx \l_stex_symdecl_name_str { #1 }
3718    }
3719    %\tl_set:Nx \l_stex_symdecl_definiens_tl {
3720    %  \stex_annotate:nnn{ OMID }{
3721    %    \l_stex_module_mmtfor_str?\l_stex_symdecl_name_str
3722    %  }{}
3723    %}
3724    \stex_symdecl_do:n{#2}
3725    \MMTrule{rules.stex.mmt.kwarc.info?SubstitutionRule}{
3726      \stex_annotate:nnn{ OMID }{
3727        \l_stex_current_module_str ? \l_stex_symdecl_name_str
3728      }{},
3729      \stex_annotate:nnn{ OMID }{
3730        \l_stex_module_mmtfor_str?\l_stex_symdecl_name_str
3731      }{}
3732    }
3733    \stex_smsmode_do:
3734  }
3735
3736  \stex_deactivate_macro:Nn \mmtdecl {mmtinterface~environments}
3737  \stex_deactivate_macro:Nn \mmtdef {mmtinterface~environments}
3738
3739  ⟨/package⟩
```

183

Chapter 30

SᴛᴇX
-Terms Implementation

3740 ⟨∗package⟩

3741

3742 %%%%%%%%%%%%   terms.dtx   %%%%%%%%%%%%

3743

3744 ⟨@@=stex_terms⟩

Warnings and error messages

3745 \msg_new:nnn{stex}{error/nonotation}{
3746   Symbol~#1~invoked,~but~has~no~notation#2!
3747 }
3748 \msg_new:nnn{stex}{error/notationarg}{
3749   Error~in~parsing~notation~#1
3750 }
3751 \msg_new:nnn{stex}{error/noop}{
3752   Symbol~#1~has~no~operator~notation~for~notation~#2
3753 }
3754 \msg_new:nnn{stex}{error/notallowed}{
3755   Symbol~invokation~#1~not~allowed~in~notation~component~of~#2
3756 }
3757 \msg_new:nnn{stex}{error/doubleargument}{
3758   Argument~#1~of~symbol~#2~already~assigned
3759 }
3760 \msg_new:nnn{stex}{error/overarity}{
3761   Argument~#1~invalid~for~symbol~#2~with~arity~#3
3762 }
3763

30.1   Symbol Invocations

\stex_invoke_symbol:n   Invokes a semantic macro

3764

3765

3766 \bool_new:N \l_stex_allow_semantic_bool
3767 \bool_set_true:N \l_stex_allow_semantic_bool

3768

```
3769  \cs_new_protected:Nn \stex_invoke_symbol:n {
3770    \ifvmode\indent\fi
3771    \bool_if:NTF \l_stex_allow_semantic_bool {
3772      \str_if_eq:eeF {
3773        \prop_item:cn {
3774          l_stex_symdecl_#1_prop
3775        }{ deprecate }
3776      }{}{
3777        \msg_warning:nnxx{stex}{warning/deprecated}{
3778          Symbol~#1
3779        }{
3780          \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
3781        }
3782      }
3783      \if_mode_math:
3784        \exp_after:wN \__stex_terms_invoke_math:n
3785      \else:
3786        \exp_after:wN \__stex_terms_invoke_text:n
3787      \fi: { #1 }
3788    }{
3789      \msg_error:nnxx{stex}{error/notallowed}{#1}{\STEXInternalCurrentSymbolStr}
3790    }
3791  }
3792
3793  \cs_new_protected:Nn \__stex_terms_invoke_text:n {
3794    \peek_charcode_remove:NTF ! {
3795      \__stex_terms_invoke_op_custom:nn {#1}
3796    }{
3797      \__stex_terms_invoke_custom:nn {#1}
3798    }
3799  }
3800
3801  \cs_new_protected:Nn \__stex_terms_invoke_math:n {
3802    \peek_charcode_remove:NTF ! {
3803      % operator
3804      \peek_charcode_remove:NTF * {
3805        % custom op
3806        \__stex_terms_invoke_op_custom:nn {#1}
3807      }{
3808        % op notation
3809        \peek_charcode:NTF [ {
3810          \__stex_terms_invoke_op_notation:nw {#1}
3811        }{
3812          \__stex_terms_invoke_op_notation:nw {#1}[]
3813        }
3814      }
3815    }{
3816      \peek_charcode_remove:NTF * {
3817        \__stex_terms_invoke_custom:nn {#1}
3818        % custom
3819      }{
3820        % normal
3821        \peek_charcode:NTF [ {
3822          \__stex_terms_invoke_notation:nw {#1}
```

```
3823        }{
3824          \__stex_terms_invoke_notation:nw {#1}[]
3825        }
3826      }
3827    }
3828 }


3831 \cs_new_protected:Nn \__stex_terms_invoke_op_custom:nn {
3832    \exp_args:Nnx \use:nn {
3833      \def\comp{\_comp}
3834      \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3835      \bool_set_false:N \l_stex_allow_semantic_bool
3836      \stex_mathml_intent:nn{#1}{
3837        \_stex_term_oms:nnn {#1}{#1 \c_hash_str CUSTOM-}{
3838          \comp{ #2 }
3839        }
3840      }
3841    }{
3842      \_stex_reset:N \comp
3843      \_stex_reset:N \STEXInternalCurrentSymbolStr
3844      \bool_set_true:N \l_stex_allow_semantic_bool
3845    }
3846 }

3848 \keys_define:nn { stex / terms } {
3849 %  lang     .tl_set_x:N = \l_stex_notation_lang_str ,
3850    variant .tl_set_x:N = \l_stex_notation_variant_str ,
3851    unknown .code:n      = \str_set:Nx
3852        \l_stex_notation_variant_str \l_keys_key_str
3853 }

3855 \cs_new_protected:Nn \__stex_terms_args:n {
3856  % \str_clear:N \l_stex_notation_lang_str
3857    \str_clear:N \l_stex_notation_variant_str

3859    \keys_set:nn { stex / terms } { #1 }
3860 }

3862 \cs_new_protected:Nn \stex_find_notation:nn {
3863    \__stex_terms_args:n { #2 }
3864    \seq_if_empty:cTF {
3865      l_stex_symdecl_ #1 _notations
3866    } {
3867      \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
3868    } {
3869      \str_if_empty:NTF \l_stex_notation_variant_str {
3870        \seq_get_left:cN {l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
3871      }{
3872        \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
3873          \l_stex_notation_variant_str
3874        }{
3875        %  \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3876        }{
```

```
3877        \msg_error:nnxx{stex}{error/nonotation}{#1}{
3878          ~\l_stex_notation_variant_str
3879        }
3880      }
3881    }
3882  }
3883 }
3884
3885 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3886   \exp_args:Nnx \use:nn {
3887     \def\comp{\_comp}
3888     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3889     \stex_find_notation:nn { #1 }{ #2 }
3890     \bool_set_false:N \l_stex_allow_semantic_bool
3891     \cs_if_exist:cTF {
3892       stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3893     }{
3894       \_stex_term_oms:nnn { #1 }{
3895         #1 \c_hash_str \l_stex_notation_variant_str
3896       }{
3897         \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3898       }
3899     }{
3900       \int_compare:nNnTF {\prop_item:cn {l_stex_symdecl_#1_prop}{arity}} = 0{
3901         \cs_if_exist:cTF {
3902           stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3903         }{
3904           \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
3905             \_stex_reset:N \comp
3906             \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
3907             \_stex_reset:N \STEXInternalCurrentSymbolStr
3908             \bool_set_true:N \l_stex_allow_semantic_bool
3909           }
3910           \def\comp{\_comp}
3911           \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3912           \bool_set_false:N \l_stex_allow_semantic_bool
3913           \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3914         }{
3915           \msg_error:nnxx{stex}{error/nonotation}{#1}{
3916             ~\l_stex_notation_variant_str
3917           }
3918         }
3919       }{
3920         \msg_error:nnxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3921       }
3922     }
3923   }{
3924     \_stex_reset:N \comp
3925     \_stex_reset:N \STEXInternalCurrentSymbolStr
3926     \bool_set_true:N \l_stex_allow_semantic_bool
3927   }
3928 }
3929
3930 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {
```

```
3931    \stex_find_notation:nn { #1 }{ #2 }
3932    \cs_if_exist:cTF {
3933      stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3934    }{
3935      \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
3936        \_stex_reset:N \comp
3937        \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
3938        \_stex_reset:N \STEXInternalCurrentSymbolStr
3939        \bool_set_true:N \l_stex_allow_semantic_bool
3940      }
3941      \def\comp{\_comp}
3942      \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3943      \bool_set_false:N \l_stex_allow_semantic_bool
3944      \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3945    }{
3946      \msg_error:nnxx{stex}{error/nonotation}{#1}{
3947        ~\l_stex_notation_variant_str
3948      }
3949    }
3950  }
3951
3952  \prop_new:N \l__stex_terms_custom_args_prop
3953  \clist_new:N \l_stex_argnames_seq
3954  \seq_new:N \l__stex_terms_tmp_seq
3955
3956  \cs_new_protected:Nn\__stex_terms_custom_comp:n{\bool_set_false:N \l_stex_allow_semantic_boo
3957
3958  \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3959    \exp_args:Nnx \use:nn {
3960      \def\comp{\__stex_terms_custom_comp:n}
3961      \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3962      \prop_clear:N \l__stex_terms_custom_args_prop
3963      \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3964      \prop_get:cnN {
3965        l_stex_symdecl_#1 _prop
3966      }{ args } \l_tmpa_str
3967      \exp_args:NNx \seq_set_from_clist:Nn \l_stex_argnames_seq {
3968        \prop_item:cn {l_stex_symdecl_#1 _prop}{argnames}
3969      }
3970      \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3971      \tl_set:Nn \arg { \__stex_terms_arg: }
3972      \str_if_empty:NTF \l_tmpa_str {
3973        \stex_mathml_intent:nn{#1}{
3974          \_stex_term_oms:nnn {#1}{#1\c_hash_str CUSTOM-}{\ignorespaces#2}
3975        }
3976      }{
3977        \seq_clear:N \l__stex_terms_tmp_seq
3978        \exp_args:Nx\int_step_inline:nn{\prop_item:cn{l_stex_symdecl_#1 _prop}{arity}}{
3979          \tl_set:Nx \l__stex_terms_tmp_tl {\seq_item:Nn \l_stex_argnames_seq {##1}}
3980          \bool_lazy_or:nnT{
3981            \str_if_eq_p:nn{a}{\str_item:Nn\l_tmpa_str{##1}}
3982          }{
3983            \str_if_eq_p:nn{B}{\str_item:Nn\l_tmpa_str{##1}}
3984          }{
```

188

```
3985        \tl_put_right:Nn \l__stex_terms_tmp_tl +
3986      }
3987      \seq_put_right:No \l__stex_terms_tmp_seq \l__stex_terms_tmp_tl
3988    }
3989    \stex_mathml_intent:nn{
3990      #1[\prop_item:cn {l_stex_symdecl_#1 _prop}{ args }](
3991        \seq_use:Nn \l__stex_terms_tmp_seq ,
3992      )
3993    }{
3994      \str_if_in:NnTF \l_tmpa_str b {
3995        \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
3996      }{
3997        \str_if_in:NnTF \l_tmpa_str B {
3998          \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
3999        }{
4000          \_stex_term_oma:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
4001        }
4002      }
4003    }
4004  }
4005  % TODO check that all arguments exist
4006  }{
4007    \_stex_reset:N \l_stex_argnames_seq
4008    \_stex_reset:N \STEXInternalCurrentSymbolStr
4009    \_stex_reset:N \arg
4010    \_stex_reset:N \comp
4011    \_stex_reset:N \l__stex_terms_custom_args_prop
4012    %\bool_set_true:N \l_stex_allow_semantic_bool
4013  }
4014 }
4015
4016 \NewDocumentCommand \__stex_terms_arg: { s O{} m}{
4017   \tl_if_empty:nTF {#2}{
4018     \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
4019     \bool_set_true:N \l_tmpa_bool
4020     \bool_do_while:Nn \l_tmpa_bool {
4021       \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int
4022         \int_incr:N \l_tmpa_int
4023       }{
4024         \bool_set_false:N \l_tmpa_bool
4025       }
4026     }
4027   }{
4028     \int_set:Nn \l_tmpa_int { #2 }
4029   }
4030   \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
4031   \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
4032     \msg_error:nnxxx{stex}{error/overarity}
4033       {\int_use:N \l_tmpa_int}
4034       {\STEXInternalCurrentSymbolStr}
4035       {\str_count:N \l_tmpa_str}
4036   }
4037   \str_set:Nx \l_tmpa_str {\str_item:Nn \l_tmpa_str \l_tmpa_int}
4038   \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
```

```
4039      \bool_lazy_any:nF {
4040        {\str_if_eq_p:Vn \l_tmpa_str {a}}
4041        {\str_if_eq_p:Vn \l_tmpa_str {B}}
4042      }{
4043        \msg_error:nnxx{stex}{error/doubleargument}
4044          {\int_use:N \l_tmpa_int}
4045          {\STEXInternalCurrentSymbolStr}
4046      }
4047    }
4048    \exp_args:NNx \prop_put:Nnn \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {\ign
4049    \bool_if:NTF \l_stex_allow_semantic_bool \use_i:nn {
4050      \bool_set_true:N \l_stex_allow_semantic_bool
4051      \use:nn
4052    }
4053    {
4054    \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq \l_tmpa_int}{
4055      \IfBooleanTF#1{
4056        \stex_annotate_invisible:n { %TODO
4057          \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{\ignorespaces#3}
4058        }
4059      }{ %TODO
4060        \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{\ignorespaces#3}
4061      }
4062    }}
4063    {\bool_set_false:N \l_stex_allow_semantic_bool}
4064 }


4065

4066
4067 \cs_new_protected:Nn \_stex_term_arg:nn {
4068    \bool_set_true:N \l_stex_allow_semantic_bool
4069    \stex_annotate:nnn{ arg }{ #1 }{ #2 }
4070    \bool_set_false:N \l_stex_allow_semantic_bool
4071 }

4072
4073 \cs_new_protected:Npn \STEXInternalTermMathArgiii #1#2#3 {
4074    \exp_args:Nnx \use:nn
4075      { \int_set:Nn \l__stex_terms_downprec { #2 }
4076        \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq \l_tmpa_int}{
4077          \_stex_term_arg:nn { #1 }{ #3 }
4078        }
4079      }
4080      { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
4081 }
```

(*End definition for* \stex_invoke_symbol:n. *This function is documented on page 87.*)

```
4082 \cs_new_protected:Npn \STEXInternalTermMathAssocArgiiiii #1#2#3#4#5 {
4083    \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
4084    \tl_set:Nn \l_tmpb_tl {\STEXInternalTermMathArgiii{#5#1}{#2}}
4085    \tl_if_empty:nTF { #3 }{
4086      \STEXInternalTermMathArgiii{#5#1}{#2}{}
4087    }{
4088      \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 }}{
```

190

```
4089        \expandafter\if\expandafter\relax\noexpand#3
4090          \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_maybe_sequence:Nnn#3{#1}{#5}}
4091        \else
4092          \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_simple:nnn{#1}{#3}{#5}}
4093        \fi
4094        \l_tmpa_tl
4095      }{
4096        \__stex_terms_math_assoc_arg_simple:nnn{#1}{#3}{#5}
4097      }
4098    }
4099  }
4100
4101  \cs_new_protected:Nn \__stex_terms_math_assoc_arg_maybe_sequence:Nnn {
4102    \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
4103    \str_if_empty:NTF \l_tmpa_str {
4104      \exp_args:Nx \cs_if_eq:NNTF {
4105        \tl_head:N #1
4106      } \stex_invoke_sequence:n {
4107        \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
4108        \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
4109        \tl_set:Nx \l_tmpa_tl {\prop_item:cn {l_stex_symdecl_varseq://\l_tmpa_str _prop}{notat
4110        \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
4111        \tl_set:Nx \l_tmpa_tl {{\exp_not:N \exp_not:n{
4112          \exp_not:n{\exp_args:Nnx \use:nn} {
4113            \exp_not:n {
4114              \def\comp{\_varcomp}
4115              \str_set:Nn \STEXInternalCurrentSymbolStr
4116            } {varseq://\l_tmpa_str}
4117            \exp_not:n{ ##1 }
4118          }{
4119            \exp_not:n {
4120              \_stex_reset:N \comp
4121              \_stex_reset:N \STEXInternalCurrentSymbolStr
4122            }
4123          }
4124        }}}
4125        \exp_args:Nno \use:n {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
4126        \seq_reverse:N \l_tmpa_seq
4127        \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
4128        \seq_map_inline:Nn \l_tmpa_seq {
4129          \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
4130            \exp_args:Nno
4131            \l_tmpa_cs { ##1 } \l_tmpa_tl
4132          }
4133        }
4134        \tl_set:Nx \l_tmpa_tl {
4135          \_stex_term_omv:nn {varseq://\l_tmpa_str}{
4136            \exp_args:No \exp_not:n \l_tmpa_tl
4137          }
4138        }
4139        \exp_args:No\l_tmpb_tl\l_tmpa_tl
4140      }{
4141        \__stex_terms_math_assoc_arg_simple:nnn{#2} { #1 }{#3}
4142      }
```

```
4143        } {
4144          \__stex_terms_math_assoc_arg_simple:nnn{#2} { #1 }{#3}
4145        }
4146
4147    }
4148
4149    \cs_new_protected:Nn \__stex_terms_math_assoc_arg_simple:nnn {
4150      \clist_set:Nn \l_tmpa_clist{ #2 }
4151      \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
4152        \tl_set:Nn \l_tmpa_tl {
4153          \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq #1}{
4154            \_stex_term_arg:nn{A#3#1}{ #2 } }
4155        }
4156      }{
4157        \clist_reverse:N \l_tmpa_clist
4158        \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
4159        \tl_set:Nx \l_tmpa_tl {
4160          \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq #1}{
4161            \_stex_term_arg:nn{A#3#1}{
4162              \exp_args:No \exp_not:n \l_tmpa_tl
4163            }
4164        }}
4165        \clist_map_inline:Nn \l_tmpa_clist {
4166          \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
4167            \exp_args:Nno
4168            \l_tmpa_cs {
4169              \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq #1}{
4170                \_stex_term_arg:nn{A#3#1}{##1}
4171              }
4172            } \l_tmpa_tl
4173          }
4174        }
4175      }
4176      \exp_args:No\l_tmpb_tl\l_tmpa_tl
4177    }
```

(*End definition for* `\STEXInternalTermMathAssocArgiiiii`*. This function is documented on page 88.*)

30.2 Terms

Precedences:

```
4178    \tl_const:Nx \infprec {\int_use:N \c_max_int}
4179    \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
4180    \int_new:N \l__stex_terms_downprec
4181    \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 88.*)

Bracketing:

\l__stex_terms_left_bracket_str
\l__stex_terms_right_bracket_str

```
4182    \tl_set:Nn \l__stex_terms_left_bracket_str (
4183    \tl_set:Nn \l__stex_terms_right_bracket_str )
```

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

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

```
4184 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
4185   \bool_if:NTF \l__stex_terms_brackets_done_bool {
4186     \bool_set_false:N \l__stex_terms_brackets_done_bool
4187     #2
4188   } {
4189     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
4190       \bool_if:NTF \l_stex_inparray_bool { #2 }{
4191         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
4192         \dobrackets { #2 }
4193       }
4194     }{ #2 }
4195   }
4196 }
```

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

`\dobrackets`

```
4197 \bool_new:N \l__stex_terms_brackets_done_bool
4198 %\RequirePackage{scalerel}
4199 \cs_new_protected:Npn \dobrackets #1 {
4200   %\ThisStyle{\if D\m@switch
4201   %  \exp_args:Nnx \use:nn
4202   %  { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
4203   %  { \exp_not:N\right\l__stex_terms_right_bracket_str }
4204   %  \else
4205     \exp_args:Nnx \use:nn
4206     {
4207       \bool_set_true:N \l__stex_terms_brackets_done_bool
4208       \int_set:Nn \l__stex_terms_downprec \infprec
4209       \l__stex_terms_left_bracket_str
4210       #1
4211     }
4212     {
4213       \bool_set_false:N \l__stex_terms_brackets_done_bool
4214       \l__stex_terms_right_bracket_str
4215       \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
4216     }
4217   %\fi}
4218 }
```

(End definition for `\dobrackets`. *This function is documented on page 88.*)*

`\withbrackets`

```
4219 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
4220   \exp_args:Nnx \use:nn
4221   {
4222     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
4223     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
4224     #3
4225   }
4226   {
```

193

```
4227        \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
4228          {\l__stex_terms_left_bracket_str}
4229        \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
4230          {\l__stex_terms_right_bracket_str}
4231    }
4232  }
```

(*End definition for* `\withbrackets`*. This function is documented on page 88.*)

\STEXinvisible

```
4233  \cs_new_protected:Npn \STEXinvisible #1 {
4234    \stex_annotate_invisible:n { #1 }
4235  }
```

(*End definition for* `\STEXinvisible`*. This function is documented on page 88.*)

OMDoc terms:

\STEXInternalTermMathOMSiiii

```
4236  \cs_new_protected:Nn \_stex_term_oms:nnn {
4237    \stex_annotate:nnn{ OMID }{ #2 }{
4238      #3
4239    }
4240  }
4241
4242  \cs_new_protected:Npn \STEXInternalTermMathOMSiiii #1#2#3#4 {
4243    \__stex_terms_maybe_brackets:nn { #3 }{
4244      \stex_mathml_intent:nn{#1} {
4245        \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
4246      }
4247    }
4248  }
```

(*End definition for* `\STEXInternalTermMathOMSiiii`*. This function is documented on page 87.*)

_stex_term_math_omv:nn

```
4249  \cs_new_protected:Nn \_stex_term_omv:nn {
4250    \stex_annotate:nnn{ OMV }{ #1 }{
4251      #2
4252    }
4253  }
```

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

\STEXInternalTermMathOMAiiii

```
4254  \cs_new_protected:Nn \_stex_term_oma:nnn {
4255    \stex_annotate:nnn{ OMA }{ #2 }{
4256      #3
4257    }
4258  }
4259
4260  \cs_new_protected:Npn \STEXInternalTermMathOMAiiii #1#2#3#4 {
4261    \exp_args:Nnx \use:nn {
4262      \seq_clear:N \l__stex_terms_tmp_seq
4263      \prop_if_exist:cT{l_stex_symdecl_#1 _prop}{
4264      \exp_args:NNx \seq_set_from_clist:Nn \l_stex_argnames_seq {
```

194

```
4265          \prop_item:cn {l_stex_symdecl_#1 _prop}{argnames}
4266        }
4267      \exp_args:Nx\int_step_inline:nn{\prop_item:cn{l_stex_symdecl_#1 _prop}{arity}}{
4268        \tl_set:Nx \l__stex_terms_tmp_tl {\seq_item:Nn \l_stex_argnames_seq {##1}}
4269        \bool_lazy_or:nnT{
4270          \str_if_eq_p:nn{a}{\str_item:Nn\l_tmpa_str{##1}}
4271        }{
4272          \str_if_eq_p:nn{B}{\str_item:Nn\l_tmpa_str{##1}}
4273        }{
4274          \tl_put_right:Nn \l__stex_terms_tmp_tl +
4275        }
4276        \seq_put_right:No \l__stex_terms_tmp_seq \l__stex_terms_tmp_tl
4277      }
4278    }
4279    \__stex_terms_maybe_brackets:nn { #3 }{
4280      \stex_mathml_intent:nn{
4281        #1[\prop_item:cn {l_stex_symdecl_#1 _prop}{ args }](
4282          \seq_use:Nn \l__stex_terms_tmp_seq ,
4283        )
4284      }{
4285        \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
4286      }
4287    }
4288    }{
4289      \_stex_reset:N \l_stex_argnames_seq
4290    }
4291 }
```

(*End definition for* \STEXInternalTermMathOMAiiii. *This function is documented on page 87.*)

```
4292 \cs_new_protected:Nn \_stex_term_ombind:nnn {
4293    \stex_annotate:nnn{ OMBIND }{ #2 }{
4294      #3
4295    }
4296 }
4297
4298 \cs_new_protected:Npn \STEXInternalTermMathOMBiiii #1#2#3#4 {
4299    \exp_args:Nnx \use:nn {
4300      \seq_clear:N \l__stex_terms_tmp_seq
4301      \prop_if_exist:cT{l_stex_symdecl_#1 _prop}{
4302      \exp_args:NNx \seq_set_from_clist:Nn \l_stex_argnames_seq {
4303        \prop_item:cn {l_stex_symdecl_#1 _prop}{argnames}
4304      }
4305      \exp_args:Nx\int_step_inline:nn{\prop_item:cn{l_stex_symdecl_#1 _prop}{arity}}{
4306        \tl_set:Nx \l__stex_terms_tmp_tl {\seq_item:Nn \l_stex_argnames_seq {##1}}
4307        \bool_lazy_or:nnT{
4308          \str_if_eq_p:nn{a}{\str_item:Nn\l_tmpa_str{##1}}
4309        }{
4310          \str_if_eq_p:nn{B}{\str_item:Nn\l_tmpa_str{##1}}
4311        }{
4312          \tl_put_right:Nn \l__stex_terms_tmp_tl +
4313        }
4314        \seq_put_right:No \l__stex_terms_tmp_seq \l__stex_terms_tmp_tl
```

```
4315        }
4316      }
4317      \__stex_terms_maybe_brackets:nn { #3 }{
4318        \stex_mathml_intent:nn{
4319          #1[\prop_item:cn {l_stex_symdecl_#1 _prop}{ args }](
4320            \seq_use:Nn \l__stex_terms_tmp_seq ,
4321          )
4322        }{
4323          \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
4324        }
4325      }
4326    }{
4327      \_stex_reset:N \l_stex_argnames_seq
4328    }
4329 }
```

(End definition for `\STEXInternalTermMathOMBiiii`*. This function is documented on page 87.)*

