The sTeX3 Package Collection *

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

2022-09-13

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 detailed documentation of the sTeX package implementation.

---

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

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 toolchain of knowledge management systems.

Luckily, the sTeX-IDE will take care of much of the setup required for the full toolchain, if you are willing to use it.

2.1 Setting up the sTeX Package

2.1.1 Minimal Setup for the sTeX Package

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**. sTeX requires a TeX kernel newer than February 2022.

Alternatively, you can install sTeX from CTAN, the Comprehensive TeX Archive Network; see [ST] for details. We assume you have the sTeX package in at least version 3.2 (September 2022).

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. Make sure to either clone the sTeX repository into a local texmf-tree or to update your `TEXINPUTS` environment variable, e.g. by placing the following line in your `.bashrc`:

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

2.1.3   Setting your MathHub Directory

One of sTeX's features is a proper *module system* of interconnected document snippets for mathematical content. Analogously to *object-oriented programming*, it allows for "object-oriented mathematics" via individual combinable and, importantly, *reusable* modules, developed collaboratively.

To make use of such modules, the sTeX system needs to be told where to find them. There are several ways to do so (see subsection 5.2.1), but the most convenient way to do so is via a system variable.

To do so, create a directory `MathHub` somewhere on your local file system and set the environment variable `MATHHUB` to the file path to that directory.

In linux, you can do so by writing

```
        export MATHHUB="/path/to/your/MathHub"
```

in your `~/.profile` (for all shells) or `~/.bashrc` (for the bash terminal only) file.

2.2   Setting up the sTeX IDE

The sTeX IDE consists of two components using the *Language Server Protocol (LSP)*: A *client* in the form of a VSCode extension, and a *server* included in the Mmt system. Installing the extension will open up a setup routine that will guide you through the rest.

2.2.1   The sTeX VSCode Extension

If you have not already, you should first install the VSCode editor available at https://code.visualstudio.com/.

Next, open VSCode and install the sTeX extension by clicking on the *extensions* menu on the very left of the VSCode window and searching for "sTeX" in the *"Search Extensions in Marketplace"* field, as in Figure 1, and clicking the *Install*-button of the sTeX extension by KWARC.

2.2.2   Setting up Mmt

Next, open any directory (`File → Open Folder...`) that contains a `.tex`-file, and a setup window as in Figure 2 will pop up. Clik on the highlighted link '*here*' and download the latest version of the `MMT.jar` file (at least version 23.0.0) anywhere you like. Then click the *"Browse..."*-button and select your freshly downloaded `MMT.jar`.

If you have already set a system variable for your MathHub-directory, you are now done and can click *"Finish"*. If you have not, you can now also enter a directory path in the lower text field, and the VSCode extension will attempt to globally set one up for you, depending on your operating system.

Once you click *"Finish"*, the client will connect to https://stexmmt.mathhub.info/:sTeX, query for available archives, download the core libraries required for all (or most) semantic services (`MMT/urtheories` and `sTeX/meta-inf`) and set up RusTeX for you automatically.
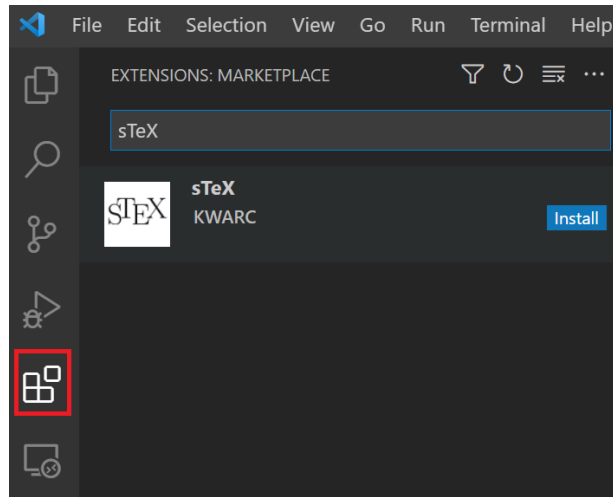
Figure 1: Installing the sTeX extension for VSCode
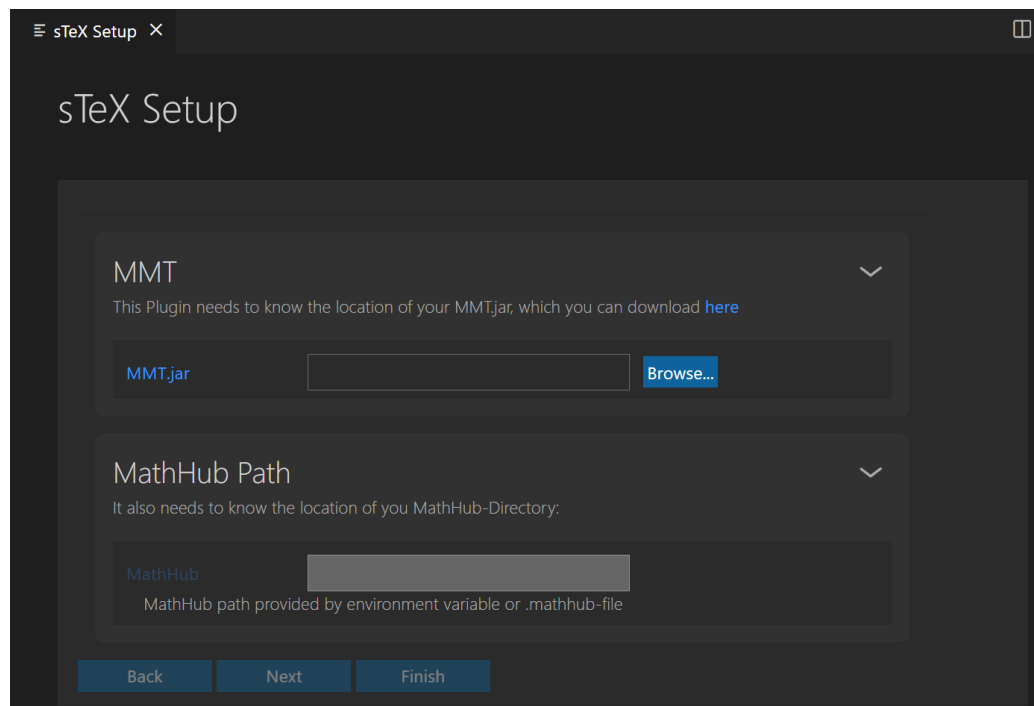
Figure 2: sTeX Setup Routine

2.3 Manual Setup

In lieu of using the sTeX IDE, we can do the following:

2.3.1 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 5.2).

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

2.3.2 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.

Chapter 3

The sTeX IDE

Chapter 4

A First sTeX Document

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

```
 1 \documentclass{article}
 2 \usepackage{stex,xcolor,stexthm}
 3
 4 \begin{document}
 5 \begin{smodule}{GeometricSeries}
 6     \importmodule[smglom/calculus]{series}
 7     \importmodule[smglom/arithmetics]{realarith}
 8
 9     \symdef{geometricSeries}[name=geometric-series]{\comp{S}}
10
11     \begin{sdefinition}[for=geometricSeries]
12         The \definame{geometricSeries} is the \symname{series}
13         \[\defeq{\geometricSeries}{\definiens{
14             \infinitesum{\svar{n}}{1}{
15                 \realdivide[frac]{1}{
16                     \realpower{2}{\svar{n}}
17             }}
18         }}.\]
19     \end{sdefinition}
20
21     \begin{sassertion}[name=geometricSeriesConverges,type=theorem]
22     The \symname{geometricSeries} \symname{converges} towards $1$.
23     \end{sassertion}
24 \end{smodule}
25 \end{document}
```

Compiling this document with `pdflatex` should yield the output

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

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

8

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 4.0.1:

> 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 section 7.3.

Let's investigate this document in detail to understand the respective parts of the SIEX 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 SIEX archive `smglom/calculus`, and `realarith` from the SIEX 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 SIEX 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.

9

\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.

4.1 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">
```

---

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

```
        <mi resource="1" property="stex:arg">2</mi>
        <mrow resource="2" property="stex:arg">
         <mi resource="var://n" property="stex:OMV">n</mi>
        </mrow>
      </msup>
     </mrow>
    </mfrac>
   </mrow>
  </mrow>
</mrow>
```

...containing all the semantic information. The MMT system can extract from this the following OPENMATH snippet:

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

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

> **Remark 4.1.1:**
>
> > 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.

4.2   Mmt/OMDoc Conversion

Another way to convert our document to *actual* MMT/OMDoc is to put it in an SₜᴇX **archive** (see section 5.2) and have MMT take care of everything.

Assuming the above file is `source/demo.tex` in an SₜᴇX 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 5

Creating sTEX Content

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

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

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

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

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

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

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

debug ($\langle log\text{-}prefix \rangle *$) Logs debugging information with the given prefixes to the terminal, or all if `all` is given. Largely irrelevant for the majority of users.

5.1 How Knowledge is Organized in sTEX

sTEX content is organized on multiple levels:

1. sTEX **archives** (see section 5.2) contain individual `.tex`-files.

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

3. Modules contain ſTEX **symbol declarations**, introduced via `\symdecl{symbolname}`, `\symdef{symbolname}` and some other constructions. Most symbols have a *notation* that can be used via a *semantic macro* `\symbolname` generated by symbol declarations.

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

⟵M⟶
⟶M⟶
∼T↝

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

5.2 ſTEX Archives

5.2.1 The Local MathHub-Directory

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

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

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

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

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

4. Finally, if all else fails, ſTEX will look for a file `~/.stex/mathhub.path`. If this file exists, ſTEX will assume that it contains the path to the local `MathHub`-directory. This method is recommended on systems where it is difficult to set environment variables.

5.2.2 The Structure of � Archives

An ꟍTEX 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 ꟍTEX 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 ꟍTEX 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 ꟍTEX archive:

- `/source/mod/` – individual ꟍTEX modules, containing symbol declarations, notations, and `\begin{sparagraph}[type=symdoc,for=...]` environments for "encyclopaedic" symbol documentations

- `/source/def/` – definitions

- `/source/ex/` – examples

- `/source/thm/` – theorems, lemmata and proofs; preferably proofs in separate files to allow for multiple proofs for the same statement

- `/source/snip/` – individual text snippets such as remarks, explanations etc.

- `/source/frag/` – individual document fragments, ideally only `\inputref`ing snippets, definitions, examples etc. in some desirable order

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

- `/source/PIC/` – image files.

5.2.3 MANIFEST.MF-Files

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

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

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

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

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

source-base or

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

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

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

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

5.2.4 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.

17

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

- the `lib`-directory of the current archive, and

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

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

`\libinput` will throw an error if *no* candidate for `some/file` is found.

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

`\libusepackage` throws an error if not *exactly one* candidate for `some/file` is found.

> **Remark 5.2.1:**
>
> A good practice is to have individual $\mathcal{S}T_{E}X$ 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.

5.3 Module, Symbol and Notation Declarations

5.3.1 The `smodule`-Environment

`smodule` (*env.*) A new module is declared using the basic syntax

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

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.

18

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

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

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

Example 1
Input:

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

Output:

```
Hello World
```

.

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

For example:

Example 2
Input:

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

Output:

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

.

5.3.2 Declaring New Symbols and Notations

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

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

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

> ↪M→ `\symdecl` introduces a new OMDoc/Mᴍᴛ constant in the current mod-
> —M→ ule (=OMDoc/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 SₜᴇX 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→ `\symbolname!` is translated to OMDoc/Mmt as `<OMS name="...?symbolname"/>`
> —M→ directly.
> ⤳T↝

5.3.3 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:

$\forall x,y,z.P$

.

5.3.4 Type and Definiens Components

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

> ↪M➔ The type and def keys correspond to the type and **definiens** components of OMDoc/Mmt constants.
> —M➔ Correspondingly, the name "type" should be taken with a grain of salt, since
> ⤳T⤳ OMDoc/Mmt– being foundation-independent – does not a priori implement a fixed typing system.

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

Example 13
Input:

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

Output:

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

.

The def-key allows for declaring symbols as abbreviations:

Example 14

Input:

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

Output:

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

.

5.3.5 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:

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

.

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

Example 17
Input:

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

Output:

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

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

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

We will investigate the precise meaning of `<opprec>` and the `<argprec>`s shortly – in the vast majority of cases, it is 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

29

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.

5.3.6 Variables

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

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

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

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

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

> **Example 19**
> Input:
>
> ```
>  1 \vardef{varf}[
>  2     name=f,
>  3     type=\funtype{\Nat}{\Nat},
>  4     op=f,
>  5     args=1,
>  6     prec=0;\neginfprec
>  7 ]{\comp{f}#1}
>  8 \vardef{varn}[name=n,type=\Nat]{\comp{n}}
>  9 \vardef{varx}[name=x,type=\Nat]{\comp{x}}
> 10
> 11 Given a function $\varf!:\funtype{\Nat}{\Nat}$,
> 12 by $\addition{\varf!,\varn}$ we mean the function\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

5.3.7 Variable Sequences

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

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

32

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:

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

.

5.4   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.

5.4.1   Multilinguality and Translations

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

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

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

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

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

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

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

```
1 \begin{smodule}[sig=en]{lcm}
2   \notation*{lcm}[de]{\comp{\mathtt{kgV}}}(#1,#2)}
3
4   Das \symref{lcm}{kleinste gemeinsame Vielfache}
5   $\lcm{a,b}$ von zwei Zahlen $a,b$ ist...
6 \end{smodule}
```

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

5.4.2 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 Mᴍᴛ 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

34

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 (S$\TeX$) 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 T$_E$Xin the L^{A}T$_E$X3-category code scheme. This means that the characters `_` and `:` are considered *letters* and valid parts of control sequence names, and space characters are

ignored entirely. For spaces, use the character ~ instead, and keep in mind, that if you want to use subscripts, you should use `\c_math_subscript_token` instead of _!

Also note, that **\newcommand** defines macros *globally* and throws an error if the macro already exists, potentially leading to low-level LaTeX errors if we put a **\newcommand** in an `\STEXexport` and the `<code>` is executed more than once in a document – which can happen easily.

A safer alternative is to use macro definition principles, that are safe to use even if the macro being defined already exists, and ideally are local to the current TeX group, such as **\def** or **\let**.

5.4.3 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 $\circ : U \times U \rightarrow U$ ...

.

and

Example 28

Input:

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

Output:

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

.

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

usestructure (*env.*) The usestructure{<struct>} environment is used in multilingual settings as a parallel to the mathstructure. It opens a group and then issues a \usemodule{.../<struct>-structure} that gives the body access to all the semantic macros in the referenced structure.

5.4.4 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: $a \cdot (c + d \cdot e)$

.

5.4.5   The `interpretmodule` Environment

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:

.

5.5 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 6

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?

6.1 \symref and its variants

`\symref`
`\symname`

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

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

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

Example 32

Input:

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

Output:

A natural number is...

.

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

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

Example 33
Input:

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

Output:

> Natural numbers are...

.

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

6.2 Marking Up Text and On-the-Fly Notations

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

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

Example 34
Input:

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

Output:

> The sum of n and m is...

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

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

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

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

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

Example 35
Input:

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

Output:

```
  Addition is...
```

.

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

`\arg` also allows us to switch the order of arguments around and "hide" arguments: For example, `\arg[3]{<code>}` signifies that `<code>` represents the *third* argument to the current operator, and `\arg*[i]{<code>}` signifies that `<code>` represents the ith argument, but it should not produce any output (it is exported in the `xhtml` however, so that Mmt and other systems can pick up on it).[1]

EdN:1

Example 36
Input:

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

---

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

44

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 9.2.1) 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 ꟫TEX3 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 align="center" style="color:red">Part III</div>

will say "Part III" (and link accordingly) in the full documentation, and "Part III (Extensions) in the full ꟫TEX3 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

sTEX Statements

7.1 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 section 7.3 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 chapter 8), type= for customization (see section 7.3) 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 7.1.1** (An Example)**.** 2+3 is 5, 2·3 is 6.

.

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

> ←M→
> —M→
> ⤳T⤳
> 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.

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

All four statement environments – i.e. sdefinition, sassertion, sexample, and sparagraph – also take an optional parameter `name=` – if this one is given a value, the environment will generate a *symbol* by that name (but with no semantic macro). Not only does this allow for `\symref` et al, it allows us to resume our earlier example for monoids much more nicely:[3]

> **Example 39**
> Input:

EdN:3

[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 7.1.2 (Associativity). $\circ$ *is associative*

Axiom 7.1.3 (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]

7.2 Proofs

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

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 Mᴍᴛ), 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, SₜₑX can not check that the assertions are the "correct" ones – but if the assertions (both in `monoid` as well as those for addition and zero) are properly marked up, Mᴍᴛ 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 7.2.1. $\sqrt{2}$ *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
>
> $\Rightarrow$ 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
>
> $\Rightarrow$ Hence b is even
>
> *By the same argument as above*
>
> $\Rightarrow$ Contradiction to a, b being coprime.
>
> $\square$

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

> **Theorem 7.2.2.** $\sqrt{2}$ *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
>
> $\Rightarrow$ 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{}`.

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

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

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

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

```
1 \stexpatchassertion[theorem]
2   {\ifx\sassertiontitle\@empty
3       \begin{theorem}
4   \else
5       \begin{theorem}[\sassertiontitle]
6   \fi}
7 {\end{theorem}}
```

Or, if we want *all kinds of* sdefinitions to use a predefined definition-environment irrespective of their `type=`, then we can issue the following customization patch:

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

\compemph
\varemph
\symrefemph
\defemph

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

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

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

For each of the four macros, there exists an additional macro that takes the full URI of the relevant symbol currently being highlighted as a second argument. That allows us to e.g. use pdf tooltips and links. For example, this document uses[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 8

Cross References

If we take features like `\inputref` and `\mhinput` (and the `sfragment`-environment, see subsection 9.2.1) 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`).

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 9

Additional Packages

9.1 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
\cmhtikzinput \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.

9.2 Modular Document Structuring

9.2.1 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.

9.2.2 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

9.2.3 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.

9.2.4 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.

9.2.5 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.

9.3 Slides and Course Notes

9.3.1 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.

9.3.2 Package Options

The notesslides class takes a variety of class options:

<u>slides</u>
<u>notes</u> The options `slides` and `notes` switch between slides mode and notes mode (see subsection 9.3.3).

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

<u>frameimages</u>
<u>fiboxed</u> 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.

9.3.3 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  ...
```

```
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.*)

9.3.4 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
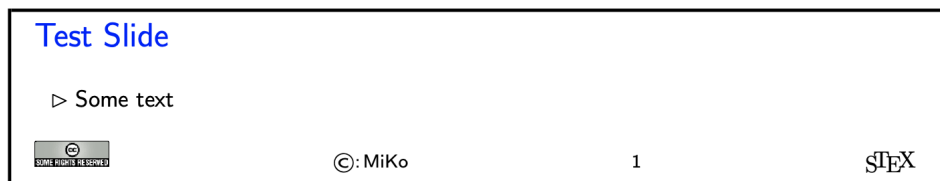
The footer line can be customized. In particular the logos.

\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.

9.3.5 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: ⚠

9.3.6 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.

9.4 Representing Problems and Solutions

9.4.1 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.

9.4.2 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 problempackage 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 9.4.1 (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.

9.4.3 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=Nooooooooo,feedback=that is for Java]{public static void}
 9   \end{mcb}
10 \end{sproblem}
```

Output:

Problem 9.4.2 (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`)

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 9.4.3 (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

The `\fillinsol` macro takes[6] an a single 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 9.4.4 (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 9.4.5 (Fitting Elefants)
How many Elefants can you fit into a Volkswagen beetle?   !

˙If we do not want to leak information about the solution by the size of the blank we can also give `\fillinsol` an optional argument with a size: `\fillinsol[3cm]{12}` makes a box three cm wide.

Obviously, the required 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]

9.4.4   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

---

[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.

screen after each run. This is useful in preparing exams, where we want to make sure that the students can indeed solve the problems in an allotted time period.

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

9.4.5 Testing and Spacing

The problem package is often used by the hwexam package, which is used to create homework assignments and exams. Both of these have a "test mode" (invoked by the package option test), where certain information –master solutions or feedback – is not shown in the presentation.

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

\testspace \testspace takes an argument that expands to a dimension, and leaves vertical space accordingly. Specific instances exist: \testsmallspace, \testsmallspace, \testsmallspace give small (1cm), medium (2cm), and big (3cm) vertical space.

\testnewpage \testnewpage makes a new page in test mode, and \testemptypage generates an empty page with the cautionary message that this page was intentionally left empty.

</div>

(margin notes: \testspace, \testsmallspace, \testsmallspace, \testsmallspace, \testnewpage, \testemptypage)

9.5 Homeworks, Quizzes and Exams

9.5.1 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.

9.5.2 Package Options

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

(margin note: 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.).

(margin note: 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.

9.5.3 Assignments

<div style="float:left">assignment (*env.*)
number
title
type
given
due</div>

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

9.5.4 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.