\symref
\symname

```
4330 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
4331
4332 \keys_define:nn { stex / symname } {
4333    pre      .tl_set_x:N    = \l__stex_terms_pre_tl ,
4334    post     .tl_set_x:N    = \l__stex_terms_post_tl ,
4335    root     .tl_set_x:N    = \l__stex_terms_root_tl
4336 }
4337
4338 \cs_new_protected:Nn \stex_symname_args:n {
4339    \tl_clear:N \l__stex_terms_post_tl
4340    \tl_clear:N \l__stex_terms_pre_tl
4341    \tl_clear:N \l__stex_terms_root_str
4342    \keys_set:nn { stex / symname } { #1 }
4343 }
4344
4345 \NewDocumentCommand \symref { m m }{
4346    \let\compemph_uri_prev:\compemph@uri
4347    \let\compemph@uri\symrefemph@uri
4348    \STEXsymbol{#1}!{ #2 }
4349    \let\compemph@uri\compemph_uri_prev:
4350 }
4351
4352 \NewDocumentCommand \synonym { O{} m m}{
4353    \stex_symname_args:n { #1 }
4354    \let\compemph_uri_prev:\compemph@uri
4355    \let\compemph@uri\symrefemph@uri
4356    % TODO
4357    \STEXsymbol{#2}!{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
4358    \let\compemph@uri\compemph_uri_prev:
4359 }
4360
4361 \NewDocumentCommand \symname { O{} m }{
4362    \stex_symname_args:n { #1 }
4363    \stex_get_symbol:n { #2 }
4364    \str_set:Nx \l_tmpa_str {
```

```
4365        \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4366      }
4367      \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4368
4369      \let\compemph_uri_prev:\compemph@uri
4370      \let\compemph@uri\symrefemph@uri
4371      \exp_args:NNx \use:nn
4372      \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmmode*\fi{
4373        \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
4374      } }
4375      \let\compemph@uri\compemph_uri_prev:
4376    }
4377
4378    \NewDocumentCommand \Symname { O{} m }{
4379      \stex_symname_args:n { #1 }
4380      \stex_get_symbol:n { #2 }
4381      \str_set:Nx \l_tmpa_str {
4382        \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4383      }
4384      \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4385      \let\compemph_uri_prev:\compemph@uri
4386      \let\compemph@uri\symrefemph@uri
4387      \exp_args:NNx \use:nn
4388      \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmmode*\fi{
4389        \exp_after:wN \stex_capitalize:n \l_tmpa_str
4390          \l__stex_terms_post_tl
4391      } }
4392      \let\compemph@uri\compemph_uri_prev:
4393    }
```

(*End definition for* `\symref` *and* `\symname`. *These functions are documented on page 87.*)

30.3   Notation Components

```
4394 ⟨@@=stex_notationcomps⟩
```

<div style="color:red">
\comp
\compemph@uri
\compemph
\defemph
\defemph@uri
\symrefemph
\symrefemph@uri
\varemph
\varemph@uri
</div>

```
4395 \cs_new_protected:Npn \_comp #1 {
4396    \str_if_empty:NF \STEXInternalCurrentSymbolStr {
4397      \stex_html_backend:TF {
4398        \stex_annotate:nnn { comp }{ \STEXInternalCurrentSymbolStr }{ #1 }
4399      }{
4400        \exp_args:Nnx \compemph@uri { #1 } { \STEXInternalCurrentSymbolStr }
4401      }
4402    }
4403 }
4404
4405 \cs_new_protected:Npn \_varcomp #1 {
4406    \str_if_empty:NF \STEXInternalCurrentSymbolStr {
4407      \stex_html_backend:TF {
4408        \stex_annotate:nnn { varcomp }{ \STEXInternalCurrentSymbolStr }{ #1 }
4409      }{
4410        \exp_args:Nnx \varemph@uri { #1 } { \STEXInternalCurrentSymbolStr }
4411      }
```

```
4412      }
4413  }
4414
4415  \def\comp{\_comp}
4416
4417  \cs_new_protected:Npn \compemph@uri #1 #2 {
4418      \compemph{ #1 }
4419  }
4420
4421
4422  \cs_new_protected:Npn \compemph #1 {
4423      #1
4424  }
4425
4426  \cs_new_protected:Npn \defemph@uri #1 #2 {
4427      \defemph{#1}
4428  }
4429
4430  \cs_new_protected:Npn \defemph #1 {
4431      \textbf{#1}
4432  }
4433
4434  \cs_new_protected:Npn \symrefemph@uri #1 #2 {
4435      \symrefemph{#1}
4436  }
4437
4438  \cs_new_protected:Npn \symrefemph #1 {
4439      \emph{#1}
4440  }
4441
4442  \cs_new_protected:Npn \varemph@uri #1 #2 {
4443      \varemph{#1}
4444  }
4445
4446  \cs_new_protected:Npn \varemph #1 {
4447      #1
4448  }
```

(*End definition for* \comp *and others. These functions are documented on page 88.*)

\ellipses

```
4449  \NewDocumentCommand \ellipses {} { \ldots }
```

(*End definition for* \ellipses*. This function is documented on page 88.*)

\parray
\prmatrix
\parrayline
\parraylineh
\parraycell

```
4450  \bool_new:N \l_stex_inparray_bool
4451  \bool_set_false:N \l_stex_inparray_bool
4452  \NewDocumentCommand \parray { m m } {
4453      \begingroup
4454      \bool_set_true:N \l_stex_inparray_bool
4455      \begin{array}{#1}
4456          #2
4457      \end{array}
4458      \endgroup
```

```
4459 }
4460
4461 \NewDocumentCommand \prmatrix { m } {
4462   \begingroup
4463   \bool_set_true:N \l_stex_inparray_bool
4464   \begin{matrix}
4465     #1
4466   \end{matrix}
4467   \endgroup
4468 }
4469
4470 \def \maybephline {
4471   \bool_if:NT \l_stex_inparray_bool {\hline}
4472 }
4473
4474 \def \parrayline #1 #2 {
4475   #1 #2 \bool_if:NT \l_stex_inparray_bool {\\}
4476 }
4477
4478 \def \pmrow #1 { \parrayline{}{ #1 } }
4479
4480 \def \parraylineh #1 #2 {
4481   #1 #2 \bool_if:NT \l_stex_inparray_bool {\\\hline}
4482 }
4483
4484 \def \parraycell #1 {
4485   #1 \bool_if:NT \l_stex_inparray_bool {&}
4486 }
```

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

30.4   Variables

```
4487 ⟨@@=stex_variables⟩
```

\stex_invoke_variable:n Invokes a variable

```
4488 \cs_new_protected:Nn \stex_invoke_variable:n {
4489   \if_mode_math:
4490     \exp_after:wN \__stex_variables_invoke_math:n
4491   \else:
4492     \exp_after:wN \__stex_variables_invoke_text:n
4493   \fi: {#1}
4494 }
4495
4496 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
4497   \peek_charcode_remove:NTF ! {
4498     \__stex_variables_invoke_op_custom:nn {#1}
4499   }{
4500     \__stex_variables_invoke_custom:nn {#1}
4501   }
4502 }
4503
4504
4505 \cs_new_protected:Nn \__stex_variables_invoke_math:n {
```

```
4506    \peek_charcode_remove:NTF ! {
4507      \peek_charcode_remove:NTF ! {
4508        \peek_charcode:NTF [ {
4509          % TODO throw error
4510        }{
4511          \__stex_variables_invoke_op_custom:nn
4512        }
4513      }{
4514        \__stex_variables_invoke_op:n { #1 }
4515      }
4516    }{
4517      \peek_charcode_remove:NTF * {
4518        \__stex_variables_invoke_custom:nn { #1 }
4519      }{
4520        \__stex_variables_invoke_math_ii:n { #1 }
4521      }
4522    }
4523  }
4524
4525  \cs_new_protected:Nn \__stex_variables_invoke_op_custom:nn {
4526    \exp_args:Nnx \use:nn {
4527      \def\comp{\_varcomp}
4528      \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4529      \bool_set_false:N \l_stex_allow_semantic_bool
4530      \_stex_term_omv:nn {var://#1}{
4531        \comp{ #2 }
4532      }
4533    }{
4534      \_stex_reset:N \comp
4535      \_stex_reset:N \STEXInternalCurrentSymbolStr
4536      \bool_set_true:N \l_stex_allow_semantic_bool
4537    }
4538  }
4539
4540  \cs_new_protected:Nn \__stex_variables_invoke_op:n {
4541    \cs_if_exist:cTF {
4542      stex_var_op_notation_ #1 _cs
4543    }{
4544      \exp_args:Nnx \use:nn {
4545        \def\comp{\_varcomp}
4546        \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4547        \_stex_term_omv:nn { var://#1 }{
4548          \use:c{stex_var_op_notation_ #1 _cs }
4549        }
4550      }{
4551        \_stex_reset:N \comp
4552        \_stex_reset:N \STEXInternalCurrentSymbolStr
4553      }
4554    }{
4555      \int_compare:nNnTF {\prop_item:cn {l_stex_symdecl_var://#1_prop}{arity}} = 0{
4556        \__stex_variables_invoke_math_ii:n {#1}
4557      }{
4558        \msg_error:nnxx{stex}{error/noop}{variable~#1}{}
4559      }
```

```
4560    }
4561  }
4562
4563  \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n  #1 {
4564    \cs_if_exist:cTF {
4565      stex_var_notation_#1_cs
4566    }{
4567      \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4568        \_stex_reset:N \comp
4569        \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
4570        \_stex_reset:N \STEXInternalCurrentSymbolStr
4571        \bool_set_true:N \l_stex_allow_semantic_bool
4572      }
4573      \def\comp{\_varcomp}
4574      \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4575      \bool_set_false:N \l_stex_allow_semantic_bool
4576      \use:c{stex_var_notation_#1_cs}
4577    }{
4578      \msg_error:nnxx{stex}{error/nonotation}{variable~#1}{s}
4579    }
4580  }
4581
4582  \cs_new_protected:Nn \__stex_variables_invoke_custom:nn {
4583    \exp_args:Nnx \use:nn {
4584      \def\comp{\_varcomp}
4585      \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4586      \prop_clear:N \l__stex_terms_custom_args_prop
4587      \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
4588      \prop_get:cnN {
4589        l_stex_symdecl_var://#1 _prop
4590      }{ args } \l_tmpa_str
4591      \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
4592      \tl_set:Nn \arg { \__stex_terms_arg: }
4593      \str_if_empty:NTF \l_tmpa_str {
4594        \_stex_term_omv:nn {var://#1}{\ignorespaces#2}
4595      }{
4596        \str_if_in:NnTF \l_tmpa_str b {
4597          \_stex_term_ombind:nnn {var://#1}{}{\ignorespaces#2}
4598        }{
4599          \str_if_in:NnTF \l_tmpa_str B {
4600            \_stex_term_ombind:nnn {var://#1}{}{\ignorespaces#2}
4601          }{
4602            \_stex_term_oma:nnn {var://#1}{}{\ignorespaces#2}
4603          }
4604        }
4605      }
4606      % TODO check that all arguments exist
4607    }{
4608      \_stex_reset:N \STEXInternalCurrentSymbolStr
4609      \_stex_reset:N \arg
4610      \_stex_reset:N \comp
4611      \_stex_reset:N \l__stex_terms_custom_args_prop
4612      %\bool_set_true:N \l_stex_allow_semantic_bool
4613    }
```

```
4614 }
```

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

30.5   Sequences

```
4615 ⟨@@=stex_sequences⟩
4616
4617 \cs_new_protected:Nn \stex_invoke_sequence:n {
4618   \peek_charcode_remove:NTF ! {
4619     \_stex_term_omv:nn {varseq://#1}{
4620       \exp_args:Nnx \use:nn {
4621         \def\comp{\_varcomp}
4622         \str_set:Nn \STEXInternalCurrentSymbolStr {varseq://#1}
4623         \prop_item:cn{l_stex_symdecl_varseq://#1_prop}{notation}
4624       }{
4625         \_stex_reset:N \comp
4626         \_stex_reset:N \STEXInternalCurrentSymbolStr
4627       }
4628     }
4629   }{
4630     \bool_set_false:N \l_stex_allow_semantic_bool
4631     \def\comp{\_varcomp}
4632     \str_set:Nn \STEXInternalCurrentSymbolStr {varseq://#1}
4633     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4634       \_stex_reset:N \comp
4635       \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
4636       \_stex_reset:N \STEXInternalCurrentSymbolStr
4637       \bool_set_true:N \l_stex_allow_semantic_bool
4638     }
4639     \use:c { stex_varseq_#1_cs }
4640   }
4641 }
4642 ⟨/package⟩
```

Chapter 31

sTeX -Structural Features Implementation

```
4643  ⟨∗package⟩
4644
4645  %%%%%%%%%%%%   features.dtx   %%%%%%%%%%%%
4646
```

Warnings and error messages

```
4647  \msg_new:nnn{stex}{error/copymodule/notallowed}{
4648    Symbol~#1~can~not~be~assigned~in~copymodule~#2
4649  }
4650  \msg_new:nnn{stex}{error/interpretmodule/nodefiniens}{
4651    Symbol~#1~not~assigned~in~interpretmodule~#2
4652  }
4653
4654  \msg_new:nnn{stex}{error/unknownstructure}{
4655    No~structure~#1~found!
4656  }
4657
4658  \msg_new:nnn{stex}{error/unknownfield}{
4659    No~field~#1~in~instance~#2~found!\\#3
4660  }
4661
4662  \msg_new:nnn{stex}{error/keyval}{
4663    Invalid~key=value~pair:#1
4664  }
4665  \msg_new:nnn{stex}{error/instantiate/missing}{
4666    Assignments~missing~in~instantiate:~#1
4667  }
4668  \msg_new:nnn{stex}{error/incompatible}{
4669    Incompatible~signature:~#1~(#2)~and~#3~(#4)
4670  }
4671
```

31.1 Imports with modification

```
4672 ⟨@@=stex_copymodule⟩
4673 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
4674   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4675     \tl_set:Nn \l_tmpa_tl { #1 }
4676     \__stex_copymodule_get_symbol_from_cs:
4677   }{
4678     % argument is a string
4679     % is it a command name?
4680     \cs_if_exist:cTF { #1 }{
4681       \cs_set_eq:Nc \l_tmpa_tl { #1 }
4682       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4683       \str_if_empty:NTF \l_tmpa_str {
4684         \exp_args:Nx \cs_if_eq:NNTF {
4685           \tl_head:N \l_tmpa_tl
4686         } \stex_invoke_symbol:n {
4687           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
4688         }{
4689           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4690         }
4691       } {
4692         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4693       }
4694     }{
4695       % argument is not a command name
4696       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4697       % \l_stex_all_symbols_seq
4698     }
4699   }
4700 }
4701
4702 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
4703   \str_set:Nn \l_tmpa_str { #1 }
4704   \bool_set_false:N \l_tmpa_bool
4705   \bool_if:NF \l_tmpa_bool {
4706     \tl_set:Nn \l_tmpa_tl {
4707       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
4708     }
4709   \str_set:Nn \l_tmpa_str { #1 }
4710   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4711   \seq_map_inline:Nn #2 {
4712     \str_set:Nn \l_tmpb_str { ##1 }
4713     \str_if_eq:eeT { \l_tmpa_str } {
4714       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4715     } {
4716       \seq_map_break:n {
4717         \tl_set:Nn \l_tmpa_tl {
4718           \str_set:Nn \l_stex_get_symbol_uri_str {
4719             ##1
4720           }
4721         }
4722       }
4723     }
```

204

```
4724        }
4725      \l_tmpa_tl
4726    }
4727 }
4728
4729 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
4730    \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4731      { \tl_tail:N \l_tmpa_tl }
4732    \tl_if_single:NTF \l_tmpa_tl {
4733      \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
4734        \exp_after:wN \str_set:Nn \exp_after:wN
4735          \l_stex_get_symbol_uri_str \l_tmpa_tl
4736        \__stex_copymodule_get_symbol_check:n { #1 }
4737      }{
4738        % TODO
4739        % tail is not a single group
4740      }
4741    }{
4742      % TODO
4743      % tail is not a single group
4744    }
4745 }
4746
4747 \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
4748    \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
4749      \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
4750        :~\seq_use:Nn #1 {,~}
4751      }
4752    }
4753 }
4754
4755 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
4756    % import module
4757    \stex_import_module_uri:nn { #1 } { #2 }
4758    \str_set:Nx \l_stex_current_copymodule_name_str {#3}
4759    \stex_import_require_module:nnnn
4760      { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4761      { \l_stex_import_path_str } { \l_stex_import_name_str }
4762
4763    \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
4764    \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
4765
4766    % fields
4767    \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
4768    \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4769      \seq_map_inline:cn {c_stex_module_##1_constants}{
4770        \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
4771          ##1 ? ####1
4772        }
4773      }
4774    }
4775
4776    % setup prop
4777    \seq_clear:N \l_tmpa_seq
```

```
4778    \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
4779      name     = \l_stex_current_copymodule_name_str ,
4780      module   = \l_stex_current_module_str ,
4781      from     = \l_stex_import_ns_str ?\l_stex_import_name_str ,
4782      includes = \l_tmpa_seq %,
4783  %  fields   = \l_tmpa_seq
4784    }
4785    \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
4786      as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
4787    \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_se
4788    \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,

4790    \stex_if_do_html:T {
4791      \begin{stex_annotate_env} {#4} {
4792        \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4793      }
4794      \stex_annotate_invisible:nnn{domain}{\l_stex_import_ns_str ?\l_stex_import_name_str}{}
4795    }
4796  }

4798  \cs_new_protected:Nn \stex_copymodule_end:n {
4799    % apply to every field
4800    \def \l_tmpa_cs ##1 ##2 {#1}

4802    \tl_clear:N \l__stex_copymodule_module_tl
4803    \tl_clear:N \l__stex_copymodule_exec_tl

4805    %\prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
4806    \seq_clear:N \l__stex_copymodule_fields_seq

4808    \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4809      \seq_map_inline:cn {c_stex_module_##1_constants}{

4811        \tl_clear:N \l__stex_copymodule_curr_symbol_tl % <- wrap in current symbol html
4812        \l_tmpa_cs{##1}{####1}

4814        \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
4815          \str_set_eq:Nc \l__stex_copymodule_curr_name_str {l__stex_copymodule_copymodule_##1?#
4816          \stex_if_do_html:T {
4817            \tl_put_right:Nx \l__stex_copymodule_curr_symbol_tl {
4818              \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?###
4819            }
4820          }
4821        }{
4822          \str_set:Nx \l__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str /
4823        }

4825        \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
4826        \prop_put:Nnx \l_tmpa_prop { name } \l__stex_copymodule_curr_name_str
4827        \prop_put:Nnx \l_tmpa_prop { module } \l_stex_current_module_str

4829        \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
4830          \stex_if_do_html:T {
4831            \tl_put_right:Nx \l__stex_copymodule_curr_symbol_tl {
```

```
                    $\stex_annotate_invisible:nnn{definiens}{}{\exp_after:wN \exp_not:N\csname l__st
            }
          }
          \prop_put:Nnn \l_tmpa_prop { defined } { true }
        }

        \stex_add_constant_to_current_module:n \__stex_copymodule_curr_name_str
        \tl_put_right:Nx \__stex_copymodule_module_tl {
          \seq_clear:c {l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_n
          \prop_set_from_keyval:cn {
            l_stex_symdecl_\l_stex_current_module_str ? \__stex_copymodule_curr_name_str _prop
          }{
            \prop_to_keyval:N \l_tmpa_prop
          }
        }

        \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
          \stex_if_do_html:T {
            \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
              \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1
            }
          }
          \tl_put_right:Nx \__stex_copymodule_module_tl {
            \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
              \stex_invoke_symbol:n {
                \l_stex_current_module_str ? \__stex_copymodule_curr_name_str
              }
            }
          }
        }

        \seq_put_right:Nx \__stex_copymodule_fields_seq {\l_stex_current_module_str ? \__stex_

        \tl_put_right:Nx \__stex_copymodule_exec_tl {
          \stex_copy_notations:nn {\l_stex_current_module_str ? \__stex_copymodule_curr_name_s
        }

        \tl_put_right:Nx \__stex_copymodule_exec_tl {
          \stex_if_do_html:TF{
            \stex_annotate_invisible:nnn{assignment} {##1?####1} { \exp_after:wN \exp_not:n \e
          }{
            \exp_after:wN \exp_not:n \exp_after:wN {\__stex_copymodule_curr_symbol_tl}
          }
        }
      }
    }
  }


  \prop_put:Nno \l_stex_current_copymodule_prop {fields} \__stex_copymodule_fields_seq
  \tl_put_left:Nx \__stex_copymodule_module_tl {
    \prop_set_from_keyval:cn {
      l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _pro
    }{
      \prop_to_keyval:N \l_stex_current_copymodule_prop
```

```
4886        }
4887      }
4888
4889      \seq_gput_right:cx{c_stex_module_\l_stex_current_module_str _copymodules}{
4890        \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4891      }
4892
4893      \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4894      \stex_debug:nn{copymodule}{result:\meaning \__stex_copymodule_module_tl}
4895      \stex_debug:nn{copymodule}{output:\meaning \__stex_copymodule_exec_tl}
4896
4897      \__stex_copymodule_exec_tl
4898      \stex_if_do_html:T {
4899        \end{stex_annotate_env}
4900      }
4901  }
4902
4903  \NewDocumentEnvironment {copymodule} { O{} m m}{
4904    \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ copymodule }
4905    \stex_deactivate_macro:Nn \symdecl {module~environments}
4906    \stex_deactivate_macro:Nn \symdef {module~environments}
4907    \stex_deactivate_macro:Nn \notation {module~environments}
4908    \stex_reactivate_macro:N \assign
4909    \stex_reactivate_macro:N \renamedecl
4910    \stex_reactivate_macro:N \donotcopy
4911    \stex_smsmode_do:
4912  }{
4913    \stex_copymodule_end:n {}
4914  }
4915
4916  \NewDocumentEnvironment {interpretmodule} { O{} m m}{
4917    \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ interpretmodule }
4918    \stex_deactivate_macro:Nn \symdecl {module~environments}
4919    \stex_deactivate_macro:Nn \symdef {module~environments}
4920    \stex_deactivate_macro:Nn \notation {module~environments}
4921    \stex_reactivate_macro:N \assign
4922    \stex_reactivate_macro:N \renamedecl
4923    \stex_reactivate_macro:N \donotcopy
4924    \stex_smsmode_do:
4925  }{
4926    \stex_copymodule_end:n {
4927      \tl_if_exist:cF {
4928        l__stex_copymodule_copymodule_##1?##2_def_tl
4929      }{
4930        \str_if_eq:eeF {
4931          \prop_item:cn{
4932            l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
4933        }{ true }{
4934          \msg_error:nnxx{stex}{error/interpretmodule/nodefiniens}{
4935            ##1?##2
4936          }{\l_stex_current_copymodule_name_str}
4937        }
4938      }
4939    }
```

208

```
4940 }
4941
4942 \iffalse \begin{stex_annotate_env} \fi
4943 \NewDocumentEnvironment {realization} { O{} m}{
4944   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #2 }{ realize }
4945   \stex_deactivate_macro:Nn \symdecl {module~environments}
4946   \stex_deactivate_macro:Nn \symdef {module~environments}
4947   \stex_deactivate_macro:Nn \notation {module~environments}
4948   \stex_reactivate_macro:N \donotcopy
4949   \stex_reactivate_macro:N \assign
4950   \stex_smsmode_do:
4951 }{
4952   \stex_import_module_uri:nn { #1 } { #2 }
4953   \tl_clear:N \__stex_copymodule_exec_tl
4954   \tl_set:Nx \__stex_copymodule_module_tl {
4955     \stex_import_require_module:nnnn
4956       { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4957       { \l_stex_import_path_str } { \l_stex_import_name_str }
4958   }
4959
4960   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4961     \seq_map_inline:cn {c_stex_module_##1_constants}{
4962       \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str / #
4963       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
4964         \stex_if_do_html:T {
4965           \tl_put_right:Nx \__stex_copymodule_exec_tl {
4966             \stex_annotate_invisible:nnn{assignment} {##1?####1} {
4967               $\stex_annotate_invisible:nnn{definiens}{}{\exp_after:wN \exp_not:N\csname l__
4968             }
4969           }
4970         }
4971         \tl_put_right:Nx \__stex_copymodule_module_tl {
4972           \prop_put:cnn {l_stex_symdecl_##1?####1_prop}{ defined }{ true }
4973         }
4974       }
4975   }}
4976
4977   \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4978
4979   \__stex_copymodule_exec_tl
4980   \stex_if_do_html:T {\end{stex_annotate_env}}
4981 }
4982
4983 \NewDocumentCommand \donotcopy { m }{
4984   \str_clear:N \l_stex_import_name_str
4985   \str_set:Nn \l_tmpa_str { #1 }
4986   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4987   \seq_map_inline:Nn \l_stex_all_modules_seq {
4988     \str_set:Nn \l_tmpb_str { ##1 }
4989     \str_if_eq:eeT { \l_tmpa_str } {
4990       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4991     } {
4992       \seq_map_break:n {
4993         \stex_if_do_html:T {
```

209

```
4994          \stex_if_smsmode:F {
4995            \stex_annotate_invisible:nnn{donotcopy}{##1}{
4996              \stex_annotate:nnn{domain}{##1}{}
4997            }
4998          }
4999        }
5000        \str_set_eq:NN \l_stex_import_name_str \l_tmpb_str
5001      }
5002    }
5003    \seq_map_inline:cn {c_stex_module_##1_copymodules}{
5004      \str_set:Nn \l_tmpb_str { ####1 }
5005      \str_if_eq:eeT { \l_tmpa_str } {
5006        \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
5007      } {
5008        \seq_map_break:n {\seq_map_break:n {
5009          \stex_if_do_html:T {
5010            \stex_if_smsmode:F {
5011              \stex_annotate_invisible:nnn{donotcopy}{####1}{
5012                \stex_annotate:nnn{domain}{
5013                  \prop_item:cn {l_stex_copymodule_ ####1 _prop}{module}
5014                }{}
5015              }
5016            }
5017          }
5018          \str_set:Nx \l_stex_import_name_str {
5019            \prop_item:cn {l_stex_copymodule_ ####1 _prop}{module}
5020          }
5021        }}
5022      }
5023    }
5024  }
5025  \str_if_empty:NTF \l_stex_import_name_str {
5026    % TODO throw error
5027  }{
5028    \stex_collect_imports:n {\l_stex_import_name_str }
5029    \seq_map_inline:Nn \l_stex_collect_imports_seq {
5030      \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
5031      \seq_map_inline:cn {c_stex_module_##1_constants}{
5032        \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? ####1 }
5033        \bool_lazy_any:nT {
5034          { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_name_str}}
5035          { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_macroname_str}}
5036          { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_def_tl}}
5037        }{
5038          % TODO throw error
5039        }
5040      }
5041    }
5042    \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
5043    \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_name_str }
5044    \prop_put:Nno \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
5045  }
5046  \stex_smsmode_do:
5047 }
```

```
5048
5049 \NewDocumentCommand \assign { m m }{
5050   \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
5051   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
5052   \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _def_tl}{#2}
5053   \stex_smsmode_do:
5054 }
5055
5056 \keys_define:nn { stex / renamedecl } {
5057   name          .str_set_x:N  = \l_stex_renamedecl_name_str
5058 }
5059 \cs_new_protected:Nn \__stex_copymodule_renamedecl_args:n {
5060   \str_clear:N \l_stex_renamedecl_name_str
5061   \keys_set:nn { stex / renamedecl } { #1 }
5062 }
5063
5064 \NewDocumentCommand \renamedecl { O{} m m }{
5065   \__stex_copymodule_renamedecl_args:n { #1 }
5066   \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
5067   \stex_debug:nn{renamedecl}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
5068   \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
5069   \str_if_empty:NTF \l_stex_renamedecl_name_str {
5070     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
5071       \l_stex_get_symbol_uri_str
5072     } }
5073   } {
5074     \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex
5075     \stex_debug:nn{renamedecl}{@~\l_stex_current_module_str ? \l_stex_renamedecl_name_str}
5076     \prop_set_eq:cc {l_stex_symdecl_
5077       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
5078       _prop
5079     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
5080     \seq_set_eq:cc {l_stex_symdecl_
5081       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
5082       _notations
5083     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
5084     \prop_put:cnx {l_stex_symdecl_
5085       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
5086       _prop
5087     }{ name }{ \l_stex_renamedecl_name_str }
5088     \prop_put:cnx {l_stex_symdecl_
5089       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
5090       _prop
5091     }{ module }{ \l_stex_current_module_str }
5092     \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
5093       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
5094     }
5095     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
5096       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
5097     } }
5098   }
5099   \stex_smsmode_do:
5100 }
5101
```

211

```
5102  \stex_deactivate_macro:Nn \assign {copymodules}
5103  \stex_deactivate_macro:Nn \renamedecl {copymodules}
5104  \stex_deactivate_macro:Nn \donotcopy {copymodules}
5105
5106
```

31.2   The feature environment

structural@feature (*env.*)

```
5107  ⟨@@=stex_features⟩
5108
5109  \NewDocumentEnvironment{structural_feature_module}{ m m m }{
5110    \stex_if_in_module:F {
5111      \msg_set:nnn{stex}{error/nomodule}{
5112        Structural~Feature~has~to~occur~in~a~module:\\
5113        Feature~#2~of~type~#1\\
5114        In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
5115      }
5116      \msg_error:nn{stex}{error/nomodule}
5117    }
5118
5119    \str_set_eq:NN \l_stex_feature_parent_str \l_stex_current_module_str
5120
5121    \stex_module_setup:nn{meta=NONE}{#2 - #1}
5122
5123    \stex_if_do_html:T {
5124      \begin{stex_annotate_env}{ feature:#1 }{\l_stex_feature_parent_str ? #2 - #1}
5125        \stex_annotate_invisible:nnn{header}{}{ #3 }
5126    }
5127  }{
5128    \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
5129    \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
5130    \stex_debug:nn{features}{
5131      Feature: \l_stex_last_feature_str
5132    }
5133    \stex_if_do_html:T {
5134      \end{stex_annotate_env}
5135    }
5136  }
```

31.3   Structure

structure (*env.*)

```
5137  ⟨@@=stex_structures⟩
5138  \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
5139    \prop_if_exist:cF {c_stex_module_\l_stex_current_module_str _structures}{
5140      \prop_new:c {c_stex_module_\l_stex_current_module_str _structures}
5141    }
5142    \prop_gput:cxx{c_stex_module_\l_stex_current_module_str _structures}
5143      {#1}{#2}
5144  }
5145
```

```
5146 \keys_define:nn { stex / features / structure } {
5147   name          .str_set_x:N  = \l__stex_structures_name_str ,
5148 }
5149
5150 \cs_new_protected:Nn \__stex_structures_structure_args:n {
5151   \str_clear:N \l__stex_structures_name_str
5152   \keys_set:nn { stex / features / structure } { #1 }
5153 }
5154
5155 \NewDocumentEnvironment{mathstructure}{m O{}}{
5156   \__stex_structures_structure_args:n { #2 }
5157   \str_if_empty:NT \l__stex_structures_name_str {
5158     \str_set:Nx \l__stex_structures_name_str { #1 }
5159   }
5160   \stex_suppress_html:n {
5161     \bool_set_true:N \l_stex_symdecl_make_macro_bool
5162     \exp_args:Nx \stex_symdecl_do:nn {
5163       name = \l__stex_structures_name_str ,
5164       def  = {\STEXsymbol{module-type}{
5165         \STEXInternalTermMathOMSiiii {
5166           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
5167             { ns } ?
5168             \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
5169               { name } / \l__stex_structures_name_str - structure
5170         }{}{0}{}
5171       }}
5172     }{ #1 }
5173   }
5174   \exp_args:Nnnx
5175   \begin{structural_feature_module}{ structure }
5176     { \l__stex_structures_name_str }{}
5177   \stex_smsmode_do:
5178 }{
5179   \end{structural_feature_module}
5180   \_stex_reset_up_to_module:n \l_stex_last_feature_str
5181   \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
5182   \seq_clear:N \l_tmpa_seq
5183   \seq_map_inline:Nn \l_stex_collect_imports_seq {
5184     \seq_map_inline:cn{c_stex_module_##1_constants}{
5185       \seq_put_right:Nn \l_tmpa_seq { ##1 ? ####1 }
5186     }
5187   }
5188   \exp_args:Nnno
5189   \prop_gput:cnn {c_stex_module_ \l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
5190   \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
5191   \stex_add_structure_to_current_module:nn
5192     \l__stex_structures_name_str
5193     \l_stex_last_feature_str
5194
5195   \stex_execute_in_module:x {
5196     \tl_set:cn { #1 }{
5197       \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structure
5198     }
5199   }
```

```
5200 }
5201
5202 \cs_new:Nn \stex_invoke_structure:nn {
5203   \stex_invoke_symbol:n { #1?#2 }
5204 }
5205
5206 \cs_new_protected:Nn \stex_get_structure:n {
5207   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
5208     \tl_set:Nn \l_tmpa_tl { #1 }
5209     \__stex_structures_get_from_cs:
5210   }{
5211     \cs_if_exist:cTF { #1 }{
5212       \cs_set_eq:Nc \l_tmpa_cs { #1 }
5213       \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
5214       \str_if_empty:NTF \l_tmpa_str {
5215         \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs} \stex_invoke_structure:nn {
5216           \__stex_structures_get_from_cs:
5217         }{
5218           \__stex_structures_get_from_string:n { #1 }
5219         }
5220       }{
5221         \__stex_structures_get_from_string:n { #1 }
5222       }
5223     }{
5224       \__stex_structures_get_from_string:n { #1 }
5225     }
5226   }
5227 }
5228
5229 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
5230   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
5231     { \tl_tail:N \l_tmpa_tl }
5232   \str_set:Nx \l_tmpa_str {
5233     \exp_after:wN \use_i:nn \l_tmpa_tl
5234   }
5235   \str_set:Nx \l_tmpb_str {
5236     \exp_after:wN \use_ii:nn \l_tmpa_tl
5237   }
5238   \str_set:Nx \l_stex_get_structure_str {
5239     \l_tmpa_str ? \l_tmpb_str
5240   }
5241   \str_set:Nx \l_stex_get_structure_module_str {
5242     \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
5243   }
5244 }
5245
5246 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
5247   \tl_set:Nn \l_tmpa_tl {
5248     \msg_error:nnn{stex}{error/unknownstructure}{#1}
5249   }
5250   \str_set:Nn \l_tmpa_str { #1 }
5251   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
5252
5253   \seq_map_inline:Nn \l_stex_all_modules_seq {
```

```
5254        \prop_if_exist:cT {c_stex_module_##1_structures} {
5255          \prop_map_inline:cn {c_stex_module_##1_structures} {
5256            \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
5257              \prop_map_break:n{\seq_map_break:n{
5258                \tl_set:Nn \l_tmpa_tl {
5259                  \str_set:Nn \l_stex_get_structure_str {##1?####1}
5260                  \str_set:Nn \l_stex_get_structure_module_str {####2}
5261                }
5262              }}
5263            }
5264          }
5265        }
5266      }
5267    \l_tmpa_tl
5268 }
```

```
5269
5270 \keys_define:nn { stex / instantiate } {
5271    name         .str_set_x:N  = \l__stex_structures_name_str
5272 }
5273 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
5274    \str_clear:N \l__stex_structures_name_str
5275    \keys_set:nn { stex / instantiate } { #1 }
5276 }
5277
5278 \NewDocumentCommand \instantiate {m O{} m m O{}}{
5279    \begingroup
5280      \stex_get_structure:n {#3}
5281      \__stex_structures_instantiate_args:n { #2 }
5282      \str_if_empty:NT \l__stex_structures_name_str {
5283        \str_set:Nn \l__stex_structures_name_str { #1 }
5284      }
5285      \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5286      \seq_clear:N \l__stex_structures_fields_seq
5287      \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
5288      \seq_map_inline:Nn \l_stex_collect_imports_seq {
5289        \seq_map_inline:cn {c_stex_module_##1_constants}{
5290          \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
5291        }
5292      }
5293
5294      \tl_if_empty:nF{#5}{
5295        \seq_set_split:Nnn \l_tmpa_seq , {#5}
5296        \prop_clear:N \l_tmpa_prop
5297        \seq_map_inline:Nn \l_tmpa_seq {
5298          \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
5299          \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
5300            \msg_error:nnn{stex}{error/keyval}{##1}
5301          }
5302          \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
5303          \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
5304          \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
5305          \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}
```

```
5306            \exp_args:Nxx \str_if_eq:nnF
5307              {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5308              {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
5309              \msg_error:nnxxxx{stex}{error/incompatible}
5310                {\l__stex_structures_dom_str}
5311                {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5312                {\l_stex_get_symbol_uri_str}
5313                {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
5314            }
5315            \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str
5316         }
5317       }
5318
5319       \seq_map_inline:Nn \l__stex_structures_fields_seq {
5320         \str_set:Nx \l_tmpa_str {field:\l__stex_structures_name_str . \prop_item:cn {l_stex_sy
5321         \stex_debug:nn{instantiate}{Field~\l_tmpa_str :~##1}
5322
5323         \stex_add_constant_to_current_module:n {\l_tmpa_str}
5324         \stex_execute_in_module:x {
5325           \prop_set_from_keyval:cn { l_stex_symdecl_ \l_stex_current_module_str?\l_tmpa_str _p
5326             name   = \l_tmpa_str ,
5327             args   = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
5328             arity  = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
5329             assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs} ,
5330             argnames = {\prop_item:cn {l_stex_symdecl_##1_prop}{argnames}}
5331           }
5332           \seq_clear:c {l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _notations}
5333         }
5334
5335         \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
5336           \stex_find_notation:nn{##1}{}
5337           \stex_execute_in_module:x {
5338             \seq_put_right:cn {l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _notation
5339           }
5340
5341           \stex_copy_control_sequence_ii:ccN
5342             {stex_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_notation_
5343             {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5344             \l_tmpa_tl
5345           \exp_args:No \stex_execute_in_module:n \l_tmpa_tl
5346
5347
5348           \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
5349             \tl_set_eq:Nc \l_tmpa_cs {stex_op_notation_##1\c_hash_str \l_stex_notation_variant
5350             \stex_execute_in_module:x {
5351               \tl_set:cn
5352               {stex_op_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_nota
5353               { \exp_args:No \exp_not:n \l_tmpa_cs}
5354             }
5355           }
5356
5357         }
5358
5359         \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\l_stex_cur
```

216

```
5360        }
5361
5362        \stex_execute_in_module:x {
5363          \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
5364            domain = \l_stex_get_structure_module_str ,
5365            \prop_to_keyval:N \l_tmpa_prop
5366          }
5367          \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
5368        }
5369        \stex_debug:nn{instantiate}{
5370          Instance~\l_stex_current_module_str?\l__stex_structures_name_str \\
5371          \prop_to_keyval:N \l_tmpa_prop
5372        }
5373        \exp_args:Nxx \stex_symdecl_do:nn {
5374          type={\STEXsymbol{module-type}{
5375            \STEXInternalTermMathOMSiiii {
5376              \l_stex_get_structure_module_str
5377            }{}{0}{}
5378          }}
5379        }{\l__stex_structures_name_str}
5380 %    {
5381        \str_set:Nx \l_stex_get_symbol_uri_str {\l_stex_current_module_str?\l__stex_structures
5382        \tl_set:Nn \l_stex_notation_after_do_tl {\__stex_notation_final:}
5383        \stex_notation_do:nnnnn{}{0}{}{}{\comp{#4}}
5384 %    }
5385        %\exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}
5386      \endgroup
5387      \stex_smsmode_do:\ignorespacesandpars
5388 }
5389
5390 \cs_new_protected:Nn \stex_symbol_or_var:n {
5391    \cs_if_exist:cTF{#1}{
5392      \cs_set_eq:Nc \l_tmpa_tl { # 1 }
5393      \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
5394      \str_if_empty:NTF \l_tmpa_str {
5395        \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
5396          \stex_invoke_variable:n {
5397            \bool_set_true:N \l_stex_symbol_or_var_bool
5398            \bool_set_false:N \l_stex_instance_or_symbol_bool
5399            \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
5400            \tl_set:Nx \l_tmpa_tl {\exp_after:wN \use:n \l_tmpa_tl}
5401            \str_set:Nx \l_stex_get_symbol_uri_str {
5402              \exp_after:wN \use:n \l_tmpa_tl
5403            }
5404          }{ % TODO \stex_invoke_varinstance:n
5405            \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl } \stex_invoke_varinstance:n {
5406              \bool_set_true:N \l_stex_symbol_or_var_bool
5407              \bool_set_true:N \l_stex_instance_or_symbol_bool
5408              \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
5409              \tl_set:Nx \l_tmpa_tl {\exp_after:wN \use:n \l_tmpa_tl}
5410              \str_set:Nx \l_stex_get_symbol_uri_str {
5411                \exp_after:wN \use:n \l_tmpa_tl
5412              }
5413            }{
```

```
5414            \bool_set_false:N \l_stex_symbol_or_var_bool
5415            \stex_get_symbol:n{#1}
5416          }
5417        }
5418    }{
5419      \__stex_structures_symbolorvar_from_string:n{ #1 }
5420    }
5421  }{
5422    \__stex_structures_symbolorvar_from_string:n{ #1 }
5423  }
5424 }
5425
5426 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
5427   \prop_if_exist:cTF {l_stex_symdecl_var://#1 _prop}{
5428     \bool_set_true:N \l_stex_symbol_or_var_bool
5429     \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
5430   }{
5431     \bool_set_false:N \l_stex_symbol_or_var_bool
5432     \stex_get_symbol:n{#1}
5433   }
5434 }
5435
5436 \keys_define:nn { stex / varinstantiate } {
5437   name          .str_set_x:N  = \l__stex_structures_name_str,
5438   bind          .choices:nn   =
5439       {forall,exists}
5440       {\str_set:Nx \l__stex_structures_bind_str {\l_keys_choice_tl}}
5441
5442 }
5443 \cs_new_protected:Nn \__stex_structures_varinstantiate_args:n {
5444   \str_clear:N \l__stex_structures_name_str
5445   \str_clear:N \l__stex_structures_bind_str
5446   \keys_set:nn { stex / varinstantiate } { #1 }
5447 }
5448
5449 \NewDocumentCommand \varinstantiate {m O{} m m O{}}{
5450   \begingroup
5451     \stex_get_structure:n {#3}
5452     \__stex_structures_varinstantiate_args:n { #2 }
5453     \str_if_empty:NT \l__stex_structures_name_str {
5454       \str_set:Nn \l__stex_structures_name_str { #1 }
5455     }
5456     \stex_if_do_html:TF{
5457       \stex_annotate:nnn{varinstance}{\l__stex_structures_name_str}
5458     }{\use:n}
5459     {
5460       \stex_if_do_html:T{
5461         \stex_annotate_invisible:nnn{domain}{\l_stex_get_structure_module_str}{}
5462       }
5463       \seq_clear:N \l__stex_structures_fields_seq
5464       \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
5465       \seq_map_inline:Nn \l_stex_collect_imports_seq {
5466         \seq_map_inline:cn {c_stex_module_##1_constants}{
5467           \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
```

```
5468              }
5469            }
5470          \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5471          \prop_clear:N \l_tmpa_prop
5472          \tl_if_empty:nF {#5} {
5473            \seq_set_split:Nnn \l_tmpa_seq , {#5}
5474            \seq_map_inline:Nn \l_tmpa_seq {
5475              \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
5476              \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
5477                \msg_error:nnn{stex}{error/keyval}{##1}
5478              }
5479              \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_stru
5480              \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
5481              \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol
5482              \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
5483              \stex_if_do_html:T{
5484                \stex_annotate:nnn{assign}{\l__stex_structures_dom_str,
5485                \bool_if:NTF\l_stex_symbol_or_var_bool{var://}{}\l_stex_get_symbol_uri_str}{}
5486              }
5487              \bool_if:NTF \l_stex_symbol_or_var_bool {
5488                \exp_args:Nxx \str_if_eq:nnF
5489                  {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5490                  {\prop_item:cn{l_stex_symdecl_var://\l_stex_get_symbol_uri_str _prop}{args}}{
5491                  \msg_error:nnxxxx{stex}{error/incompatible}
5492                    {\l__stex_structures_dom_str}
5493                    {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5494                    {\l_stex_get_symbol_uri_str}
5495                    {\prop_item:cn{l_stex_symdecl_var://\l_stex_get_symbol_uri_str _prop}{args}}
5496                }
5497                \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n
5498              }{
5499                \exp_args:Nxx \str_if_eq:nnF
5500                  {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5501                  {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
5502                  \msg_error:nnxxxx{stex}{error/incompatible}
5503                    {\l__stex_structures_dom_str}
5504                    {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5505                    {\l_stex_get_symbol_uri_str}
5506                    {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
5507                }
5508                \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n {
5509              }
5510            }
5511          }
5512          \tl_gclear:N \g__stex_structures_aftergroup_tl
5513          \seq_map_inline:Nn \l__stex_structures_fields_seq {
5514            \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdec
5515            \stex_debug:nn{varinstantiate}{Field~\l_tmpa_str :~##1}
5516            \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
5517              \stex_find_notation:nn{##1}{}
5518              \cs_gset_eq:cc{g__stex_structures_tmpa_\l_tmpa_str _cs}
5519                {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5520              \stex_debug:nn{varinstantiate}{Notation:~\cs_meaning:c{g__stex_structures_tmpa_\l_
5521              \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
```

219

```
5522              \cs_gset_eq:cc {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
5523                {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5524                \stex_debug:nn{varinstantiate}{Operator~Notation:~\cs_meaning:c{g__stex_struct
5525            }
5526          }
5527
5528          \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
5529            \prop_set_from_keyval:cn { l_stex_symdecl_ var://\l_tmpa_str _prop}{
5530              name  = \l_tmpa_str ,
5531              args  = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
5532              arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
5533              assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs} ,
5534              argnames = {\prop_item:cn {l_stex_symdecl_##1_prop}{argnames}} ,
5535            }
5536            \cs_set_eq:cc {stex_var_notation_\l_tmpa_str _cs}
5537              {g__stex_structures_tmpa_\l_tmpa_str _cs}
5538            \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str _cs}
5539              {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
5540          }
5541          \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\stex_inv
5542        }
5543        \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
5544          \prop_set_from_keyval:cn {l_stex_varinstance_\l__stex_structures_name_str _prop }{
5545            domain = \l_stex_get_structure_module_str ,
5546            \prop_to_keyval:N \l_tmpa_prop
5547          }
5548          \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
5549          \tl_set:cn {l_stex_varinstance_\l__stex_structures_name_str _op_tl}{
5550            \exp_args:Nnx \exp_not:N \use:nn {
5551              \str_set:Nn \exp_not:N \STEXInternalCurrentSymbolStr {var://\l__stex_structures_
5552              \_stex_term_omv:nn {var://\l__stex_structures_name_str}{
5553                \exp_not:n{
5554                  \_varcomp{#4}
5555                }
5556              }
5557            }{
5558              \exp_not:n{\_stex_reset:N \STEXInternalCurrentSymbolStr}
5559            }
5560          }
5561        }
5562      }
5563      \stex_debug:nn{varinstantiate}{\expandafter\detokenize\expandafter{\g__stex_structures_a
5564      \aftergroup\g__stex_structures_aftergroup_tl
5565    \endgroup
5566    \stex_smsmode_do:\ignorespacesandpars
5567 }
5568
5569 \cs_new_protected:Nn \stex_invoke_instance:n {
5570    \peek_charcode_remove:NTF ! {
5571      \stex_invoke_symbol:n{#1}
5572    }{
5573      \_stex_invoke_instance:nn {#1}
5574    }
5575 }
```

220

```
5576
5577
5578 \cs_new_protected:Nn \stex_invoke_varinstance:n {
5579   \peek_charcode_remove:NTF ! {
5580     \exp_args:Nnx \use:nn {
5581       \def\comp{\_varcomp}
5582       \use:c{l_stex_varinstance_#1_op_tl}
5583     }{
5584       \_stex_reset:N \comp
5585     }
5586   }{
5587     \_stex_invoke_varinstance:nn {#1}
5588   }
5589 }
5590
5591 \cs_new_protected:Nn \_stex_invoke_instance:nn {
5592   \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
5593     \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
5594   }{
5595     \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ # 1 _prop}
5596     \msg_error:nnxxx{stex}{error/unknownfield}{#2}{#1}{
5597       \prop_to_keyval:N \l_tmpa_prop
5598     }
5599   }
5600 }
5601
5602 \cs_new_protected:Nn \_stex_invoke_varinstance:nn {
5603   \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
5604     \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
5605     \l_tmpa_tl
5606   }{
5607     \msg_error:nnnnn{stex}{error/unknownfield}{#2}{#1}{}
5608   }
5609 }
```

(*End definition for* \instantiate. *This function is documented on page* *34*.)