9.5.5 Typesetting Exams

<div style="float:left">testheading (*env.*)
duration
min
reqpts</div>

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

```
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)

2022-09-13

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.	9.4.1	9.4.2	9.4.3	9.4.4	9.4.5	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

Part II
Documentation

Chapter 10

sTeX-Basics

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

10.1 Macros and Environments

\sTeX	Both print this sTeX logo.
\stex	

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

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

10.1.1 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

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

`\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⟩}`.

10.1.2   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`.

10.1.3   Auxiliary Methods

`\stex_deactivate_macro:Nn`	`\stex_deactivate_macro:Nn⟨cs⟩{⟨environments⟩}`
`\stex_reactivate_macro:N`	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 11

sTEX-MathHub

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

11.1 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.

11.1.1 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` The inverse; turns a path into a string and stores it in the second argument variable, or
`\stex_path_to_string:N` 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` Store the current working directory as path-sequence and string, respectively, and the
`\c_stex_pwd_str` (heuristically guessed) full path to the main file, based on the PWD and `\jobname`.
`\c_stex_mainfile_seq`
`\c_stex_mainfile_str`

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

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

11.1.2 MathHub Archives

`\mathhub`
`\c_stex_mathhub_seq` We determine the path to the local MathHub folder via one of four means, in order of
`\c_stex_mathhub_str` 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*⟩}.

11.1.3 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 12

sTEX-References

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

12.1   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`

12.1.1   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*⟩.

12.1.2 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 13

sTeX-Modules

This sub package contains code related to Modules

13.1 Macros and Environments

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

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

name The *name* of the module,

ns the *namespace* in field `ns`,

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

lang the module's *language*,

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

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

meta the metatheory of the module.

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

`\c_stex_module_<URI>_constants`

The names of all constants declared in the module

`\c_stex_module_<URI>_constants`

The full URIs of all modules imported in this module

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

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

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

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

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

`\stex_add_to_current_module:n`
`\STEXexport`

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

`\stex_add_constant_to_current_module:n`

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

`\stex_add_import_to_current_module:n`

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

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

`\stex_do_up_to_module:n` Code that is *exported* from module (such as symbol declarations) should be local *to the current module*. For that reason, ideally all symbol declarations and similar commands should be called directly in the module environment, however, that is not always feasible, e.g. in structural features or `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`.

13.1.1 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 14

sTEX-Module Inheritance

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

14.1 Macros and Environments

14.1.1 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.

14.1.2 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 15

sTeX-Symbols

Code related to symbol declarations and notations

15.1 Macros and Environments

$\underline{\texttt{\symdecl}}$ `\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.

- `args`: Specifies the "signature" of the semantic macro. Can be either an integer $0 \leq n \leq 9$, or a (more precise) sequence of the following characters:

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

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

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

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

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

- name (string),

- module (string),

- notations (sequence of strings; initially empty),

- 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 16

sTEX-Terms

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

16.1 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 Maximal and minimal notation precedences.
\neginfprec

\dobrackets \dobrackets {⟨*body*⟩}

Puts ⟨*body*⟩ in parentheses; scaled if in display mode unscaled otherwise. Uses the current SIₑX 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 SIₑX 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 \comp{⟨args⟩}
\compemph
\compemph@uri Marks ⟨*args*⟩ as a notation component of the current symbol for highlighting, linking,
\defemph etc.
\defemph@uri   The precise behavior is governed by \@comp, which takes as additional argument the
\symrefemph URI of the current symbol. By default, \@comp adds the URI as a PDF tooltip and colors
\symrefemph@uri the highlighted part in blue.
\varemph   \@defemph behaves like \@comp, and can be similarly redefined, but marks an ex-
\varemph@uri pression 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 17

sTEX-Structural Features

Code related to structural features

17.1 Macros and Environments

17.1.1 Structures

`mathstructure` (*env.*) TODO

Chapter 18

sTeX-Statements

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

18.1 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 19

sTeX-Proofs: Structural Markup for Proofs

Chapter 20

sTeX-Metatheory

20.1 Symbols

Part III

Extensions

Chapter 21

Tikzinput: Treating TIKZ code as images

21.1 Macros and Environments

Chapter 22

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

Chapter 23

NotesSlides – Slides and Course Notes

Chapter 24

`problem.sty`: An Infrastructure for formatting Problems

Chapter 25

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

Part IV

Implementation

Chapter 26

sTEX -Basics Implementation

26.1 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⟩
```

26.2   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* 76.)

26.3   Messages and logging

```
72 ⟨@@=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 76.*)

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}
```

26.4   HTML Annotations

```
107 ⟨@@=stex_annotate⟩
```

\l_stex_html_arg_tl   Used by annotation macros to ensure that the HTML output to annotate is not empty.
\c_stex_html_emptyarg_tl
```
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* *76*.)

`\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* *76*.)

`stex\annotate enve:nnn)`
`\stex_annotate_invisible:n`
`\stex_annotate_invisible:nnn`

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

The pdflatex-macros largely do nothing; the RᴜꜱTᴇX-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 77.*)

26.5   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 77.*)

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 {
```

109

```
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
```

26.6   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 }
```

26.7   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 77.*)

\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 77.*)

\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   }}
```

```
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 77.*)

\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 27

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 }
```

27.1 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 78.*)

`\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 78.*)

`\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 78.)

```
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 78.)

27.2 PWD and kpsewhich

```
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 78.*)

We determine the PWD

```
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 78.*)

27.3   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`*.*)

```
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 78.*)

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

(*End definition for* `\g_stex_currentfile_seq`*. This variable is documented on page 79.*)

```
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     }
```

118

```
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 79.*)

\stex_filestack_pop:

```
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 79.*)

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 }
```

27.4   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 79.*)

__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
```

(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 {
```

```
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* *79.*)

<code>\stex_require_repository:n</code>

```
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* *79.*)

<code>\l_stex_current_repository_prop</code> 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* *79.*)

<code>\stex_in_repository:nn</code> 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    }{
```

123

```
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* *79.)*

27.5   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* *80.)*

`\inputref`
`\mhinput`

```
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 }
```

124

```
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 80.*)

\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* *80.*)

\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* *80.*)

\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    }
```

126

```
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 80.*)

<div align="right">

`\mhgraphics`
`\cmhgraphics`
</div>

```
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 80.*)

<div align="right">

`\lstinputmhlisting`
`\clstinputmhlisting`
</div>

```
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  }
```

127

933
934 ⟨/package⟩

(*End definition for* `\lstinputmhlisting` *and* `\clstinputmhlisting`. *These functions are documented on page 80.*)

Chapter 28

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 }
```

28.1 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 81.*)

\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 81.)

\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 81.)

\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* *81.*)

28.2   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
```

131

Restoring references from `.sref`-files

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

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

```
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     \stex_debug:nn{sref}{New~label~\l__stex_refs_curr_label_str~at~\use:c{\use:c{@currentcou
1094     \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
1095     \immediate\write\@auxout{\STEXInternalAuxAddDocRef{\l_stex_current_docns_str}{\l_tmpa_st
1096     %\str_gset:cx {sref_\l__stex_refs_curr_label_str _type}\c__stex_refs_ref_str
1097   }
1098 }
1099 \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 81.)*

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

```
1100 \cs_new_protected:Npn \STEXInternalAuxAddDocRef #1 #2 {
1101   \exp_args:NNx \seq_if_in:NnTF \g_stex_ref_files_seq {\detokenize{#1}} {
1102     \exp_args:Nnx \seq_if_in:cnF{g_stex_ref_ #1 _seq}{\detokenize{#2}}{
1103       \exp_args:Nnx \seq_gput_left:cn{g_stex_ref_ #1 _seq}{\detokenize{#2}}
1104     }
1105   }{
1106       \exp_args:NNx \seq_gput_right:Nn \g_stex_ref_files_seq {\detokenize{#1}}
1107       %\seq_if_exist:cF{g_stex_ref_ #1 _seq}{
1108         \seq_new:c{g_stex_ref_ #1 _seq} % <- seq_new throws errors??
1109       %}
1110       \exp_args:Nnx \seq_gput_left:cn{g_stex_ref_ #1 _seq}{\detokenize{#2}}
1111   }
1112
1113   %\str_set:Nn \l_tmpa_str {#1?#2}
1114   %\str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
1115   %\seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
1116   %  \seq_new:c {g__stex_refs_labels_#2_seq}
1117   %}
1118   %\seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
1119   %  \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
1120   %}
1121 }
```

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

```
1122 \AtEndDocument{
1123   \def\STEXInternalAuxAddDocRef#1 #2 {}{}
1124 }
```

`\stex_ref_new_sym_target:n`

```
1125 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
1126
1127 %  \stex_if_smsmode:TF {
1128 %    \str_if_exist:cF{sref_sym_#1_type}{
1129 %      \stex_get_document_url:
1130 %      \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str
1131 %      \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
1132 %    }
1133 %  }{
1134 %    \str_if_empty:NF \l__stex_refs_curr_label_str {
1135 %      \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
1136 %      \immediate\write\@auxout{
1137 %        \exp_not:N\expandafter\def\exp_not:N\csname \exp_not:N\detokenize{sref_sym_#1_label
1138 %          \l__stex_refs_curr_label_str
1139 %        }
1140 %      }
1141 %    }
1142 %  }
1143 }
```

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

28.3   Using References

`\sref`   Optional arguments:

```
1144
1145 \keys_define:nn { stex / sref / 1 } {
1146   archive    .str_set_x:N  = \l__stex_refs_repo_str,
1147   file       .str_set_x:N  = \l__stex_refs_file_str,
1148   % TODO get rid of this
1149   fallback   .code:n = {},
1150   pre        .code:n = {},
1151   post       .code:n = {}
1152 }
1153 \cs_new_protected:Nn \__stex_refs_args_i:n {
1154   \str_clear:N \l__stex_refs_repo_str
1155   \str_clear:N \l__stex_refs_file_str
1156   \keys_set:nn { stex / sref / 1 } { #1 }
1157 }
1158 \keys_define:nn { stex / sref / 2 } {
1159   in       .str_set_x:N  = \l__stex_refs_in_str,
1160   archive    .str_set_x:N  = \l__stex_refs_repob_str,
1161   title     .tl_set:N  = \l__stex_refs_title_tl
1162 }
1163 \cs_new_protected:Nn \__stex_refs_args_ii:n {
1164   \str_clear:N \l__stex_refs_in_str
1165   \tl_clear:N \l__stex_refs_title_tl
1166   \str_clear:N \l__stex_refs_repob_str
1167   \keys_set:nn { stex / sref / 2 } { #1 }
1168 }
```

The actual macro:

```
1169 \NewDocumentCommand \sref { O{} m O{}}{
1170   \__stex_refs_args_i:n{#1}
1171   \__stex_refs_args_ii:n{#3}
1172   \str_clear:N \l__stex_refs_uri_str
1173   \__stex_refs_find_uri:n{#2}
1174   \__stex_refs_do_sref:n{#2}
1175 }
1176 \NewDocumentCommand \extref { O{} m m}{
1177   \__stex_refs_args_i:n{#1}
1178   \__stex_refs_args_ii:n{#3}
1179   \str_if_empty:NT \l__stex_refs_in_str {
1180     \msg_error:nn{stex}{error/extrefmissing}
1181   }
1182   \str_clear:N \l__stex_refs_uri_str
1183   \__stex_refs_find_uri:n{#2}
1184   \__stex_refs_do_sref_in:n{#2}
1185 }
1186
1187 \cs_new_protected:Nn \__stex_refs_find_uri:n {
1188   \stex_debug:nn{sref}{File:~\l__stex_refs_file_str^^JRepo:\l__stex_refs_repo_str}
1189   \str_if_empty:NTF \l__stex_refs_file_str {
1190     \stex_debug:nn{sref}{Empty.~Checking~current~file~for~#1}
1191     \seq_if_exist:cT{g_stex_ref_\l_stex_current_docns_str _seq}{
1192       \seq_map_inline:cn{g_stex_ref_\l_stex_current_docns_str _seq}{
```

```
1193        \str_if_eq:nnT{#1}{##1}{
1194          \str_set_eq:NN \l__stex_refs_uri_str \l_stex_current_docns_str
1195          \stex_debug:nn{sref}{Found.}
1196          \seq_map_break:
1197        }
1198      }
1199    }
1200    \str_if_empty:NT \l__stex_refs_uri_str {
1201      \stex_debug:nn{sref}{Checking~other~files}
1202      \seq_map_inline:Nn \g_stex_ref_files_seq {
1203        \stex_debug:nn{sref}{##1...}
1204        \seq_map_inline:cn{g_stex_ref_##1_seq}{
1205          \str_if_eq:nnT{#1}{####1}{
1206            \stex_debug:nn{sref}{Found~##1}
1207            \str_set:Nn \l__stex_refs_uri_str {##1}
1208            \seq_map_break:n{\seq_map_break:}
1209          }
1210        }
1211      }
1212    }
1213  }{
1214    \str_if_empty:NTF \l__stex_refs_repo_str {
1215      \prop_if_exist:NTF \l_stex_current_repository_prop {
1216        \prop_get:NnN \l_stex_current_repository_prop { ns } \l__stex_refs_uri_str
1217        \str_set:Nx \l__stex_refs_uri_str {\l__stex_refs_uri_str / \l__stex_refs_file_str}
1218        \stex_path_from_string:Nn \l_tmpb_seq \l__stex_refs_uri_str
1219        \str_set:Nx \l__stex_refs_uri_str {\stex_path_to_string:N \l_tmpb_seq}
1220      }{
1221        \stex_path_from_string:Nn \l_tmpb_seq {
1222          \stex_path_to_string:N \g_stex_currentfile_seq/ .. / \l__stex_refs_file_str
1223        }
1224        \str_set:Nx \l__stex_refs_uri_str {file:/\stex_path_to_string:N \l_tmpb_seq}
1225      }
1226    }{
1227      \stex_require_repository:n \l__stex_refs_repo_str
1228      \prop_get:cnN { c_stex_mathhub_\l__stex_refs_repo_str _manifest_prop } { ns } \l__stex
1229      \str_set:Nx \l__stex_refs_uri_str {\l__stex_refs_uri_str / \l__stex_refs_file_str}
1230      \stex_path_from_string:Nn \l_tmpb_seq \l__stex_refs_uri_str
1231      \str_set:Nx \l__stex_refs_uri_str {\stex_path_to_string:N \l_tmpb_seq}
1232    }
1233  }
1234 }
1235
1236 \cs_new_protected:Nn \__stex_refs_do_autoref:n{
1237   \cs_if_exist:cTF{autoref}{
1238     \exp_args:Nx\autoref{sref_#1}
1239   }{
1240     \exp_args:Nx\ref{sref_#1}
1241   }
1242 }
1243
1244 \cs_new_protected:Nn \__stex_refs_do_sref:n {
1245   \str_if_empty:NTF \l__stex_refs_uri_str {
1246     \str_if_empty:NTF \l__stex_refs_in_str {
```

```
1247      \stex_debug:nn{sref}{autoref~on~#1}
1248      \__stex_refs_do_autoref:n{#1}
1249    }{
1250      \stex_debug:nn{sref}{srefin~on~#1}
1251      \__stex_refs_do_sref_in:n{#1}
1252    }
1253  }{
1254    \exp_args:NNo \seq_if_in:NnTF \g_stex_ref_files_seq \l__stex_refs_uri_str {
1255      \exp_args:Nnx \seq_if_in:cnTF{g_stex_ref_\l__stex_refs_uri_str _seq}{\detokenize{#1}}{
1256        \stex_debug:nn{sref}{Reference~found~in~ref~files;~autoref~on~#1}
1257        \__stex_refs_do_autoref:n{\l__stex_refs_uri_str?#1}
1258      }{
1259        \str_if_empty:NTF \l__stex_refs_in_str {
1260          \stex_debug:nn{sref}{in~empty;~autoref~on~#1}
1261          \__stex_refs_do_autoref:n{#1}
1262        }{
1263          \stex_debug:nn{sref}{in~non-empty;~srefin~on~#1}
1264          \__stex_refs_do_sref_in:n{#1}
1265        }
1266      }
1267    }{
1268      \str_if_empty:NTF \l__stex_refs_in_str {
1269        \stex_debug:nn{sref}{in~empty;~autoref~on~#1}
1270        \__stex_refs_do_autoref:n{#1}
1271      }{
1272        \stex_debug:nn{sref}{in~non-empty;~srefin~on~#1}
1273        \__stex_refs_do_sref_in:n{#1}
1274      }
1275    }
1276  }
1277 }
1278
1279 \cs_new_protected:Nn \__stex_refs_restore_target:nnnnn {
1280   \str_if_empty:NTF \l__stex_refs_uri_str {
1281     \exp_args:No \str_if_eq:nnT \l__stex_refs_id_str {#2}{
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     }
1289   }{
1290     \stex_debug:nn{sref}{\l__stex_refs_uri_str{}~ == ~ #1 ~ ?}
1291     \exp_args:No \str_if_eq:nnT \l__stex_refs_uri_str {#1}{
1292       \stex_debug:nn{sref}{\l__stex_refs_id_str~ == ~ #2 ~ ?}
1293       \exp_args:No \str_if_eq:nnT \l__stex_refs_id_str {#2}{
1294         \stex_debug:nn{sref}{success!}
1295         \tl_set:Nn \l__stex_refs_return_tl {
1296           \use:c{#3autorefname}~#4\tl_if_empty:nF{#5}{~(#5)}~in~
1297           \tl_if_empty:nTF\l__stex_refs_title_tl{
1298             ???
1299           }\l__stex_refs_title_tl
1300         }
```

```
1301        \endinput
1302      }
1303    }
1304  }
1305 }
1306
1307 \cs_new_protected:Nn \__stex_refs_do_sref_in:n {
1308   \stex_debug:nn{sref}{In: \l__stex_refs_in_str^^JRepo:\l__stex_refs_repo_str}
1309   \stex_debug:nn{sref}{URI: \l__stex_refs_uri_str?#1}
1310   %\msg_warning:nnn{stex}{warning/smsmissing}{<filename>}
1311   \begingroup\catcode13=9\relax\catcode10=9\relax
1312     \str_if_empty:NTF \l__stex_refs_repob_str {
1313       \prop_if_exist:NTF \l_stex_current_repository_prop {
1314         \str_set:Nx \l_tmpa_str {
1315           \c_stex_mathhub_str /
1316           \prop_item:Nn \l_stex_current_repository_prop { id }
1317           / source / \l__stex_refs_in_str .sref
1318         }
1319       }{
1320         \str_set:Nx \l_tmpa_str {
1321           \stex_path_to_string:N \g_stex_currentfile_seq/ .. / \l__stex_refs_in_str . sref
1322         }
1323       }
1324     }{
1325       \str_set:Nx \l_tmpa_str {
1326         \c_stex_mathhub_str / \l__stex_refs_repob_str
1327         / source / \l__stex_refs_in_str . sref
1328       }
1329     }
1330     \stex_path_from_string:Nn \l_tmpb_seq \l_tmpa_str
1331     \stex_path_to_string:NN \l_tmpb_seq \l_tmpa_str
1332     \stex_debug:nn{sref}{File: \l_tmpa_str}
1333     \exp_args:No \IfFileExists \l_tmpa_str {
1334       \tl_clear:N \l__stex_refs_return_tl
1335       \str_set:Nn \l__stex_refs_id_str {#1}
1336       \let\STEXInternalSrefRestoreTarget\__stex_refs_restore_target:nnnnn
1337       \use:c{@ @ input}{\l_tmpa_str}
1338       \exp_args:No \tl_if_empty:nTF \l__stex_refs_return_tl {
1339         \exp_args:Nnno \msg_warning:nnnn{stex}{warning/smslabelmissing}\l_tmpa_str{#1}
1340         \__stex_refs_do_autoref:n{
1341           \str_if_empty:NF\l__stex_refs_uri_str{\l__stex_refs_uri_str?}#1
1342         }
1343       }{
1344         \l__stex_refs_return_tl
1345       }
1346     }{
1347       \exp_args:Nnno \msg_warning:nnn{stex}{warning/smsmissing}\l_tmpa_str
1348       \__stex_refs_do_autoref:n{
1349         \str_if_empty:NF\l__stex_refs_uri_str{\l__stex_refs_uri_str?}#1
1350       }
1351     }
1352   \endgroup
1353 }
1354
```

```
1355 % \__stex_refs_args:n { #1 }
1356 % \str_if_empty:NTF \l__stex_refs_indocument_str {
1357 %   \str_set:Nx \l_tmpa_str { #2 }
1358 %   \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
1359 %   \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
1360 %     \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
1361 %       \seq_get_left:cNF {g__stex_refs_labels_\l_tmpa_str _seq} \l_tmpa_str {
1362 %         \str_clear:N \l_tmpa_str
1363 %       }
1364 %     }{
1365 %       \str_clear:N \l_tmpa_str
1366 %     }
1367 %   }{
1368 %     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1369 %     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1370 %    \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
1371 %     \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
1372 %       \str_set_eq:NN \l_tmpc_str \l_tmpa_str
1373 %       \str_clear:N \l_tmpa_str
1374 %       \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str _seq} {
1375 %         \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
1376 %           \str_range:nnn { ##1 }{ -\l_tmpa_int}{ -1 }
1377 %         }{
1378 %           \seq_map_break:n {
1379 %             \str_set:Nn \l_tmpa_str { ##1 }
1380 %           }
1381 %         }
1382 %       }
1383 %     }{
1384 %       \str_clear:N \l_tmpa_str
1385 %     }
1386 %   }
1387 %   \str_if_empty:NTF \l_tmpa_str {
1388 %     \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_li
1389 %   }{
1390 %     \str_if_eq:cNTF {sref_\l_tmpa_str _type} \c__stex_refs_ref_str {
1391 %       \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1392 %         \cs_if_exist:cTF{autoref}{
1393 %           \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1394 %         }{
1395 %           \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1396 %         }
1397 %       }{
1398 %         \ltx@ifpackageloaded{hyperref}{
1399 %           \hyperref[sref_\l_tmpa_str]\l__stex_refs_linktext_tl
1400 %         }{
1401 %           \l__stex_refs_linktext_tl
1402 %         }
1403 %       }
1404 %     }{
1405 %       \ltx@ifpackageloaded{hyperref}{
1406 %         \href{\use:c{sref_url_\l_tmpa_str _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_
1407 %       }{
1408 %         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_ref
```

138

```
1409 %        }
1410 %      }
1411 %    }
1412 % }{
1413 %    % TODO
1414 % }
1415 %}
```

(*End definition for* `\sref`. *This function is documented on page* *82*.)