\stex_invoke_structure:nnn

```
5610 % #1: URI of the instance
5611 % #2: URI of the instantiated module
5612 \cs_new_protected:Nn \stex_invoke_structure:nnn {
5613   \tl_if_empty:nTF{ #3 }{
5614     \prop_set_eq:Nc \l__stex_structures_structure_prop {
5615       c_stex_feature_ #2 _prop
5616     }
5617     \tl_clear:N \l_tmpa_tl
5618     \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
5619     \seq_map_inline:Nn \l_tmpa_seq {
5620       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
5621       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
5622       \cs_if_exist:cT {
5623         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
5624       }{
5625         \tl_if_empty:NF \l_tmpa_tl {
```

```
5626            \tl_put_right:Nn \l_tmpa_tl {,}
5627          }
5628          \tl_put_right:Nx \l_tmpa_tl {
5629            \stex_invoke_symbol:n {#1/\l_tmpa_str}!
5630          }
5631        }
5632      }
5633      \exp_args:No \mathstruct \l_tmpa_tl
5634    }{
5635      \stex_invoke_symbol:n{#1/#3}
5636    }
5637 }
```

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

5638 ⟨/package⟩

Chapter 32

STEX
-Statements Implementation

```
5639 ⟨∗package⟩
5640
5641 %%%%%%%%%%%%    features.dtx    %%%%%%%%%%%%
5642
5643 ⟨@@=stex_statements⟩
```

Warnings and error messages

```
5644
```

\titleemph

```
5645 \def\titleemph#1{\textbf{#1}}
```

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

32.1 Definitions

definiendum

```
5646 \keys_define:nn {stex / definiendum }{
5647   pre      .tl_set:N     = \l__stex_statements_definiendum_pre_tl,
5648   post     .tl_set:N     = \l__stex_statements_definiendum_post_tl,
5649   root     .str_set_x:N  = \l__stex_statements_definiendum_root_str,
5650   gfa      .str_set_x:N  = \l__stex_statements_definiendum_gfa_str
5651 }
5652 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
5653   \str_clear:N \l__stex_statements_definiendum_root_str
5654   \tl_clear:N \l__stex_statements_definiendum_post_tl
5655   \str_clear:N \l__stex_statements_definiendum_gfa_str
5656   \keys_set:nn { stex / definiendum }{ #1 }
5657 }
5658 \NewDocumentCommand \definiendum { O{} m m} {
5659   \__stex_statements_definiendum_args:n { #1 }
5660   \stex_get_symbol:n { #2 }
5661   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5662   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
5663     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

```
5664        \tl_set:Nn \l_tmpa_tl { #3 }
5665      } {
5666        \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
5667        \tl_set:Nn \l_tmpa_tl {
5668          \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
5669        }
5670      }
5671    } {
5672      \tl_set:Nn \l_tmpa_tl { #3 }
5673    }
5674
5675    % TODO root
5676    \stex_html_backend:TF {
5677      \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
5678    } {
5679      \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
5680    }
5681 }
5682 \stex_deactivate_macro:Nn \definiendum {definition~environments}
```

(*End definition for* `definiendum`. *This function is documented on page* *44.*)

definame

```
5683
5684 \NewDocumentCommand \definame { O{} m } {
5685    \__stex_statements_definiendum_args:n { #1 }
5686    % TODO: root
5687    \stex_get_symbol:n { #2 }
5688    \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5689    \str_set:Nx \l_tmpa_str {
5690      \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5691    }
5692    \str_replace_all:Nnn \l_tmpa_str {-} {~}
5693    \stex_html_backend:TF {
5694      \stex_if_do_html:T {
5695        \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5696          \l_tmpa_str\l__stex_statements_definiendum_post_tl
5697        }
5698      }
5699    } {
5700      \exp_args:Nnx \defemph@uri {
5701        \l_tmpa_str\l__stex_statements_definiendum_post_tl
5702      } { \l_stex_get_symbol_uri_str }
5703    }
5704 }
5705 \stex_deactivate_macro:Nn \definame {definition~environments}
5706
5707 \NewDocumentCommand \Definame { O{} m } {
5708    \__stex_statements_definiendum_args:n { #1 }
5709    \stex_get_symbol:n { #2 }
5710    \str_set:Nx \l_tmpa_str {
5711      \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5712    }
5713    \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
```

```
5714    \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5715    \stex_html_backend:TF {
5716      \stex_if_do_html:T {
5717        \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5718          \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5719        }
5720      }
5721    } {
5722      \exp_args:Nnx \defemph@uri {
5723        \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5724      } { \l_stex_get_symbol_uri_str }
5725    }
5726  }
5727  \stex_deactivate_macro:Nn \Definame {definition~environments}
5728
5729  \NewDocumentCommand \premise { m }{
5730    \noindent\stex_annotate:nnn{ premise }{}{\ignorespaces #1 }
5731  }
5732  \NewDocumentCommand \conclusion { m }{
5733    \noindent\stex_annotate:nnn{ conclusion }{}{\ignorespaces #1 }
5734  }
5735  \NewDocumentCommand \definiens { O{} m }{
5736    \str_clear:N \l_stex_get_symbol_uri_str
5737    \tl_if_empty:nF {#1} {
5738      \stex_get_symbol:n { #1 }
5739    }
5740    \str_if_empty:NT \l_stex_get_symbol_uri_str {
5741      \int_compare:nNnTF {\clist_count:N \l__stex_statements_sdefinition_for_clist} = 1 {
5742        \str_set:Nx \l_stex_get_symbol_uri_str {\clist_item:Nn \l__stex_statements_sdefinition
5743      }{
5744        % TODO throw error
5745      }
5746    }
5747    \str_if_eq:eeT {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{module}}
5748      {\l_stex_current_module_str}{
5749        \str_if_eq:eeF {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defin
5750        {true}{
5751          \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5752          \exp_args:Nx \stex_add_to_current_module:n {
5753            \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5754          }
5755        }
5756    }
5757    \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
5758  }
5759
5760  \NewDocumentCommand \varbindforall {m}{
5761    \stex_symbol_or_var:n {#1}
5762    \bool_if:NTF\l_stex_symbol_or_var_bool{
5763      \stex_if_do_html:T {
5764        \stex_annotate_invisible:nnn {bindtype}{forall,\l_stex_get_symbol_uri_str}{}
5765      }
5766    }{
5767      % todo throw error
```

```
5768      }
5769 }
5770
5771 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
5772 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
5773 \stex_deactivate_macro:Nn \definiens {definition~environments}
5774 \stex_deactivate_macro:Nn \varbindforall {definition~or~assertion~environments}
5775
```

(*End definition for* `definame`. *This function is documented on page* *44.*)

sdefinition (*env.*)

```
5776
5777 \keys_define:nn {stex / sdefinition }{
5778   type     .str_set_x:N  = \sdefinitiontype,
5779   id       .str_set_x:N  = \sdefinitionid,
5780   name     .str_set_x:N  = \sdefinitionname,
5781   for      .clist_set:N   = \l__stex_statements_sdefinition_for_clist ,
5782   title    .tl_set:N      = \sdefinitiontitle
5783 }
5784 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
5785   \str_clear:N \sdefinitiontype
5786   \str_clear:N \sdefinitionid
5787   \str_clear:N \sdefinitionname
5788   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5789   \tl_clear:N \sdefinitiontitle
5790   \keys_set:nn { stex / sdefinition }{ #1 }
5791 }
5792
5793 \NewDocumentEnvironment{sdefinition}{O{}}{
5794   \__stex_statements_sdefinition_args:n{ #1 }
5795   \stex_reactivate_macro:N \definiendum
5796   \stex_reactivate_macro:N \definame
5797   \stex_reactivate_macro:N \Definame
5798   \stex_reactivate_macro:N \premise
5799   \stex_reactivate_macro:N \definiens
5800   \stex_reactivate_macro:N \varbindforall
5801   \stex_if_smsmode:F{
5802     \seq_clear:N \l_tmpb_seq
5803     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5804       \tl_if_empty:nF{ ##1 }{
5805         \stex_get_symbol:n { ##1 }
5806         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5807           \l_stex_get_symbol_uri_str
5808         }
5809       }
5810     }
5811     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5812     \exp_args:Nnnx
5813     \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpb_seq {,}}
5814     \str_if_empty:NF \sdefinitiontype {
5815       \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5816     }
5817     \str_if_empty:NF \sdefinitionname {
```

226

```
5818        \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5819      }
5820      \clist_set:No \l_tmpa_clist \sdefinitiontype
5821      \tl_clear:N \l_tmpa_tl
5822      \clist_map_inline:Nn \l_tmpa_clist {
5823        \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
5824          \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
5825        }
5826      }
5827      \tl_if_empty:NTF \l_tmpa_tl {
5828        \__stex_statements_sdefinition_start:
5829      }{
5830        \l_tmpa_tl
5831      }
5832    }
5833    \stex_ref_new_doc_target:n \sdefinitionid
5834    \stex_smsmode_do:
5835 }{
5836    \stex_suppress_html:n {
5837      \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5838    }
5839    \stex_if_smsmode:F {
5840      \clist_set:No \l_tmpa_clist \sdefinitiontype
5841      \tl_clear:N \l_tmpa_tl
5842      \clist_map_inline:Nn \l_tmpa_clist {
5843        \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
5844          \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
5845        }
5846      }
5847      \tl_if_empty:NTF \l_tmpa_tl {
5848        \__stex_statements_sdefinition_end:
5849      }{
5850        \l_tmpa_tl
5851      }
5852      \end{stex_annotate_env}
5853    }
5854 }
```

```
5855 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
5856    \stex_par:\noindent\titleemph{Definition\tl_if_empty:NF \sdefinitiontitle {
5857      ~(\sdefinitiontitle)
5858    }~}
5859 }
5860 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\stex_par:\medskip}
5861
5862 \newcommand\stexpatchdefinition[3][] {
5863    \str_set:Nx \l_tmpa_str{ #1 }
5864    \str_if_empty:NTF \l_tmpa_str {
5865      \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
5866      \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
5867    }{
5868      \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2
5869      \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
```

```
5870        }
5871    }
```

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

\inlinedef   inline:
```
5872  \keys_define:nn {stex / inlinedef }{
5873    type     .str_set_x:N  = \sdefinitiontype,
5874    id       .str_set_x:N  = \sdefinitionid,
5875    for      .clist_set:N   = \l__stex_statements_sdefinition_for_clist ,
5876    name     .str_set_x:N  = \sdefinitionname
5877  }
5878  \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
5879    \str_clear:N \sdefinitiontype
5880    \str_clear:N \sdefinitionid
5881    \str_clear:N \sdefinitionname
5882    \clist_clear:N \l__stex_statements_sdefinition_for_clist
5883    \keys_set:nn { stex / inlinedef }{ #1 }
5884  }
5885  \NewDocumentCommand \inlinedef { O{} m } {
5886    \begingroup
5887    \__stex_statements_inlinedef_args:n{ #1 }
5888    \stex_reactivate_macro:N \definiendum
5889    \stex_reactivate_macro:N \definame
5890    \stex_reactivate_macro:N \Definame
5891    \stex_reactivate_macro:N \premise
5892    \stex_reactivate_macro:N \definiens
5893    \stex_reactivate_macro:N \varbindforall
5894    \stex_ref_new_doc_target:n \sdefinitionid
5895    \stex_if_smsmode:TF{\stex_suppress_html:n {
5896      \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5897    }}{
5898      \seq_clear:N \l_tmpb_seq
5899      \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5900        \tl_if_empty:nF{ ##1 }{
5901          \stex_get_symbol:n { ##1 }
5902          \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5903            \l_stex_get_symbol_uri_str
5904          }
5905        }
5906      }
5907      \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5908      \exp_args:Nnx
5909      \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpb_seq {,}}{
5910        \str_if_empty:NF \sdefinitiontype {
5911          \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5912        }
5913        #2
5914        \str_if_empty:NF \sdefinitionname {
5915          \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sdefinitionname}}
5916          \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5917        }
5918      }
5919    }
```

```
5920    \endgroup
5921    \stex_smsmode_do:
5922 }
```

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

32.2   Assertions

sassertion (*env.*)

```
5923
5924 \keys_define:nn {stex / sassertion }{
5925   type    .str_set_x:N  = \sassertiontype,
5926   id      .str_set_x:N  = \sassertionid,
5927   title   .tl_set:N     = \sassertiontitle ,
5928   for     .clist_set:N  = \l__stex_statements_sassertion_for_clist ,
5929   name    .str_set_x:N  = \sassertionname
5930 }
5931 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
5932   \str_clear:N \sassertiontype
5933   \str_clear:N \sassertionid
5934   \str_clear:N \sassertionname
5935   \clist_clear:N \l__stex_statements_sassertion_for_clist
5936   \tl_clear:N \sassertiontitle
5937   \keys_set:nn { stex / sassertion }{ #1 }
5938 }
5939
5940 %\tl_new:N \g__stex_statements_aftergroup_tl
5941
5942 \NewDocumentEnvironment{sassertion}{O{}}{
5943   \__stex_statements_sassertion_args:n{ #1 }
5944   \stex_reactivate_macro:N \premise
5945   \stex_reactivate_macro:N \conclusion
5946   \stex_reactivate_macro:N \varbindforall
5947   \stex_if_smsmode:F {
5948     \seq_clear:N \l_tmpb_seq
5949     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5950       \tl_if_empty:nF{ ##1 }{
5951         \stex_get_symbol:n { ##1 }
5952         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5953           \l_stex_get_symbol_uri_str
5954         }
5955       }
5956     }
5957     \exp_args:Nnnx
5958     \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
5959     \str_if_empty:NF \sassertiontype {
5960       \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
5961     }
5962     \str_if_empty:NF \sassertionname {
5963       \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
5964     }
5965     \clist_set:No \l_tmpa_clist \sassertiontype
5966     \tl_clear:N \l_tmpa_tl
```

229

```
5967        \clist_map_inline:Nn \l_tmpa_clist {
5968          \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
5969            \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
5970          }
5971        }
5972        \tl_if_empty:NTF \l_tmpa_tl {
5973          \__stex_statements_sassertion_start:
5974        }{
5975          \l_tmpa_tl
5976        }
5977      }
5978      \str_if_empty:NTF \sassertionid {
5979        \str_if_empty:NF \sassertionname {
5980          \stex_ref_new_doc_target:n {}
5981        }
5982      } {
5983        \stex_ref_new_doc_target:n \sassertionid
5984      }
5985      \stex_smsmode_do:
5986    }{
5987      \str_if_empty:NF \sassertionname {
5988        \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5989        \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5990      }
5991      \stex_if_smsmode:F {
5992        \clist_set:No \l_tmpa_clist \sassertiontype
5993        \tl_clear:N \l_tmpa_tl
5994        \clist_map_inline:Nn \l_tmpa_clist {
5995          \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
5996            \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
5997          }
5998        }
5999        \tl_if_empty:NTF \l_tmpa_tl {
6000          \__stex_statements_sassertion_end:
6001        }{
6002          \l_tmpa_tl
6003        }
6004        \end{stex_annotate_env}
6005      }
6006    }
```

\stexpatchassertion

```
6007
6008  \cs_new_protected:Nn \__stex_statements_sassertion_start: {
6009    \stex_par:\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {
6010      (\sassertiontitle)
6011    }~}
6012  }
6013  \cs_new_protected:Nn \__stex_statements_sassertion_end: {\stex_par:\medskip}
6014
6015  \newcommand\stexpatchassertion[3][] {
6016      \str_set:Nx \l_tmpa_str{ #1 }
6017      \str_if_empty:NTF \l_tmpa_str {
6018        \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
```

```
6019        \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
6020      }{
6021        \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
6022        \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
6023      }
6024 }
```

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

\inlineass   inline:

```
6025 \keys_define:nn {stex / inlineass }{
6026    type     .str_set_x:N  = \sassertiontype,
6027    id       .str_set_x:N  = \sassertionid,
6028    for      .clist_set:N   = \l__stex_statements_sassertion_for_clist ,
6029    name     .str_set_x:N  = \sassertionname
6030 }
6031 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
6032    \str_clear:N \sassertiontype
6033    \str_clear:N \sassertionid
6034    \str_clear:N \sassertionname
6035    \clist_clear:N \l__stex_statements_sassertion_for_clist
6036    \keys_set:nn { stex / inlineass }{ #1 }
6037 }
6038 \NewDocumentCommand \inlineass { O{} m } {
6039    \begingroup
6040    \stex_reactivate_macro:N \premise
6041    \stex_reactivate_macro:N \conclusion
6042    \stex_reactivate_macro:N \varbindforall
6043    \__stex_statements_inlineass_args:n{ #1 }
6044    \str_if_empty:NTF \sassertionid {
6045      \str_if_empty:NF \sassertionname {
6046        \stex_ref_new_doc_target:n {}
6047      }
6048    } {
6049      \stex_ref_new_doc_target:n \sassertionid
6050    }
6051
6052    \stex_if_smsmode:TF{
6053      \str_if_empty:NF \sassertionname {
6054        \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
6055        \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
6056      }
6057    }{
6058      \seq_clear:N \l_tmpb_seq
6059      \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
6060        \tl_if_empty:nF{ ##1 }{
6061          \stex_get_symbol:n { ##1 }
6062          \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6063            \l_stex_get_symbol_uri_str
6064          }
6065        }
6066      }
6067      \exp_args:Nnx
6068      \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpb_seq {,}}{
```

231

```
6069        \str_if_empty:NF \sassertiontype {
6070          \stex_annotate_invisible:nnn{typestrings}{\sassertiontype}{}
6071        }
6072        #2
6073        \str_if_empty:NF \sassertionname {
6074          \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
6075          \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
6076          \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
6077        }
6078      }
6079    }
6080    \endgroup
6081    \stex_smsmode_do:
6082  }
```

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

32.3   Examples

sexample (*env.*)

```
6083
6084  \keys_define:nn {stex / sexample }{
6085    type     .str_set_x:N  = \exampletype,
6086    id       .str_set_x:N  = \sexampleid,
6087    title    .tl_set:N     = \sexampletitle,
6088    name     .str_set_x:N  = \sexamplename ,
6089    for      .clist_set:N  = \l__stex_statements_sexample_for_clist,
6090  }
6091  \cs_new_protected:Nn \__stex_statements_sexample_args:n {
6092    \str_clear:N \sexampletype
6093    \str_clear:N \sexampleid
6094    \str_clear:N \sexamplename
6095    \tl_clear:N \sexampletitle
6096    \clist_clear:N \l__stex_statements_sexample_for_clist
6097    \keys_set:nn { stex / sexample }{ #1 }
6098  }
6099
6100  \NewDocumentEnvironment{sexample}{O{}}{
6101    \__stex_statements_sexample_args:n{ #1 }
6102    \stex_reactivate_macro:N \premise
6103    \stex_reactivate_macro:N \conclusion
6104    \stex_if_smsmode:F {
6105      \seq_clear:N \l_tmpb_seq
6106      \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
6107        \tl_if_empty:nF{ ##1 }{
6108          \stex_get_symbol:n { ##1 }
6109          \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6110            \l_stex_get_symbol_uri_str
6111          }
6112        }
6113      }
6114      \exp_args:Nnnx
6115      \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpb_seq {,}}
```

```
6116        \str_if_empty:NF \sexampletype {
6117          \stex_annotate_invisible:nnn{typestrings}{\sexampletype}{}
6118        }
6119        \str_if_empty:NF \sexamplename {
6120          \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}
6121        }
6122        \clist_set:No \l_tmpa_clist \sexampletype
6123        \tl_clear:N \l_tmpa_tl
6124        \clist_map_inline:Nn \l_tmpa_clist {
6125          \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
6126            \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
6127          }
6128        }
6129        \tl_if_empty:NTF \l_tmpa_tl {
6130          \__stex_statements_sexample_start:
6131        }{
6132          \l_tmpa_tl
6133        }
6134      }
6135      \str_if_empty:NF \sexampleid {
6136        \stex_ref_new_doc_target:n \sexampleid
6137      }
6138      \stex_smsmode_do:
6139    }{
6140      \str_if_empty:NF \sexamplename {
6141        \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
6142      }
6143      \stex_if_smsmode:F {
6144        \clist_set:No \l_tmpa_clist \sexampletype
6145        \tl_clear:N \l_tmpa_tl
6146        \clist_map_inline:Nn \l_tmpa_clist {
6147          \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
6148            \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
6149          }
6150        }
6151        \tl_if_empty:NTF \l_tmpa_tl {
6152          \__stex_statements_sexample_end:
6153        }{
6154          \l_tmpa_tl
6155        }
6156        \end{stex_annotate_env}
6157      }
6158  }
```

\stexpatchexample

```
6159
6160  \cs_new_protected:Nn \__stex_statements_sexample_start: {
6161    \stex_par:\noindent\titleemph{Example~\tl_if_empty:NF \sexampletitle {
6162      (\sexampletitle)
6163    }~}
6164  }
6165  \cs_new_protected:Nn \__stex_statements_sexample_end: {\stex_par:\medskip}
6166
6167  \newcommand\stexpatchexample[3][] {
```

233

```
6168        \str_set:Nx \l_tmpa_str{ #1 }
6169        \str_if_empty:NTF \l_tmpa_str {
6170           \tl_set:Nn \__stex_statements_sexample_start: { #2 }
6171           \tl_set:Nn \__stex_statements_sexample_end: { #3 }
6172        }{
6173           \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
6174           \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
6175        }
6176     }
```

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

\inlineex   inline:

```
6177  \keys_define:nn {stex / inlineex }{
6178    type     .str_set_x:N  = \sexampletype,
6179    id       .str_set_x:N  = \sexampleid,
6180    for      .clist_set:N   = \l__stex_statements_sexample_for_clist ,
6181    name     .str_set_x:N  = \sexamplename
6182  }
6183  \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
6184    \str_clear:N \sexampletype
6185    \str_clear:N \sexampleid
6186    \str_clear:N \sexamplename
6187    \clist_clear:N \l__stex_statements_sexample_for_clist
6188    \keys_set:nn { stex / inlineex }{ #1 }
6189  }
6190  \NewDocumentCommand \inlineex { O{} m } {
6191    \begingroup
6192    \stex_reactivate_macro:N \premise
6193    \stex_reactivate_macro:N \conclusion
6194    \__stex_statements_inlineex_args:n{ #1 }
6195    \str_if_empty:NF \sexampleid {
6196      \stex_ref_new_doc_target:n \sexampleid
6197    }
6198    \stex_if_smsmode:TF{
6199      \str_if_empty:NF \sexamplename {
6200        \stex_suppress_html:n{\stex_symdecl_do:nn{}{\examplename}}
6201      }
6202    }{
6203      \seq_clear:N \l_tmpb_seq
6204      \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
6205        \tl_if_empty:nF{ ##1 }{
6206          \stex_get_symbol:n { ##1 }
6207          \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6208            \l_stex_get_symbol_uri_str
6209          }
6210        }
6211      }
6212      \exp_args:Nnx
6213      \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpb_seq {,}}{
6214        \str_if_empty:NF \sexampletype {
6215          \stex_annotate_invisible:nnn{typestrings}{\sexampletype}{}
6216        }
6217        #2
```

```
6218        \str_if_empty:NF \sexamplename {
6219            \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
6220            \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}
6221        }
6222     }
6223   }
6224   \endgroup
6225   \stex_smsmode_do:
6226 }
```

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

32.4   Logical Paragraphs

sparagraph (*env.*)

```
6227 \keys_define:nn { stex / sparagraph} {
6228   id      .str_set_x:N    = \sparagraphid ,
6229   title   .tl_set:N       = \l_stex_sparagraph_title_tl ,
6230   type    .str_set_x:N    = \sparagraphtype ,
6231   for     .clist_set:N    = \l__stex_statements_sparagraph_for_clist ,
6232   from    .tl_set:N       = \sparagraphfrom ,
6233   to      .tl_set:N       = \sparagraphto ,
6234   start   .tl_set:N       = \l_stex_sparagraph_start_tl ,
6235   name    .str_set:N      = \sparagraphname ,
6236   imports .tl_set:N       = \l__stex_statements_sparagraph_imports_tl
6237 }
6238
6239 \cs_new_protected:Nn \stex_sparagraph_args:n {
6240   \tl_clear:N \l_stex_sparagraph_title_tl
6241   \tl_clear:N \sparagraphfrom
6242   \tl_clear:N \sparagraphto
6243   \tl_clear:N \l_stex_sparagraph_start_tl
6244   \tl_clear:N \l__stex_statements_sparagraph_imports_tl
6245   \str_clear:N \sparagraphid
6246   \str_clear:N \sparagraphtype
6247   \clist_clear:N \l__stex_statements_sparagraph_for_clist
6248   \str_clear:N \sparagraphname
6249   \keys_set:nn { stex / sparagraph }{ #1 }
6250 }
6251 \newif\if@in@omtext\@in@omtextfalse
6252
6253 \NewDocumentEnvironment {sparagraph} { O{} } {
6254   \stex_sparagraph_args:n { #1 }
6255   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
6256     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
6257   }{
6258     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
6259   }
6260   \@in@omtexttrue
6261   \stex_if_smsmode:F {
6262     \seq_clear:N \l_tmpb_seq
6263     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
6264       \tl_if_empty:nF{ ##1 }{
```

```
6265        \stex_get_symbol:n { ##1 }
6266        \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6267          \l_stex_get_symbol_uri_str
6268        }
6269      }
6270    }
6271    \exp_args:Nnnx
6272    \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}
6273    \str_if_empty:NF \sparagraphtype {
6274      \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
6275    }
6276    \str_if_empty:NF \sparagraphfrom {
6277      \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
6278    }
6279    \str_if_empty:NF \sparagraphto {
6280      \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
6281    }
6282    \str_if_empty:NF \sparagraphname {
6283      \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
6284    }
6285    \clist_set:No \l_tmpa_clist \sparagraphtype
6286    \tl_clear:N \l_tmpa_tl
6287    \clist_map_inline:Nn \sparagraphtype {
6288      \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
6289        \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
6290      }
6291    }
6292    \stex_csl_to_imports:No \usemodule \l__stex_statements_sparagraph_imports_tl
6293    \tl_if_empty:NTF \l_tmpa_tl {
6294      \__stex_statements_sparagraph_start:
6295    }{
6296      \l_tmpa_tl
6297    }
6298  }
6299  \clist_set:No \l_tmpa_clist \sparagraphtype
6300  \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}
6301  {
6302    \stex_reactivate_macro:N \definiendum
6303    \stex_reactivate_macro:N \definame
6304    \stex_reactivate_macro:N \Definame
6305    \stex_reactivate_macro:N \premise
6306    \stex_reactivate_macro:N \definiens
6307  }
6308  \str_if_empty:NTF \sparagraphid {
6309    \str_if_empty:NTF \sparagraphname {
6310      \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
6311        \stex_ref_new_doc_target:n {}
6312      }
6313    } {
6314      \stex_ref_new_doc_target:n {}
6315    }
6316  } {
6317    \stex_ref_new_doc_target:n \sparagraphid
6318  }
```

236

```
6319      \exp_args:NNx
6320      \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
6321        \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
6322          \tl_if_empty:nF{ ##1 }{
6323            \stex_get_symbol:n { ##1 }
6324            \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
6325          }
6326        }
6327      }
6328      \stex_smsmode_do:
6329      \ignorespacesandpars
6330    }{
6331      \str_if_empty:NF \sparagraphname {
6332        \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
6333        \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
6334      }
6335      \stex_if_smsmode:F {
6336        \clist_set:No \l_tmpa_clist \sparagraphtype
6337        \tl_clear:N \l_tmpa_tl
6338        \clist_map_inline:Nn \l_tmpa_clist {
6339          \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
6340            \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
6341          }
6342        }
6343        \tl_if_empty:NTF \l_tmpa_tl {
6344          \__stex_statements_sparagraph_end:
6345        }{
6346          \l_tmpa_tl
6347        }
6348        \end{stex_annotate_env}
6349      }
6350    }
```

```
6351
6352   \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
6353     \stex_par:\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
6354       \tl_if_empty:NF \l_stex_sparagraph_title_tl {
6355         \titleemph{\l_stex_sparagraph_title_tl}:~
6356       }
6357     }{
6358       \titleemph{\l_stex_sparagraph_start_tl}~
6359     }
6360   }
6361   \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\stex_par:\medskip}
6362
6363   \newcommand\stexpatchparagraph[3][] {
6364       \str_set:Nx \l_tmpa_str{ #1 }
6365       \str_if_empty:NTF \l_tmpa_str {
6366         \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
6367         \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
6368       }{
6369         \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
6370         \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
```

237

```
6371        }
6372 }
6373
6374 \keys_define:nn { stex / inlinepara} {
6375   id      .str_set_x:N   = \sparagraphid ,
6376   type    .str_set_x:N   = \sparagraphtype ,
6377   for     .clist_set:N   = \l__stex_statements_sparagraph_for_clist ,
6378   from    .tl_set:N      = \sparagraphfrom ,
6379   to      .tl_set:N      = \sparagraphto ,
6380   name    .str_set:N     = \sparagraphname
6381 }
6382 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
6383   \tl_clear:N \sparagraphfrom
6384   \tl_clear:N \sparagraphto
6385   \str_clear:N \sparagraphid
6386   \str_clear:N \sparagraphtype
6387   \clist_clear:N \l__stex_statements_sparagraph_for_clist
6388   \str_clear:N \sparagraphname
6389   \keys_set:nn { stex / inlinepara }{ #1 }
6390 }
6391 \NewDocumentCommand \inlinepara { O{} m } {
6392   \begingroup
6393   \__stex_statements_inlinepara_args:n{ #1 }
6394   \clist_set:No \l_tmpa_clist \sparagraphtype
6395   \str_if_empty:NTF \sparagraphid {
6396     \str_if_empty:NTF \sparagraphname {
6397       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
6398         \stex_ref_new_doc_target:n {}
6399       }
6400     } {
6401       \stex_ref_new_doc_target:n {}
6402     }
6403   } {
6404     \stex_ref_new_doc_target:n \sparagraphid
6405   }
6406   \stex_if_smsmode:TF{
6407     \str_if_empty:NF \sparagraphname {
6408       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
6409       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
6410     }
6411   }{
6412     \seq_clear:N \l_tmpb_seq
6413     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
6414       \tl_if_empty:nF{ ##1 }{
6415         \stex_get_symbol:n { ##1 }
6416         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6417           \l_stex_get_symbol_uri_str
6418         }
6419       }
6420     }
6421     \exp_args:Nnx
6422     \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}{
6423       \str_if_empty:NF \sparagraphtype {
6424         \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
```

238

```
6425        }
6426        \str_if_empty:NF \sparagraphfrom {
6427          \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
6428        }
6429        \str_if_empty:NF \sparagraphto {
6430          \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
6431        }
6432        \str_if_empty:NF \sparagraphname {
6433          \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
6434          \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
6435          \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
6436        }
6437        \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
6438          \clist_map_inline:Nn \l_tmpb_seq {
6439            \stex_ref_new_sym_target:n {##1}
6440          }
6441        }
6442        #2
6443      }
6444    }
6445    \endgroup
6446    \stex_smsmode_do:
6447 }
6448
```

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

6449 ⟨/package⟩

Chapter 33

The Implementation

```
6450  ⟨*package⟩
6451  ⟨@@=stex_sproof⟩
6452
6453  %%%%%%%%%%%%   sproof.dtx   %%%%%%%%%%%%
6454
```

33.1 Proofs

We first define some keys for the proof environment.