`\srefsym`

```
1416 \NewDocumentCommand \srefsym { O{} m}{
1417   \stex_get_symbol:n { #2 }
1418   \__stex_refs_sym_aux:nn{#1}{\l_stex_get_symbol_uri_str}
1419 }
1420
1421 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1422
1423 %  \str_if_exist:cTF {sref_sym_#2 _label_str }{
1424 %     \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1425 %  }{
1426 %     \__stex_refs_args:n { #1 }
1427 %     \str_if_empty:NTF \l__stex_refs_indocument_str {
1428 %       \tl_if_exist:cTF{sref_sym_#2 _type}{
1429 %         % doc uri in \l_tmpb_str
1430 %         \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1431 %         \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1432 %           % reference
1433 %           \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1434 %             \cs_if_exist:cTF{autoref}{
1435 %               \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1436 %             }{
1437 %               \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1438 %             }
1439 %           }{
1440 %             \ltx@ifpackageloaded{hyperref}{
1441 %               \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1442 %             }{
1443 %               \l__stex_refs_linktext_tl
1444 %             }
1445 %           }
1446 %         }{
1447 %           % URL
1448 %           \ltx@ifpackageloaded{hyperref}{
1449 %             \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl
1450 %           }{
1451 %             \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_r
1452 %           }
1453 %         }
1454 %       }{
1455 %         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_
1456 %       }
1457 %     }{
1458 %       % TODO
```

```
1459  %      }
1460  %   }
1461  }
```

(*End definition for* `\srefsym`*. This function is documented on page* *82.*)

\srefsymuri

```
1462  \cs_new_protected:Npn \srefsymuri #1 #2 { % TODO
1463    #2%\__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1464  }
```

(*End definition for* `\srefsymuri`*. This function is documented on page* *82.*)

```
1465  ⟨/package⟩
```

Chapter 29

sTeX
-Modules Implementation

```
1466 ⟨*package⟩
1467
1468 %%%%%%%%%%%%   modules.dtx   %%%%%%%%%%%%
1469
1470 ⟨@@=stex_modules⟩
```

Warnings and error messages
```
1471 \msg_new:nnn{stex}{error/unknownmodule}{
1472   No~module~#1~found
1473 }
1474 \msg_new:nnn{stex}{error/syntax}{
1475   Syntax~error:~#1
1476 }
1477 \msg_new:nnn{stex}{error/siglanguage}{
1478   Module~#1~declares~signature~#2,~but~does~not~
1479   declare~its~language
1480 }
1481 \msg_new:nnn{stex}{warning/deprecated}{
1482   #1~is~deprecated;~please~use~#2~instead!
1483 }
1484
1485 \msg_new:nnn{stex}{error/conflictingmodules}{
1486   Conflicting~imports~for~module~#1
1487 }
```

`\l_stex_current_module_str`   The current module:
```
1488 \str_new:N \l_stex_current_module_str
```

(*End definition for* `\l_stex_current_module_str`*. This variable is documented on page 84.*)

`\l_stex_all_modules_seq`   Stores all available modules
```
1489 \seq_new:N \l_stex_all_modules_seq
```

(*End definition for* `\l_stex_all_modules_seq`*. This variable is documented on page 84.*)

```
                      \stex_if_in_module_p:
                      \stex_if_in_module:TF   1490 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
                                              1491    \str_if_empty:NTF \l_stex_current_module_str
                                              1492       \prg_return_false: \prg_return_true:
                                              1493 }
```

(*End definition for* `\stex_if_in_module:TF`. *This function is documented on page 84.*)

```
\stex_if_module_exists_p:n
\stex_if_module_exists:nTF   1494 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
                             1495    \prop_if_exist:cTF { c_stex_module_#1_prop }
                             1496       \prg_return_true: \prg_return_false:
                             1497 }
```

(*End definition for* `\stex_if_module_exists:nTF`. *This function is documented on page 84.*)

```
\stex_add_to_current_module:n   Only allowed within modules:
             \STEXexport   1498 \cs_new_protected:Nn \stex_execute_in_module:n { \stex_if_in_module:T {
                           1499    \stex_add_to_current_module:n { #1 }
                           1500    \stex_do_up_to_module:n { #1 }
                           1501 }}
                           1502 \cs_generate_variant:Nn \stex_execute_in_module:n {x}
                           1503
                           1504 \cs_new_protected:Nn \stex_add_to_current_module:n {
                           1505    \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
                           1506 }
                           1507 \cs_generate_variant:Nn \stex_add_to_current_module:n {x}
                           1508 \cs_new_protected:Npn \STEXexport {
                           1509    \ExplSyntaxOn
                           1510    \__stex_modules_export:n
                           1511 }
                           1512 \cs_new_protected:Nn \__stex_modules_export:n {
                           1513    \ignorespacesandpars#1\ExplSyntaxOff
                           1514    \stex_add_to_current_module:n { \ignorespacesandpars#1}
                           1515    \stex_smsmode_do:
                           1516 }
                           1517 \let \stex_module_export_helper:n \use:n
                           1518 \stex_deactivate_macro:Nn \STEXexport {module~environments}
```

(*End definition for* `\stex_add_to_current_module:n` *and* `\STEXexport`. *These functions are documented on page 84.*)

```
\stex_add_constant_to_current_module:n
                             1519 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
                             1520    \str_set:Nx \l_tmpa_str { #1 }
                             1521    \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
                             1522 }
```

(*End definition for* `\stex_add_constant_to_current_module:n`. *This function is documented on page 84.*)

```
\stex_add_import_to_current_module:n
                             1523 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
                             1524    \str_set:Nx \l_tmpa_str { #1 }
                             1525    \exp_args:Nno
```

```
1526    \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1527      \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1528    }
1529  }
```

(*End definition for* `\stex_add_import_to_current_module:n`. *This function is documented on page* *84.*)

```
1530  \cs_new_protected:Nn \stex_collect_imports:n {
1531    \seq_clear:N \l_stex_collect_imports_seq
1532    \__stex_modules_collect_imports:n {#1}
1533  }
1534  \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1535    \seq_map_inline:cn {c_stex_module_#1_imports} {
1536      \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1537        \__stex_modules_collect_imports:n { ##1 }
1538      }
1539    }
1540    \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1541      \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1542    }
1543  }
```

(*End definition for* `\stex_collect_imports:n`. *This function is documented on page* *84.*)

```
1544  \int_new:N \l__stex_modules_group_depth_int
1545  \cs_new_protected:Nn \stex_do_up_to_module:n {
1546    \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1547      #1
1548    }{
1549      #1
1550      \expandafter \tl_gset:Nn
1551      \csname l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1552      \expandafter\expandafter\expandafter\endcsname
1553      \expandafter\expandafter\expandafter { \csname
1554        l__stex_modules_aftergroup_\l_stex_current_module_str _tl\endcsname #1 }
1555      \aftergroup\__stex_modules_aftergroup_do:
1556    }
1557  }
1558  \cs_generate_variant:Nn \stex_do_up_to_module:n {x}
1559  \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1560    \stex_debug:nn{aftergroup}{\cs_meaning:c{
1561      l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1562    }}
1563    \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1564      \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1565      \tl_gclear:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1566    }{
1567      \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1568      \aftergroup\__stex_modules_aftergroup_do:
1569    }
1570  }
1571  \cs_new_protected:Nn \_stex_reset_up_to_module:n {
1572    \expandafter\let\csname l__stex_modules_aftergroup_#1_tl\endcsname\undefined
```

```
1573 }
```

(*End definition for* `\stex_do_up_to_module:n`. *This function is documented on page* *84.*)

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

```
1574
```

(*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.

```
1575 \str_new:N \l_stex_module_ns_str
1576 \str_new:N \l_stex_module_subpath_str
1577 \cs_new_protected:Nn \__stex_modules_compute_namespace:nN {
1578   \seq_set_eq:NN \l_tmpa_seq #2
1579   % split off file extension
1580   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str % <- filename
1581   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1582   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str % <- filename without suffixes
1583   \seq_put_right:No \l_tmpa_seq \l_tmpb_str % <- file path including name without suffixes
1584
1585   \bool_set_true:N \l_tmpa_bool
1586   \bool_while_do:Nn \l_tmpa_bool {
1587     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1588     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1589       {source} { \bool_set_false:N \l_tmpa_bool }
1590     }{}{
1591       \seq_if_empty:NT \l_tmpa_seq {
1592         \bool_set_false:N \l_tmpa_bool
1593       }
1594     }
1595   }
1596
1597   \stex_path_to_string:NN \l_tmpa_seq \l_stex_module_subpath_str
1598   % \l_tmpa_seq <- sub-path relative to archive
1599   \str_if_empty:NTF \l_stex_module_subpath_str {
1600     \str_set:Nx \l_stex_module_ns_str {#1}
1601   }{
1602     \str_set:Nx \l_stex_module_ns_str {
1603       #1/\l_stex_module_subpath_str
1604     }
1605   }
1606 }
1607
1608 \cs_new_protected:Nn \stex_modules_current_namespace: {
1609   \str_clear:N \l_stex_module_subpath_str
1610   \prop_if_exist:NTF \l_stex_current_repository_prop {
1611     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1612     \__stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1613   }{
1614     % split off file extension
1615     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1616     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
```

144

```
1617    \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1618    \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1619    \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1620    \str_set:Nx \l_stex_module_ns_str {
1621      file:/\stex_path_to_string:N \l_tmpa_seq
1622    }
1623  }
1624 }
```

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

29.1 The `smodule` environment

`smodule` arguments:

```
1625 \keys_define:nn { stex / module } {
1626   title         .tl_set:N   = \smoduletitle ,
1627   type          .str_set_x:N = \smoduletype ,
1628   id            .str_set_x:N = \smoduleid ,
1629   deprecate     .str_set_x:N = \l_stex_module_deprecate_str ,
1630   ns            .str_set_x:N = \l_stex_module_ns_str ,
1631   lang          .str_set_x:N = \l_stex_module_lang_str ,
1632   sig           .str_set_x:N = \l_stex_module_sig_str ,
1633   creators      .str_set_x:N = \l_stex_module_creators_str ,
1634   contributors  .str_set_x:N = \l_stex_module_contributors_str ,
1635   meta          .str_set_x:N = \l_stex_module_meta_str ,
1636   srccite       .str_set_x:N = \l_stex_module_srccite_str
1637 }
1638
1639 \cs_new_protected:Nn \__stex_modules_args:n {
1640   \str_clear:N \smoduletitle
1641   \str_clear:N \smoduletype
1642   \str_clear:N \smoduleid
1643   \str_clear:N \l_stex_module_ns_str
1644   \str_clear:N \l_stex_module_deprecate_str
1645   \str_clear:N \l_stex_module_lang_str
1646   \str_clear:N \l_stex_module_sig_str
1647   \str_clear:N \l_stex_module_creators_str
1648   \str_clear:N \l_stex_module_contributors_str
1649   \str_clear:N \l_stex_module_meta_str
1650   \str_clear:N \l_stex_module_srccite_str
1651   \keys_set:nn { stex / module } { #1 }
1652 }
1653
1654 % module parameters here? In the body?
1655
```

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

```
1656 \cs_new_protected:Nn \stex_module_setup:nn {
1657   \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1658   \str_set:Nx \l_stex_module_name_str { #2 }
1659   \__stex_modules_args:n { #1 }
```

145

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

```
1660    \stex_if_in_module:TF {
1661      % Nested module
1662      \prop_get:cnN {c_stex_module_\l_stex_current_module_str _prop}
1663        { ns } \l_stex_module_ns_str
1664      \str_set:Nx \l_stex_module_name_str {
1665        \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
1666          { name } / \l_stex_module_name_str
1667      }
1668      \str_if_empty:NT \l_stex_module_lang_str {
1669        \str_set:Nx \l_stex_module_lang_str {
1670          \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
1671            { lang }
1672        }
1673      }
1674    }{
1675      % not nested:
1676      \str_if_empty:NT \l_stex_module_ns_str {
1677        \stex_modules_current_namespace:
1678        \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1679          / {\l_stex_module_ns_str}
1680        \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1681        \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1682          \str_set:Nx \l_stex_module_ns_str {
1683            \stex_path_to_string:N \l_tmpa_seq
1684          }
1685        }
1686      }
1687    }
```

Next, we determine the language of the module:

```
1688    \str_if_empty:NT \l_stex_module_lang_str {
1689      \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1690      \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1691      \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1692      \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
1693        \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
1694          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
1695        }
1696      }
1697      \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1698      \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
1699        \seq_pop_right:NN \l_tmpa_seq \l_stex_module_lang_str
1700        \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1701          inferred~from~file~name}
1702      }
1703    }
1704
1705    \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
1706      \exp_args:NNo \stex_set_language:Nn \l_tmpa_str \l_stex_module_lang_str
1707    }}
```

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

```
1708    \str_if_empty:NTF \l_stex_module_sig_str {
1709      \exp_args:Nnx \prop_gset_from_keyval:cn {
1710        c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1711      } {
1712        name      = \l_stex_module_name_str ,
1713        ns        = \l_stex_module_ns_str ,
1714        file      = \exp_not:o { \g_stex_currentfile_seq } ,
1715        lang      = \l_stex_module_lang_str ,
1716        sig       = \l_stex_module_sig_str ,
1717        deprecate = \l_stex_module_deprecate_str ,
1718        meta      = \l_stex_module_meta_str
1719      }
1720      \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1721      \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1722      \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _copymodules}
1723      \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1724      \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
```

We load the metatheory:

```
1725      \str_if_empty:NT \l_stex_module_meta_str {
1726        \str_set:Nx \l_stex_module_meta_str {
1727          \c_stex_metatheory_ns_str ? Metatheory
1728        }
1729      }
1730      \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1731        \bool_set_true:N \l_stex_in_meta_bool
1732        \exp_args:Nx \stex_add_to_current_module:n {
1733          \bool_set_true:N \l_stex_in_meta_bool
1734          \stex_activate_module:n {\l_stex_module_meta_str}
1735          \bool_set_false:N \l_stex_in_meta_bool
1736        }
1737        \stex_activate_module:n {\l_stex_module_meta_str}
1738        \bool_set_false:N \l_stex_in_meta_bool
1739      }
1740    }{
1741      \str_if_empty:NT \l_stex_module_lang_str {
1742        \msg_error:nnxx{stex}{error/siglanguage}{
1743          \l_stex_module_ns_str?\l_stex_module_name_str
1744        }{\l_stex_module_sig_str}
1745      }
1746      \stex_debug:nn{modules}{Signature~\l_stex_module_sig_str~for~\l_stex_module_ns_str?\l_st
1747      \stex_if_module_exists:nTF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1748        \stex_debug:nn{modules}{(already exists)}
1749      }{
1750        \stex_debug:nn{modules}{(needs loading)}
1751        \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1752        \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1753        \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1754        \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1755        \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1756        \str_set:Nx \l_tmpa_str {
1757          \stex_path_to_string:N \l_tmpa_seq /
```

```
1758                \l_tmpa_str . \l_stex_module_sig_str .tex
1759              }
1760          \IfFileExists \l_tmpa_str {
1761            \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1762              \str_clear:N \l_stex_current_module_str
1763              \seq_clear:N \l_stex_all_modules_seq
1764              \stex_debug:nn{modules}{Loading~signature}
1765            }
1766          }{
1767            \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1768          }
1769        }
1770        \stex_if_smsmode:F {
1771          \stex_activate_module:n {
1772            \l_stex_module_ns_str ? \l_stex_module_name_str
1773          }
1774        }
1775        \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1776      }
1777      \str_if_empty:NF \l_stex_module_deprecate_str {
1778        \msg_warning:nnxx{stex}{warning/deprecated}{
1779          Module~\l_stex_current_module_str
1780        }{
1781          \l_stex_module_deprecate_str
1782        }
1783      }
1784      \seq_put_right:Nx \l_stex_all_modules_seq {
1785        \l_stex_module_ns_str ? \l_stex_module_name_str
1786      }
1787      \tl_clear:c{l__stex_modules_aftergroup_\l_stex_module_ns_str ? \l_stex_module_name_str _tl
1788 }
```

(*End definition for* \stex_module_setup:nn. *This function is documented on page 85.*)

smodule (*env.*) The module environment.

__stex_modules_begin_module: implements \begin{smodule}

```
1789 \cs_new_protected:Nn \__stex_modules_begin_module: {
1790    \stex_reactivate_macro:N \STEXexport
1791    \stex_reactivate_macro:N \importmodule
1792    \stex_reactivate_macro:N \symdecl
1793    \stex_reactivate_macro:N \notation
1794    \stex_reactivate_macro:N \symdef
1795
1796    \stex_debug:nn{modules}{
1797      New~module:\\
1798      Namespace:~\l_stex_module_ns_str\\
1799      Name:~\l_stex_module_name_str\\
1800      Language:~\l_stex_module_lang_str\\
1801      Signature:~\l_stex_module_sig_str\\
1802      Metatheory:~\l_stex_module_meta_str\\
1803      File:~\stex_path_to_string:N \g_stex_currentfile_seq
1804    }
1805
```

```
1806    \stex_if_do_html:T{
1807      \begin{stex_annotate_env} {theory} {
1808        \l_stex_module_ns_str ? \l_stex_module_name_str
1809      }
1810
1811      \stex_annotate_invisible:nnn{header}{} {
1812        \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1813        \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1814        \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1815          \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1816        }
1817        \str_if_empty:NF \smoduletype {
1818          \stex_annotate:nnn{type}{\smoduletype}{}
1819        }
1820      }
1821    }
1822    % TODO: Inherit metatheory for nested modules?
1823 }
1824 \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}

```
1825 \cs_new_protected:Nn \__stex_modules_end_module: {
1826    \stex_debug:nn{modules}{Closing~module~\prop_item:cn {c_stex_module_\l_stex_current_module
1827    \_stex_reset_up_to_module:n \l_stex_current_module_str
1828    \stex_if_smsmode:T {
1829      \stex_persist:x {
1830        \prop_set_from_keyval:cn{c_stex_module_\l_stex_current_module_str _prop}{
1831          \exp_after:wN \prop_to_keyval:N \csname c_stex_module_\l_stex_current_module_str _pr
1832        }
1833        \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _constants}{
1834          \seq_use:cn{c_stex_module_\l_stex_current_module_str _constants},
1835        }
1836        \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _imports}{
1837          \seq_use:cn{c_stex_module_\l_stex_current_module_str _imports},
1838        }
1839        \tl_set:cn {c_stex_module_\l_stex_current_module_str _code}
1840      }
1841      \exp_after:wN \let \exp_after:wN \l_tmpa_tl \csname c_stex_module_\l_stex_current_module
1842      \exp_after:wN \stex_persist:n \exp_after:wN { \exp_after:wN { \l_tmpa_tl } }
1843    }
1844 }
```

(*End definition for* __stex_modules_end_module:.)

   The core environment

```
1845 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1846 \NewDocumentEnvironment { smodule } { O{} m } {
1847    \stex_module_setup:nn{#1}{#2}
1848    %\par
1849    \stex_if_smsmode:F{
1850      \tl_if_empty:NF \smoduletitle {
1851        \exp_args:No \stex_document_title:n \smoduletitle
1852      }
```

```
1853        \tl_clear:N \l_tmpa_tl
1854        \clist_map_inline:Nn \smoduletype {
1855          \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1856            \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1857          }
1858        }
1859        \tl_if_empty:NTF \l_tmpa_tl {
1860          \__stex_modules_smodule_start:
1861        }{
1862          \l_tmpa_tl
1863        }
1864      }
1865      \__stex_modules_begin_module:
1866      \str_if_empty:NF \smoduleid {
1867        \stex_ref_new_doc_target:n \smoduleid
1868      }
1869      \stex_smsmode_do:
1870    } {
1871      \__stex_modules_end_module:
1872      \stex_if_smsmode:F {
1873        \end{stex_annotate_env}
1874        \clist_set:No \l_tmpa_clist \smoduletype
1875        \tl_clear:N \l_tmpa_tl
1876        \clist_map_inline:Nn \l_tmpa_clist {
1877          \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1878            \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1879          }
1880        }
1881        \tl_if_empty:NTF \l_tmpa_tl {
1882          \__stex_modules_smodule_end:
1883        }{
1884          \l_tmpa_tl
1885        }
1886      }
1887    }
```

\stexpatchmodule

```
1888 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1889 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1890
1891 \newcommand\stexpatchmodule[3][] {
1892    \str_set:Nx \l_tmpa_str{ #1 }
1893    \str_if_empty:NTF \l_tmpa_str {
1894      \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1895      \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1896    }{
1897      \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1898      \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1899    }
1900 }
```

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

29.2 Invoking modules

```
1901 \NewDocumentCommand \STEXModule { m } {
1902   \exp_args:NNx \str_set:Nn \l_tmpa_str { # 1 }
1903   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1904   \tl_set:Nn \l_tmpa_tl {
1905     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1906   }
1907   \seq_map_inline:Nn \l_stex_all_modules_seq {
1908     \str_set:Nn \l_tmpb_str { ##1 }
1909     \str_if_eq:eeT { \l_tmpa_str } {
1910       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1911     } {
1912       \seq_map_break:n {
1913         \tl_set:Nn \l_tmpa_tl {
1914           \stex_invoke_module:n { ##1 }
1915         }
1916       }
1917     }
1918   }
1919   \l_tmpa_tl
1920 }
1921
1922 \cs_new_protected:Nn \stex_invoke_module:n {
1923   \stex_debug:nn{modules}{Invoking~module~#1}
1924   \peek_charcode_remove:NTF ! {
1925     \__stex_modules_invoke_uri:nN { #1 }
1926   } {
1927     \peek_charcode_remove:NTF ? {
1928       \__stex_modules_invoke_symbol:nn { #1 }
1929     } {
1930       \msg_error:nnx{stex}{error/syntax}{
1931         ?~or~!~expected~after~
1932         \c_backslash_str STEXModule{#1}
1933       }
1934     }
1935   }
1936 }
1937
1938 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1939   \str_set:Nn #2 { #1 }
1940 }
1941
1942 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1943   \stex_invoke_symbol:n{#1?#2}
1944 }
```

(*End definition for* \STEXModule *and* \stex_invoke_module:n. *These functions are documented on page* *85*.)