```
6455  \keys_define:nn { stex / spf } {
6456    id          .str_set_x:N  = \spfid,
6457    for         .clist_set:N  = \l__stex_sproof_spf_for_clist ,
6458    from        .tl_set:N     = \l__stex_sproof_spf_from_tl ,
6459    proofend    .tl_set:N     = \l__stex_sproof_spf_proofend_tl,
6460    type        .str_set_x:N  = \spftype,
6461    title       .tl_set:N     = \spftitle,
6462    continues   .tl_set:N     = \l__stex_sproof_spf_continues_tl,
6463    functions   .tl_set:N     = \l__stex_sproof_spf_functions_tl,
6464    term        .tl_set:N     = \l__stex_sproof_spf_term_tl,
6465    method      .tl_set:N     = \l__stex_sproof_spf_method_tl,
6466    hide        .bool_set:N   = \l__stex_sproof_spf_hide_bool
6467  }
6468  \cs_new_protected:Nn \__stex_sproof_spf_args:n {
6469  \str_clear:N \spfid
6470  \tl_clear:N \l__stex_sproof_spf_for_tl
6471  \tl_clear:N \l__stex_sproof_spf_from_tl
6472  \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
6473  \str_clear:N \spftype
6474  \tl_clear:N \spftitle
6475  \tl_clear:N \l__stex_sproof_spf_continues_tl
6476  \tl_clear:N \l__stex_sproof_spf_term_tl
6477  \tl_clear:N \l__stex_sproof_spf_functions_tl
6478  \tl_clear:N \l__stex_sproof_spf_method_tl
6479    \bool_set_false:N \l__stex_sproof_spf_hide_bool
6480  \keys_set:nn { stex / spf }{ #1 }
6481  }
6482  \bool_set_true:N \l__stex_sproof_inc_counter_bool
```

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

```
6483 \str_set:Nn\c__stex_sproof_flow_str{inline}
```

(*End definition for* \c__stex_sproof_flow_str.)

For proofs, we will have to have deeply nested structures of enumerated list-like environments. However, LaTeX only allows `enumerate` environments up to nesting depth 4 and general list environments up to listing depth 6. This is not enough for us. Therefore we have decided to go along the route proposed by Leslie Lamport to use a single top-level list with dotted sequences of numbers to identify the position in the proof tree. Unfortunately, we could not use his `pf.sty` package directly, since it does not do automatic numbering, and we have to add keyword arguments all over the place, to accomodate semantic information.

```
6484 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
6485 \cs_new_protected:Npn \sproofnumber {
6486   \int_set:Nn \l_tmpa_int {1}
6487   \bool_while_do:nn {
6488     \int_compare_p:nNn {
6489       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6490     } > 0
6491   }{
6492     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
6493     \int_incr:N \l_tmpa_int
6494   }
6495 }
6496 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
6497   \int_set:Nn \l_tmpa_int {1}
6498   \bool_while_do:nn {
6499     \int_compare_p:nNn {
6500       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6501     } > 0
6502   }{
6503     \int_incr:N \l_tmpa_int
6504   }
6505   \int_compare:nNnF \l_tmpa_int = 1 {
6506     \int_decr:N \l_tmpa_int
6507   }
6508   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
6509     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1
6510   }
6511 }
6512
6513 \cs_new_protected:Npn \__stex_sproof_add_counter: {
6514   \int_set:Nn \l_tmpa_int {1}
6515   \bool_while_do:nn {
6516     \int_compare_p:nNn {
6517       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6518     } > 0
6519   }{
6520     \int_incr:N \l_tmpa_int
6521   }
6522   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 1 }
6523 }
6524
```

```
6525 \cs_new_protected:Npn \__stex_sproof_remove_counter: {
6526   \int_set:Nn \l_tmpa_int {1}
6527   \bool_while_do:nn {
6528     \int_compare_p:nNn {
6529       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6530     } > 0
6531   }{
6532     \int_incr:N \l_tmpa_int
6533   }
6534   \int_decr:N \l_tmpa_int
6535   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 0 }
6536 }
```

\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

```
6537 \def\sproof@box{
6538   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
6539 }
6540 \def\sproofend{
6541   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
6542     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
6543   }
6544 }
```

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

spf@*@kw

```
6545 \def\spf@proofsketch@kw{Proof~Sketch}
6546 \def\spf@proof@kw{Proof}
6547 \def\spf@step@kw{Step}
```

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

For the other languages, we set up triggers

```
6548 \AddToHook{begindocument}{
6549   \ltx@ifpackageloaded{babel}{
6550     \makeatletter
6551     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6552     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6553       \input{sproof-ngerman.ldf}
6554     }
6555     \clist_if_in:NnT \l_tmpa_clist {finnish}{
6556       \input{sproof-finnish.ldf}
6557     }
6558     \clist_if_in:NnT \l_tmpa_clist {french}{
6559       \input{sproof-french.ldf}
6560     }
6561     \clist_if_in:NnT \l_tmpa_clist {russian}{
6562       \input{sproof-russian.ldf}
6563     }
6564     \makeatother
6565   }{}
6566 }
```

```
6567 \newcommand\spfsketch[2][]{
6568   \begingroup
6569   \let \premise \stex_proof_premise:
6570   \__stex_sproof_spf_args:n{#1}
6571   \stex_if_smsmode:TF {
6572     \str_if_empty:NF \spfid {
6573       \stex_ref_new_doc_target:n \spfid
6574     }
6575   }{
6576     \seq_clear:N \l_tmpa_seq
6577     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6578       \tl_if_empty:nF{ ##1 }{
6579         \stex_get_symbol:n { ##1 }
6580         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6581           \l_stex_get_symbol_uri_str
6582         }
6583       }
6584     }
6585     \exp_args:Nnx
6586     \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
6587       \str_if_empty:NF \spftype {
6588         \stex_annotate_invisible:nnn{type}{\spftype}{}
6589       }
6590       \clist_set:No \l_tmpa_clist \spftype
6591       \tl_set:Nn \l_tmpa_tl {
6592         \titleemph{
6593           \tl_if_empty:NTF \spftitle {
6594             \spf@proofsketch@kw
6595           }{
6596             \spftitle
6597           }
6598         }:~
6599       }
6600       \clist_map_inline:Nn \l_tmpa_clist {
6601         \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6602           \tl_clear:N \l_tmpa_tl
6603         }
6604       }
6605       \str_if_empty:NF \spfid {
6606         \stex_ref_new_doc_target:n \spfid
6607       }
6608       \l_tmpa_tl #2 \sproofend
6609     }
6610   }
6611   \endgroup
6612   \stex_smsmode_do:
6613 }
6614
```

(*End definition for* spfsketch*. This function is documented on page 50.*)

```
6615 \bool_set_false:N \l__stex_sproof_in_spfblock_bool
```

```
6616
6617 \cs_new_protected:Nn \__stex_sproof_maybe_comment: {
6618   \bool_if:NF \l__stex_sproof_in_spfblock_bool {
6619     \par \setbox \l_tmpa_box \vbox \bgroup \everypar{\__stex_sproof_start_comment:}
6620   }
6621 }
6622 \cs_new_protected:Nn \__stex_sproof_maybe_comment_end: {
6623   \bool_if:NF \l__stex_sproof_in_spfblock_bool { \egroup }
6624 }
6625 \cs_new_protected:Nn \__stex_sproof_start_comment: {
6626   \csname @ @ par\endcsname\egroup\item[]\bgroup\stexcommentfont
6627 }
6628
```

(*End definition for* __stex_sproof_maybe_comment: , __stex_sproof_maybe_comment_end: , *and* __- *stex_sproof_start_comment:.*)

\stexcommentfont

```
6629 \cs_new_protected:Npn \stexcommentfont {
6630   \small\itshape
6631 }
```

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

sproof (*env.*) 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.

```
6632 \cs_new_protected:Nn \__stex_sproof_start_env:nnn {
6633   \seq_clear:N \l_tmpa_seq
6634   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6635     \tl_if_empty:nF{ ##1 }{
6636       \stex_get_symbol:n { ##1 }
6637       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6638         \l_stex_get_symbol_uri_str
6639       }
6640     }
6641   }
6642   \exp_args:Nnnx
6643   \begin{stex_annotate_env}{#1}{\seq_use:Nn \l_tmpa_seq {,}}
6644   \str_if_empty:NF \spftype {
6645     \stex_annotate_invisible:nnn{type}{\spftype}{}
6646   }
6647   #3 {~\stex_annotate:nnn{spftitle}{}{#2}}
6648   \str_if_empty:NF \spfid {
6649     \stex_ref_new_doc_target:n \spfid
6650   }
6651   \begin{stex_annotate_env}{spfbody}{\bool_if:NTF \l__stex_sproof_spf_hide_bool {false}{true
6652   \bool_if:NT \l__stex_sproof_spf_hide_bool{
6653     \stex_html_backend:F{\setbox\l_tmpa_box\vbox\bgroup}
6654   }
6655   \begin{list}{}{
6656     \setlength\topsep{0pt}
6657     \setlength\parsep{0pt}
6658     \setlength\rightmargin{0pt}
```

```
6659
6660    }\__stex_sproof_maybe_comment:
6661 }
6662 \cs_new_protected:Nn \__stex_sproof_end_env:n {
6663    \stex_if_smsmode:F{
6664       \__stex_sproof_maybe_comment_end:
6665       \end{list}
6666       \bool_if:NT \l__stex_sproof_spf_hide_bool{
6667          \stex_html_backend:F{\egroup}
6668       }
6669       \clist_set:No \l_tmpa_clist \spftype
6670       #1
6671       \end{stex_annotate_env}
6672       \end{stex_annotate_env}
6673    }
6674 }
6675 \NewDocumentEnvironment{sproof}{s O{} m}{
6676    \intarray_gzero:N \l__stex_sproof_counter_intarray
6677    \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
6678    \stex_reactivate_macro:N \yield
6679    \stex_reactivate_macro:N \eqstep
6680    \stex_reactivate_macro:N \assumption
6681    \stex_reactivate_macro:N \conclude
6682    \stex_reactivate_macro:N \spfstep
6683    \__stex_sproof_spf_args:n{#2}
6684    \stex_if_smsmode:TF {
6685       \str_if_empty:NF \spfid {
6686          \stex_ref_new_doc_target:n \spfid
6687       }
6688    }{
6689       \__stex_sproof_start_env:nnn{sproof}{#3}{
6690          \clist_set:No \l_tmpa_clist \spftype
6691          \tl_clear:N \l_tmpa_tl
6692          \clist_map_inline:Nn \l_tmpa_clist {
6693             \tl_if_exist:cT {__stex_sproof_sproof_##1_start:}{
6694                \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_start:}}
6695             }
6696             \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6697                \tl_set:Nn \l_tmpa_tl {\use:n{}}
6698             }
6699          }
6700          \tl_if_empty:NTF \l_tmpa_tl {
6701             \__stex_sproof_sproof_start:
6702          }{
6703             \l_tmpa_tl
6704          }
6705       }
6706    }
6707    \stex_smsmode_do:
6708 }{\__stex_sproof_end_env:n{
6709    \tl_clear:N \l_tmpa_tl
6710    \clist_map_inline:Nn \l_tmpa_clist {
6711       \tl_if_exist:cT {__stex_sproof_sproof_##1_end:}{
6712          \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_end:}}
```

```
6713              }
6714          }
6715          \tl_if_empty:NTF \l_tmpa_tl {
6716            \__stex_sproof_sproof_end:
6717          }{
6718            \l_tmpa_tl
6719          }
6720 }}
6721 \NewDocumentEnvironment{subproof}{s O{} m}{
6722    \__stex_sproof_spf_args:n{#2}
6723    \stex_if_smsmode:TF {
6724      \str_if_empty:NF \spfid {
6725        \stex_ref_new_doc_target:n \spfid
6726      }
6727    }{
6728      \__stex_sproof_start_env:nnn{subproof}{\item[\sproofnumber]\ignorespacesandpars #3}{}
6729    }
6730    \__stex_sproof_add_counter:
6731    \stex_smsmode_do:
6732 }{\__stex_sproof_remove_counter:\__stex_sproof_end_env:n{}
6733    \bool_if:NT \l__stex_sproof_inc_counter_bool {
6734      \__stex_sproof_inc_counter:
6735    }
6736    \aftergroup\__stex_sproof_maybe_comment:
6737 }
6738 \AddToHook{env/subproof/before}{\__stex_sproof_maybe_comment_end:}
6739
6740 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
6741    \par\noindent\titleemph{
6742      \tl_if_empty:NTF \spftype {
6743        \spf@proof@kw
6744      }{
6745        \spftype
6746      }
6747    }:
6748 }
6749 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
6750
6751 \newcommand\stexpatchproof[3][] {
6752    \str_set:Nx \l_tmpa_str{ #1 }
6753    \str_if_empty:NTF \l_tmpa_str {
6754      \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
6755      \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
6756    }{
6757      \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
6758      \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
6759    }
6760 }
```

\pstep
\conclude
\assumption
\have
\eqstep

```
6761
6762 \keys_define:nn { stex / spfsteps } {
6763    id          .str_set_x:N  = \spfstepid,
6764    for          .clist_set:N  = \l__stex_sproof_spf_for_clist ,
```

246

```
6765   type        .str_set_x:N  = \spftype,
6766   title       .tl_set:N     = \spftitle,
6767   method      .tl_set:N     = \l__stex_sproof_spf_method_tl,
6768   term        .tl_set:N     = \l__stex_sproof_spf_term_tl
6769 }
6770 \cs_new_protected:Nn \__stex_sproof_spfstep_args:n {
6771 \str_clear:N \spfstepid
6772 \clist_clear:N \l__stex_sproof_spf_for_clist
6773 \str_clear:N \spftype
6774 \tl_clear:N \l__stex_sproof_spf_method_tl
6775 \tl_clear:N \l__stex_sproof_spf_term_tl
6776   %\bool_set_false:N \l__stex_sproof_inc_counter_bool
6777 \keys_set:nn { stex / spfsteps }{ #1 }
6778 }
6779
6780 \cs_new_protected:Nn \__stex_sproof_make_step_macro:Nnnnn {
6781   \NewDocumentCommand #1 {s O{} +m} {
6782     \__stex_sproof_maybe_comment_end:
6783
6784     \__stex_sproof_spfstep_args:n{##2}
6785     \stex_annotate:nnn{spfstep}{#2}{
6786       \tl_if_empty:NF \l__stex_sproof_spf_term_tl {
6787         \stex_annotate_invisible:nnn{spfyield}{}{$\l__stex_sproof_spf_term_tl$}
6788       }
6789       \bool_if:NTF \l__stex_sproof_in_spfblock_bool {
6790         #4
6791       }{
6792         \item[\IfBooleanTF ##1 {}{#3}]
6793       }
6794       \ignorespacesandpars ##3
6795     }
6796     \bool_if:NF \l__stex_sproof_in_spfblock_bool { \IfBooleanTF ##1 {}{ #5 } }
6797     \__stex_sproof_maybe_comment:
6798   }
6799   \stex_deactivate_macro:Nn #1 {sproof~environments}
6800 }
6801
6802 \__stex_sproof_make_step_macro:Nnnnn \assumption {assumption} \sproofnumber {} \__stex_sproo
6803 \__stex_sproof_make_step_macro:Nnnnn \conclude {conclusion} {$\Rightarrow$} {} {}
6804 \__stex_sproof_make_step_macro:Nnnnn \spfstep {} \sproofnumber {} \__stex_sproof_inc_counter
6805
6806 \NewDocumentCommand \eqstep {s m}{
6807   \__stex_sproof_maybe_comment_end:
6808   \bool_if:NTF \l__stex_sproof_in_spfblock_bool {
6809     $=$
6810   }{
6811     \item[$=$]
6812   }
6813   $\stex_annotate:nnn{spfstep}{eq}{ #2 }$
6814   \__stex_sproof_maybe_comment:
6815 }
6816 \stex_deactivate_macro:Nn \eqstep {sproof~environments}
6817
6818 \NewDocumentCommand \yield {+m}{
```

247

```
6819        \stex_annotate:nnn{spfyield}{}{ #1 }
6820    }
6821    \stex_deactivate_macro:Nn \yield {sproof~environments}
6822
6823    \NewDocumentEnvironment{spfblock}{}{
6824        \item[]
6825        \bool_set_true:N \l__stex_sproof_in_spfblock_bool
6826    }{
6827        \aftergroup\__stex_sproof_maybe_comment:
6828    }
6829    \AddToHook{env/spfblock/before}{\__stex_sproof_maybe_comment_end:}
6830
```

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

\spfidea

```
6831    \NewDocumentCommand\spfidea{O{} +m}{
6832        \__stex_sproof_spf_args:n{#1}
6833        \titleemph{
6834            \tl_if_empty:NTF \spftype {Proof~Idea}{
6835                \spftype
6836            }:
6837        }~#2
6838        \sproofend
6839    }
```

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

```
6840    \newcommand\spfjust[1]{
6841        #1
6842    }
6843    ⟨/package⟩
```

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

Chapter 34

STEX -Others Implementation

```
6844 ⟨*package⟩
6845
6846 %%%%%%%%%%%%   others.dtx   %%%%%%%%%%%%
6847
6848 ⟨@@=stex_others⟩
```

Warnings and error messages
```
6849   % None
```

Math subject classifier

```
6850 \NewDocumentCommand \MSC {m} {
6851   % TODO
6852 }
```

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

Patching tikzinput, if loaded

```
6853 \@ifpackageloaded{tikzinput}{
6854   \RequirePackage{stex-tikzinput}
6855 }{}
6856
6857 \bool_if:NT \c_stex_persist_mode_bool {
6858   \let\__stex_notation_restore_notation_old:nnnnn
6859     \__stex_notation_restore_notation:nnnnn
6860   \def\__stex_notation_restore_notation_new:nnnnn#1#2#3#4#5{
6861     \__stex_notation_restore_notation_old:nnnnn{#1}{#2}{#3}{#4}{#5}
6862     \ExplSyntaxOn
6863   }
6864   \def\__stex_notation_restore_notation:nnnnn{
6865     \ExplSyntaxOff
6866     \catcode'~10
6867     \__stex_notation_restore_notation_new:nnnnn
6868   }
6869   \input{\jobname.sms}
6870   \let\__stex_notation_restore_notation:nnnnn
6871     \__stex_notation_restore_notation_old:nnnnn
6872   \prop_if_exist:NT\c_stex_mathhub_main_manifest_prop{
```

```
6873      \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
6874        \l_tmpa_str
6875      \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str _manifest_prop }
6876        \c_stex_mathhub_main_manifest_prop
6877      \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
6878    }
6879  }
6880
6881  \stex_get_document_uri:
6882  ⟨/package⟩
```

Chapter 35

sTeX
-Metatheory Implementation

```
6883  ⟨∗package⟩
6884  ⟨@@=stex_modules⟩
6885
6886  %%%%%%%%%%%%  metatheory.dtx  %%%%%%%%%%%%
6887
6888  \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX/meta}
6889  \begingroup
6890  \stex_module_setup:nn{
6891    ns=\c_stex_metatheory_ns_str,
6892    meta=NONE
6893  }{Metatheory}
6894  \stex_reactivate_macro:N \symdecl
6895  \stex_reactivate_macro:N \notation
6896  \stex_reactivate_macro:N \symdef
6897  \ExplSyntaxOff
6898  \csname stex_suppress_html:n\endcsname{
6899    % is-a (a:A, a \in A, a is an A, etc.)
6900    \symdecl{isa}[args=ai]
6901    \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
6902    \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
6903    \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
6904
6905    % bind (\forall, \Pi, \lambda etc.)
6906    \symdecl{bind}[args=Bi,assoc=pre]
6907    \notation{bind}[depfun,prec=nobrackets,op={(\cdot)\;\to\;\cdot}]{\comp( #1 \comp{)\;\to\;}
6908    \notation{bind}[forall]{\comp\forall #1.\;#2}{##1 \comp, ##2}
6909    \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6910
6911    % implicit bind
6912    \symdecl{implicitbind}[args=Bi,assoc=pre]
6913    \notation{implicitbind}[braces,prec=nobrackets,op={\{\cdot\}_I\;\cdot}]{\comp\{ #1 \comp\
6914    \notation{implicitbind}[depfun,prec=nobrackets]{\comp( #1 \comp)\;\to_I\;} #2}{##1 \comp,
6915    \notation{implicitbind}[Pi]{\comp\prod^I_{#1}#2}{##1\comp,##2}
6916
6917    % dummy variable
```

```
6918    \symdecl{dummyvar}
6919    \notation{dummyvar}[underscore]{\comp\_}
6920    \notation{dummyvar}[dot]{\comp\cdot}
6921    \notation{dummyvar}[dash]{\comp{{\rm --}}}
6922
6923    %fromto (function space, Hom-set, implication etc.)
6924    \symdecl{fromto}[args=ai]
6925    \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
6926    \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
6927
6928    % mapto (lambda etc.)
6929    %\symdecl{mapto}[args=Bi]
6930    %\notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
6931    %\notation{mapto}[lambda]{\comp\lambda #1 \comp.\; #2}{#1 \comp, #2}
6932    %\notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.\; #2}{#1 \comp, #2}
6933
6934    % function/operator application
6935    \symdecl{apply}[args=ia]
6936    \notation{apply}[prec=0;0x\infprec,parens,op=\cdot(\cdot)]{#1 \comp( #2 \comp)}{##1 \comp,
6937    \notation{apply}[prec=0;0x\infprec,lambda]{#1 \; #2 }{##1 \; ##2}
6938
6939    % collection of propositions/booleans/truth values
6940    \symdecl{prop}[name=proposition]
6941    \notation{prop}[prop]{\comp{{\rm prop}}}
6942    \notation{prop}[BOOL]{\comp{{\rm BOOL}}}
6943
6944    \symdecl{judgmentholds}[args=1]
6945    \notation{judgmentholds}[vdash,op=\vdash]{\comp\vdash\; #1}
6946
6947    % sequences
6948    \symdecl{seqtype}[args=1]
6949    \notation{seqtype}[kleene]{#1^{\comp\ast}}
6950
6951    \symdecl{seqexpr}[args=a]
6952    \notation{seqexpr}[angle,prec=nobrackets]{\comp\langle #1\comp\rangle}{##1\comp,##2}
6953
6954    \symdef{seqmap}[args=abi,setlike]{\comp\{#3 \comp| #2\comp\in \dobrackets{#1} \comp\}}{##1
6955    \symdef{seqprepend}[args=ia]{#1 \comp{::} #2}{##1 \comp, ##2}
6956    \symdef{seqappend}[args=ai]{#1 \comp{::} #2}{##1 \comp, ##2}
6957    \symdef{seqfoldleft}[args=iabbi]{ \comp{foldl}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\com
6958    \symdef{seqfoldright}[args=iabbi,op=foldr]{ \comp{foldr}\dobrackets{#1,#2}\dobrackets{#3\c
6959    \symdef{seqhead}[args=a]{\comp{head}\dobrackets{#1}}{##1 \comp, ##2}
6960    \symdef{seqtail}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6961    \symdef{seqlast}[args=a]{\comp{last}\dobrackets{#1}}{##1 \comp, ##2}
6962    \symdef{seqinit}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6963
6964    \symdef{sequence-index}[args=2,li,prec=nobrackets]{{#1}_{#2}}
6965    \notation{sequence-index}[ui,prec=nobrackets]{{#1}^{#2}}
6966
6967    \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{,\ellipses}}{##1\comp,##2}
6968    \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{,\ellipses,}#2}{##1\comp,##2}
6969    \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{,\ellipses,}#2\comp{,\ellipses,}#
6970
6971    % nat literals
```

```
6972    \symdef{natliteral}{\comp{\mathtt{Ord}}}}

6973

6974    % letin (''let'', local definitions, variable substitution)
6975    \symdecl{letin}[args=bii]
6976    \notation{letin}[let]{\comp{{\rm let}}\;#1\comp{=}#2\;\comp{{\rm in}}\;#3}
6977    \notation{letin}[subst]{#3 \comp[ #1 \comp/ #2 \comp]}
6978    \notation{letin}[frac]{#3 \comp[ \frac{#2}{#1} \comp]}

6979

6980    % structures
6981    \symdecl*{module-type}[args=1]
6982    \notation{module-type}{\comp{\mathtt{MOD}} #1}
6983    \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
6984    \notation{mathstruct}[angle,prec=nobrackets]{\comp\langle #1 \comp\rangle}{##1 \comp, ##2}

6985

6986    % objects
6987    \symdecl{object}
6988    \notation{object}{\comp{\mathtt{OBJECT}}}}

6989

6990    }

6991

6992    % The following are abbreviations in the sTeX corpus that are left over from earlier
6993    % developments. They will eventually be phased out.

6994

6995    \ExplSyntaxOn
6996    \stex_add_to_current_module:n{
6997        \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
6998        \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
6999        \def\livar{\csname sequence-index\endcsname[li]}
7000        \def\uivar{\csname sequence-index\endcsname[ui]}
7001        \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
7002        \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
7003    }
7004 \__stex_modules_end_module:
7005 \endgroup

7006 ⟨/package⟩
```

Chapter 36

Tikzinput Implementation

```
7007 ⟨@@=tikzinput⟩
7008 ⟨*package⟩
7009
7010 %%%%%%%%%%%%    tikzinput.dtx    %%%%%%%%%%%%
7011
7012 \ProvidesExplPackage{tikzinput}{2022/08/08}{3.2.0}{tikzinput package}
7013 \RequirePackage{l3keys2e}
7014
7015 \keys_define:nn { tikzinput } {
7016   image   .bool_set:N   = \c_tikzinput_image_bool,
7017   image   .default:n    = false ,
7018   unknown   .code:n        = {}
7019 }
7020
7021 \ProcessKeysOptions { tikzinput }
7022
7023 \bool_if:NTF \c_tikzinput_image_bool {
7024   \RequirePackage{graphicx}
7025
7026   \providecommand\usetikzlibrary[]{}
7027   \newcommand\tikzinput[2][]{\includegraphics[#1]{#2}}
7028 }{
7029   \RequirePackage{tikz}
7030   \RequirePackage{standalone}
7031
7032   \newcommand \tikzinput [2] [] {
7033     \setkeys{Gin}{#1}
7034     \ifx \Gin@ewidth \Gin@exclamation
7035       \ifx \Gin@eheight \Gin@exclamation
7036         \input { #2 }
7037       \else
7038         \resizebox{!}{ \Gin@eheight }{
7039           \input { #2 }
7040         }
7041       \fi
7042     \else
7043       \ifx \Gin@eheight \Gin@exclamation
7044         \resizebox{ \Gin@ewidth }{!}{
```

```
7045        \input { #2 }
7046      }
7047    \else
7048      \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
7049        \input { #2 }
7050      }
7051    \fi
7052  \fi
7053  }
7054 }

7055
7056 \newcommand \ctikzinput [2] [] {
7057  \begin{center}
7058    \tikzinput [#1] {#2}
7059  \end{center}
7060 }

7061
7062 \@ifpackageloaded{stex}{
7063  \RequirePackage{stex-tikzinput}
7064 }{}

7065
7066 ⟨/package⟩
7067 ⟨∗stex⟩

7068 \ProvidesExplPackage{stex-tikzinput}{2022/08/08}{3.2.0}{stex-tikzinput}
7069 \RequirePackage{stex}
7070 \RequirePackage{tikzinput}

7071
7072 \newcommand\mhtikzinput[2][]{%
7073  \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
7074  \stex_in_repository:nn\Gin@mhrepos{
7075    \tikzinput[#1]{\mhpath{##1}{#2}}
7076  }
7077 }
7078 \newcommand\cmhtikzinput[2][]{\begin{center}\mhtikzinput[#1]{#2}\end{center}}

7079
7080 \cs_new_protected:Nn \__tikzinput_usetikzlibrary:nn {
7081  \pgfkeys@spdef\pgf@temp{#1}
7082  \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
7083  \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgfut
7084  \expandafter\edef\csname tikz@library@#1@atcode\endcsname{\the\catcode`\@}
7085  \expandafter\edef\csname tikz@library@#1@barcode\endcsname{\the\catcode`\|}
7086  \expandafter\edef\csname tikz@library@#1@dollarcode\endcsname{\the\catcode`\$}
7087  \catcode`\@=11
7088  \catcode`\|=12
7089  \catcode`\$=3
7090  \pgfutil@InputIfFileExists{#2}{}{}
7091  \catcode`\@=\csname tikz@library@#1@atcode\endcsname
7092  \catcode`\|=\csname tikz@library@#1@barcode\endcsname
7093  \catcode`\$=\csname tikz@library@#1@dollarcode\endcsname
7094 }

7095

7096
7097 \newcommand\libusetikzlibrary[1]{
```

```
7098    \prop_if_exist:NF \l_stex_current_repository_prop {
7099      \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
7100    }
7101    \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
7102      \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
7103    }
7104    \seq_clear:N \l__tikzinput_libinput_files_seq
7105    \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
7106    \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
7107
7108    \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
7109      \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / tikzlibra
7110      \IfFileExists{ \l_tmpa_str }{
7111        \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
7112      }{}
7113      \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
7114      \seq_put_right:No \l_tmpa_seq \l_tmpa_str
7115    }
7116
7117    \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / tikzlibrary #1 .code.t
7118    \IfFileExists{ \l_tmpa_str }{
7119      \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
7120    }{}
7121
7122    \seq_if_empty:NTF \l__tikzinput_libinput_files_seq {
7123      \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .code.t
7124    }{
7125      \int_compare:nNnTF {\seq_count:N \l__tikzinput_libinput_files_seq} = 1 {
7126        \seq_map_inline:Nn \l__tikzinput_libinput_files_seq {
7127          \__tikzinput_usetikzlibrary:nn{#1}{ ##1 }
7128        }
7129      }{
7130        \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .co
7131      }
7132    }
7133 }
7134 ⟨/stex⟩
```

256

Chapter 37

document-structure.sty Implementation

```
7135 ⟨∗package⟩
7136 ⟨@@=document_structure⟩
7137 \ProvidesExplPackage{document-structure}{2022/08/08}{3.2.0}{Modular Document Structure}
7138 \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).

```
7139
7140 \keys_define:nn{ document-structure }{
7141   class       .str_set_x:N  = \c_document_structure_class_str,
7142   topsect     .str_set_x:N  = \c_document_structure_topsect_str,
7143   unknown     .code:n       = {
7144     \PassOptionsToClass{\CurrentOption}{stex}
7145     \PassOptionsToClass{\CurrentOption}{tikzinput}
7146   }
7147 %   showignores .bool_set:N  = \c_document_structure_showignores_bool,
7148 }
7149 \ProcessKeysOptions{ document-structure }
7150 \str_if_empty:NT \c_document_structure_class_str {
7151   \str_set:Nn \c_document_structure_class_str {article}
7152 }
7153 \str_if_empty:NT \c_document_structure_topsect_str {
7154   \str_set:Nn \c_document_structure_topsect_str {section}
7155 }
```

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

```
7156 \RequirePackage{xspace}
7157 \RequirePackage{comment}
7158 \RequirePackage{stex}
7159 \AddToHook{begindocument}{
```

```
7160  \ltx@ifpackageloaded{babel}{
7161     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7162     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
7163        \makeatletter\input{document-structure-ngerman.ldf}\makeatother
7164     }
7165  }{}
7166 }
```

\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.

```
7167 \int_new:N \l_document_structure_section_level_int
7168 \str_case:VnF \c_document_structure_topsect_str {
7169   {part}{
7170      \int_set:Nn \l_document_structure_section_level_int {0}
7171   }
7172   {chapter}{
7173      \int_set:Nn \l_document_structure_section_level_int {1}
7174   }
7175 }{
7176   \str_case:VnF \c_document_structure_class_str {
7177     {book}{
7178        \int_set:Nn \l_document_structure_section_level_int {0}
7179     }
7180     {report}{
7181        \int_set:Nn \l_document_structure_section_level_int {0}
7182     }
7183   }{
7184      \int_set:Nn \l_document_structure_section_level_int {2}
7185   }
7186 }
```

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:9   element of class omdoc_currentsectionlevel, wich will be instantiated by CSS later.[9]

```
7187 \def\current@section@level{document}%
7188 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
7189 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

\skipfragment

```
7190 \cs_new_protected:Npn \skipfragment {
```

---

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

```
7191    \ifcase\l_document_structure_section_level_int
7192    \or\stepcounter{part}
7193    \or\stepcounter{chapter}
7194    \or\stepcounter{section}
7195    \or\stepcounter{subsection}
7196    \or\stepcounter{subsubsection}
7197    \or\stepcounter{paragraph}
7198    \or\stepcounter{subparagraph}
7199    \fi
7200 }
```

(*End definition for* \skipfragment. *This function is documented on page* *57.*)

blindfragment (*env.*)

```
7201 \newcommand\at@begin@blindsfragment[1]{}
7202 \newenvironment{blindfragment}
7203 {
7204    \int_incr:N\l_document_structure_section_level_int
7205    \at@begin@blindsfragment\l_document_structure_section_level_int
7206 }{}
```

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

```
7207 \newcommand\sfragment@nonum[2]{
7208    \ifx\hyper@anchor\@undefined\else\phantomsection\fi
7209    \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
7210 }
```

(*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.

```
7211 \newcommand\sfragment@num[2]{
7212    \tl_if_empty:NTF \l__document_structure_sfragment_short_tl {
7213       \@nameuse{#1}{#2}
7214    }{
7215       \cs_if_exist:NTF\rdfmeta@sectioning{
7216          \@nameuse{rdfmeta@#1@old}[\l__document_structure_sfragment_short_tl]{#2}
7217       }{
7218          \@nameuse{#1}[\l__document_structure_sfragment_short_tl]{#2}
7219       }
7220    }
7221 %\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\sfragment@id
7222 }
```

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

sfragment (*env.*)

```
7223 \keys_define:nn { document-structure / sfragment }{
7224    id            .str_set_x:N = \l__document_structure_sfragment_id_str,
7225    date          .str_set_x:N = \l__document_structure_sfragment_date_str,
```

```
7226    creators      .clist_set:N = \l__document_structure_sfragment_creators_clist,
7227    contributors  .clist_set:N = \l__document_structure_sfragment_contributors_clist,
7228    srccite       .tl_set:N    = \l__document_structure_sfragment_srccite_tl,
7229    type          .tl_set:N    = \l__document_structure_sfragment_type_tl,
7230    short         .tl_set:N    = \l__document_structure_sfragment_short_tl,
7231    intro         .tl_set:N    = \l__document_structure_sfragment_intro_tl,
7232    imports       .tl_set:N    = \l__document_structure_sfragment_imports_tl,
7233    loadmodules   .bool_set:N  = \l__document_structure_sfragment_loadmodules_bool
7234 }
7235 \cs_new_protected:Nn \__document_structure_sfragment_args:n {
7236    \str_clear:N \l__document_structure_sfragment_id_str
7237    \str_clear:N \l__document_structure_sfragment_date_str
7238    \clist_clear:N \l__document_structure_sfragment_creators_clist
7239    \clist_clear:N \l__document_structure_sfragment_contributors_clist
7240    \tl_clear:N \l__document_structure_sfragment_srccite_tl
7241    \tl_clear:N \l__document_structure_sfragment_type_tl
7242    \tl_clear:N \l__document_structure_sfragment_short_tl
7243    \tl_clear:N \l__document_structure_sfragment_imports_tl
7244    \tl_clear:N \l__document_structure_sfragment_intro_tl
7245    \bool_set_false:N \l__document_structure_sfragment_loadmodules_bool
7246    \keys_set:nn { document-structure / sfragment } { #1 }
7247 }
```

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.

```
7248 \newif\if@mainmatter\@mainmattertrue
7249 \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.

```
7250 \keys_define:nn { document-structure / sectioning }{
7251    name    .str_set_x:N  = \l__document_structure_sect_name_str   ,
7252    ref     .str_set_x:N  = \l__document_structure_sect_ref_str    ,
7253    clear   .bool_set:N   = \l__document_structure_sect_clear_bool ,
7254    clear   .default:n    = {true}                                 ,
7255    num     .bool_set:N   = \l__document_structure_sect_num_bool   ,
7256    num     .default:n    = {true}
7257 }
7258 \cs_new_protected:Nn \__document_structure_sect_args:n {
7259    \str_clear:N \l__document_structure_sect_name_str
7260    \str_clear:N \l__document_structure_sect_ref_str
7261    \bool_set_false:N \l__document_structure_sect_clear_bool
7262    \bool_set_false:N \l__document_structure_sect_num_bool
7263    \keys_set:nn { document-structure / sectioning } { #1 }
7264 }
7265 \newcommand\omdoc@sectioning[3][]{
7266    \__document_structure_sect_args:n {#1 }
7267    \let\omdoc@sect@name\l__document_structure_sect_name_str
7268    \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
7269    \if@mainmatter% numbering not overridden by frontmatter, etc.
7270       \bool_if:NTF \l__document_structure_sect_num_bool {
7271          \sfragment@num{#2}{#3}
7272       }{
```

```
7273        \sfragment@nonum{#2}{#3}
7274      }
7275      \def\current@section@level{\omdoc@sect@name}
7276    \else
7277      \sfragment@nonum{#2}{#3}
7278    \fi
7279 }% if@mainmatter
```

and another one, if redefines the `\addtocontensline` macro of LaTeX to import the respective macros. It takes as an argument a list of module names.

```
7280 \newcommand\sfragment@redefine@addtocontents[1]{%
7281 %\edef\__document_structureimport{#1}%
7282 %\@for\@I:=\__document_structureimport\do{%
7283 %\edef\@path{\csname module@\@I  @path\endcsname}%
7284 %\@ifundefined{tf@toc}\relax%
7285 %      {\protected@write\tf@toc{}{\string\@requiremodules{\@path}}}}}
7286 %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
7287 %\def\addcontentsline##1##2##3{%
7288 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}
7289 %\else% hyperref.sty not loaded
7290 %\def\addcontentsline##1##2##3{%
7291 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}{
7292 %\fi
7293 }% 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.

```
7294 \newenvironment{sfragment}[2][]% keys, title
7295 {
7296    \__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.

```
7297    \stex_csl_to_imports:No \usemodule \l__document_structure_sfragment_imports_tl
7298
7299    \bool_if:NT \l__document_structure_sfragment_loadmodules_bool {
7300      \sfragment@redefine@addtocontents{
7301        %\@ifundefined{module@id}\used@modules%
7302        %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}
7303      }
7304    }
```

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

```
7305
7306    \stex_document_title:n { #2 }
7307
7308    \int_incr:N\l_document_structure_section_level_int
7309    \ifcase\l_document_structure_section_level_int
7310      \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
7311      \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
7312      \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
7313      \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
```

```
7314     \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
7315     \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#
7316     \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{paragr
7317   \fi
7318   \at@begin@sfragment[#1]\l_document_structure_section_level_int{#2}
7319   \str_if_empty:NF \l__document_structure_sfragment_id_str {
7320     \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
7321   }
7322 }% for customization
7323 {}
```

and finally, we localize the sections

```
7324 \newcommand\omdoc@part@kw{Part}
7325 \newcommand\omdoc@chapter@kw{Chapter}
7326 \newcommand\omdoc@section@kw{Section}
7327 \newcommand\omdoc@subsection@kw{Subsection}
7328 \newcommand\omdoc@subsubsection@kw{Subsubsection}
7329 \newcommand\omdoc@paragraph@kw{paragraph}
7330 \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

```
7331 \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).

```
7332 \cs_if_exist:NTF\frontmatter{
7333   \let\__document_structure_orig_frontmatter\frontmatter
7334   \let\frontmatter\relax
7335 }{
7336   \tl_set:Nn\__document_structure_orig_frontmatter{
7337     \clearpage
7338     \@mainmatterfalse
7339     \pagenumbering{roman}
7340   }
7341 }
7342 \cs_if_exist:NTF\backmatter{
7343   \let\__document_structure_orig_backmatter\backmatter
7344   \let\backmatter\relax
7345 }{
7346   \tl_set:Nn\__document_structure_orig_backmatter{
7347     \clearpage
7348     \@mainmatterfalse
7349     \pagenumbering{roman}
7350   }
```

```
7351 }
```

Using these, we can now define the `frontmatter` and `backmatter` environments

frontmatter (*env.*) we use the `\orig@frontmatter` macro defined above and `\mainmatter` if it exists, otherwise we define it.

```
7352 \newenvironment{frontmatter}{
7353   \__document_structure_orig_frontmatter
7354 }{
7355   \cs_if_exist:NTF\mainmatter{
7356     \mainmatter
7357   }{
7358     \clearpage
7359     \@mainmattertrue
7360     \pagenumbering{arabic}
7361   }
7362 }
```

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

```
7363 \newenvironment{backmatter}{
7364   \__document_structure_orig_backmatter
7365 }{
7366   \cs_if_exist:NTF\mainmatter{
7367     \mainmatter
7368   }{
7369     \clearpage
7370     \@mainmattertrue
7371     \pagenumbering{arabic}
7372   }
7373 }
```

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

```
7374 \@mainmattertrue\pagenumbering{arabic}
```

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

```
7375 \def \c__document_structure_document_str{document}
7376 \newcommand\afterprematurestop{}
7377 \def\prematurestop@endsfragment{
7378   \unless\ifx\@currenvir\c__document_structure_document_str
7379     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafte
7380     \expandafter\prematurestop@endsfragment
7381   \fi
7382 }
7383 \providecommand\prematurestop{
7384   \message{Stopping~sTeX~processing~prematurely}
7385   \prematurestop@endsfragment
7386   \afterprematurestop
7387   \end{document}
7388 }
```

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

37.4 Global Variables

\setSGvar set a global variable

```
7389 \RequirePackage{etoolbox}
7390 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}
```

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

\useSGvar use a global variable

```
7391 \newrobustcmd\useSGvar[1]{%
7392   \@ifundefined{sTeX@Gvar@#1}
7393   {\PackageError{document-structure}
7394     {The sTeX Global variable #1 is undefined}
7395     {set it with \protect\setSGvar}}
7396   \@nameuse{sTeX@Gvar@#1}}
```

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

\ifSGvar execute something conditionally based on the state of the global variable.

```
7397 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
7398   \@ifundefined{sTeX@Gvar@#1}
7399   {\PackageError{document-structure}
7400     {The sTeX Global variable #1 is undefined}
7401     {set it with \protect\setSGvar}}
7402   {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}
```

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

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.

```
7403  ⟨∗cls⟩
7404  ⟨@@=notesslides⟩
7405  \ProvidesExplClass{notesslides}{2022/08/08}{3.2.0}{notesslides Class}
7406  \RequirePackage{l3keys2e}
7407
7408  \keys_define:nn{notesslides / cls}{
7409    class    .str_set_x:N = \c__notesslides_class_str,
7410    notes    .bool_set:N  = \c__notesslides_notes_bool ,
7411    slides   .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
7412    docopt   .str_set_x:N = \c__notesslides_docopt_str,
7413    unknown  .code:n      = {
7414      \PassOptionsToPackage{\CurrentOption}{document-structure}
7415      \PassOptionsToClass{\CurrentOption}{beamer}
7416      \PassOptionsToPackage{\CurrentOption}{notesslides}
7417      \PassOptionsToPackage{\CurrentOption}{stex}
7418    }
7419  }
7420  \ProcessKeysOptions{ notesslides / cls }
7421
7422  \str_if_empty:NF \c__notesslides_class_str {
7423    \PassOptionsToPackage{class=\c__notesslides_class_str}{document-structure}
7424  }
7425
7426  \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{book}{
7427    \PassOptionsToPackage{defaulttopsect=part}{notesslides}
7428  }
7429  \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{report}{
7430    \PassOptionsToPackage{defaulttopsect=part}{notesslides}
7431  }
7432
7433  \RequirePackage{stex}
```

```
7434  \stex_html_backend:T {
7435    \bool_set_true:N\c__notesslides_notes_bool
7436  }
7437
7438  \bool_if:NTF \c__notesslides_notes_bool {
7439    \PassOptionsToPackage{notes=true}{notesslides}
7440    \message{notesslides.cls:~Formatting~course~materials~in~notes~mode}
7441  }{
7442    \PassOptionsToPackage{notes=false}{notesslides}
7443    \message{notesslides.cls:~Formatting~course~materials~in~slides~mode}
7444  }
7445  ⟨/cls⟩
```

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

```
7446  ⟨*package⟩
7447  \ProvidesExplPackage{notesslides}{2022/08/08}{3.2.0}{notesslides Package}
7448  \RequirePackage{l3keys2e}
7449
7450  \keys_define:nn{notesslides / pkg}{
7451    topsect          .str_set_x:N  = \c__notesslides_topsect_str,
7452    defaulttopsect   .str_set_x:N  = \c__notesslides_defaulttopsec_str,
7453    notes            .bool_set:N   = \c__notesslides_notes_bool ,
7454    slides           .code:n       = { \bool_set_false:N \c__notesslides_notes_bool },
7455    sectocframes     .bool_set:N   = \c__notesslides_sectocframes_bool ,
7456    frameimages      .bool_set:N   = \c__notesslides_frameimages_bool ,
7457    fiboxed          .bool_set:N   = \c__notesslides_fiboxed_bool ,
7458    noproblems       .bool_set:N   = \c__notesslides_noproblems_bool,
7459    unknown          .code:n       = {
7460      \PassOptionsToClass{\CurrentOption}{stex}
7461      \PassOptionsToClass{\CurrentOption}{tikzinput}
7462    }
7463  }
7464  \ProcessKeysOptions{ notesslides / pkg }
7465
7466  \RequirePackage{stex}
7467  \stex_html_backend:T {
7468    \bool_set_true:N\c__notesslides_notes_bool
7469  }
7470
7471  \newif\ifnotes
7472  \bool_if:NTF \c__notesslides_notes_bool {
7473    \notestrue
7474  }{
7475    \notesfalse
7476  }
7477
```

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

```
7478  \str_if_empty:NTF \c__notesslides_topsect_str {
7479    \str_set_eq:NN \__notesslidestopsect \c__notesslides_defaulttopsec_str
7480  }{
7481    \str_set_eq:NN \__notesslidestopsect \c__notesslides_topsect_str
7482  }
7483  \PassOptionsToPackage{topsect=\__notesslidestopsect}{document-structure}
```

₇₄₈₄ ⟨/package⟩

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

₇₄₈₅ ⟨*cls⟩
₇₄₈₆ *\bool_if:NTF \c__notesslides_notes_bool {*
₇₄₈₇ * \str_if_empty:NT \c__notesslides_class_str {*
₇₄₈₈ *   \str_set:Nn \c__notesslides_class_str {article}*
₇₄₈₉ * }*
₇₄₉₀ * \exp_after:wN\LoadClass\exp_after:wN[\c__notesslides_docopt_str]*
₇₄₉₁ *   {\c__notesslides_class_str}*
₇₄₉₂ *}{*
₇₄₉₃ * \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}*
₇₄₉₄ * \newcounter{Item}*
₇₄₉₅ * \newcounter{paragraph}*
₇₄₉₆ * \newcounter{subparagraph}*
₇₄₉₇ * \newcounter{Hfootnote}*
₇₄₉₈ *}*
₇₄₉₉ *\RequirePackage{document-structure}*

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

₇₅₀₀ *\RequirePackage{notesslides}*
₇₅₀₁ ⟨/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 SᴛEX packages. The first batch of packages we want are loaded on `notesslides.sty`. These are the general ones, we will load the SᴛEX-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.

₇₅₀₂ ⟨*package⟩
₇₅₀₃ *\bool_if:NT \c__notesslides_notes_bool {*
₇₅₀₄ * \RequirePackage{a4wide}*
₇₅₀₅ * \RequirePackage{marginnote}*
₇₅₀₆ * \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}*
₇₅₀₇ * \RequirePackage{mdframed}*
₇₅₀₈ * \RequirePackage[noxcolor,noamsthm]{beamerarticle}*
₇₅₀₉ * \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}*
₇₅₁₀ *}*
₇₅₁₁ *\RequirePackage{stex-tikzinput}*
₇₅₁₂ *\RequirePackage{comment}*
₇₅₁₃ *\RequirePackage{url}*
₇₅₁₄ *\RequirePackage{graphicx}*
₇₅₁₅ *\RequirePackage{pgf}*
₇₅₁₆ *\RequirePackage{bookmark}*

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.