```
1945 \bool_new:N \l_stex_in_meta_bool
1946 \bool_set_false:N \l_stex_in_meta_bool
```

151

```
1947 \cs_new_protected:Nn \stex_activate_module:n {
1948   \stex_debug:nn{modules}{Activating~module~#1}
1949   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1950     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1951     \use:c{ c_stex_module_#1_code }
1952   }
1953 }
```

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

mmtinterface (*env.*)

```
1954 \NewDocumentEnvironment { mmtinterface } { O{} m m } {
1955   \begin{smodule}[#1]{#3}
1956     \str_set:Nx \l_stex_module_mmtfor_str {#2}
1957     \MMTinclude{#2}
1958     \stex_reactivate_macro:N \mmtdecl
1959     \stex_reactivate_macro:N \mmtdef
1960 }{
1961   \end{smodule}
1962 }
```

```
1963 ⟨/package⟩
```

Chapter 30

SТEX -Module Inheritance Implementation

```
1964 ⟨∗package⟩
1965
1966 %%%%%%%%%%%%    inheritance.dtx    %%%%%%%%%%%%
1967
```

30.1 SMS Mode

```
1968 ⟨@@=stex_smsmode⟩
```

```
1969 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1970 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1971 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1972
1973 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1974    \makeatletter
1975    \makeatother
1976    \ExplSyntaxOn
1977    \ExplSyntaxOff
1978    \rustexBREAK
1979 }
1980
1981 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1982    \symdef
1983    \importmodule
1984    \notation
1985    \symdecl
1986    \STEXexport
1987    \inlineass
1988    \inlinedef
1989    \inlineex
1990    \endinput
1991    \setnotation
```

```
1992    \copynotation
1993    \assign
1994    \renamedecl
1995    \donotcopy
1996    \instantiate
1997    \textsymdecl
1998 }
1999
2000 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
2001    \tl_to_str:n {
2002      smodule,
2003      copymodule,
2004      interpretmodule,
2005      realization,
2006      sdefinition,
2007      sexample,
2008      sassertion,
2009      sparagraph,
2010      mathstructure,
2011      extstructure,
2012      extstructure*
2013    }
2014 }
```

(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 87.)*

`\stex_if_smsmode_p:`
`\stex_if_smsmode:TF`

```
2015 \bool_new:N \g__stex_smsmode_bool
2016 \bool_set_false:N \g__stex_smsmode_bool
2017 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
2018    \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
2019 }
```

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

`\__stex_smsmode_in_smsmode:nn`

```
2020 \cs_new_protected:Nn \__stex_smsmode_in_smsmode:nn { \stex_suppress_html:n {
2021    \vbox_set:Nn \l_tmpa_box {
2022      \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
2023      \bool_gset_true:N \g__stex_smsmode_bool
2024      #2
2025      \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
2026    }
2027    \box_clear:N \l_tmpa_box
2028 } }
```

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

`\stex_file_in_smsmode:nn`

```
2029 \quark_new:N \q__stex_smsmode_break
2030
2031 \NewDocumentCommand \__stex_smsmode_importmodule: { O{} m} {
2032    \seq_gput_right:Nn \l__stex_smsmode_importmodules_seq {{#1}{#2}}
2033    \stex_smsmode_do:
```

```
2034 }
2035
2036 \cs_new_protected:Nn \__stex_smsmode_module:nn {
2037   \__stex_modules_args:n{#1}
2038   \stex_if_in_module:F {
2039     \str_if_empty:NF \l_stex_module_sig_str {
2040       \stex_modules_current_namespace:
2041       \str_set:Nx \l_stex_module_name_str { #2 }
2042       \stex_if_module_exists:nF{\l_stex_module_ns_str?\l_stex_module_name_str}{
2043         \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
2044         \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
2045         \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
2046         \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
2047         \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
2048         \str_set:Nx \l_tmpa_str {
2049           \stex_path_to_string:N \l_tmpa_seq /
2050           \l_tmpa_str . \l_stex_module_sig_str .tex
2051         }
2052         \IfFileExists \l_tmpa_str {
2053           \exp_args:NNx \seq_gput_right:Nn \l__stex_smsmode_sigmodules_seq \l_tmpa_str
2054         }{
2055           \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
2056         }
2057       }
2058     }
2059   }
2060 }
2061
2062 \prg_new_conditional:Nnn \__stex_smsmode_check_import_pair:nn {T,F,TF} {
2063   %\stex_debug:nn{import-pair}{\detokenize{{#1}~{#2}}}
2064   \tl_if_empty:nTF{#1}{
2065     \prop_if_exist:NTF \l_stex_current_repository_prop
2066       {
2067         %\stex_debug:nn{import-pair}{in repository \prop_item:Nn \l_stex_current_repository_
2068         \prg_return_true:
2069       } {
2070         \seq_set_split:Nnn \l_tmpa_seq ? {#2}
2071         \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
2072         \tl_if_empty:NT \l_tmpa_tl {
2073           \seq_pop_left:NN \l_tmpa_seq \l_tmpa_tl
2074         }
2075         %\stex_debug:nn{import-pair}{\seq_use:Nn \l_tmpa_seq,~of~length~\seq_count:N \l_tmpa
2076         \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1
2077           \prg_return_true: \prg_return_false:
2078       }
2079   }\prg_return_true:
2080 }
2081
2082 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
2083   \stex_filestack_push:n{#1}
2084   \seq_gclear:N \l__stex_smsmode_importmodules_seq
2085   \seq_gclear:N \l__stex_smsmode_sigmodules_seq
2086   % ----- new ---------------------------
2087   \__stex_smsmode_in_smsmode:nn{#1}{
```

```
2088        \let\importmodule\__stex_smsmode_importmodule:
2089        \let\stex_module_setup:nn\__stex_smsmode_module:nn
2090        \let\__stex_modules_begin_module:\relax
2091        \let\__stex_modules_end_module:\relax
2092        \seq_clear:N \g_stex_smsmode_allowedenvs_seq
2093        \exp_args:NNx \seq_put_right:Nn \g_stex_smsmode_allowedenvs_seq {\tl_to_str:n{smodule}}
2094        \tl_clear:N \g_stex_smsmode_allowedmacros_tl
2095        \tl_clear:N \g_stex_smsmode_allowedmacros_escape_tl
2096        \tl_put_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\importmodule}
2097        \everyeof{\q__stex_smsmode_break\noexpand}
2098        \expandafter\expandafter\expandafter
2099        \stex_smsmode_do:
2100        \csname @ @ input\endcsname "#1"\relax
2101
2102        \seq_map_inline:Nn \l__stex_smsmode_sigmodules_seq {
2103          \stex_filestack_push:n{##1}
2104          \expandafter\expandafter\expandafter
2105          \stex_smsmode_do:
2106          \csname @ @ input\endcsname "##1"\relax
2107          \stex_filestack_pop:
2108        }
2109      }
2110      % ----- new ---------------------------
2111      \__stex_smsmode_in_smsmode:nn{#1} {
2112        #2
2113        % ----- new ---------------------------
2114        \begingroup
2115        %\stex_debug:nn{smsmode}{Here:~\seq_use:Nn\l__stex_smsmode_importmodules_seq, }
2116        \seq_map_inline:Nn \l__stex_smsmode_importmodules_seq {
2117          \__stex_smsmode_check_import_pair:nnT ##1 { \begingroup
2118            \stex_import_module_uri:nn ##1
2119            \stex_import_require_module:nnnn
2120              \l_stex_import_ns_str
2121              \l_stex_import_archive_str
2122              \l_stex_import_path_str
2123              \l_stex_import_name_str \endgroup
2124          }
2125        }
2126        \endgroup
2127        \stex_debug:nn{smsmode}{Actually~loading~file~#1}
2128        % ----- new ---------------------------
2129        \everyeof{\q__stex_smsmode_break\noexpand}
2130        \expandafter\expandafter\expandafter
2131        \stex_smsmode_do:
2132        \csname @ @ input\endcsname "#1"\relax
2133      }
2134      \stex_filestack_pop:
2135    }
```

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

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

```
2136 \cs_new_protected:Npn \stex_smsmode_do: {
```

```
2137    \stex_if_smsmode:T {
2138      \__stex_smsmode_do:w
2139    }
2140 }
2141 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
2142    \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 }}{
2143      \expandafter\if\expandafter\relax\noexpand#1
2144        \expandafter\__stex_smsmode_do_aux:N\expandafter#1
2145      \else\expandafter\__stex_smsmode_do:w\fi
2146    }{
2147      \__stex_smsmode_do:w %#1
2148    }
2149 }
2150 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
2151    \cs_if_eq:NNF #1 \q__stex_smsmode_break {
2152      \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
2153        #1\__stex_smsmode_do:w
2154      }{
2155        \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
2156          #1
2157        }{
2158          \cs_if_eq:NNTF \begin #1 {
2159            \__stex_smsmode_check_begin:n
2160          }{
2161            \cs_if_eq:NNTF \end #1 {
2162              \__stex_smsmode_check_end:n
2163            }{
2164              \__stex_smsmode_do:w
2165            }
2166          }
2167        }
2168      }
2169    }
2170 }
2171
2172 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
2173    \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
2174      \begin{#1}
2175    }{
2176      \__stex_smsmode_do:w
2177    }
2178 }
2179 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
2180    \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
2181      \end{#1}\__stex_smsmode_do:w
2182    }{
2183      \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
2184    }
2185 }
```

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

30.2 Inheritance

`\stex_import_module_uri:nn`

```
2187 \cs_new_protected:Nn \stex_import_module_uri:nn {
2188   \str_set:Nx \l_stex_import_archive_str { #1 }
2189   \str_set:Nn \l_stex_import_path_str { #2 }
2190
2191   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
2192   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
2193   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
2194
2195   \stex_modules_current_namespace:
2196   \bool_lazy_all:nTF {
2197     {\str_if_empty_p:N \l_stex_import_archive_str}
2198     {\str_if_empty_p:N \l_stex_import_path_str}
2199     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
2200   }{
2201     \str_set_eq:NN \l_stex_import_path_str \l_stex_module_subpath_str
2202     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
2203   }{
2204     \str_if_empty:NT \l_stex_import_archive_str {
2205       \prop_if_exist:NT \l_stex_current_repository_prop {
2206         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
2207       }
2208     }
2209     \str_if_empty:NTF \l_stex_import_archive_str {
2210       \str_if_empty:NF \l_stex_import_path_str {
2211         \stex_path_from_string:Nn \l_tmpb_seq {
2212           \l_stex_module_ns_str  / .. / \l_stex_import_path_str
2213         }
2214         \str_set:Nx \l_stex_import_ns_str {\stex_path_to_string:N \l_tmpb_seq}
2215         \str_replace_once:Nnn \l_stex_import_ns_str {file:/} {file://}
2216       }
2217     }{
2218       \stex_require_repository:n \l_stex_import_archive_str
2219       \prop_get:cnN { c_stex_mathhub_\l_stex_import_archive_str _manifest_prop } { ns }
2220         \l_stex_import_ns_str
2221       \str_if_empty:NF \l_stex_import_path_str {
2222         \str_set:Nx \l_stex_import_ns_str {
2223           \l_stex_import_ns_str / \l_stex_import_path_str
2224         }
2225       }
2226     }
2227   }
2228 }
```

(*End definition for* `\stex_import_module_uri:nn`. *This function is documented on page 89.*)

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

Store the return values of `\stex_import_module_uri:nn`.

```
2229 \str_new:N \l_stex_import_name_str
2230 \str_new:N \l_stex_import_archive_str
2231 \str_new:N \l_stex_import_path_str
2232 \str_new:N \l_stex_import_ns_str
```

(*End definition for* `\l_stex_import_name_str` *and others. These variables are documented on page 89.*)

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

```
2233 \cs_new_protected:Nn \stex_import_require_module:nnnn {
2234   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
2235
2236     \stex_debug:nn{requiremodule}{Here:\\~~1:~#1\\~~2:~#2\\~~3:~#3\\~~4:~#4}
2237
2238     \exp_args:NNxx \seq_set_split:Nnn \l_tmpa_seq {\tl_to_str:n{/}} {#4}
2239     \seq_get_left:NN \l_tmpa_seq \l_tmpc_str
2240
2241     %\stex_debug:nn{requiremodule}{Top~module:\l_tmpc_str}
2242
2243     % archive
2244     \str_set:Nx \l_tmpa_str { #2 }
2245     \str_if_empty:NTF \l_tmpa_str {
2246       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
2247       \seq_put_right:Nn \l_tmpa_seq {..}
2248     } {
2249       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
2250       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
2251       \seq_put_right:Nn \l_tmpa_seq { source }
2252     }
2253
2254     % path
2255     \str_set:Nx \l_tmpb_str { #3 }
2256     \str_if_empty:NTF \l_tmpb_str {
2257       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / \l_tmpc_str }
2258
2259       \ltx@ifpackageloaded{babel} {
2260         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
2261           { \languagename } \l_tmpb_str {
2262             \msg_error:nnx{stex}{error/unknownlanguage}{\languagename}
2263           }
2264       } {
2265         \str_clear:N \l_tmpb_str
2266       }
2267
2268       \stex_debug:nn{modules}{Checking~a1~\l_tmpa_str.\l_tmpb_str.tex}
2269       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2270         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2271       }{
2272         \stex_debug:nn{modules}{Checking~a2~\l_tmpa_str.tex}
2273         \IfFileExists{ \l_tmpa_str.tex }{
2274           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2275         }{
2276           % try english as default
2277           \stex_debug:nn{modules}{Checking~a3~\l_tmpa_str.en.tex}
2278           \IfFileExists{ \l_tmpa_str.en.tex }{
2279             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2280           }{
2281             \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2282           }
2283         }
2284       }
2285
```

```
2286        } {
2287          \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
2288          \seq_concat:NNN \l_tmpb_seq \l_tmpa_seq \l_tmpb_seq
2289
2290          \ltx@ifpackageloaded{babel} {
2291            \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
2292                { \languagename } \l_tmpb_str {
2293                    \msg_error:nnx{stex}{error/unknownlanguage}{\languagename}
2294                }
2295          } {
2296            \str_clear:N \l_tmpb_str
2297          }
2298
2299          \stex_path_canonicalize:N \l_tmpb_seq
2300          \stex_path_to_string:NN \l_tmpb_seq \l_tmpa_str
2301
2302          \stex_debug:nn{modules}{Checking~b1~\l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex}
2303          \IfFileExists{ \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }{
2304            \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.te
2305          }{
2306            \stex_debug:nn{modules}{Checking~b2~\l_tmpa_str/\l_tmpc_str.tex}
2307            \IfFileExists{ \l_tmpa_str/\l_tmpc_str.tex }{
2308              \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.tex }
2309            }{
2310              % try english as default
2311              \stex_debug:nn{modules}{Checking~b3~\l_tmpa_str/\l_tmpc_str.en.tex}
2312              \IfFileExists{ \l_tmpa_str/\l_tmpc_str.en.tex }{
2313                \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.en.tex }
2314              }{
2315                \stex_debug:nn{modules}{Checking~b4~\l_tmpa_str.\l_tmpb_str.tex}
2316                \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2317                  \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2318                }{
2319                  \stex_debug:nn{modules}{Checking~b4~\l_tmpa_str.tex}
2320                  \IfFileExists{ \l_tmpa_str.tex }{
2321                    \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2322                  }{
2323                    % try english as default
2324                    \stex_debug:nn{modules}{Checking~b5~\l_tmpa_str.en.tex}
2325                    \IfFileExists{ \l_tmpa_str.en.tex }{
2326                      \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2327                    }{
2328                      \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2329                    }
2330                  }
2331                }
2332              }
2333            }
2334          }
2335        }
2336
2337        \str_if_eq:eeF{\g__stex_importmodule_file_str}{\seq_use:Nn \g_stex_currentfile_seq /}{
2338          \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
2339            \seq_clear:N \l_stex_all_modules_seq
```

160

```
2340        \str_clear:N \l_stex_current_module_str
2341        \str_set:Nx \l_tmpb_str { #2 }
2342        \str_if_empty:NF \l_tmpb_str {
2343          \stex_set_current_repository:n { #2 }
2344        }
2345        \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
2346      }
2347
2348      \stex_if_module_exists:nF { #1 ? #4 } {
2349        \msg_error:nnx{stex}{error/unknownmodule}{
2350          #1?#4~(in~file~\g__stex_importmodule_file_str)
2351        }
2352      }
2353    }
2354
2355  }
2356  \stex_activate_module:n { #1 ? #4 }
2357 }
```

(*End definition for* \stex_import_require_module:nnnn. *This function is documented on page* *89.*)

\importmodule

```
2358 \NewDocumentCommand \importmodule { O{} m } {
2359   \stex_import_module_uri:nn { #1 } { #2 }
2360   \stex_debug:nn{modules}{Importing~module:~
2361     \l_stex_import_ns_str ? \l_stex_import_name_str
2362   }
2363   \stex_if_smsmode:F {
2364     \stex_annotate_invisible:nnn
2365       {import} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
2366   }
2367   \stex_execute_in_module:x {
2368     \stex_import_require_module:nnnn
2369     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2370     { \l_stex_import_path_str } { \l_stex_import_name_str }
2371   }
2372   \exp_args:Nx \stex_add_import_to_current_module:n {
2373     \l_stex_import_ns_str ? \l_stex_import_name_str
2374   }
2375   \stex_smsmode_do:
2376   \ignorespacesandpars
2377 }
2378 \stex_deactivate_macro:Nn \importmodule {module~environments}
```

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

\usemodule

```
2379 \NewDocumentCommand \usemodule { O{} m } {
2380   \stex_if_smsmode:F {
2381     \stex_import_module_uri:nn { #1 } { #2 }
2382     \stex_import_require_module:nnnn
2383     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2384     { \l_stex_import_path_str } { \l_stex_import_name_str }
2385     \stex_annotate_invisible:nnn
2386       {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
```

```
2387       }
2388     \stex_smsmode_do:
2389     \ignorespacesandpars
2390   }
```

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

```
2391   \cs_new_protected:Nn \stex_csl_to_imports:Nn {
2392     \tl_if_empty:nF{#2}{
2393       \clist_set:Nn \l_tmpa_clist {#2}
2394       \clist_map_inline:Nn \l_tmpa_clist {
2395         \tl_if_head_eq_charcode:nNTF {##1}[{
2396           #1 ##1
2397         }{
2398           #1{##1}
2399         }
2400       }
2401     }
2402   }
2403   \cs_generate_variant:Nn \stex_csl_to_imports:Nn {No}
2404
2405
2406 ⟨/package⟩
```

Chapter 31

STEX
-Symbols Implementation

2407 ⟨∗package⟩

2408

2409 %%%%%%%%%%%% symbols.dtx %%%%%%%%%%%%

2410

Warnings and error messages

2411 \msg_new:nnn{stex}{error/wrongargs}{
2412   args~value~in~symbol~declaration~for~#1~
2413   needs~to~be~i,~a,~b~or~B,~but~#2~given
2414 }
2415 \msg_new:nnn{stex}{error/unknownsymbol}{
2416   No~symbol~#1~found!
2417 }
2418 \msg_new:nnn{stex}{error/seqlength}{
2419   Expected~#1~arguments;~got~#2!
2420 }
2421 \msg_new:nnn{stex}{error/unknownnotation}{
2422   Unknown~notation~#1~for~#2!
2423 }

31.1 Symbol Declarations

2424 ⟨@@=stex_symdecl⟩

\stex_all_symbols:n Map over all available symbols

2425 \cs_new_protected:Nn \stex_all_symbols:n {
2426   \def __stex_symdecl_all_symbols_cs ##1 {#1}
2427   \seq_map_inline:Nn \l_stex_all_modules_seq {
2428   \seq_map_inline:cn{c_stex_module_##1_constants}{
2429   __stex_symdecl_all_symbols_cs{##1?####1}
2430   }
2431   }
2432 }

(*End definition for* \stex_all_symbols:n. *This function is documented on page 91.*)

163

```
2433 \NewDocumentCommand \STEXsymbol { m } {
2434   \stex_get_symbol:n { #1 }
2435   \exp_args:No
2436   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
2437 }
```

(*End definition for* \STEXsymbol. *This function is documented on page* *92.*)

symdecl arguments:

```
2438 \keys_define:nn { stex / symdecl } {
2439   name        .str_set_x:N  = \l_stex_symdecl_name_str ,
2440   args        .str_set_x:N  = \l_stex_symdecl_args_str ,
2441   type        .tl_set:N     = \l_stex_symdecl_type_tl ,
2442   deprecate   .str_set_x:N  = \l_stex_symdecl_deprecate_str ,
2443   align       .str_set:N    = \l_stex_symdecl_align_str , % TODO(?)
2444   gfc         .str_set:N    = \l_stex_symdecl_gfc_str , % TODO(?)
2445   def         .tl_set:N     = \l_stex_symdecl_definiens_tl ,
2446   reorder     .str_set_x:N  = \l_stex_symdecl_reorder_str ,
2447   argnames    .clist_set:N  = \l_stex_symdecl_argnames_clist ,
2448   assoc       .choices:nn   =
2449     {bin,binl,binr,pre,conj,pwconj}
2450     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}
2451 }
2452
2453 \bool_new:N \l_stex_symdecl_make_macro_bool
2454
2455 \cs_new_protected:Nn \__stex_symdecl_args:n {
2456   \str_clear:N \l_stex_symdecl_name_str
2457   \str_clear:N \l_stex_symdecl_args_str
2458   \str_clear:N \l_stex_symdecl_deprecate_str
2459   \str_clear:N \l_stex_symdecl_reorder_str
2460   \str_clear:N \l_stex_symdecl_assoctype_str
2461   \bool_set_false:N \l_stex_symdecl_local_bool
2462   \tl_clear:N \l_stex_symdecl_type_tl
2463   \tl_clear:N \l_stex_symdecl_definiens_tl
2464   \clist_clear:N \l_stex_symdecl_argnames_clist
2465
2466   \keys_set:nn { stex / symdecl } { #1 }
2467 }
```

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

```
2468
2469 \NewDocumentCommand \symdecl { s m O{} } {
2470   \__stex_symdecl_args:n { #3 }
2471   \IfBooleanTF #1 {
2472     \bool_set_false:N \l_stex_symdecl_make_macro_bool
2473   } {
2474     \bool_set_true:N \l_stex_symdecl_make_macro_bool
2475   }
2476   \stex_symdecl_do:n { #2 }
2477   \stex_smsmode_do:
2478 }
```

164

```
2479
2480 \cs_new_protected:Nn \stex_symdecl_do:nn {
2481   \__stex_symdecl_args:n{#1}
2482   \bool_set_false:N \l_stex_symdecl_make_macro_bool
2483   \stex_symdecl_do:n{#2}
2484 }
2485
2486 \stex_deactivate_macro:Nn \symdecl {module~environments}
```

(*End definition for* \symdecl. *This function is documented on page* 90.)

\stex_symdecl_do:n

```
2487 \cs_new_protected:Nn \stex_symdecl_do:n {
2488   \stex_if_in_module:F {
2489     % TODO throw error? some default namespace?
2490   }
2491
2492   \str_if_empty:NT \l_stex_symdecl_name_str {
2493     \str_set:Nx \l_stex_symdecl_name_str { #1 }
2494   }
2495
2496   \prop_if_exist:cT { l_stex_symdecl_
2497       \l_stex_current_module_str ?
2498       \l_stex_symdecl_name_str
2499     _prop
2500   }{
2501     % TODO throw error (beware of circular dependencies)
2502   }
2503
2504   \prop_clear:N \l_tmpa_prop
2505   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
2506   \seq_clear:N \l_tmpa_seq
2507   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
2508   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
2509
2510   \str_if_empty:NT \l_stex_symdecl_deprecate_str {
2511     \str_if_empty:NF \l_stex_module_deprecate_str {
2512       \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
2513     }
2514   }
2515   \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
2516
2517   \exp_args:No \stex_add_constant_to_current_module:n {
2518     \l_stex_symdecl_name_str
2519   }
2520
2521   % arity/args
2522   \int_zero:N \l_tmpb_int
2523
2524   \bool_set_true:N \l_tmpa_bool
2525   \str_map_inline:Nn \l_stex_symdecl_args_str {
2526     \token_case_meaning:NnF ##1 {
2527       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2528       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
```

```
      {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
      {\tl_to_str:n a} {
        \bool_set_false:N \l_tmpa_bool
        \int_incr:N \l_tmpb_int
      }
      {\tl_to_str:n B} {
        \bool_set_false:N \l_tmpa_bool
        \int_incr:N \l_tmpb_int
      }
    }{
      \msg_error:nnxx{stex}{error/wrongargs}{
        \l_stex_current_module_str ?
        \l_stex_symdecl_name_str
      }{##1}
    }
  }

  \bool_if:NTF \l_tmpa_bool {
    % possibly numeric
    \str_if_empty:NTF \l_stex_symdecl_args_str {
      \prop_put:Nnn \l_tmpa_prop { args } {}
      \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
    }{
      \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
      \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
      \str_clear:N \l_tmpa_str
      \int_step_inline:nn \l_tmpa_int {
        \str_put_right:Nn \l_tmpa_str i
      }
      \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
    }
  } {
    \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
    \prop_put:Nnx \l_tmpa_prop { arity }
      { \str_count:N \l_stex_symdecl_args_str }
  }
  \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }

  \tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
    \prop_put:Nnx \l_tmpa_prop { defined }{ false }
  }{
    \prop_put:Nnx \l_tmpa_prop { defined }{ true }
  }

  % argnames

  \clist_clear:N \l_tmpa_clist
  \int_step_inline:nn {\prop_item:Nn \l_tmpa_prop {arity}} {
    \clist_if_empty:NTF \l_stex_symdecl_argnames_clist {
      \clist_put_right:Nn \l_tmpa_clist {##1}
    }{
      \clist_pop:NN \l_stex_symdecl_argnames_clist \l_tmpa_tl
      \exp_args:NNx \clist_put_right:Nn \l_tmpa_clist {\c_dollar_str\l_tmpa_tl}
    }
```

166

```
2583    }
2584    \prop_put:Nnx \l_tmpa_prop {argnames} {\clist_use:Nn \l_tmpa_clist ,}

2585

2586    % semantic macro

2587

2588    \bool_if:NT \l_stex_symdecl_make_macro_bool {
2589      \exp_args:Nx \stex_do_up_to_module:n {
2590        \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2591          \l_stex_current_module_str ? \l_stex_symdecl_name_str
2592        }}
2593      }
2594    }

2595

2596    \stex_debug:nn{symbols}{New~symbol:~
2597      \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2598      Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2599      Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
2600      Definiens:~\exp_not:o {\l_stex_symdecl_definiens_tl}
2601    }

2602

2603    % circular dependencies require this:
2604    \stex_if_do_html:T {
2605      \stex_annotate_invisible:nnn {symdecl} {
2606        \l_stex_current_module_str ? \l_stex_symdecl_name_str
2607      } {
2608        \tl_if_empty:NF \l_stex_symdecl_type_tl {
2609          \stex_annotate_invisible:nnn{type}{}{$\l_stex_symdecl_type_tl$}
2610        }
2611        \stex_annotate_invisible:nnn{args}{\prop_item:Nn \l_tmpa_prop { args }}{}
2612        \stex_annotate_invisible:nnn{macroname}{#1}{}
2613        \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2614          \stex_annotate_invisible:nnn{definiens}{}
2615            {$\l_stex_symdecl_definiens_tl$}
2616        }
2617        \str_if_empty:NF \l_stex_symdecl_assoctype_str {
2618          \stex_annotate_invisible:nnn{assoctype}{\l_stex_symdecl_assoctype_str}{}
2619        }
2620        \str_if_empty:NF \l_stex_symdecl_reorder_str {
2621          \stex_annotate_invisible:nnn{reorderargs}{\l_stex_symdecl_reorder_str}{}
2622        }
2623      }
2624    }
2625    \prop_if_exist:cF {
2626      l_stex_symdecl_
2627      \l_stex_current_module_str ? \l_stex_symdecl_name_str
2628      _prop
2629    } {
2630      \bool_if:NTF \l_stex_symdecl_local_bool \stex_do_up_to_module:x \stex_execute_in_module:
2631        \__stex_symdecl_restore_symbol:nnnnnnnn
2632          {\l_stex_symdecl_name_str}
2633          { \prop_item:Nn \l_tmpa_prop {args} }
2634          { \prop_item:Nn \l_tmpa_prop {arity} }
2635          { \prop_item:Nn \l_tmpa_prop {assocs} }
2636          { \prop_item:Nn \l_tmpa_prop {defined} }
```

```
2637          {\bool_if:NT \l_stex_symdecl_make_macro_bool {#1} }
2638          {\l_stex_current_module_str}
2639          { \prop_item:Nn \l_tmpa_prop {argnames} }
2640      }
2641    }
2642 }
2643 \cs_new_protected:Nn \__stex_symdecl_restore_symbol:nnnnnnnn {
2644    \prop_clear:N \l_tmpa_prop
2645    \prop_put:Nnn \l_tmpa_prop { module } { #7 }
2646    \prop_put:Nnn \l_tmpa_prop { name } { #1}
2647    \prop_put:Nnn \l_tmpa_prop { args } {#2}
2648    \prop_put:Nnn \l_tmpa_prop { arity } { #3 }
2649    \prop_put:Nnn \l_tmpa_prop { assocs } { #4 }
2650    \prop_put:Nnn \l_tmpa_prop { defined } { #5 }
2651    \prop_put:Nnn \l_tmpa_prop { argnames } { #8 }
2652    \tl_if_empty:nF{#6}{
2653      \tl_set:cx{#6}{\stex_invoke_symbol:n{\detokenize{#7 ? #1}}}
2654    }
2655    \prop_set_eq:cN{l_stex_symdecl_ \detokenize{#7 ? #1} _prop}\l_tmpa_prop
2656    \seq_clear:c{l_stex_symdecl_ \detokenize{#7 ? #1} _notations}
2657 }
```

(*End definition for* `\stex_symdecl_do:n`. *This function is documented on page 91.*)

\textsymdecl

```
2658
2659 \keys_define:nn { stex / textsymdecl } {
2660   name     .str_set_x:N = \l__stex_symdecl_name_str ,
2661   type     .tl_set:N    = \l__stex_symdecl_type_tl
2662 }
2663
2664 \cs_new_protected:Nn \_stex_textsymdecl_args:n {
2665   \str_clear:N \l__stex_symdecl_name_str
2666   \tl_clear:N \l__stex_symdecl_type_tl
2667   \clist_clear:N \l_stex_symdecl_argnames_clist
2668   \keys_set:nn { stex / textsymdecl } { #1 }
2669 }
2670
2671 \NewDocumentCommand \textsymdecl {m O{} m} {
2672   \_stex_textsymdecl_args:n { #2 }
2673   \str_if_empty:NTF \l__stex_symdecl_name_str {
2674     \__stex_symdecl_args:n{name=#1,#2}
2675   }{
2676     \__stex_symdecl_args:n{#2}
2677   }
2678   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2679   \stex_symdecl_do:n{#1-sym}
2680   \stex_execute_in_module:n{
2681     \cs_set_nopar:cpn{#1name}{
2682       \ifvmode\hbox_unpack:N\c_empty_box\fi
2683       \ifmmode\hbox{#3}\else#3\fi\xspace
2684     }
2685     \cs_set_nopar:cpn{#1}{
2686       \ifmmode\csname#1-sym\expandafter\endcsname\else
```

```
2687        \ifvmode\hbox_unpack:N\c_empty_box\fi
2688        \symref{#1-sym}{#3}\expandafter\xspace
2689        \fi
2690      }
2691    }
2692    \stex_execute_in_module:x{
2693      \__stex_notation_restore_notation:nnnnn
2694      {\l_stex_current_module_str?\tl_if_empty:NTF\l__stex_symdecl_name_str{#1}\l__stex_symdec
2695      {}{0}
2696      {\exp_not:n{\STEXInternalTermMathOMSiiii{\STEXInternalCurrentSymbolStr}{}{\neginfprec}{
2697        \comp{\hbox{#3}}\STEXInternalSymbolAfterInvokationTL
2698      }}}
2699      {}
2700    }
2701    \stex_smsmode_do:
2702 }
```

(*End definition for* \textsymdecl. *This function is documented on page 23.*)

\stex_get_symbol:n

```
2703 \str_new:N \l_stex_get_symbol_uri_str
2704
2705 \cs_new_protected:Nn \stex_get_symbol:n {
2706    \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2707      \tl_set:Nn \l_tmpa_tl { #1 }
2708      \__stex_symdecl_get_symbol_from_cs:
2709    }{
2710      % argument is a string
2711      % is it a command name?
2712      \cs_if_exist:cTF { #1 }{
2713        \cs_set_eq:Nc \l_tmpa_tl { #1 }
2714        \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2715        \str_if_empty:NTF \l_tmpa_str {
2716          \exp_args:Nx \cs_if_eq:NNTF {
2717            \tl_head:N \l_tmpa_tl
2718          } \stex_invoke_symbol:n {
2719            \__stex_symdecl_get_symbol_from_cs:
2720          }{
2721            \__stex_symdecl_get_symbol_from_string:n { #1 }
2722          }
2723        } {
2724          \__stex_symdecl_get_symbol_from_string:n { #1 }
2725        }
2726      }{
2727        % argument is not a command name
2728        \__stex_symdecl_get_symbol_from_string:n { #1 }
2729        % \l_stex_all_symbols_seq
2730      }
2731    }
2732    \str_if_eq:eeF {
2733      \prop_item:cn {
2734        l_stex_symdecl_\l_stex_get_symbol_uri_str _prop
2735      }{ deprecate }
2736    }{}{
```

169

```
2737        \msg_warning:nnxx{stex}{warning/deprecated}{
2738          Symbol~\l_stex_get_symbol_uri_str
2739        }{
2740          \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{ deprecate }
2741        }
2742     }
2743 }
2744
2745 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2746    \tl_set:Nn \l_tmpa_tl {
2747      \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2748    }
2749    \str_set:Nn \l_tmpa_str { #1 }
2750
2751    %\int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2752
2753    \str_if_in:NnTF \l_tmpa_str ? {
2754      \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
2755      \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
2756      \str_set:Nx \l_tmpb_str {\seq_use:Nn \l_tmpa_seq ?}
2757    }{
2758      \str_clear:N \l_tmpb_str
2759    }
2760    \str_if_empty:NTF \l_tmpb_str {
2761      \seq_map_inline:Nn \l_stex_all_modules_seq {
2762        \seq_map_inline:cn{c_stex_module_##1_constants}{
2763          \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2764            \seq_map_break:n{\seq_map_break:n{
2765              \tl_set:Nn \l_tmpa_tl {
2766                \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2767              }
2768            }}
2769          }
2770        }
2771      }
2772    }{
2773      \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpb_str }
2774      \seq_map_inline:Nn \l_stex_all_modules_seq {
2775        \str_if_eq:eeT{ \l_tmpb_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2776          \seq_map_inline:cn{c_stex_module_##1_constants}{
2777            \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2778              \seq_map_break:n{\seq_map_break:n{
2779                \tl_set:Nn \l_tmpa_tl {
2780                  \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2781                }
2782              }}
2783            }
2784          }
2785        }
2786      }
2787    }
2788
2789    \l_tmpa_tl
2790 }
```

```
2791
2792  \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
2793    \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2794      { \tl_tail:N \l_tmpa_tl }
2795    \tl_if_single:NTF \l_tmpa_tl {
2796      \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2797        \exp_after:wN \str_set:Nn \exp_after:wN
2798          \l_stex_get_symbol_uri_str \l_tmpa_tl
2799      }{
2800        % TODO
2801        % tail is not a single group
2802      }
2803    }{
2804      % TODO
2805      % tail is not a single group
2806    }
2807  }
```

(*End definition for* \stex_get_symbol:n. *This function is documented on page* *91.*)

31.2   Notations

2808 ⟨@@=stex_notation⟩

notation arguments:
```
2809  \keys_define:nn { stex / notation } {
2810  % lang    .tl_set_x:N  = \l__stex_notation_lang_str ,
2811    variant .tl_set_x:N  = \l__stex_notation_variant_str ,
2812    prec    .str_set_x:N  = \l__stex_notation_prec_str ,
2813    op      .tl_set:N     = \l__stex_notation_op_tl ,
2814    primary .bool_set:N   = \l__stex_notation_primary_bool ,
2815    primary .default:n    = {true} ,
2816    hints   .str_set_x:N = \l__stex_notation_hints_str,
2817    unknown .code:n       = \str_set:Nx
2818        \l__stex_notation_variant_str \l_keys_key_str
2819  }
2820
2821  \cs_new_protected:Nn \_stex_notation_args:n {
2822  %  \str_clear:N \l__stex_notation_lang_str
2823    \str_clear:N \l__stex_notation_variant_str
2824    \str_clear:N \l__stex_notation_prec_str
2825    \str_clear:N \l__stex_notation_hints_str
2826    \tl_clear:N \l__stex_notation_op_tl
2827    \bool_set_false:N \l__stex_notation_primary_bool
2828
2829    \keys_set:nn { stex / notation } { #1 }
2830  }
```

\notation

```
2831  \NewDocumentCommand \notation { s m O{}} {
2832    \_stex_notation_args:n { #3 }
2833    \tl_clear:N \l_stex_symdecl_definiens_tl
2834    \stex_get_symbol:n { #2 }
2835    \tl_set:Nn \l_stex_notation_after_do_tl {
```

```
2836        \__stex_notation_final:
2837        \IfBooleanTF#1{
2838          \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2839        }{}
2840        \stex_smsmode_do:\ignorespacesandpars
2841     }
2842     \stex_notation_do:nnnnn
2843        { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } }
2844        { \prop_item:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } }
2845        { \l__stex_notation_variant_str }
2846        { \l__stex_notation_prec_str}
2847   }
2848   \stex_deactivate_macro:Nn \notation {module~environments}
```

(*End definition for* `\notation`. *This function is documented on page* *91*.)

\stex_notation_do:nnnnn

```
2849   \seq_new:N \l__stex_notation_precedences_seq
2850   \tl_new:N \l__stex_notation_opprec_tl
2851   \int_new:N \l__stex_notation_currarg_int
2852   \tl_new:N \STEXInternalSymbolAfterInvokationTL
2853
2854   \cs_new_protected:Nn \stex_notation_do:nnnnn {
2855     \let\STEXInternalCurrentSymbolStr\relax
2856     \seq_clear:N \l__stex_notation_precedences_seq
2857     \tl_clear:N \l__stex_notation_opprec_tl
2858     \str_set:Nx \l__stex_notation_args_str { #1 }
2859     \str_set:Nx \l__stex_notation_arity_str { #2 }
2860     \str_set:Nx \l__stex_notation_suffix_str { #3 }
2861     \str_set:Nx \l__stex_notation_prec_str { #4 }
2862
2863     % precedences
2864     \str_if_empty:NTF \l__stex_notation_prec_str {
2865       \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2866         \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2867       }{
2868         \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2869       }
2870     } {
2871       \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2872         \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2873         \int_step_inline:nn { \l__stex_notation_arity_str } {
2874           \exp_args:NNo
2875           \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2876         }
2877       }{
2878         \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2879         \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2880           \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2881           \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2882             \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2883               \l_tmpa_seq {\tl_to_str:n{x} } { \l_tmpa_str }
2884             \seq_map_inline:Nn \l_tmpa_seq {
2885               \seq_put_right:Nn \l__stex_notation_precedences_seq { ##1 }
```

```
2886              }
2887            }
2888          }{
2889            \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2890              \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2891            }{
2892              \tl_set:No \l__stex_notation_opprec_tl { 0 }
2893            }
2894          }
2895        }
2896      }
2897
2898      \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2899      \int_step_inline:nn { \l__stex_notation_arity_str } {
2900        \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
2901          \exp_args:NNo
2902          \seq_put_right:No \l__stex_notation_precedences_seq {
2903            \l__stex_notation_opprec_tl
2904          }
2905        }
2906      }
2907      \tl_clear:N \l_stex_notation_dummyargs_tl
2908
2909      \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2910        \exp_args:NNe
2911        \cs_set:Npn \l_stex_notation_macrocode_cs {
2912          \STEXInternalTermMathOMSiiii { \STEXInternalCurrentSymbolStr }
2913            { \l__stex_notation_suffix_str }
2914            { \l__stex_notation_opprec_tl }
2915            { \exp_not:n { #5 } }
2916        }
2917        \l_stex_notation_after_do_tl
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          \str_if_in:NnTF \l__stex_notation_args_str B {
2931            \exp_args:Nne \use:nn
2932            {
2933            \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2934            \cs_set:Npn \l__stex_notation_arity_str } { {
2935              \STEXInternalTermMathOMBiiii { \STEXInternalCurrentSymbolStr }
2936                { \l__stex_notation_suffix_str }
2937                { \l__stex_notation_opprec_tl }
2938                { \exp_not:n { #5 } }
2939            } }
```

173

```
2940        }{
2941          \exp_args:Nne \use:nn
2942          {
2943          \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2944          \cs_set:Npn \l__stex_notation_arity_str } { {
2945            \STEXInternalTermMathOMAiiii { \STEXInternalCurrentSymbolStr }
2946              { \l__stex_notation_suffix_str }
2947              { \l__stex_notation_opprec_tl }
2948              { \exp_not:n { #5 } } }
2949        } }
2950        }
2951      }
2952
2953      \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2954      \int_zero:N \l__stex_notation_currarg_int
2955      \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2956      \__stex_notation_arguments:
2957    }
2958 }
```

(*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

```
2959 \cs_new_protected:Nn \__stex_notation_arguments: {
2960    \int_incr:N \l__stex_notation_currarg_int
2961    \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2962      \l_stex_notation_after_do_tl
2963    }{
2964      \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2965      \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2966      \str_if_eq:VnTF \l_tmpa_str a {
2967        \__stex_notation_argument_assoc:nn{a}
2968      }{
2969        \str_if_eq:VnTF \l_tmpa_str B {
2970          \__stex_notation_argument_assoc:nn{B}
2971        }{
2972          \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpb_str
2973          \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2974            { \STEXInternalTermMathArgiii
2975              { \l_tmpa_str\int_use:N \l__stex_notation_currarg_int }
2976              { \l_tmpb_str }
2977              { ####\int_use:N \l__stex_notation_currarg_int }
2978          }
2979        }
2980        \__stex_notation_arguments:
2981      }
2982    }
2983  }
2984 }
```

(*End definition for* __stex_notation_arguments:.)

__stex_notation_argument_assoc:nn

```
2985 \cs_new_protected:Nn \__stex_notation_argument_assoc:nn {
```

174

```
2986
2987  \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2988    {\l__stex_notation_arity_str}{
2989    #2
2990  }
2991  \int_zero:N \l_tmpa_int
2992  \tl_clear:N \l_tmpa_tl
2993  \str_map_inline:Nn \l__stex_notation_args_str {
2994    \int_incr:N \l_tmpa_int
2995    \tl_put_right:Nx \l_tmpa_tl {
2996      \str_if_eq:nnTF {##1}{a}{ {} }{
2997        \str_if_eq:nnTF {##1}{B}{ {} }{
2998          {\_stex_term_arg:nn{##1\int_use:N \l_tmpa_int}{############### \int_use:N \l_tmpa
2999        }
3000      }
3001    }
3002  }
3003  \exp_after:wN\exp_after:wN\exp_after:wN \def
3004  \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
3005  \exp_after:wN\exp_after:wN\exp_after:wN ##
3006  \exp_after:wN\exp_after:wN\exp_after:wN 1
3007  \exp_after:wN\exp_after:wN\exp_after:wN ##
3008  \exp_after:wN\exp_after:wN\exp_after:wN 2
3009  \exp_after:wN\exp_after:wN\exp_after:wN {
3010    \exp_after:wN \exp_after:wN \exp_after:wN
3011    \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
3012      \exp_after:wN \l_tmpa_cs \l_tmpa_tl
3013    }
3014  }
3015
3016  \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
3017  \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
3018    \STEXInternalTermMathAssocArgiiiii
3019      { \int_use:N \l__stex_notation_currarg_int }
3020      { \l_tmpa_str }
3021      { ####\int_use:N \l__stex_notation_currarg_int }
3022      { \l_tmpa_cs {####1} {####2} }
3023      {#1}
3024  } }
3025  \__stex_notation_arguments:
3026 }
```

(*End definition for* __stex_notation_argument_assoc:nn.)

__stex_notation_final: Called after processing all notation arguments