₇₅₁₇ *\bool_if:NT \c__notesslides_notes_bool {*
₇₅₁₈ * \renewcommand\usetheme[2][]{\usepackage[#1]{beamertheme#2}}*
₇₅₁₉ *}*

```
7520  \NewDocumentCommand \libusetheme {O{} m} {
7521    \libusepackage[#1]{beamertheme#2}
7522  }
7523
```

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

```
7524  \newcounter{slide}
7525  \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
7526  \newlength{\slideheight}\setlength{\slideheight}{9cm}
```

note (*env.*) 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.

```
7527  \bool_if:NTF \c__notesslides_notes_bool {
7528    \renewenvironment{note}{\ignorespaces}{}
7529  }{
7530    \excludecomment{note}
7531  }
```

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.

```
7532  \bool_if:NT \c__notesslides_notes_bool {
7533    \newlength{\slideframewidth}
7534    \setlength{\slideframewidth}{1.5pt}
```

frame (*env.*) We first define the keys.

```
7535  \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
7536    \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
7537      \bool_set_true:N #1
7538    }{
7539      \bool_set_false:N #1
7540    }
7541  }
7542  \keys_define:nn{notesslides / frame}{
7543    label                 .str_set_x:N = \l__notesslides_frame_label_str,
7544    allowframebreaks    .code:n       = {
7545      \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
7546    },
7547    allowdisplaybreaks  .code:n       = {
7548      \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
7549    },
7550    fragile               .code:n       = {
7551      \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
7552    },
7553    shrink                .code:n       = {
7554      \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
7555    },
7556    squeeze               .code:n       = {
7557      \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
7558    },
7559    t                     .code:n       = {
```

```
7560        \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
7561      },
7562      unknown    .code:n        = {}
7563    }
7564    \cs_new_protected:Nn \__notesslides_frame_args:n {
7565      \str_clear:N \l__notesslides_frame_label_str
7566      \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
7567      \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
7568      \bool_set_true:N \l__notesslides_frame_fragile_bool
7569      \bool_set_true:N \l__notesslides_frame_shrink_bool
7570      \bool_set_true:N \l__notesslides_frame_squeeze_bool
7571      \bool_set_true:N \l__notesslides_frame_t_bool
7572      \keys_set:nn { notesslides / frame }{ #1 }
7573    }
```

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

```
7574    \renewenvironment{frame}[1][]{
7575      \__notesslides_frame_args:n{#1}
7576      \sffamily
7577      \stepcounter{slide}
7578      \def\@currentlabel{\theslide}
7579      \str_if_empty:NF \l__notesslides_frame_label_str {
7580        \label{\l__notesslides_frame_label_str}
7581      }
```

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

```
7582      \def\itemize@level{outer}
7583      \def\itemize@outer{outer}
7584      \def\itemize@inner{inner}
7585      \renewcommand\newpage{\addtocounter{framenumber}{1}}
7586      %\newcommand\metakeys@show@keys[2]{\marginnote{{\scriptsize ##2}}}
7587      \renewenvironment{itemize}{
7588        \ifx\itemize@level\itemize@outer
7589          \def\itemize@label{$\rhd$}
7590        \fi
7591        \ifx\itemize@level\itemize@inner
7592          \def\itemize@label{$\scriptstyle\rhd$}
7593        \fi
7594        \begin{list}
7595        {\itemize@label}
7596        {\setlength{\labelsep}{.3em}
7597         \setlength{\labelwidth}{.5em}
7598         \setlength{\leftmargin}{1.5em}
7599        }
7600        \edef\itemize@level{\itemize@inner}
7601      }{
7602        \end{list}
7603      }
```

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

```
7604      \stex_html_backend:TF {
7605        \begin{stex_annotate_env}{frame}{}\vbox\bgroup
7606          \mdf@patchamsthm
7607      }{
7608        \begin{mdframed}[[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwid
```

269

```
7609        }
7610   }{
7611      \stex_html_backend:TF {
7612         \miko@slidelabel\egroup\end{stex_annotate_env}
7613      }{\medskip\miko@slidelabel\end{mdframed}}
7614   }
```

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

\frametitle

```
7615   \renewcommand{\frametitle}[1]{
7616      \stex_document_title:n { #1 }
7617      {\Large\bf\sf\color{blue}{#1}}\medskip
7618   }
7619 }
```

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

EdN:10   \pause   10

```
7620 \bool_if:NT \c__notesslides_notes_bool {
7621   \newcommand\pause{}
7622 }
```

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

nparagraph (*env.*)

```
7623 \bool_if:NTF \c__notesslides_notes_bool {
7624   \newenvironment{nparagraph}[1][]{\begin{sparagraph}[#1]}{\end{sparagraph}}
7625 }{
7626   \excludecomment{nparagraph}
7627 }
```

nfragment (*env.*)

```
7628 \bool_if:NTF \c__notesslides_notes_bool {
7629   \newenvironment{nfragment}[2][]{\begin{sfragment}[#1]{#2}}{\end{sfragment}}
7630 }{
7631   \excludecomment{nfragment}
7632 }
```

ndefinition (*env.*)

```
7633 \bool_if:NTF \c__notesslides_notes_bool {
7634   \newenvironment{ndefinition}[1][]{\begin{sdefinition}[#1]}{\end{sdefinition}}
7635 }{
7636   \excludecomment{ndefinition}
7637 }
```

nassertion (*env.*)

```
7638 \bool_if:NTF \c__notesslides_notes_bool {
7639   \newenvironment{nassertion}[1][]{\begin{sassertion}[#1]}{\end{sassertion}}
7640 }{
7641   \excludecomment{nassertion}
7642 }
```

---

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

270

```
7643  \bool_if:NTF \c__notesslides_notes_bool {
7644    \newenvironment{nproof}[2][]{\begin{sproof}[#1]{#2}}{\end{sproof}}
7645  }{
7646    \excludecomment{nproof}
7647  }
```

```
7648  \bool_if:NTF \c__notesslides_notes_bool {
7649    \newenvironment{nexample}[1][]{\begin{sexample}[#1]}{\end{sexample}}
7650  }{
7651    \excludecomment{nexample}
7652  }
```

We customize the hooks for in \inputref.

```
7653  \def\inputref@preskip{\smallskip}
7654  \def\inputref@postskip{\medskip}
```

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

```
7655  \let\orig@inputref\inputref
7656  \def\inputref{\@ifstar\ninputref\orig@inputref}
7657  \newcommand\ninputref[2][]{
7658    \bool_if:NT \c__notesslides_notes_bool {
7659      \orig@inputref[#1]{#2}
7660    }
7661  }
```

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

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.

The default logo is the SₜₑX logo. Customization can be done by \setslidelogo{⟨*logo name*⟩}.

```
7662  \newlength{\slidelogoheight}
7663
7664  \RequirePackage{graphicx}
7665
7666  \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7667  \providecommand\mhgraphics[2][]{
7668    \def\Gin@mhrepos{}\setkeys{Gin}{#1}
7669    \includegraphics[#1]{\mhpath\Gin@mhrepos{#2}}
7670  }
7671
7672  \bool_if:NTF \c__notesslides_notes_bool {
7673    \setlength{\slidelogoheight}{.4cm}
7674  }{
7675    \setlength{\slidelogoheight}{.25cm}
7676  }
```

271

```
7677  \ifcsname slidelogo\endcsname\else
7678    \newsavebox{\slidelogo}
7679    \sbox{\slidelogo}{\sTeX}
7680  \fi
7681  \newrobustcmd{\setslidelogo}[2][]{
7682    \tl_if_empty:nTF{#1}{
7683      \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#2}}
7684    }{
7685      \sbox{\slidelogo}{\mhgraphics[height=\slidelogoheight,mhrepos=#1]{#2}}
7686    }
7687  }
```

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

\author   In notes mode, we redefine the \author macro so that it does not disregard the optional
          argument (as beamerarticle does). We want to use it to set the source later.

```
7688  \bool_if:NT \c__notesslides_notes_bool {
7689    \def\author{\@dblarg\ns@author}
7690    \long\def\ns@author[#1]#2{%
7691      \def\c__notesslides_shortauthor{#1}%
7692      \def\@author{#2}
7693    }
7694  }
```

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

\setsource   \source stores the writer's name. By default it is *Michael Kohlhase* since he is the main
             user and designer of this package. \setsource{⟨*name*⟩} can change the writer's name.

```
7695  \newrobustcmd{\setsource}[1]{\def\source{#1}}
```

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

\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.

```
7696  \def\copyrightnotice{%
7697    \footnotesize\copyright :\hspace{.3ex}%
7698    \ifcsname source\endcsname\source\else%
7699    \ifcsname c__notesslides_shortauthor\endcsname\c__notesslides_shortauthor\else%
7700    \PackageWarning{notesslides}{Author/Source~undefined~in~copyright~notice}%
7701    ?source/author?\fi%
7702    \fi}
7703  \newsavebox{\cclogo}
7704  \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{stex-cc_somerights}}
7705  \newif\ifcchref\cchreffalse
7706  \AtBeginDocument{
7707    \@ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
7708  }
7709  \def\licensing{
7710    \ifcchref
7711      \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
7712    \else
7713      {\usebox{\cclogo}}
```

```
7714        \fi
7715    }
7716    \newrobustcmd{\setlicensing}[2][]{
7717        \def\@url{#1}
7718        \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
7719        \ifx\@url\@empty
7720            \def\licensing{{\usebox{\cclogo}}}
7721        \else
7722            \def\licensing{
7723                \ifcchref
7724                \href{#1}{\usebox{\cclogo}}
7725                \else
7726                {\usebox{\cclogo}}
7727                \fi
7728            }
7729        \fi
7730    }
```

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

EdN:11   `\slidelabel`   Now, we set up the slide label for the `article` mode.[11]

```
7731    \newrobustcmd\miko@slidelabel{
7732        \vbox to \slidelogoheight{
7733            \vss\hbox to \slidewidth
7734            {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
7735        }
7736    }
```

(*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.

```
7737    \def\Gin@mhrepos{}
7738    \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7739    \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
7740    \newrobustcmd\frameimage[2][]{
7741        \stepcounter{slide}
7742        \bool_if:NT \c__notesslides_frameimages_bool {
7743            \def\Gin@ewidth{}\setkeys{Gin}{#1}
7744            \bool_if:NF \c__notesslides_notes_bool { \vfill }
7745            \begin{center}
7746                \bool_if:NTF \c__notesslides_fiboxed_bool {
7747                    \fbox{
7748                        \ifx\Gin@ewidth\@empty
7749                            \ifx\Gin@mhrepos\@empty
7750                                \mhgraphics[width=\slidewidth,#1]{#2}
7751                            \else
7752                                \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7753                            \fi
7754                        \else% Gin@ewidth empty
```

---

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

```
7755        \ifx\Gin@mhrepos\@empty
7756          \mhgraphics[#1]{#2}
7757        \else
7758          \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7759        \fi
7760      \fi% Gin@ewidth empty
7761    }
7762  }{
7763    \ifx\Gin@ewidth\@empty
7764      \ifx\Gin@mhrepos\@empty
7765        \mhgraphics[width=\slidewidth,#1]{#2}
7766      \else
7767        \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7768      \fi
7769      \ifx\Gin@mhrepos\@empty
7770        \mhgraphics[#1]{#2}
7771      \else
7772        \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7773      \fi
7774    \fi% Gin@ewidth empty
7775    }
7776    \end{center}
7777    \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
7778    \bool_if:NF \c__notesslides_notes_bool { \vfill }
7779  }
7780 } % ifmks@sty@frameimages
```

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

38.5   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`.

```
7781 \stex_html_backend:F {
7782   \bool_if:NT \c__notesslides_sectocframes_bool {
7783     \str_if_eq:VnTF \__notesslidestopsect{part}{
7784       \newcounter{chapter}\counterwithin*{section}{chapter}
7785     }{
7786       \str_if_eq:VnT\__notesslidestopsect{chapter}{
7787         \newcounter{chapter}\counterwithin*{section}{chapter}
7788       }
7789     }
7790   }
7791 }
```

\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

```
7792 \def\part@prefix{}
7793 \@ifpackageloaded{document-structure}{}{
7794   \str_case:VnF \__notesslidestopsect {
```

274

```
7795    {part}{
7796        \int_set:Nn \l_document_structure_section_level_int {0}
7797        \def\thesection{\arabic{chapter}.\arabic{section}}
7798        \def\part@prefix{\arabic{chapter}.}
7799    }
7800    {chapter}{
7801        \int_set:Nn \l_document_structure_section_level_int {1}
7802        \def\thesection{\arabic{chapter}.\arabic{section}}
7803        \def\part@prefix{\arabic{chapter}.}
7804    }
7805    }{
7806        \int_set:Nn \l_document_structure_section_level_int {2}
7807        \def\part@prefix{}
7808    }
7809 }
7810
7811 \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 (*env.*)

```
7812  \renewenvironment{sfragment}[2][]{
7813      \__document_structure_sfragment_args:n { #1 }
7814      \int_incr:N \l_document_structure_section_level_int
7815      \bool_if:NT \c__notesslides_sectocframes_bool {
7816          \stepcounter{slide}
7817          \begin{frame}[noframenumbering]
7818          \vfill\Large\centering
7819          \red{
7820              \ifcase\l_document_structure_section_level_int\or
7821                  \stepcounter{part}
7822                  \def\__notesslideslabel{{\omdoc@part@kw}~\Roman{part}}
7823                  \addcontentsline{toc}{part}{\protect\numberline{\thepart}#2}
7824                  \pdfbookmark[0]{\thepart\ #2}{part.\thepart}
7825                  \def\currentsectionlevel{\omdoc@part@kw}
7826              \or
7827                  \stepcounter{chapter}
7828                  \def\__notesslideslabel{{\omdoc@chapter@kw}~\arabic{chapter}}
7829                  \addcontentsline{toc}{chapter}{\protect\numberline{\thechapter}#2}
7830                  \pdfbookmark[1]{\thechapter\ #2}{chapter.\cs_if_exist:cT{thepart}\thepart.\thechap
7831                  \def\currentsectionlevel{\omdoc@chapter@kw}
7832              \or
7833                  \stepcounter{section}
7834                  \def\__notesslideslabel{\part@prefix\arabic{section}}
7835                  \addcontentsline{toc}{section}{\protect\numberline{\thesection}#2}
7836                  \pdfbookmark[2]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection\ #2}
7837                  {section.\cs_if_exist:cT{thepart}{\thepart.}\cs_if_exist:cT{thechapter}{\thechapte
7838                  \def\currentsectionlevel{\omdoc@section@kw}
7839              \or
7840                  \stepcounter{subsection}
7841                  \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
7842                  \addcontentsline{toc}{subsection}{\protect\numberline{\thesubsection}#2}
```

```
7843          \pdfbookmark[3]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsectio
7844          {subsection.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thecha
7845          \def\currentsectionlevel{\omdoc@subsection@kw}
7846        \or
7847          \stepcounter{subsubsection}
7848          \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
7849          \addcontentsline{toc}{subsubsection}{\protect\numberline{\thesubsubsection}#2}
7850          \pdfbookmark[4]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsectio
7851          {subsubsection.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\the
7852          \def\currentsectionlevel{\omdoc@subsubsection@kw}
7853        \or
7854          \stepcounter{paragraph}
7855          \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
7856          \addcontentsline{toc}{paragraph}{\protect\numberline{\theparagraph}#2}
7857          \pdfbookmark[5]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsectio
7858          {paragraph.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechap
7859          \def\currentsectionlevel{\omdoc@paragraph@kw}
7860        \else
7861          \def\__notesslideslabel{}
7862          \def\currentsectionlevel{\omdoc@paragraph@kw}
7863        \fi% end ifcase
7864        \__notesslideslabel\quad #2%
7865      }%
7866      \vfill%
7867      \end{frame}%
7868    }
7869    \str_if_empty:NF \l__document_structure_sfragment_id_str {
7870      \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
7871    }
7872  }{}
7873 }
```

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

```
7874 \def\inserttheorembodyfont{\normalfont}
7875 %\bool_if:NF \c__notesslides_notes_bool {
7876 %   \defbeamertemplate{theorem begin}{miko}
7877 %   {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
7878 %     \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
7879 %     \inserttheorempunctuation\inserttheorembodyfont\xspace}
7880 %   \defbeamertemplate{theorem end}{miko}{}
```

and we set it as the default one.

```
7881 %   \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.

```
7882 %   \expandafter\def\csname Parent2\endcsname{}
7883 %}
7884
7885 \AddToHook{begindocument}{ % this does not work for some reasone
7886   \setbeamertemplate{theorems}[ams style]
7887 }
7888 \bool_if:NT \c__notesslides_notes_bool {
7889   \renewenvironment{columns}[1][]{%
```

```
7890        \par\noindent%
7891        \begin{minipage}%
7892        \slidewidth\centering\leavevmode%
7893    }{%
7894        \end{minipage}\par\noindent%
7895    }%
7896    \newsavebox\columnbox%
7897    \renewenvironment<>{column}[2][]{%
7898        \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
7899    }{%
7900        \end{minipage}\end{lrbox}\usebox\columnbox%
7901    }%
7902 }

7903 \bool_if:NTF \c__notesslides_noproblems_bool {
7904    \newenvironment{problems}{}{}
7905 }{
7906    \excludecomment{problems}
7907 }
```

38.6   Excursions

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.

```
7908 \gdef\printexcursions{}
7909 \newcommand\excursionref[2]{% label, text
7910    \bool_if:NT \c__notesslides_notes_bool {
7911        \begin{sparagraph}[title=Excursion]
7912            #2 \sref[fallback=the appendix]{#1}.
7913        \end{sparagraph}
7914    }
7915 }
7916 \newcommand\activate@excursion[2][]{
7917    \gappto\printexcursions{\inputref[#1]{#2}}
7918 }
7919 \newcommand\excursion[4][]{% repos, label, path, text
7920    \bool_if:NT \c__notesslides_notes_bool {
7921        \activate@excursion[#1]{#3}\excursionref{#2}{#4}
7922    }
7923 }
```

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

```
7924 \keys_define:nn{notesslides / excursiongroup }{
7925    id       .str_set_x:N  = \l__notesslides_excursion_id_str,
7926    intro    .tl_set:N     = \l__notesslides_excursion_intro_tl,
7927    mhrepos   .str_set_x:N  = \l__notesslides_excursion_mhrepos_str
7928 }
7929 \cs_new_protected:Nn \__notesslides_excursion_args:n {
7930    \tl_clear:N \l__notesslides_excursion_intro_tl
7931    \str_clear:N \l__notesslides_excursion_id_str
```

```
7932    \str_clear:N \l__notesslides_excursion_mhrepos_str
7933    \keys_set:nn {notesslides / excursiongroup }{ #1 }
7934 }
7935 \newcommand\excursiongroup[1][]{
7936    \__notesslides_excursion_args:n{ #1 }
7937    \ifdefempty\printexcursions{}% only if there are excursions
7938    {\begin{note}
7939      \begin{sfragment}[#1]{Excursions}%
7940        \ifdefempty\l__notesslides_excursion_intro_tl{}{
7941          \inputref[\l__notesslides_excursion_mhrepos_str]{
7942            \l__notesslides_excursion_intro_tl
7943          }
7944        }
7945        \printexcursions%
7946      \end{sfragment}
7947    \end{note}}
7948 }
7949 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
7950 ⟨/package⟩
```

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

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.

```
7951 ⟨*package⟩
7952 ⟨@@=problems⟩
7953 \ProvidesExplPackage{problem}{2022/08/08}{3.2.0}{Semantic Markup for Problems}
7954 \RequirePackage{l3keys2e}
7955 \RequirePackage{amssymb}% for \Box
7956
7957 \keys_define:nn { problem / pkg }{
7958   notes      .default:n    = { true },
7959   notes      .bool_set:N   = \c__problems_notes_bool,
7960   gnotes     .default:n    = { true },
7961   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
7962   hints      .default:n    = { true },
7963   hints      .bool_set:N   = \c__problems_hints_bool,
7964   solutions .default:n    = { true },
7965   solutions .bool_set:N   = \c__problems_solutions_bool,
7966   pts        .default:n    = { true },
7967   pts        .bool_set:N   = \c__problems_pts_bool,
7968   min        .default:n    = { true },
7969   min        .bool_set:N   = \c__problems_min_bool,
7970   boxed      .default:n    = { true },
7971   boxed      .bool_set:N   = \c__problems_boxed_bool,
7972   unknown      .code:n       = {
7973     \PassOptionsToPackage{\CurrentOption}{stex}
7974   }
7975 }
7976 \newif\ifsolutions
7977
7978 \ProcessKeysOptions{ problem / pkg }
7979 \bool_if:NTF \c__problems_solutions_bool {
7980   \solutionstrue
7981 }{
7982   \solutionsfalse
```

```
7983  }
7984  \RequirePackage{stex}
```

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

```
7985  \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.

```
7986  \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.

```
7987  \def\prob@problem@kw{Problem}
7988  \def\prob@solution@kw{Solution}
7989  \def\prob@hint@kw{Hint}
7990  \def\prob@note@kw{Note}
7991  \def\prob@gnote@kw{Grading}
7992  \def\prob@pt@kw{pt}
7993  \def\prob@min@kw{min}
7994  \def\prob@correct@kw{Correct}
7995  \def\prob@wrong@kw{Wrong}
```

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

For the other languages, we set up triggers

```
7996  \AddToHook{begindocument}{
7997    \ltx@ifpackageloaded{babel}{
7998        \makeatletter
7999        \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
8000        \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
8001          \input{problem-ngerman.ldf}
8002        }
8003        \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
8004          \input{problem-finnish.ldf}
8005        }
8006        \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
8007          \input{problem-french.ldf}
8008        }
8009        \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
8010          \input{problem-russian.ldf}
8011        }
8012        \makeatother
8013    }{}
8014  }
```

39.2   Problems and Solutions

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

```
8015  \keys_define:nn{ problem / problem }{
8016    id       .str_set_x:N  = \l__problems_prob_id_str,
8017    pts      .tl_set:N      = \l__problems_prob_pts_tl,
8018    min      .tl_set:N      = \l__problems_prob_min_tl,
```

```
8019   title   .tl_set:N     = \l__problems_prob_title_tl,
8020   type    .tl_set:N     = \l__problems_prob_type_tl,
8021   imports .tl_set:N     = \l__problems_prob_imports_tl,
8022   name    .str_set_x:N  = \l__problems_prob_name_str,
8023   refnum  .int_set:N    = \l__problems_prob_refnum_int
8024 }
8025 \cs_new_protected:Nn \__problems_prob_args:n {
8026   \str_clear:N \l__problems_prob_id_str
8027   \str_clear:N \l__problems_prob_name_str
8028   \tl_clear:N \l__problems_prob_pts_tl
8029   \tl_clear:N \l__problems_prob_min_tl
8030   \tl_clear:N \l__problems_prob_title_tl
8031   \tl_clear:N \l__problems_prob_type_tl
8032   \tl_clear:N \l__problems_prob_imports_tl
8033   \int_zero_new:N \l__problems_prob_refnum_int
8034   \keys_set:nn { problem / problem }{ #1 }
8035   \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
8036     \let\l__problems_prob_refnum_int\undefined
8037   }
8038 }
```

Then we set up a counter for problems.

\numberproblemsin

```
8039 \newcounter{sproblem}[section]
8040 \newcommand\numberproblemsin[1]{\@addtoreset{sproblem}{#1}}
8041 \def\theplainsproblem{\arabic{sproblem}}
8042 \def\thesproblem{\thesection.\theplainsproblem}
```

(*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.

```
8043 \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

```
8044 \newcommand\prob@number{
8045   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
8046     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
8047   }{
8048     \int_if_exist:NTF \l__problems_prob_refnum_int {
8049       \prob@label{\int_use:N \l__problems_prob_refnum_int }
8050     }{
8051         \prob@label\theplainsproblem
8052     }
8053   }
8054 }
8055 \def\sproblemautorefname{\prob@problem@kw}
```

(*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.

```
8056  \newcommand\prob@title[3]{%
8057    \tl_if_exist:NTF \l__problems_inclprob_title_tl {
8058      #2 \l__problems_inclprob_title_tl #3
8059    }{
8060      \tl_if_empty:NTF \l__problems_prob_title_tl {
8061        #1
8062      }{
8063        #2 \l__problems_prob_title_tl #3
8064      }
8065    }
8066  }
```

(*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.

```
8067  \def\prob@heading{
8068    {\prob@problem@kw}\ \prob@number\prob@title{~}{~(}{)}\strut}
8069    %\sref@label@id{\prob@problem@kw~\prob@number}{}
8070  }
```

(*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 (*env.*)

```
8071  \newenvironment{sproblem}[1][]{
8072    \__problems_prob_args:n{#1}%\sref@target%
8073    \@in@omtexttrue% we are in a statement (for inline definitions)
8074    \refstepcounter{sproblem}\record@problem
8075    \def\current@section@level{\prob@problem@kw}
8076
8077    \str_if_empty:NT \l__problems_prob_name_str {
8078      \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
8079      \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
8080      \seq_get_left:NN \l_tmpa_seq \l__problems_prob_name_str
8081    }
8082
8083    \stex_if_do_html:T{
8084      \tl_if_empty:NF \l__problems_prob_title_tl {
8085        \exp_args:No \stex_document_title:n \l__problems_prob_title_tl
8086      }
8087    }
8088
8089    \exp_args:Nno\stex_module_setup:nn{type=problem}\l__problems_prob_name_str
8090
8091    \stex_reactivate_macro:N \STEXexport
8092    \stex_reactivate_macro:N \importmodule
8093    \stex_reactivate_macro:N \symdecl
8094    \stex_reactivate_macro:N \notation
8095    \stex_reactivate_macro:N \symdef
```

```
8096
8097    \stex_if_do_html:T{
8098      \begin{stex_annotate_env} {problem} {
8099        \l_stex_module_ns_str ? \l_stex_module_name_str
8100      }
8101
8102      \stex_annotate_invisible:nnn{header}{} {
8103        \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
8104        \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
8105        \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
8106          \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
8107        }
8108      }
8109    }
8110
8111    \stex_csl_to_imports:No \importmodule \l__problems_prob_imports_tl
8112
8113
8114    \tl_if_exist:NTF \l__problems_inclprob_type_tl {
8115      \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
8116    }{
8117      \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
8118    }
8119    \str_if_exist:NTF \l__problems_inclprob_id_str {
8120      \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
8121    }{
8122      \str_set_eq:NN \sproblemid \l__problems_prob_id_str
8123    }
8124
8125
8126    \stex_if_smsmode:F {
8127      \clist_set:No \l_tmpa_clist \sproblemtype
8128      \tl_clear:N \l_tmpa_tl
8129      \clist_map_inline:Nn \l_tmpa_clist {
8130        \tl_if_exist:cT {__problems_sproblem_##1_start:}{
8131          \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_start:}}
8132        }
8133      }
8134      \tl_if_empty:NTF \l_tmpa_tl {
8135        \__problems_sproblem_start:
8136      }{
8137        \l_tmpa_tl
8138      }
8139    }
8140    \stex_ref_new_doc_target:n \sproblemid
8141    \stex_if_smsmode:TF \stex_smsmode_do: \ignorespacesandpars
8142 }{
8143    \__stex_modules_end_module:
8144    \stex_if_smsmode:F{
8145      \clist_set:No \l_tmpa_clist \sproblemtype
8146      \tl_clear:N \l_tmpa_tl
8147      \clist_map_inline:Nn \l_tmpa_clist {
8148        \tl_if_exist:cT {__problems_sproblem_##1_end:}{
8149          \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_end:}}
```

```
8150              }
8151            }
8152        \tl_if_empty:NTF \l_tmpa_tl {
8153          \__problems_sproblem_end:
8154        }{
8155          \l_tmpa_tl
8156        }
8157      }
8158      \stex_if_do_html:T{
8159        \end{stex_annotate_env}
8160      }
8161
8162      \smallskip
8163 }
8164
8165 \seq_put_right:Nx\g_stex_smsmode_allowedenvs_seq{\tl_to_str:n{sproblem}}
8166
8167
8168
8169 \cs_new_protected:Nn \__problems_sproblem_start: {
8170    \par\noindent\textbf\prob@heading\show@pts\show@min\\ignorespacesandpars
8171 }
8172 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
8173
8174 \newcommand\stexpatchproblem[3][] {
8175      \str_set:Nx \l_tmpa_str{ #1 }
8176      \str_if_empty:NTF \l_tmpa_str {
8177        \tl_set:Nn \__problems_sproblem_start: { #2 }
8178        \tl_set:Nn \__problems_sproblem_end: { #3 }
8179      }{
8180        \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
8181        \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
8182      }
8183 }
8184
8185
8186 \bool_if:NT \c__problems_boxed_bool {
8187    \surroundwithmdframed{problem}
8188 }
```

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

```
8189 \def\record@problem{
8190    \protected@write\@auxout{}
8191    {
8192      \string\@problem{\prob@number}
8193      {
8194        \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
8195          \l__problems_inclprob_pts_tl
8196        }{
8197          \l__problems_prob_pts_tl
8198        }
8199      }%
8200      {
8201        \tl_if_exist:NTF \l__problems_inclprob_min_tl {
```

284

```
8202          \l__problems_inclprob_min_tl
8203       }{
8204          \l__problems_prob_min_tl
8205       }
8206     }
8207   }
8208 }
```

(*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).