```
3027 \cs_new_protected:Nn \__stex_notation_restore_notation:nnnnn {
3028  \cs_generate_from_arg_count:cNnn{stex_notation_\detokenize{#1} \c_hash_str \detokenize{#2}
3029  \cs_set_nopar:Npn {#3}{#4}
3030  \tl_if_empty:nF {#5}{
3031    \tl_set:cn{stex_op_notation_\detokenize{#1} \c_hash_str \detokenize{#2}_cs}{ \comp{ #5 }
3032  }
3033  \seq_if_exist:cT { l_stex_symdecl_\detokenize{#1} _notations }{
3034    \seq_put_right:cx { l_stex_symdecl_\detokenize{#1} _notations } { \detokenize{#2} }
3035  }
```

```
3036 }
3037
3038 \cs_new_protected:Nn \__stex_notation_final: {
3039
3040    \stex_execute_in_module:x {
3041      \__stex_notation_restore_notation:nnnnn
3042        {\l_stex_get_symbol_uri_str}
3043        {\l__stex_notation_suffix_str}
3044        {\l__stex_notation_arity_str}
3045        {
3046          \exp_after:wN \exp_after:wN \exp_after:wN
3047          \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
3048          { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \STEXInt
3049        }
3050        {\exp_args:No \exp_not:n \l__stex_notation_op_tl }
3051    }
3052
3053    \stex_debug:nn{symbols}{
3054      Notation~\l__stex_notation_suffix_str
3055      ~for~\l_stex_get_symbol_uri_str^^J
3056      Operator~precedence:~\l__stex_notation_opprec_tl^^J
3057      Argument~precedences:~
3058        \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
3059      Notation: \cs_meaning:c {
3060        stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
3061        \l__stex_notation_suffix_str
3062        _cs
3063      }
3064    }
3065      % HTML annotations
3066    \stex_if_do_html:T {
3067      \stex_annotate_invisible:nnn { notation }
3068      { \l_stex_get_symbol_uri_str } {
3069        \stex_annotate_invisible:nnn { notationfragment }
3070          { \l__stex_notation_suffix_str }{}
3071        \stex_annotate_invisible:nnn { precedence }
3072          { \l__stex_notation_prec_str }{}
3073
3074      \int_zero:N \l_tmpa_int
3075      \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
3076      \tl_clear:N \l_tmpa_tl
3077      \int_step_inline:nn { \l__stex_notation_arity_str }{
3078        \int_incr:N \l_tmpa_int
3079        \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
3080        \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_rem
3081        \str_if_eq:VnTF \l_tmpb_str a {
3082          \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3083            \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3084            \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3085          } }
3086        }{
3087          \str_if_eq:VnTF \l_tmpb_str B {
3088            \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3089              \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
```

176

```
3090              \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3091           } }
3092        }{
3093          \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3094              \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
3095          } }
3096        }
3097      }
3098    }
3099    \stex_annotate_invisible:nnn { notationcomp }{}{
3100      \str_set:Nx \STEXInternalCurrentSymbolStr {\l_stex_get_symbol_uri_str }
3101      $ \exp_args:Nno \use:nn { \use:c {
3102        stex_notation_ \STEXInternalCurrentSymbolStr
3103        \c_hash_str \l__stex_notation_suffix_str _cs
3104      } } { \l_tmpa_tl } $
3105    }
3106    \tl_if_empty:NF \l__stex_notation_op_tl {
3107      \stex_annotate_invisible:nnn { notationopcomp }{}{
3108        $\l__stex_notation_op_tl$
3109      }
3110    }
3111   }
3112  }
3113 }
```

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

```
3114 \keys_define:nn { stex / setnotation } {
3115 %  lang    .tl_set_x:N  = \l__stex_notation_lang_str ,
3116   variant .tl_set_x:N  = \l__stex_notation_variant_str ,
3117   unknown .code:n      = \str_set:Nx
3118      \l__stex_notation_variant_str \l_keys_key_str
3119 }
3120
3121 \cs_new_protected:Nn \_stex_setnotation_args:n {
3122 % \str_clear:N \l__stex_notation_lang_str
3123   \str_clear:N \l__stex_notation_variant_str
3124   \keys_set:nn { stex / setnotation } { #1 }
3125 }
3126
3127 \cs_new_protected:Nn \__stex_notation_setnotation:nn {
3128   \seq_if_exist:cT{l_stex_symdecl_#1_notations}{
3129     \seq_remove_all:cn { l_stex_symdecl_#1 _notations }{ #2 }
3130     \seq_put_left:cn { l_stex_symdecl_#1 _notations }{ #2 }
3131   }
3132 }
3133
3134 \cs_new_protected:Nn \stex_setnotation:n {
3135   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1 _notations }
3136     { \l__stex_notation_variant_str }{
3137       \stex_execute_in_module:x{ \__stex_notation_setnotation:nn {#1}{\l__stex_notation_vari
3138       \stex_debug:nn {notations}{
3139         Setting~default~notation~
```

177

```
3140          {\l__stex_notation_variant_str }~for~
3141          #1 \\
3142          \expandafter\meaning\csname
3143          l_stex_symdecl_#1 _notations\endcsname
3144        }
3145      }{
3146        \msg_error:nnxx{stex}{unknownnotation}{\l__stex_notation_variant_str}{#1}
3147      }
3148 }
3149
3150 \NewDocumentCommand \setnotation {m m} {
3151   \stex_get_symbol:n { #1 }
3152   \_stex_setnotation_args:n { #2 }
3153   \stex_setnotation:n{\l_stex_get_symbol_uri_str}
3154   \stex_smsmode_do:\ignorespacesandpars
3155 }
3156
3157 \cs_new_protected:Nn \stex_copy_notations:nn {
3158   \stex_debug:nn {notations}{
3159     Copying~notations~from~#2~to~#1\\
3160     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
3161   }
3162   \tl_clear:N \l_tmpa_tl
3163   \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
3164     \tl_put_right:Nn \l_tmpa_tl { {######## ##1} }
3165   }
3166   \seq_map_inline:cn {l_stex_symdecl_#2_notations}{\begingroup
3167     \stex_debug:nn{Here}{Here:~##1}
3168     \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
3169     \edef \l_tmpa_tl {
3170       \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
3171       \exp_after:wN\exp_after:wN\exp_after:wN {
3172         \exp_after:wN \l_tmpa_cs \l_tmpa_tl
3173       }
3174     }
3175
3176     \exp_after:wN \def \exp_after:wN \l_tmpa_tl
3177     \exp_after:wN ####\exp_after:wN 1 \exp_after:wN ####\exp_after:wN 2
3178     \exp_after:wN  { \l_tmpa_tl }
3179
3180     \edef \l_tmpa_tl {
3181       \exp_after:wN \exp_not:n \exp_after:wN {
3182         \l_tmpa_tl {######## 1}{######## 2}
3183       }
3184     }
3185
3186     \stex_debug:nn{Here}{Here:~\expandafter\detokenize\expandafter{\l_tmpa_tl}}
3187
3188     \stex_execute_in_module:x {
3189       \__stex_notation_restore_notation:nnnnn
3190         {#1}{##1}
3191         { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }
3192         { \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl} }
3193         {
```

```
3194            \cs_if_exist:cT{stex_op_notation_ #2\c_hash_str ##1 _cs}{
3195              \exp_args:NNo\exp_args:No\exp_not:n{\csname stex_op_notation_ #2\c_hash_str ##1
3196            }
3197          }
3198      }\endgroup
3199    }
3200 }
3201
3202 \NewDocumentCommand \copynotation {m m} {
3203    \stex_get_symbol:n { #1 }
3204    \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
3205    \stex_get_symbol:n { #2 }
3206    \exp_args:Noo
3207    \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
3208    \stex_smsmode_do:\ignorespacesandpars
3209 }
3210
```

(End definition for \setnotation*. This function is documented on page 23.)*

\symdef

```
3211 \keys_define:nn { stex / symdef } {
3212   name     .str_set_x:N = \l_stex_symdecl_name_str ,
3213   args     .str_set_x:N = \l_stex_symdecl_args_str ,
3214   type     .tl_set:N    = \l_stex_symdecl_type_tl ,
3215   def      .tl_set:N    = \l_stex_symdecl_definiens_tl ,
3216   reorder  .str_set_x:N = \l_stex_symdecl_reorder_str ,
3217   op       .tl_set:N    = \l__stex_notation_op_tl ,
3218 % lang     .str_set_x:N = \l__stex_notation_lang_str ,
3219   variant  .str_set_x:N = \l__stex_notation_variant_str ,
3220   prec     .str_set_x:N = \l__stex_notation_prec_str ,
3221   argnames     .clist_set:N  = \l_stex_symdecl_argnames_clist ,
3222   assoc    .choices:nn =
3223       {bin,binl,binr,pre,conj,pwconj}
3224       {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
3225   unknown .code:n      = \str_set:Nx
3226       \l__stex_notation_variant_str \l_keys_key_str
3227 }
3228
3229 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
3230   \str_clear:N \l_stex_symdecl_name_str
3231   \str_clear:N \l_stex_symdecl_args_str
3232   \str_clear:N \l_stex_symdecl_assoctype_str
3233   \str_clear:N \l_stex_symdecl_reorder_str
3234   \bool_set_false:N \l_stex_symdecl_local_bool
3235   \tl_clear:N \l_stex_symdecl_type_tl
3236   \tl_clear:N \l_stex_symdecl_definiens_tl
3237   \clist_clear:N \l_stex_symdecl_argnames_clist
3238 % \str_clear:N \l__stex_notation_lang_str
3239   \str_clear:N \l__stex_notation_variant_str
3240   \str_clear:N \l__stex_notation_prec_str
3241   \tl_clear:N \l__stex_notation_op_tl
3242
3243   \keys_set:nn { stex / symdef } { #1 }
```

```
3244 }
3245
3246 \NewDocumentCommand \symdef { m O{} } {
3247   \__stex_notation_symdef_args:n { #2 }
3248   \bool_set_true:N \l_stex_symdecl_make_macro_bool
3249   \stex_symdecl_do:n { #1 }
3250   \tl_set:Nn \l_stex_notation_after_do_tl {
3251     \__stex_notation_final:
3252     \stex_smsmode_do:\ignorespacesandpars
3253   }
3254   \str_set:Nx \l_stex_get_symbol_uri_str {
3255     \l_stex_current_module_str ? \l_stex_symdecl_name_str
3256   }
3257   \exp_args:Nx \stex_notation_do:nnnnn
3258     { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } } }
3259     { \prop_item:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } } }
3260     { \l__stex_notation_variant_str }
3261     { \l__stex_notation_prec_str}
3262 }
3263 \stex_deactivate_macro:Nn \symdef {module~environments}
3264
3265 \keys_define:nn { stex / mmtdef } {
3266   name     .str_set_x:N = \l_stex_symdecl_name_str ,
3267   args     .str_set_x:N = \l_stex_symdecl_args_str ,
3268   reorder .str_set_x:N = \l_stex_symdecl_reorder_str ,
3269   op       .tl_set:N    = \l__stex_notation_op_tl ,
3270 % lang     .str_set_x:N = \l__stex_notation_lang_str ,
3271   variant .str_set_x:N = \l__stex_notation_variant_str ,
3272   prec     .str_set_x:N = \l__stex_notation_prec_str ,
3273   argnames     .clist_set:N  = \l_stex_symdecl_argnames_clist ,
3274   assoc    .choices:nn  =
3275       {bin,binl,binr,pre,conj,pwconj}
3276       {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
3277   unknown .code:n      = \str_set:Nx
3278       \l__stex_notation_variant_str \l_keys_key_str
3279 }
3280 \cs_new_protected:Nn \_stex_mmtdef_args:n {
3281   \str_clear:N \l_stex_symdecl_name_str
3282   \str_clear:N \l_stex_symdecl_args_str
3283   \str_clear:N \l_stex_symdecl_assoctype_str
3284   \str_clear:N \l_stex_symdecl_reorder_str
3285   \bool_set_false:N \l_stex_symdecl_local_bool
3286   \clist_clear:N \l_stex_symdecl_argnames_clist
3287 % \str_clear:N \l__stex_notation_lang_str
3288   \str_clear:N \l__stex_notation_variant_str
3289   \str_clear:N \l__stex_notation_prec_str
3290   \tl_clear:N \l__stex_notation_op_tl
3291
3292   \keys_set:nn { stex / mmtdef } { #1 }
3293 }
3294
3295 \NewDocumentCommand \mmtdef {m O{} }{
3296   \_stex_mmtdef_args:n{ #2 }
3297   \bool_set_true:N \l_stex_symdecl_make_macro_bool
```

```
3298    \str_if_empty:NT \l_stex_symdecl_name_str {
3299      \str_set:Nx \l_stex_symdecl_name_str { #1 }
3300    }
3301    %\tl_set:Nx \l_stex_symdecl_definiens_tl {
3302    %  \stex_annotate:nnn{ OMID }{
3303    %    \l_stex_module_mmtfor_str?\l_stex_symdecl_name_str
3304    %  }{}
3305    %}
3306    \stex_symdecl_do:n { #1 }
3307    \MMTrule{rules.stex.mmt.kwarc.info?SubstitutionRule}{
3308      \stex_annotate:nnn{ OMID }{
3309        \l_stex_current_module_str ? \l_stex_symdecl_name_str
3310      }{},
3311      \stex_annotate:nnn{ OMID }{
3312        \l_stex_module_mmtfor_str?\l_stex_symdecl_name_str
3313      }{}
3314    }
3315    \tl_set:Nn \l_stex_notation_after_do_tl {
3316      \__stex_notation_final:
3317      \stex_smsmode_do:\ignorespacesandpars
3318    }
3319    \str_set:Nx \l_stex_get_symbol_uri_str {
3320      \l_stex_current_module_str ? \l_stex_symdecl_name_str
3321    }
3322    \exp_args:Nx \stex_notation_do:nnnnn
3323      { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } }
3324      { \prop_item:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } }
3325      { \l__stex_notation_variant_str }
3326      { \l__stex_notation_prec_str}
3327  }
```

(*End definition for* `\symdef`*. This function is documented on page* *91*.)

31.3 Variables

```
3328  ⟨@@=stex_variables⟩
3329
3330  \keys_define:nn { stex / vardef } {
3331    name    .str_set_x:N  = \l__stex_variables_name_str ,
3332    args    .str_set_x:N  = \l__stex_variables_args_str ,
3333    type    .tl_set:N     = \l__stex_variables_type_tl ,
3334    def     .tl_set:N     = \l__stex_variables_def_tl ,
3335    op      .tl_set:N     = \l__stex_variables_op_tl ,
3336    prec    .str_set_x:N  = \l__stex_variables_prec_str ,
3337    reorder .str_set_x:N  = \l__stex_variables_reorder_str ,
3338    argnames    .clist_set:N  = \l__stex_variables_argnames_clist ,
3339    assoc   .choices:nn  =
3340        {bin,binl,binr,pre,conj,pwconj}
3341        {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
3342    bind    .choices:nn   =
3343        {forall,exists}
3344        {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
3345  }
3346
```

```
3347 \cs_new_protected:Nn \__stex_variables_args:n {
3348   \str_clear:N \l__stex_variables_name_str
3349   \str_clear:N \l__stex_variables_args_str
3350   \str_clear:N \l__stex_variables_prec_str
3351   \str_clear:N \l__stex_variables_assoctype_str
3352   \str_clear:N \l__stex_variables_reorder_str
3353   \str_clear:N \l__stex_variables_bind_str
3354   \tl_clear:N \l__stex_variables_type_tl
3355   \tl_clear:N \l__stex_variables_def_tl
3356   \tl_clear:N \l__stex_variables_op_tl
3357   \clist_clear:N \l__stex_variables_argnames_clist
3358
3359   \keys_set:nn { stex / vardef } { #1 }
3360 }
3361
3362 \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{}} {
3363   \__stex_variables_args:n {#2}
3364   \str_if_empty:NT \l__stex_variables_name_str {
3365     \str_set:Nx \l__stex_variables_name_str { #1 }
3366   }
3367   \prop_clear:N \l_tmpa_prop
3368   \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
3369
3370   \int_zero:N \l_tmpb_int
3371   \bool_set_true:N \l_tmpa_bool
3372   \str_map_inline:Nn \l__stex_variables_args_str {
3373     \token_case_meaning:NnF ##1 {
3374       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
3375       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
3376       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
3377       {\tl_to_str:n a} {
3378         \bool_set_false:N \l_tmpa_bool
3379         \int_incr:N \l_tmpb_int
3380       }
3381       {\tl_to_str:n B} {
3382         \bool_set_false:N \l_tmpa_bool
3383         \int_incr:N \l_tmpb_int
3384       }
3385     }{
3386       \msg_error:nnxx{stex}{error/wrongargs}{
3387         variable~\l__stex_variables_name_str
3388       }{##1}
3389     }
3390   }
3391   \bool_if:NTF \l_tmpa_bool {
3392     % possibly numeric
3393     \str_if_empty:NTF \l__stex_variables_args_str {
3394       \prop_put:Nnn \l_tmpa_prop { args } {}
3395       \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
3396     }{
3397       \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
3398       \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
3399       \str_clear:N \l_tmpa_str
3400       \int_step_inline:nn \l_tmpa_int {
```

```
3401        \str_put_right:Nn \l_tmpa_str i
3402      }
3403      \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
3404      \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
3405    }
3406  } {
3407    \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
3408    \prop_put:Nnx \l_tmpa_prop { arity }
3409      { \str_count:N \l__stex_variables_args_str }
3410  }
3411  \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
3412  \tl_set:cx { #1 }{ \stex_invoke_variable:n { \l__stex_variables_name_str } }
3413
3414  % argnames
3415
3416  \clist_clear:N \l_tmpa_clist
3417  \int_step_inline:nn {\prop_item:Nn \l_tmpa_prop {arity}} {
3418    \clist_if_empty:NTF \l__stex_variables_argnames_clist {
3419      \clist_put_right:Nn \l_tmpa_clist {##1}
3420    }{
3421      \clist_pop:NN \l__stex_variables_argnames_clist \l_tmpa_tl
3422      \exp_args:NNx \clist_put_right:Nn \l_tmpa_clist {\c_dollar_str\l_tmpa_tl}
3423    }
3424  }
3425  \prop_put:Nnx \l_tmpa_prop {argnames} {\clist_use:Nn \l_tmpa_clist ,}
3426
3427
3428  \prop_set_eq:cN { l_stex_symdecl_var://\l__stex_variables_name_str _prop} \l_tmpa_prop
3429
3430  \tl_if_empty:NF \l__stex_variables_op_tl {
3431    \cs_set:cpx {
3432      stex_var_op_notation_ \l__stex_variables_name_str _cs
3433    } { \exp_not:N\comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } } }
3434  }
3435
3436  \tl_set:Nn \l_stex_notation_after_do_tl {
3437    \exp_args:Nne \use:nn {
3438      \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str _cs }
3439        \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
3440    } {{
3441      \exp_after:wN \exp_after:wN \exp_after:wN
3442      \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
3443      { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \STEXInter
3444    }}
3445    \stex_if_do_html:T {
3446      \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
3447        \stex_annotate_invisible:nnn { precedence }
3448          { \l__stex_variables_prec_str }{}
3449        \tl_if_empty:NF \l__stex_variables_type_tl {\stex_annotate_invisible:nnn{type}{}{$\l
3450        \stex_annotate_invisible:nnn{args}{ \l__stex_variables_args_str }{}
3451        \stex_annotate_invisible:nnn{macroname}{#1}{}
3452        \tl_if_empty:NF \l__stex_variables_def_tl {
3453          \stex_annotate_invisible:nnn{definiens}{}
3454            {$\l__stex_variables_def_tl$}
```

183

```
3455            }
3456          \str_if_empty:NF \l__stex_variables_assoctype_str {
3457            \stex_annotate_invisible:nnn{assoctype}{\l__stex_variables_assoctype_str}{}
3458          }
3459          \str_if_empty:NF \l__stex_variables_reorder_str {
3460            \stex_annotate_invisible:nnn{reorderargs}{\l__stex_variables_reorder_str}{}
3461          }
3462          \int_zero:N \l_tmpa_int
3463          \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
3464          \tl_clear:N \l_tmpa_tl
3465          \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
3466            \int_incr:N \l_tmpa_int
3467            \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
3468            \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables
3469            \str_if_eq:VnTF \l_tmpb_str a {
3470              \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3471                \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3472                \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3473              } }
3474            }{
3475              \str_if_eq:VnTF \l_tmpb_str B {
3476                \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3477                  \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3478                  \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3479                } }
3480              }{
3481                \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3482                  \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
3483                } }
3484              }
3485            }
3486          }
3487          \stex_annotate_invisible:nnn { notationcomp }{}{
3488            \str_set:Nx \STEXInternalCurrentSymbolStr {var://\l__stex_variables_name_str }
3489            $ \exp_args:Nno \use:nn { \use:c {
3490              stex_var_notation_\l__stex_variables_name_str _cs
3491            } } { \l_tmpa_tl } $
3492          }
3493          \tl_if_empty:NF \l__stex_variables_op_tl {
3494            \stex_annotate_invisible:nnn { notationopcomp }{}{
3495              $\l__stex_variables_op_tl$
3496            }
3497          }
3498        }
3499        \str_if_empty:NF \l__stex_variables_bind_str {
3500          \stex_annotate_invisible:nnn {bindtype}{\l__stex_variables_bind_str,\l__stex_variabl
3501        }
3502      }\ignorespacesandpars
3503    }
3504
3505    \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
3506 }
3507
3508 \cs_new:Nn \_stex_reset:N {
```

```
3509    \tl_if_exist:NTF #1 {
3510        \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
3511    }{
3512        \let \exp_not:N #1 \exp_not:N \undefined
3513    }
3514 }
3515
3516 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
3517    \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
3518    \exp_args:Nnx \use:nn {
3519        % TODO
3520        \stex_annotate_invisible:nnn {vardecl}{\clist_use:Nn\l__stex_variables_names,}{
3521            #2
3522        }
3523    }{
3524        \_stex_reset:N \varnot
3525        \_stex_reset:N \vartype
3526        \_stex_reset:N \vardefi
3527    }
3528 }
3529
3530 \NewDocumentCommand \vardef { s } {
3531    \IfBooleanTF#1 {
3532        \__stex_variables_do_complex:nn
3533    }{
3534        \__stex_variables_do_simple:nnn
3535    }
3536 }
3537
3538 \NewDocumentCommand \svar { O{} m }{
3539    \tl_if_empty:nTF {#1}{
3540        \str_set:Nn \l_tmpa_str { #2 }
3541    }{
3542        \str_set:Nn \l_tmpa_str { #1 }
3543    }
3544    \_stex_term_omv:nn {
3545        var://\l_tmpa_str
3546    }{
3547        \exp_args:Nnx \use:nn {
3548            \def\comp{\_varcomp}
3549            \str_set:Nx \STEXInternalCurrentSymbolStr { var://\l_tmpa_str }
3550            \comp{ #2 }
3551        }{
3552            \_stex_reset:N \comp
3553            \_stex_reset:N \STEXInternalCurrentSymbolStr
3554        }
3555    }
3556 }
3557
3558
3559
3560 \keys_define:nn { stex / varseq } {
3561    name      .str_set_x:N  = \l__stex_variables_name_str ,
3562    args      .int_set:N    = \l__stex_variables_args_int ,
```

```
3563    type     .tl_set:N     = \l__stex_variables_type_tl   ,
3564    mid      .tl_set:N     = \l__stex_variables_mid_tl    ,
3565    bind     .choices:nn   =
3566        {forall,exists}
3567        {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
3568  }
3569
3570  \cs_new_protected:Nn \__stex_variables_seq_args:n {
3571    \str_clear:N \l__stex_variables_name_str
3572    \int_set:Nn \l__stex_variables_args_int 1
3573    \tl_clear:N \l__stex_variables_type_tl
3574    \str_clear:N \l__stex_variables_bind_str
3575
3576    \keys_set:nn { stex / varseq } { #1 }
3577  }
3578
3579  \NewDocumentCommand \varseq {m O{} m m m}{
3580    \__stex_variables_seq_args:n { #2 }
3581    \str_if_empty:NT \l__stex_variables_name_str {
3582      \str_set:Nx \l__stex_variables_name_str { #1 }
3583    }
3584    \prop_clear:N \l_tmpa_prop
3585    \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
3586
3587    \seq_set_from_clist:Nn \l_tmpa_seq {#3}
3588    \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
3589      \msg_error:nnxx{stex}{error/seqlength}
3590        {\int_use:N \l__stex_variables_args_int}
3591        {\seq_count:N \l_tmpa_seq}
3592    }
3593    \seq_set_from_clist:Nn \l_tmpb_seq {#4}
3594    \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
3595      \msg_error:nnxx{stex}{error/seqlength}
3596        {\int_use:N \l__stex_variables_args_int}
3597        {\seq_count:N \l_tmpb_seq}
3598    }
3599    \prop_put:Nnn \l_tmpa_prop {starts} {#3}
3600    \prop_put:Nnn \l_tmpa_prop {ends} {#4}
3601
3602    \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
3603      \cs_set:Npn  {\int_use:N \l__stex_variables_args_int} { #5 }
3604
3605    % argnames
3606
3607    \clist_clear:N \l_tmpa_clist
3608    \int_step_inline:nn {\l__stex_variables_args_int} {
3609        \clist_put_right:Nn \l_tmpa_clist {##1}
3610    }
3611    \prop_put:Nnx \l_tmpa_prop {argnames} {\clist_use:Nn \l_tmpa_clist ,}
3612
3613
3614
3615
3616    \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
```

```
3617    \int_step_inline:nn \l__stex_variables_args_int {
3618      \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
3619    }
3620    \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
3621    \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
3622    \tl_if_empty:NF \l__stex_variables_mid_tl {
3623      \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
3624      \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
3625    }
3626    \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3627    \int_step_inline:nn \l__stex_variables_args_int {
3628      \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
3629    }
3630    \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
3631    \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
3632

3633

3634    \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl

3635

3636    \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}

3637

3638    \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}

3639

3640    \int_step_inline:nn \l__stex_variables_args_int {
3641      \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
3642        \STEXInternalTermMathArgiii{i##1}{0}{\exp_not:n{####}##1}
3643      }}
3644    }

3645

3646    \tl_set:Nx \l_tmpa_tl {
3647      \STEXInternalTermMathOMAiiii { varseq://\l__stex_variables_name_str}{}{0}{
3648        \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
3649      }
3650    }

3651

3652    \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \STEXInternalSymbolAfterInvokationTL} }

3653

3654    \exp_args:Nno \use:nn {
3655    \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
3656      \cs_set:Npn {\int_use:N \l__stex_variables_args_int}}{\l_tmpa_tl}

3657

3658    \stex_debug:nn{sequences}{New~Sequence:~
3659      \expandafter\meaning\csname stex_varseq_\l__stex_variables_name_str _cs\endcsname\\~\\
3660      \prop_to_keyval:N \l_tmpa_prop
3661    }
3662    \prop_set_eq:cN {l_stex_symdecl_varseq://\l__stex_variables_name_str _prop}\l_tmpa_prop

3663

3664    \stex_if_do_html:T{\stex_annotate_invisible:nnn{varseq}{\l__stex_variables_name_str}{
3665      \tl_if_empty:NF \l__stex_variables_type_tl {
3666        \stex_annotate:nnn {type}{}{$\l__stex_variables_type_tl$}
3667      }
3668      \stex_annotate:nnn {args}{\int_use:N \l__stex_variables_args_int}{}
3669      \str_if_empty:NF \l__stex_variables_bind_str {
3670        \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{}
```

```
3671        }
3672      \stex_annotate:nnn{startindex}{}{$#3$}
3673      \stex_annotate:nnn{endindex}{}{$#4$}
3674
3675      \tl_clear:N \l_tmpa_tl
3676      \int_step_inline:nn \l__stex_variables_args_int {
3677        \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3678          \stex_annotate:nnn{argmarker}{##1}{}
3679        } }
3680      }
3681      \stex_annotate_invisible:nnn { notationcomp }{}{
3682        \str_set:Nx \STEXInternalCurrentSymbolStr {varseq://\l__stex_variables_name_str }
3683        $ \exp_args:Nno \use:nn { \use:c {
3684          stex_varseq_\l__stex_variables_name_str _cs
3685        } } { \l_tmpa_tl } $
3686      }
3687      \stex_annotate_invisible:nnn { notationopcomp }{}{
3688        $ \prop_item:Nn \l_tmpa_prop { notation } $
3689      }
3690
3691  }}
3692
3693    \ignorespacesandpars
3694  }
3695
3696
3697  \keys_define:nn { stex / mmtdecl } {
3698    name        .str_set_x:N  = \l_stex_symdecl_name_str ,
3699    args        .str_set_x:N  = \l_stex_symdecl_args_str ,
3700    deprecate   .str_set_x:N  = \l_stex_symdecl_deprecate_str ,
3701    reorder     .str_set_x:N  = \l_stex_symdecl_reorder_str ,
3702    argnames    .clist_set:N  = \l_stex_symdecl_argnames_clist ,
3703    assoc       .choices:nn   =
3704      {bin,binl,binr,pre,conj,pwconj}
3705      {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}
3706  }
3707
3708  \cs_new_protected:Nn \_stex_mmtdecl_args:n {
3709    \str_clear:N \l_stex_symdecl_name_str
3710    \str_clear:N \l_stex_symdecl_args_str
3711    \str_clear:N \l_stex_symdecl_deprecate_str
3712    \str_clear:N \l_stex_symdecl_reorder_str
3713    \str_clear:N \l_stex_symdecl_assoctype_str
3714    \bool_set_false:N \l_stex_symdecl_local_bool
3715    \clist_clear:N \l_stex_symdecl_argnames_clist
3716
3717    \keys_set:nn { stex / symdecl } { #1 }
3718  }
3719
3720  \NewDocumentCommand \mmtdecl { s m O{}} {
3721    \_stex_mmtdecl_args:n{#3}
3722    \IfBooleanTF #1 {
3723      \bool_set_false:N \l_stex_symdecl_make_macro_bool
3724    } {
```

```
3725        \bool_set_true:N \l_stex_symdecl_make_macro_bool
3726    }
3727    \str_if_empty:NT \l_stex_symdecl_name_str {
3728        \str_set:Nx \l_stex_symdecl_name_str { #1 }
3729    }
3730    %\tl_set:Nx \l_stex_symdecl_definiens_tl {
3731    %  \stex_annotate:nnn{ OMID }{
3732    %    \l_stex_module_mmtfor_str?\l_stex_symdecl_name_str
3733    %  }{}
3734    %}
3735    \stex_symdecl_do:n{#2}
3736    \MMTrule{rules.stex.mmt.kwarc.info?SubstitutionRule}{
3737        \stex_annotate:nnn{ OMID }{
3738            \l_stex_current_module_str ? \l_stex_symdecl_name_str
3739        }{},
3740        \stex_annotate:nnn{ OMID }{
3741            \l_stex_module_mmtfor_str?\l_stex_symdecl_name_str
3742        }{}
3743    }
3744    \stex_smsmode_do:
3745 }
3746
3747 \stex_deactivate_macro:Nn \mmtdecl {mmtinterface~environments}
3748 \stex_deactivate_macro:Nn \mmtdef {mmtinterface~environments}
3749
3750 ⟨/package⟩
```

Chapter 32

STEX
-Terms Implementation

```
3751 ⟨∗package⟩
3752
3753 %%%%%%%%%%%%   terms.dtx   %%%%%%%%%%%%
3754
3755 ⟨@@=stex_terms⟩
```

Warnings and error messages

```
3756 \msg_new:nnn{stex}{error/nonotation}{
3757   Symbol~#1~invoked,~but~has~no~notation#2!
3758 }
3759 \msg_new:nnn{stex}{error/notationarg}{
3760   Error~in~parsing~notation~#1
3761 }
3762 \msg_new:nnn{stex}{error/noop}{
3763   Symbol~#1~has~no~operator~notation~for~notation~#2
3764 }
3765 \msg_new:nnn{stex}{error/notallowed}{
3766   Symbol~invokation~#1~not~allowed~in~notation~component~of~#2
3767 }
3768 \msg_new:nnn{stex}{error/doubleargument}{
3769   Argument~#1~of~symbol~#2~already~assigned
3770 }
3771 \msg_new:nnn{stex}{error/overarity}{
3772   Argument~#1~invalid~for~symbol~#2~with~arity~#3
3773 }
3774
```

32.1   Symbol Invocations

`\stex_invoke_symbol:n`   Invokes a semantic macro

```
3775
3776
3777 \bool_new:N \l_stex_allow_semantic_bool
3778 \bool_set_true:N \l_stex_allow_semantic_bool
3779
```

```
3780 \cs_new_protected:Nn \stex_invoke_symbol:n {
3781   \ifvmode\indent\fi
3782   \bool_if:NTF \l_stex_allow_semantic_bool {
3783     \str_if_eq:eeF {
3784       \prop_item:cn {
3785         l_stex_symdecl_#1_prop
3786       }{ deprecate }
3787     }{}{
3788       \msg_warning:nnxx{stex}{warning/deprecated}{
3789         Symbol~#1
3790       }{
3791         \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
3792       }
3793     }
3794     \if_mode_math:
3795       \exp_after:wN \__stex_terms_invoke_math:n
3796     \else:
3797       \exp_after:wN \__stex_terms_invoke_text:n
3798     \fi: { #1 }
3799   }{
3800     \msg_error:nnxx{stex}{error/notallowed}{#1}{\STEXInternalCurrentSymbolStr}
3801   }
3802 }
3803
3804 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
3805   \peek_charcode_remove:NTF ! {
3806     \__stex_terms_invoke_op_custom:nn {#1}
3807   }{
3808     \__stex_terms_invoke_custom:nn {#1}
3809   }
3810 }
3811
3812 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
3813   \peek_charcode_remove:NTF ! {
3814     % operator
3815     \peek_charcode_remove:NTF * {
3816       % custom op
3817       \__stex_terms_invoke_op_custom:nn {#1}
3818     }{
3819       % op notation
3820       \peek_charcode:NTF [ {
3821         \__stex_terms_invoke_op_notation:nw {#1}
3822       }{
3823         \__stex_terms_invoke_op_notation:nw {#1}[]
3824       }
3825     }
3826   }{
3827     \peek_charcode_remove:NTF * {
3828       \__stex_terms_invoke_custom:nn {#1}
3829       % custom
3830     }{
3831       % normal
3832       \peek_charcode:NTF [ {
3833         \__stex_terms_invoke_notation:nw {#1}
```

```
3834        }{
3835            \__stex_terms_invoke_notation:nw {#1}[]
3836        }
3837      }
3838   }
3839 }
3840
3841
3842 \cs_new_protected:Nn \__stex_terms_invoke_op_custom:nn {
3843   \exp_args:Nnx \use:nn {
3844     \def\comp{\_comp}
3845     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3846     \bool_set_false:N \l_stex_allow_semantic_bool
3847     \stex_mathml_intent:nn{#1}{
3848       \_stex_term_oms:nnn {#1}{#1 \c_hash_str CUSTOM-}{
3849         \comp{ #2 }
3850       }
3851     }
3852   }{
3853     \_stex_reset:N \comp
3854     \_stex_reset:N \STEXInternalCurrentSymbolStr
3855     \bool_set_true:N \l_stex_allow_semantic_bool
3856   }
3857 }
3858
3859 \keys_define:nn { stex / terms } {
3860 %  lang     .tl_set_x:N = \l_stex_notation_lang_str ,
3861   variant .tl_set_x:N = \l_stex_notation_variant_str ,
3862   unknown .code:n      = \str_set:Nx
3863       \l_stex_notation_variant_str \l_keys_key_str
3864 }
3865
3866 \cs_new_protected:Nn \__stex_terms_args:n {
3867 % \str_clear:N \l_stex_notation_lang_str
3868   \str_clear:N \l_stex_notation_variant_str
3869
3870   \keys_set:nn { stex / terms } { #1 }
3871 }
3872
3873 \cs_new_protected:Nn \stex_find_notation:nn {
3874   \__stex_terms_args:n { #2 }
3875   \seq_if_empty:cTF {
3876     l_stex_symdecl_ #1 _notations
3877   } {
3878     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
3879   } {
3880     \str_if_empty:NTF \l_stex_notation_variant_str {
3881       \seq_get_left:cN {l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
3882     }{
3883       \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
3884         \l_stex_notation_variant_str
3885       }{
3886       %  \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3887       }{
```

```
3888        \msg_error:nnxx{stex}{error/nonotation}{#1}{
3889          ~\l_stex_notation_variant_str
3890        }
3891      }
3892    }
3893  }
3894 }
3895
3896 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3897   \exp_args:Nnx \use:nn {
3898     \def\comp{\_comp}
3899     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3900     \stex_find_notation:nn { #1 }{ #2 }
3901     \bool_set_false:N \l_stex_allow_semantic_bool
3902     \cs_if_exist:cTF {
3903       stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3904     }{
3905       \_stex_term_oms:nnn { #1 }{
3906         #1 \c_hash_str \l_stex_notation_variant_str
3907       }{
3908         \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3909       }
3910     }{
3911       \int_compare:nNnTF {\prop_item:cn {l_stex_symdecl_#1_prop}{arity}}} = 0{
3912         \cs_if_exist:cTF {
3913           stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3914         }{
3915           \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
3916             \_stex_reset:N \comp
3917             \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
3918             \_stex_reset:N \STEXInternalCurrentSymbolStr
3919             \bool_set_true:N \l_stex_allow_semantic_bool
3920           }
3921           \def\comp{\_comp}
3922           \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3923           \bool_set_false:N \l_stex_allow_semantic_bool
3924           \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3925         }{
3926           \msg_error:nnxx{stex}{error/nonotation}{#1}{
3927             ~\l_stex_notation_variant_str
3928           }
3929         }
3930       }{
3931         \msg_error:nnxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3932       }
3933     }
3934   }{
3935     \_stex_reset:N \comp
3936     \_stex_reset:N \STEXInternalCurrentSymbolStr
3937     \bool_set_true:N \l_stex_allow_semantic_bool
3938   }
3939 }
3940
3941 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {
```

193

```
3942    \stex_find_notation:nn { #1 }{ #2 }
3943    \cs_if_exist:cTF {
3944      stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3945    }{
3946      \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
3947        \_stex_reset:N \comp
3948        \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
3949        \_stex_reset:N \STEXInternalCurrentSymbolStr
3950        \bool_set_true:N \l_stex_allow_semantic_bool
3951      }
3952      \def\comp{\_comp}
3953      \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3954      \bool_set_false:N \l_stex_allow_semantic_bool
3955      \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3956    }{
3957      \msg_error:nnxx{stex}{error/nonotation}{#1}{
3958        ~\l_stex_notation_variant_str
3959      }
3960    }
3961  }
3962
3963  \prop_new:N \l__stex_terms_custom_args_prop
3964  \clist_new:N \l_stex_argnames_seq
3965  \seq_new:N \l__stex_terms_tmp_seq
3966
3967  \cs_new_protected:Nn\__stex_terms_custom_comp:n{\bool_set_false:N \l_stex_allow_semantic_boo
3968
3969  \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3970    \exp_args:Nnx \use:nn {
3971      \def\comp{\__stex_terms_custom_comp:n}
3972      \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3973      \prop_clear:N \l__stex_terms_custom_args_prop
3974      \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3975      \prop_get:cnN {
3976        l_stex_symdecl_#1 _prop
3977      }{ args } \l_tmpa_str
3978      \exp_args:NNx \seq_set_from_clist:Nn \l_stex_argnames_seq {
3979        \prop_item:cn {l_stex_symdecl_#1 _prop}{argnames}
3980      }
3981      \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3982      \tl_set:Nn \arg { \__stex_terms_arg: }
3983      \str_if_empty:NTF \l_tmpa_str {
3984        \stex_mathml_intent:nn{#1}{
3985          \_stex_term_oms:nnn {#1}{#1\c_hash_str CUSTOM-}{\ignorespaces#2}
3986        }
3987      }{
3988        \seq_clear:N \l__stex_terms_tmp_seq
3989        \exp_args:Nx\int_step_inline:nn{\prop_item:cn{l_stex_symdecl_#1 _prop}{arity}}{
3990          \tl_set:Nx \l__stex_terms_tmp_tl {\seq_item:Nn \l_stex_argnames_seq {##1}}
3991          \bool_lazy_or:nnT{
3992            \str_if_eq_p:nn{a}{\str_item:Nn\l_tmpa_str{##1}}
3993          }{
3994            \str_if_eq_p:nn{B}{\str_item:Nn\l_tmpa_str{##1}}
3995          }{
```

194

```
3996           \tl_put_right:Nn \l__stex_terms_tmp_tl +
3997         }
3998         \seq_put_right:No \l__stex_terms_tmp_seq \l__stex_terms_tmp_tl
3999       }
4000       \stex_mathml_intent:nn{
4001         #1[\prop_item:cn {l_stex_symdecl_#1 _prop}{ args }](
4002           \seq_use:Nn \l__stex_terms_tmp_seq ,
4003         )
4004       }{
4005         \str_if_in:NnTF \l_tmpa_str b {
4006           \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
4007         }{
4008           \str_if_in:NnTF \l_tmpa_str B {
4009             \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
4010           }{
4011             \_stex_term_oma:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
4012           }
4013         }
4014       }
4015     }
4016     % TODO check that all arguments exist
4017   }{
4018     \_stex_reset:N \l_stex_argnames_seq
4019     \_stex_reset:N \STEXInternalCurrentSymbolStr
4020     \_stex_reset:N \arg
4021     \_stex_reset:N \comp
4022     \_stex_reset:N \l__stex_terms_custom_args_prop
4023     %\bool_set_true:N \l_stex_allow_semantic_bool
4024   }
4025 }
4026
4027 \NewDocumentCommand \__stex_terms_arg: { s O{} m}{
4028   \tl_if_empty:nTF {#2}{
4029     \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
4030     \bool_set_true:N \l_tmpa_bool
4031     \bool_do_while:Nn \l_tmpa_bool {
4032       \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int
4033         \int_incr:N \l_tmpa_int
4034       }{
4035         \bool_set_false:N \l_tmpa_bool
4036       }
4037     }
4038   }{
4039     \int_set:Nn \l_tmpa_int { #2 }
4040   }
4041   \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
4042   \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
4043     \msg_error:nnxxx{stex}{error/overarity}
4044       {\int_use:N \l_tmpa_int}
4045       {\STEXInternalCurrentSymbolStr}
4046       {\str_count:N \l_tmpa_str}
4047   }
4048   \str_set:Nx \l_tmpa_str {\str_item:Nn \l_tmpa_str \l_tmpa_int}
4049   \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
```