```
8209 \def\@problem#1#2#3{}
```

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

solution (*env.*) 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.

```
8210 \keys_define:nn { problem / solution }{
8211   id              .str_set_x:N  = \l__problems_solution_id_str ,
8212   for             .str_set_x:N  = \l__problems_solution_for_str ,
8213   type            .str_set_x:N  = \l__problems_solution_type_str ,
8214   title           .tl_set:N     = \l__problems_solution_title_tl
8215 }
8216 \cs_new_protected:Nn \__problems_solution_args:n {
8217   \str_clear:N \l__problems_solution_id_str
8218   \str_clear:N \l__problems_solution_type_str
8219   \str_clear:N \l__problems_solution_for_str
8220   \tl_clear:N \l__problems_solution_title_tl
8221   \keys_set:nn { problem / solution }{ #1 }
8222 }
```

\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.

```
8223 \box_new:N \l__problems_solution_box
8224 \newenvironment{solution}[1][]{
8225   \__problems_solution_args:n{#1}
8226   \stex_html_backend:TF{
8227     \stex_if_do_html:T{
8228       \begin{stex_annotate_env}{solution}{}
8229         \str_if_empty:NF \l__problems_solution_type_str {
8230           \stex_annotate_invisible:nnn{typestrings}{\sexampletype}{}
8231         }
8232         \noindent\textbf{Solution\tl_if_empty:NF\l__problems_solution_title_tl{~(\l__problem
8233     }
8234   }{
8235     \setbox\l__problems_solution_box\vbox\bgroup
8236       \par\smallskip\hrule\smallskip
8237       \noindent\textbf{Solution\tl_if_empty:NF\l__problems_solution_title_tl{~(\l__problems_
8238   }
8239 }{
8240   \stex_html_backend:TF{
8241     \stex_if_do_html:T{
8242       \end{stex_annotate_env}
```

285

```
8243        }
8244     }{
8245       \smallskip\hrule
8246       \egroup
8247       \bool_if:NT \c__problems_solutions_bool {
8248         \box\l__problems_solution_box
8249       }
8250     }
8251 }
8252
8253 \newcommand\startsolutions{
8254     \bool_set_true:N \c__problems_solutions_bool
8255     \solutionstrue
8256 %   \specialcomment{solution}{\@startsolution}{
8257 %     \bool_if:NF \c__problems_boxed_bool {
8258 %       \hrule\medskip
8259 %     }
8260 %     \end{small}%
8261 %   }
8262 %   \bool_if:NT \c__problems_boxed_bool {
8263 %     \surroundwithmdframed{solution}
8264 %   }
8265 }
```

(*End definition for* \startsolutions. *This function is documented on page 64.*)

\stopsolutions

```
8266 \newcommand\stopsolutions{\bool_set_false:N \c__problems_solutions_bool \solutionsfalse}%\ex
```

(*End definition for* \stopsolutions. *This function is documented on page 64.*)

exnote (*env.*)

```
8267 \bool_if:NTF \c__problems_notes_bool {
8268     \newenvironment{exnote}[1][]{
8269       \par\smallskip\hrule\smallskip
8270       \noindent\textbf{\prob@note@kw :~ }\small
8271     }{
8272       \smallskip\hrule
8273     }
8274 }{
8275     \excludecomment{exnote}
8276 }
```

hint (*env.*)

```
8277 \bool_if:NTF \c__problems_notes_bool {
8278     \newenvironment{hint}[1][]{
8279       \par\smallskip\hrule\smallskip
8280       \noindent\textbf{\prob@hint@kw :~ }\small
8281     }{
8282       \smallskip\hrule
8283     }
8284     \newenvironment{exhint}[1][]{
8285       \par\smallskip\hrule\smallskip
8286       \noindent\textbf{\prob@hint@kw :~ }\small
```

```
8287   }{
8288     \smallskip\hrule
8289   }
8290 }{
8291   \excludecomment{hint}
8292   \excludecomment{exhint}
8293 }
```

gnote (*env.*)

```
8294 \bool_if:NTF \c__problems_notes_bool {
8295   \newenvironment{gnote}[1][]{
8296     \par\smallskip\hrule\smallskip
8297     \noindent\textbf{\prob@gnote@kw :~ }\small
8298   }{
8299     \smallskip\hrule
8300   }
8301 }{
8302   \excludecomment{gnote}
8303 }
```

39.3 Marup for Added Value Services

39.4 Multiple Choice Blocks

mcb (*env.*) [12]

```
8304 \newenvironment{mcb}{
8305   \begin{enumerate}
8306 }{
8307   \end{enumerate}
8308 }
```

we define the keys for the mcc macro

```
8309 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
8310   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
8311     \bool_set_true:N #1
8312   }{
8313     \bool_set_false:N #1
8314   }
8315 }
8316 \keys_define:nn { problem / mcc }{
8317   id        .str_set_x:N  = \l__problems_mcc_id_str ,
8318   feedback  .tl_set:N     = \l__problems_mcc_feedback_tl ,
8319   T         .default:n    = { false } ,
8320   T         .bool_set:N   = \l__problems_mcc_t_bool ,
8321   F         .default:n    = { false } ,
8322   F         .bool_set:N   = \l__problems_mcc_f_bool ,
8323   Ttext     .tl_set:N     = \l__problems_mcc_Ttext_tl ,
8324   Ftext     .tl_set:N     = \l__problems_mcc_Ftext_tl
8325 }
8326 \cs_new_protected:Nn \l__problems_mcc_args:n {
```

---

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

```
8327   \str_clear:N \l__problems_mcc_id_str
8328   \tl_clear:N \l__problems_mcc_feedback_tl
8329   \bool_set_false:N \l__problems_mcc_t_bool
8330   \bool_set_false:N \l__problems_mcc_f_bool
8331   \tl_clear:N \l__problems_mcc_Ttext_tl
8332   \tl_clear:N \l__problems_mcc_Ftext_tl
8333   \str_clear:N \l__problems_mcc_id_str
8334   \keys_set:nn { problem / mcc }{ #1 }
8335 }
```

\mcc

```
8336 \def\mccTrueText{\textbf{\prob@correct@kw!~}}
8337 \def\mccFalseText{\textbf{\prob@wrong@kw!~}}
8338 \newcommand\mcc[2][]{
8339   \l__problems_mcc_args:n{ #1 }
8340   \item[$\Box$] #2
8341   \bool_if:NT \c__problems_solutions_bool{
8342     \\
8343     \bool_if:NT \l__problems_mcc_t_bool {
8344       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccTrueText\l__problems_mcc_Ttext_tl
8345     }
8346     \bool_if:NT \l__problems_mcc_f_bool {
8347       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccFalseText\l__problems_mcc_Ftext_tl
8348     }
8349     \tl_if_empty:NF \l__problems_mcc_feedback_tl {
8350       \emph{\l__problems_mcc_feedback_tl}
8351     }
8352   }
8353 } %solutions
```

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

39.5   Filling in Concrete Solutions

\includeproblem   This is embarrasingly simple, but can grow over time.

```
8354 \newcommand\fillinsol[1]{\quad%
8355   \ifsolutions\textcolor{red}{#1!}\else%
8356   \fbox{\phantom{\huge{#1}}}%
8357   \fi}
```

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

39.6   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.

```
8358
8359 \keys_define:nn{ problem / inclproblem }{
8360   id      .str_set_x:N  = \l__problems_inclprob_id_str,
8361   pts     .tl_set:N     = \l__problems_inclprob_pts_tl,
8362   min     .tl_set:N     = \l__problems_inclprob_min_tl,
```

```
8363    title    .tl_set:N     = \l__problems_inclprob_title_tl,
8364    refnum  .int_set:N    = \l__problems_inclprob_refnum_int,
8365    type     .tl_set:N     = \l__problems_inclprob_type_tl,
8366    mhrepos .str_set_x:N = \l__problems_inclprob_mhrepos_str
8367 }
8368 \cs_new_protected:Nn \__problems_inclprob_args:n {
8369    \str_clear:N \l__problems_prob_id_str
8370    \tl_clear:N \l__problems_inclprob_pts_tl
8371    \tl_clear:N \l__problems_inclprob_min_tl
8372    \tl_clear:N \l__problems_inclprob_title_tl
8373    \tl_clear:N \l__problems_inclprob_type_tl
8374    \int_zero_new:N \l__problems_inclprob_refnum_int
8375    \str_clear:N \l__problems_inclprob_mhrepos_str
8376    \keys_set:nn { problem / inclproblem }{ #1 }
8377    \tl_if_empty:NT \l__problems_inclprob_pts_tl {
8378       \let\l__problems_inclprob_pts_tl\undefined
8379    }
8380    \tl_if_empty:NT \l__problems_inclprob_min_tl {
8381       \let\l__problems_inclprob_min_tl\undefined
8382    }
8383    \tl_if_empty:NT \l__problems_inclprob_title_tl {
8384       \let\l__problems_inclprob_title_tl\undefined
8385    }
8386    \tl_if_empty:NT \l__problems_inclprob_type_tl {
8387       \let\l__problems_inclprob_type_tl\undefined
8388    }
8389    \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
8390       \let\l__problems_inclprob_refnum_int\undefined
8391    }
8392 }
8393
8394 \cs_new_protected:Nn \__problems_inclprob_clear: {
8395    \let\l__problems_inclprob_id_str\undefined
8396    \let\l__problems_inclprob_pts_tl\undefined
8397    \let\l__problems_inclprob_min_tl\undefined
8398    \let\l__problems_inclprob_title_tl\undefined
8399    \let\l__problems_inclprob_type_tl\undefined
8400    \let\l__problems_inclprob_refnum_int\undefined
8401    \let\l__problems_inclprob_mhrepos_str\undefined
8402 }
8403 \__problems_inclprob_clear:
8404
8405 \newcommand\includeproblem[2][]{
8406    \__problems_inclprob_args:n{ #1 }
8407    \exp_args:No \stex_in_repository:nn\l__problems_inclprob_mhrepos_str{
8408       \stex_html_backend:TF {
8409          \str_clear:N \l_tmpa_str
8410          \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
8411             \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
8412          }
8413          \stex_annotate_invisible:nnn{includeproblem}{
8414             \l_tmpa_str / #2
8415          }{}
8416       }{
```

```
8417        \begingroup
8418          \inputreftrue
8419          \tl_if_empty:nTF{ ##1 }{
8420            \input{#2}
8421          }{
8422            \input{ \c_stex_mathhub_str / ##1 / source / #2 }
8423          }
8424        \endgroup
8425      }
8426    }
8427    \__problems_inclprob_clear:
8428  }
```

(*End definition for* `\includeproblem`. *This function is documented on page* [67](#).)

39.7   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.

```
8429  \AddToHook{enddocument}{
8430    \bool_if:NT \c__problems_pts_bool {
8431      \message{Total:~\arabic{pts}~points}
8432    }
8433    \bool_if:NT \c__problems_min_bool {
8434      \message{Total:~\arabic{min}~minutes}
8435    }
8436  }
```

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

```
8437  \def\pts#1{
8438    \bool_if:NT \c__problems_pts_bool {
8439      \marginpar{#1~\prob@pt@kw}
8440    }
8441  }
8442  \def\min#1{
8443    \bool_if:NT \c__problems_min_bool {
8444      \marginpar{#1~\prob@min@kw}
8445    }
8446  }
```

\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.

```
8447  \newcounter{pts}
8448  \def\show@pts{
8449    \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
8450      \bool_if:NT \c__problems_pts_bool {
8451        \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
8452        \addtocounter{pts}{\l__problems_inclprob_pts_tl}
8453      }
8454    }{
8455      \tl_if_exist:NT \l__problems_prob_pts_tl {
8456        \bool_if:NT \c__problems_pts_bool {
```

```
8457        \tl_if_empty:NT\l__problems_prob_pts_tl{
8458          \tl_set:Nn \l__problems_prob_pts_tl {0}
8459        }
8460        \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
8461        \addtocounter{pts}{\l__problems_prob_pts_tl}
8462      }
8463    }
8464  }
8465 }
```

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

and now the same for the minutes

\show@min

```
8466 \newcounter{min}
8467 \def\show@min{
8468   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
8469     \bool_if:NT \c__problems_min_bool {
8470       \marginpar{\l__problems_inclprob_pts_tl\ min}
8471       \addtocounter{min}{\l__problems_inclprob_min_tl}
8472     }
8473   }{
8474     \tl_if_exist:NT \l__problems_prob_min_tl {
8475       \bool_if:NT \c__problems_min_bool {
8476         \tl_if_empty:NT\l__problems_prob_min_tl{
8477           \tl_set:Nn \l__problems_prob_min_tl {0}
8478         }
8479         \marginpar{\l__problems_prob_min_tl\ min}
8480         \addtocounter{min}{\l__problems_prob_min_tl}
8481       }
8482     }
8483   }
8484 }
8485 ⟨/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.

```
8486  ⟨*package⟩
8487  \ProvidesExplPackage{hwexam}{2022/08/08}{3.2.0}{homework assignments and exams}
8488  \RequirePackage{l3keys2e}
8489
8490  \newif\iftest\testfalse
8491  \DeclareOption{test}{\testtrue}
8492  \newif\ifmultiple\multiplefalse
8493  \DeclareOption{multiple}{\multipletrue}
8494  \DeclareOption{lang}{\PassOptionsToPackage{\CurrentOption}{problem}}
8495  \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
8496  \ProcessOptions
```

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

```
8497  \RequirePackage{keyval}[1997/11/10]
8498  \RequirePackage{problem}
```

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

```
8499  \newcommand\hwexam@assignment@kw{Assignment}
8500  \newcommand\hwexam@given@kw{Given}
8501  \newcommand\hwexam@due@kw{Due}
8502  \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~blank~for~extra~space}
8503  \newcommand\hwexam@minutes@kw{minutes}
8504  \newcommand\correction@probs@kw{prob.}
8505  \newcommand\correction@pts@kw{total}
8506  \newcommand\correction@reached@kw{reached}
8507  \newcommand\correction@sum@kw{Sum}
8508  \newcommand\correction@grade@kw{grade}
8509  \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

```
8510 \AddToHook{begindocument}{
8511 \ltx@ifpackageloaded{babel}{
8512 \makeatletter
8513 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
8514 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
8515   \input{hwexam-ngerman.ldf}
8516 }
8517 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
8518   \input{hwexam-finnish.ldf}
8519 }
8520 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
8521   \input{hwexam-french.ldf}
8522 }
8523 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
8524   \input{hwexam-russian.ldf}
8525 }
8526 \makeatother
8527 }{}
8528 }
8529
```

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.

```
8530 \newcounter{assignment}
8531 %\numberproblemsin{assignment}
```

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

```
8532 \keys_define:nn { hwexam / assignment } {
8533 id   .str_set_x:N = \l_@@_assign_id_str,
8534 number   .int_set:N = \l_@@_assign_number_int,
8535 title   .tl_set:N  = \l_@@_assign_title_tl,
8536 type   .tl_set:N  = \l_@@_assign_type_tl,
8537 given .tl_set:N  = \l_@@_assign_given_tl,
8538 due .tl_set:N  = \l_@@_assign_due_tl,
8539 loadmodules .code:n  = {
8540 \bool_set_true:N \l_@@_assign_loadmodules_bool
8541 }
8542 }
8543 \cs_new_protected:Nn \_@@_assignment_args:n {
8544 \str_clear:N \l_@@_assign_id_str
8545 \int_set:Nn \l_@@_assign_number_int {-1}
8546 \tl_clear:N \l_@@_assign_title_tl
8547 \tl_clear:N \l_@@_assign_type_tl
8548 \tl_clear:N \l_@@_assign_given_tl
8549 \tl_clear:N \l_@@_assign_due_tl
8550 \bool_set_false:N \l_@@_assign_loadmodules_bool
8551 \keys_set:nn { hwexam / assignment }{ #1 }
8552 }
```

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.

```
8553  \newcommand\given@due[2]{
8554  \bool_lazy_all:nF {
8555  {\tl_if_empty_p:V \l_@@_inclassign_given_tl}
8556  {\tl_if_empty_p:V \l_@@_assign_given_tl}
8557  {\tl_if_empty_p:V \l_@@_inclassign_due_tl}
8558  {\tl_if_empty_p:V \l_@@_assign_due_tl}
8559  }{ #1 }
8560
8561  \tl_if_empty:NTF \l_@@_inclassign_given_tl {
8562  \tl_if_empty:NF \l_@@_assign_given_tl {
8563  \hwexam@given@kw\xspace\l_@@_assign_given_tl
8564  }
8565  }{
8566  \hwexam@given@kw\xspace\l_@@_inclassign_given_tl
8567  }
8568
8569  \bool_lazy_or:nnF {
8570  \bool_lazy_and_p:nn {
8571  \tl_if_empty_p:V \l_@@_inclassign_due_tl
8572  }{
8573  \tl_if_empty_p:V \l_@@_assign_due_tl
8574  }
8575  }{
8576  \bool_lazy_and_p:nn {
8577  \tl_if_empty_p:V \l_@@_inclassign_due_tl
8578  }{
8579  \tl_if_empty_p:V \l_@@_assign_due_tl
8580  }
8581  }{ ,~ }
8582
8583  \tl_if_empty:NTF \l_@@_inclassign_due_tl {
8584  \tl_if_empty:NF \l_@@_assign_due_tl {
8585  \hwexam@due@kw\xspace \l_@@_assign_due_tl
8586  }
8587  }{
8588  \hwexam@due@kw\xspace \l_@@_inclassign_due_tl
8589  }
8590
8591  \bool_lazy_all:nF {
8592  { \tl_if_empty_p:V \l_@@_inclassign_given_tl }
8593  { \tl_if_empty_p:V \l_@@_assign_given_tl }
8594  { \tl_if_empty_p:V \l_@@_inclassign_due_tl }
8595  { \tl_if_empty_p:V \l_@@_assign_due_tl }
8596  }{ #2 }
8597  }
```

\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.

```
8598  \newcommand\assignment@title[3]{
8599  \tl_if_empty:NTF \l_@@_inclassign_title_tl {
8600  \tl_if_empty:NTF \l_@@_assign_title_tl {
8601  #1
8602  }{
8603  #2\l_@@_assign_title_tl#3
8604  }
8605  }{
8606  #2\l_@@_inclassign_title_tl#3
8607  }
8608  }
```

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

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

```
8609  \newcommand\assignment@number{
8610  \int_compare:nNnTF \l_@@_inclassign_number_int = {-1} {
8611  \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8612  \arabic{assignment}
8613  } {
8614  \int_use:N \l_@@_assign_number_int
8615  }
8616  }{
8617  \int_use:N \l_@@_inclassign_number_int
8618  }
8619  }
```

(*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 (*env.*)   For the `assignment` environment we delegate the work to the `@assignment` environment that depends on whether `multiple` option is given.

```
8620  \newenvironment{assignment}[1][]{
8621  \_@@_assignment_args:n { #1 }
8622  %\sref@target
8623  \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8624  \global\stepcounter{assignment}
8625  }{
8626  \global\setcounter{assignment}{\int_use:N\l_@@_assign_number_int}
8627  }
8628  \setcounter{sproblem}{0}
8629  \renewcommand\prob@label[1]{\assignment@number.##1}
8630  \def\current@section@level{\document@hwexamtype}
8631  %\sref@label@id{\document@hwexamtype \thesection}
8632  \begin{@assignment}
8633  }{
8634  \end{@assignment}
8635  }
```

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

```
8636  \def\ass@title{
8637  {\protect\document@hwexamtype}~\arabic{assignment}
8638  \assignment@title{}{\;(}{)\;} -- \given@due{}{}
8639  }
8640  \ifmultiple
8641  \newenvironment{@assignment}{
8642  \bool_if:NTF \l_@@_assign_loadmodules_bool {
8643  \begin{sfragment}[loadmodules]{\ass@title}
8644  }{
8645  \begin{sfragment}{\ass@title}
8646  }
8647  }{
8648  \end{sfragment}
8649  }
```

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

```
8650  \else
8651  \newenvironment{@assignment}{
8652  \begin{center}\bf
8653  \Large\@title\strut\\
8654  \document@hwexamtype~\arabic{assignment}\assignment@title{\;}{:\;}{\\}
8655  \large\given@due{--\;}{\;--}
8656  \end{center}
8657  }{}
8658  \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.

```
8659  \keys_define:nn { hwexam / inclassignment } {
8660  %id   .str_set_x:N = \l_@@_assign_id_str,
8661  number   .int_set:N  = \l_@@_inclassign_number_int,
8662  title  .tl_set:N  = \l_@@_inclassign_title_tl,
8663  type  .tl_set:N  = \l_@@_inclassign_type_tl,
8664  given .tl_set:N  = \l_@@_inclassign_given_tl,
8665  due .tl_set:N  = \l_@@_inclassign_due_tl,
8666  mhrepos   .str_set_x:N = \l_@@_inclassign_mhrepos_str
8667  }
8668  \cs_new_protected:Nn \_@@_inclassignment_args:n {
8669  \int_set:Nn \l_@@_inclassign_number_int {-1}
8670  \tl_clear:N \l_@@_inclassign_title_tl
8671  \tl_clear:N \l_@@_inclassign_type_tl
8672  \tl_clear:N \l_@@_inclassign_given_tl
8673  \tl_clear:N \l_@@_inclassign_due_tl
8674  \str_clear:N \l_@@_inclassign_mhrepos_str
8675  \keys_set:nn { hwexam / inclassignment }{ #1 }
8676  }
8677  \_@@_inclassignment_args:n {}
8678
8679  \newcommand\inputassignment[2][]{
```

```
8680  \_@@_inclassignment_args:n { #1 }
8681  \str_if_empty:NTF \l_@@_inclassign_mhrepos_str {
8682  \input{#2}
8683  }{
8684  \stex_in_repository:nn{\l_@@_inclassign_mhrepos_str}{
8685  \input{\mhpath{\l_@@_inclassign_mhrepos_str}{#2}}}
8686  }
8687  }
8688  \_@@_inclassignment_args:n {}
8689  }
8690  \newcommand\includeassignment[2][]{
8691  \newpage
8692  \inputassignment[#1]{#2}
8693  }
```

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

40.4   Typesetting Exams

\quizheading

```
8694  \ExplSyntaxOff
8695  \newcommand\quizheading[1]{%
8696  \def\@tas{#1}%
8697  \large\noindent NAME: \hspace{8cm}  MAILBOX:\\[2ex]%
8698  \ifx\@tas\@empty\else%
8699  \noindent TA:~\@for\@I:=\@tas\do{{\Large$\Box$}\@I\hspace*{1em}}\\[2ex]%
8700  \fi%
8701  }
8702  \ExplSyntaxOn
```

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

\testheading

```
8703
8704  \def\hwexamheader{\input{hwexam-default.header}}
8705
8706  \def\hwexamminutes{
8707  \tl_if_empty:NTF \testheading@duration {
8708  {\testheading@min}~\hwexam@minutes@kw
8709  }{
8710  \testheading@duration
8711  }
8712  }
8713
8714  \keys_define:nn { hwexam / testheading } {
8715  min  .tl_set:N  = \testheading@min,
8716  duration .tl_set:N  = \testheading@duration,
8717  reqpts .tl_set:N  = \testheading@reqpts,
8718  tools .tl_set:N  = \testheading@tools
8719  }
8720  \cs_new_protected:Nn \_@@_testheading_args:n {
8721  \tl_clear:N \testheading@min
8722  \tl_clear:N \testheading@duration
```

```
8723    \tl_clear:N \testheading@reqpts
8724    \tl_clear:N \testheading@tools
8725    \keys_set:nn { hwexam / testheading }{ #1 }
8726  }
8727  \newenvironment{testheading}[1][]{
8728    \_@@_testheading_args:n{ #1 }
8729    \newcount\check@time\check@time=\testheading@min
8730    \advance\check@time by -\theassignment@totalmin
8731    \newif\if@bonuspoints
8732    \tl_if_empty:NTF \testheading@reqpts {
8733      \@bonuspointsfalse
8734    }{
8735      \newcount\bonus@pts
8736      \bonus@pts=\theassignment@totalpts
8737      \advance\bonus@pts by -\testheading@reqpts
8738      \edef\bonus@pts{\the\bonus@pts}
8739      \@bonuspointstrue
8740    }
8741    \edef\check@time{\the\check@time}
8742
8743    \makeatletter\hwexamheader\makeatother
8744  }{
8745    \newpage
8746  }
```

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

`\testspace`

```
8747  \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}
```

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

`\testnewpage`

```
8748  \newcommand\testnewpage{\iftest\newpage\fi}
```

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

`\testemptypage`

```
8749  \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.

```
8750  ⟨@@=problems⟩
8751  \renewcommand\@problem[3]{
8752    \stepcounter{assignment@probs}
8753    \def\__problemspts{#2}
8754    \ifx\__problemspts\@empty\else
8755    \addtocounter{assignment@totalpts}{#2}
8756    \fi
8757    \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\f
8758    \xdef\correction@probs{\correction@probs & #1}%
8759    \xdef\correction@pts{\correction@pts & #2}
8760    \xdef\correction@reached{\correction@reached &}
```

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

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

8763 `\newcounter{assignment@probs}`
8764 `\newcounter{assignment@totalpts}`
8765 `\newcounter{assignment@totalmin}`
8766 `\def\correction@probs{\correction@probs@kw}`
8767 `\def\correction@pts{\correction@pts@kw}`
8768 `\def\correction@reached{\correction@reached@kw}`
8769 `\stepcounter{assignment@probs}`
8770 `\newcommand\correction@table{`
8771 `\resizebox{\textwidth}{!}{%`
8772 `\begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%`
8773 `&\multicolumn{\theassignment@probs}{c||}%|`
8774 `{\footnotesize\correction@forgrading@kw} &\\\hline`
8775 `\correction@probs & \correction@sum@kw & \correction@grade@kw\\\hline`
8776 `\correction@pts &\theassignment@totalpts & \\\hline`
8777 `\correction@reached & & \\[.7cm]\hline`
8778 `\end{tabular}}}`
8779 ⟨/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:13

13

[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).

---

[13]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).