```
4050        \bool_lazy_any:nF {
4051          {\str_if_eq_p:Vn \l_tmpa_str {a}}
4052          {\str_if_eq_p:Vn \l_tmpa_str {B}}
4053        }{
4054          \msg_error:nnxx{stex}{error/doubleargument}
4055            {\int_use:N \l_tmpa_int}
4056            {\STEXInternalCurrentSymbolStr}
4057        }
4058      }
4059      \exp_args:NNx \prop_put:Nnn \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {\ign
4060      \bool_if:NTF \l_stex_allow_semantic_bool \use_i:nn {
4061        \bool_set_true:N \l_stex_allow_semantic_bool
4062        \use:nn
4063      }
4064      {
4065      \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq \l_tmpa_int}{
4066        \IfBooleanTF#1{
4067          \stex_annotate_invisible:n { %TODO
4068            \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{\ignorespaces#3}
4069          }
4070        }{ %TODO
4071          \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{\ignorespaces#3}
4072        }
4073      }}
4074      {\bool_set_false:N \l_stex_allow_semantic_bool}
4075  }
4076
4077
4078  \cs_new_protected:Nn \_stex_term_arg:nn {
4079    \bool_set_true:N \l_stex_allow_semantic_bool
4080    \stex_annotate:nnn{ arg }{ #1 }{ #2 }
4081    \bool_set_false:N \l_stex_allow_semantic_bool
4082  }
4083
4084  \cs_new_protected:Npn \STEXInternalTermMathArgiii #1#2#3 {
4085    \exp_args:Nnx \use:nn
4086      { \int_set:Nn \l__stex_terms_downprec { #2 }
4087        \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq \l_tmpa_int}{
4088          \_stex_term_arg:nn { #1 }{ #3 }
4089        }
4090      }
4091      { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
4092  }
```

(*End definition for* \stex_invoke_symbol:n. *This function is documented on page 92.*)

```
4093  \cs_new_protected:Npn \STEXInternalTermMathAssocArgiiiii #1#2#3#4#5 {
4094    \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
4095    \tl_set:Nn \l_tmpb_tl {\STEXInternalTermMathArgiii{#5#1}{#2}}
4096    \tl_if_empty:nTF { #3 }{
4097      \STEXInternalTermMathArgiii{#5#1}{#2}{}
4098    }{
4099      \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 }}{
```

```
      \expandafter\if\expandafter\relax\noexpand#3
        \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_maybe_sequence:Nnn#3{#1}{#5}}
      \else
        \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_simple:nnn{#1}{#3}{#5}}
      \fi
      \l_tmpa_tl
    }{
      \__stex_terms_math_assoc_arg_simple:nnn{#1}{#3}{#5}
    }
  }
}

\cs_new_protected:Nn \__stex_terms_math_assoc_arg_maybe_sequence:Nnn {
  \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
  \str_if_empty:NTF \l_tmpa_str {
    \exp_args:Nx \cs_if_eq:NNTF {
      \tl_head:N #1
    } \stex_invoke_sequence:n {
      \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
      \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
      \tl_set:Nx \l_tmpa_tl {\prop_item:cn {l_stex_symdecl_varseq://\l_tmpa_str _prop}{notat
      \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
      \tl_set:Nx \l_tmpa_tl {{\exp_not:N \exp_not:n{
        \exp_not:n{\exp_args:Nnx \use:nn} {
          \exp_not:n {
            \def\comp{\_varcomp}
            \str_set:Nn \STEXInternalCurrentSymbolStr
          } {varseq://\l_tmpa_str}
          \exp_not:n{ ##1 }
        }{
          \exp_not:n {
            \_stex_reset:N \comp
            \_stex_reset:N \STEXInternalCurrentSymbolStr
          }
        }
      }}}
      \exp_args:Nno \use:n {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
      \seq_reverse:N \l_tmpa_seq
      \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
      \seq_map_inline:Nn \l_tmpa_seq {
        \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
          \exp_args:Nno
          \l_tmpa_cs { ##1 } \l_tmpa_tl
        }
      }
      \tl_set:Nx \l_tmpa_tl {
        \_stex_term_omv:nn {varseq://\l_tmpa_str}{
          \exp_args:No \exp_not:n \l_tmpa_tl
        }
      }
      \exp_args:No\l_tmpb_tl\l_tmpa_tl
    }{
      \__stex_terms_math_assoc_arg_simple:nnn{#2} { #1 }{#3}
    }
```

197

```
4154      } {
4155        \__stex_terms_math_assoc_arg_simple:nnn{#2} { #1 }{#3}
4156      }
4157
4158  }
4159
4160  \cs_new_protected:Nn \__stex_terms_math_assoc_arg_simple:nnn {
4161    \clist_set:Nn \l_tmpa_clist{ #2 }
4162    \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
4163      \tl_set:Nn \l_tmpa_tl {
4164        \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq #1}{
4165          \_stex_term_arg:nn{A#3#1}{ #2 } }
4166      }
4167    }{
4168      \clist_reverse:N \l_tmpa_clist
4169      \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
4170      \tl_set:Nx \l_tmpa_tl {
4171        \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq #1}{
4172          \_stex_term_arg:nn{A#3#1}{
4173            \exp_args:No \exp_not:n \l_tmpa_tl
4174          }
4175      }}
4176      \clist_map_inline:Nn \l_tmpa_clist {
4177        \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
4178          \exp_args:Nno
4179          \l_tmpa_cs {
4180            \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq #1}{
4181              \_stex_term_arg:nn{A#3#1}{##1}
4182            }
4183          } \l_tmpa_tl
4184        }
4185      }
4186    }
4187    \exp_args:No\l_tmpb_tl\l_tmpa_tl
4188  }
```

(*End definition for* \STEXInternalTermMathAssocArgiiiii. *This function is documented on page 93.*)

32.2   Terms

Precedences:

```
4189  \tl_const:Nx \infprec {\int_use:N \c_max_int}
4190  \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
4191  \int_new:N \l__stex_terms_downprec
4192  \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 93.*)

Bracketing:

\l__stex_terms_left_bracket_str
\l__stex_terms_right_bracket_str

```
4193  \tl_set:Nn \l__stex_terms_left_bracket_str (
4194  \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

```
4195 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
4196   \bool_if:NTF \l__stex_terms_brackets_done_bool {
4197     \bool_set_false:N \l__stex_terms_brackets_done_bool
4198     #2
4199   } {
4200     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
4201       \bool_if:NTF \l_stex_inparray_bool { #2 }{
4202         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
4203         \dobrackets { #2 }
4204       }
4205     }{ #2 }
4206   }
4207 }
```

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

`\dobrackets`

```
4208 \bool_new:N \l__stex_terms_brackets_done_bool
4209 %\RequirePackage{scalerel}
4210 \cs_new_protected:Npn \dobrackets #1 {
4211   %\ThisStyle{\if D\m@switch
4212   %  \exp_args:Nnx \use:nn
4213   %  { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
4214   %  { \exp_not:N\right\l__stex_terms_right_bracket_str }
4215   %  \else
4216     \exp_args:Nnx \use:nn
4217     {
4218       \bool_set_true:N \l__stex_terms_brackets_done_bool
4219       \int_set:Nn \l__stex_terms_downprec \infprec
4220       \l__stex_terms_left_bracket_str
4221       #1
4222     }
4223     {
4224       \bool_set_false:N \l__stex_terms_brackets_done_bool
4225       \l__stex_terms_right_bracket_str
4226       \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
4227     }
4228   %\fi}
4229 }
```

(End definition for `\dobrackets`. *This function is documented on page 93.*)*

`\withbrackets`

```
4230 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
4231   \exp_args:Nnx \use:nn
4232   {
4233     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
4234     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
4235     #3
4236   }
4237   {
```

199

```
4238        \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
4239            {\l__stex_terms_left_bracket_str}
4240        \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
4241            {\l__stex_terms_right_bracket_str}
4242    }
4243 }
```

(*End definition for* `\withbrackets`*. This function is documented on page 93.*)

<code>\STEXinvisible</code>

```
4244 \cs_new_protected:Npn \STEXinvisible #1 {
4245    \stex_annotate_invisible:n { #1 }
4246 }
```

(*End definition for* `\STEXinvisible`*. This function is documented on page 93.*)

OMDoc terms:

<code>\STEXInternalTermMathOMSiiii</code>

```
4247 \cs_new_protected:Nn \_stex_term_oms:nnn {
4248    \stex_annotate:nnn{ OMID }{ #2 }{
4249        #3
4250    }
4251 }
4252
4253 \cs_new_protected:Npn \STEXInternalTermMathOMSiiii #1#2#3#4 {
4254    \__stex_terms_maybe_brackets:nn { #3 }{
4255        \stex_mathml_intent:nn{#1} {
4256            \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
4257        }
4258    }
4259 }
```

(*End definition for* `\STEXInternalTermMathOMSiiii`*. This function is documented on page 92.*)

<code>_stex_term_math_omv:nn</code>

```
4260 \cs_new_protected:Nn \_stex_term_omv:nn {
4261    \stex_annotate:nnn{ OMV }{ #1 }{
4262        #2
4263    }
4264 }
```

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

<code>\STEXInternalTermMathOMAiiii</code>

```
4265 \cs_new_protected:Nn \_stex_term_oma:nnn {
4266    \stex_annotate:nnn{ OMA }{ #2 }{
4267        #3
4268    }
4269 }
4270
4271 \cs_new_protected:Npn \STEXInternalTermMathOMAiiii #1#2#3#4 {
4272    \exp_args:Nnx \use:nn {
4273        \seq_clear:N \l__stex_terms_tmp_seq
4274        \prop_if_exist:cT{l_stex_symdecl_#1 _prop}{
4275            \exp_args:NNx \seq_set_from_clist:Nn \l_stex_argnames_seq {
```

```
4276            \prop_item:cn {l_stex_symdecl_#1 _prop}{argnames}
4277          }
4278          \exp_args:Nx\int_step_inline:nn{\prop_item:cn{l_stex_symdecl_#1 _prop}{arity}}{
4279            \tl_set:Nx \l__stex_terms_tmp_tl {\seq_item:Nn \l_stex_argnames_seq {##1}}
4280            \bool_lazy_or:nnT{
4281              \str_if_eq_p:nn{a}{\str_item:Nn\l_tmpa_str{##1}}
4282            }{
4283              \str_if_eq_p:nn{B}{\str_item:Nn\l_tmpa_str{##1}}
4284            }{
4285              \tl_put_right:Nn \l__stex_terms_tmp_tl +
4286            }
4287            \seq_put_right:No \l__stex_terms_tmp_seq \l__stex_terms_tmp_tl
4288          }
4289        }
4290        \__stex_terms_maybe_brackets:nn { #3 }{
4291          \stex_mathml_intent:nn{
4292            #1[\prop_item:cn {l_stex_symdecl_#1 _prop}{ args }](
4293              \seq_use:Nn \l__stex_terms_tmp_seq ,
4294            )
4295          }{
4296            \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
4297          }
4298        }
4299      }{
4300        \_stex_reset:N \l_stex_argnames_seq
4301      }
4302 }
```

(End definition for \STEXInternalTermMathOMAiiii. *This function is documented on page 92.)*

```
4303 \cs_new_protected:Nn \_stex_term_ombind:nnn {
4304   \stex_annotate:nnn{ OMBIND }{ #2 }{
4305     #3
4306   }
4307 }
4308
4309 \cs_new_protected:Npn \STEXInternalTermMathOMBiiii #1#2#3#4 {
4310   \exp_args:Nnx \use:nn {
4311     \seq_clear:N \l__stex_terms_tmp_seq
4312     \prop_if_exist:cT{l_stex_symdecl_#1 _prop}{
4313     \exp_args:NNx \seq_set_from_clist:Nn \l_stex_argnames_seq {
4314       \prop_item:cn {l_stex_symdecl_#1 _prop}{argnames}
4315     }
4316     \exp_args:Nx\int_step_inline:nn{\prop_item:cn{l_stex_symdecl_#1 _prop}{arity}}{
4317       \tl_set:Nx \l__stex_terms_tmp_tl {\seq_item:Nn \l_stex_argnames_seq {##1}}
4318       \bool_lazy_or:nnT{
4319         \str_if_eq_p:nn{a}{\str_item:Nn\l_tmpa_str{##1}}
4320       }{
4321         \str_if_eq_p:nn{B}{\str_item:Nn\l_tmpa_str{##1}}
4322       }{
4323         \tl_put_right:Nn \l__stex_terms_tmp_tl +
4324       }
4325       \seq_put_right:No \l__stex_terms_tmp_seq \l__stex_terms_tmp_tl
```

```
4326          }
4327        }
4328      \__stex_terms_maybe_brackets:nn { #3 }{
4329        \stex_mathml_intent:nn{
4330          #1[\prop_item:cn {l_stex_symdecl_#1 _prop}{ args }](
4331            \seq_use:Nn \l__stex_terms_tmp_seq ,
4332          )
4333        }{
4334          \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
4335        }
4336      }
4337    }{
4338      \_stex_reset:N \l_stex_argnames_seq
4339    }
4340  }
```

(End definition for \STEXInternalTermMathOMBiiii. *This function is documented on page 92.)*

\symref
\symname

```
4341  \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
4342
4343  \keys_define:nn { stex / symname } {
4344    pre     .tl_set_x:N    = \l__stex_terms_pre_tl ,
4345    post    .tl_set_x:N    = \l__stex_terms_post_tl ,
4346    root    .tl_set_x:N    = \l__stex_terms_root_tl
4347  }
4348
4349  \cs_new_protected:Nn \stex_symname_args:n {
4350    \tl_clear:N \l__stex_terms_post_tl
4351    \tl_clear:N \l__stex_terms_pre_tl
4352    \tl_clear:N \l__stex_terms_root_str
4353    \keys_set:nn { stex / symname } { #1 }
4354  }
4355
4356  \NewDocumentCommand \symref { m m }{
4357    \let\compemph_uri_prev:\compemph@uri
4358    \let\compemph@uri\symrefemph@uri
4359    \STEXsymbol{#1}!{ #2 }
4360    \let\compemph@uri\compemph_uri_prev:
4361  }
4362
4363  \NewDocumentCommand \synonym { O{} m m}{
4364    \stex_symname_args:n { #1 }
4365    \let\compemph_uri_prev:\compemph@uri
4366    \let\compemph@uri\symrefemph@uri
4367    % TODO
4368    \STEXsymbol{#2}!{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
4369    \let\compemph@uri\compemph_uri_prev:
4370  }
4371
4372  \NewDocumentCommand \symname { O{} m }{
4373    \stex_symname_args:n { #1 }
4374    \stex_get_symbol:n { #2 }
4375    \str_set:Nx \l_tmpa_str {
```

```
4376          \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4377     }
4378     \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4379
4380     \let\compemph_uri_prev:\compemph@uri
4381     \let\compemph@uri\symrefemph@uri
4382     \exp_args:NNx \use:nn
4383     \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmmode*\fi{
4384       \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
4385     } }
4386     \let\compemph@uri\compemph_uri_prev:
4387  }
4388
4389  \NewDocumentCommand \Symname { O{} m }{
4390     \stex_symname_args:n { #1 }
4391     \stex_get_symbol:n { #2 }
4392     \str_set:Nx \l_tmpa_str {
4393       \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4394     }
4395     \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4396     \let\compemph_uri_prev:\compemph@uri
4397     \let\compemph@uri\symrefemph@uri
4398     \exp_args:NNx \use:nn
4399     \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmmode*\fi{
4400       \exp_after:wN \stex_capitalize:n \l_tmpa_str
4401         \l__stex_terms_post_tl
4402     } }
4403     \let\compemph@uri\compemph_uri_prev:
4404  }
```

(*End definition for* `\symref` *and* `\symname`*. These functions are documented on page* *92*.)

32.3   Notation Components

```
4405  ⟨@@=stex_notationcomps⟩
```

<div style="float:right">

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

</div>

```
4406  \cs_new_protected:Npn \_comp #1 {
4407     \str_if_empty:NF \STEXInternalCurrentSymbolStr {
4408       \stex_html_backend:TF {
4409         \stex_annotate:nnn { comp }{ \STEXInternalCurrentSymbolStr }{ #1 }
4410       }{
4411         \exp_args:Nnx \compemph@uri { #1 } { \STEXInternalCurrentSymbolStr }
4412       }
4413     }
4414  }
4415
4416  \cs_new_protected:Npn \_varcomp #1 {
4417     \str_if_empty:NF \STEXInternalCurrentSymbolStr {
4418       \stex_html_backend:TF {
4419         \stex_annotate:nnn { varcomp }{ \STEXInternalCurrentSymbolStr }{ #1 }
4420       }{
4421         \exp_args:Nnx \varemph@uri { #1 } { \STEXInternalCurrentSymbolStr }
4422       }
```

```
4423        }
4424    }
4425
4426    \def\comp{\_comp}
4427
4428    \cs_new_protected:Npn \compemph@uri #1 #2 {
4429        \compemph{ #1 }
4430    }
4431
4432
4433    \cs_new_protected:Npn \compemph #1 {
4434        #1
4435    }
4436
4437    \cs_new_protected:Npn \defemph@uri #1 #2 {
4438        \defemph{#1}
4439    }
4440
4441    \cs_new_protected:Npn \defemph #1 {
4442        \textbf{#1}
4443    }
4444
4445    \cs_new_protected:Npn \symrefemph@uri #1 #2 {
4446        \symrefemph{#1}
4447    }
4448
4449    \cs_new_protected:Npn \symrefemph #1 {
4450        \emph{#1}
4451    }
4452
4453    \cs_new_protected:Npn \varemph@uri #1 #2 {
4454        \varemph{#1}
4455    }
4456
4457    \cs_new_protected:Npn \varemph #1 {
4458        #1
4459    }
```

(*End definition for* `\comp` *and others. These functions are documented on page 93.*)

<code>\ellipses</code>

```
4460    \NewDocumentCommand \ellipses {} { \ldots }
```

(*End definition for* `\ellipses`. *This function is documented on page 93.*)

<code>\parray</code>
<code>\prmatrix</code>
<code>\parrayline</code>
<code>\parraylineh</code>
<code>\parraycell</code>

```
4461    \bool_new:N \l_stex_inparray_bool
4462    \bool_set_false:N \l_stex_inparray_bool
4463    \NewDocumentCommand \parray { m m } {
4464        \begingroup
4465        \bool_set_true:N \l_stex_inparray_bool
4466        \begin{array}{#1}
4467            #2
4468        \end{array}
4469        \endgroup
```

```
4470 }
4471
4472 \NewDocumentCommand \prmatrix { m } {
4473   \begingroup
4474   \bool_set_true:N \l_stex_inparray_bool
4475   \begin{matrix}
4476     #1
4477   \end{matrix}
4478   \endgroup
4479 }
4480
4481 \def \maybephline {
4482   \bool_if:NT \l_stex_inparray_bool {\hline}
4483 }
4484
4485 \def \parrayline #1 #2 {
4486   #1 #2 \bool_if:NT \l_stex_inparray_bool {\\}
4487 }
4488
4489 \def \pmrow #1 { \parrayline{}{ #1 } }
4490
4491 \def \parraylineh #1 #2 {
4492   #1 #2 \bool_if:NT \l_stex_inparray_bool {\\\hline}
4493 }
4494
4495 \def \parraycell #1 {
4496   #1 \bool_if:NT \l_stex_inparray_bool {&}
4497 }
```

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

32.4   Variables

```
4498 ⟨@@=stex_variables⟩
```

\stex_invoke_variable:n   Invokes a variable

```
4499 \cs_new_protected:Nn \stex_invoke_variable:n {
4500   \if_mode_math:
4501     \exp_after:wN \__stex_variables_invoke_math:n
4502   \else:
4503     \exp_after:wN \__stex_variables_invoke_text:n
4504   \fi: {#1}
4505 }
4506
4507 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
4508   \peek_charcode_remove:NTF ! {
4509     \__stex_variables_invoke_op_custom:nn {#1}
4510   }{
4511     \__stex_variables_invoke_custom:nn {#1}
4512   }
4513 }
4514
4515
4516 \cs_new_protected:Nn \__stex_variables_invoke_math:n {
```

```
4517    \peek_charcode_remove:NTF ! {
4518      \peek_charcode_remove:NTF ! {
4519        \peek_charcode:NTF [ {
4520          % TODO throw error
4521        }{
4522          \__stex_variables_invoke_op_custom:nn
4523        }
4524      }{
4525        \__stex_variables_invoke_op:n { #1 }
4526      }
4527    }{
4528      \peek_charcode_remove:NTF * {
4529        \__stex_variables_invoke_custom:nn { #1 }
4530      }{
4531        \__stex_variables_invoke_math_ii:n { #1 }
4532      }
4533    }
4534  }
4535
4536  \cs_new_protected:Nn \__stex_variables_invoke_op_custom:nn {
4537    \exp_args:Nnx \use:nn {
4538      \def\comp{\_varcomp}
4539      \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4540      \bool_set_false:N \l_stex_allow_semantic_bool
4541      \_stex_term_omv:nn {var://#1}{
4542        \comp{ #2 }
4543      }
4544    }{
4545      \_stex_reset:N \comp
4546      \_stex_reset:N \STEXInternalCurrentSymbolStr
4547      \bool_set_true:N \l_stex_allow_semantic_bool
4548    }
4549  }
4550
4551  \cs_new_protected:Nn \__stex_variables_invoke_op:n {
4552    \cs_if_exist:cTF {
4553      stex_var_op_notation_ #1 _cs
4554    }{
4555      \exp_args:Nnx \use:nn {
4556        \def\comp{\_varcomp}
4557        \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4558        \_stex_term_omv:nn { var://#1 }{
4559          \use:c{stex_var_op_notation_ #1 _cs }
4560        }
4561      }{
4562        \_stex_reset:N \comp
4563        \_stex_reset:N \STEXInternalCurrentSymbolStr
4564      }
4565    }{
4566      \int_compare:nNnTF {\prop_item:cn {l_stex_symdecl_var://#1_prop}{arity}} = 0{
4567        \__stex_variables_invoke_math_ii:n {#1}
4568      }{
4569        \msg_error:nnxx{stex}{error/noop}{variable~#1}{}
4570      }
```

```
4571     }
4572 }
4573
4574 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n  #1 {
4575   \cs_if_exist:cTF {
4576     stex_var_notation_#1_cs
4577   }{
4578     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4579       \_stex_reset:N \comp
4580       \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
4581       \_stex_reset:N \STEXInternalCurrentSymbolStr
4582       \bool_set_true:N \l_stex_allow_semantic_bool
4583     }
4584     \def\comp{\_varcomp}
4585     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4586     \bool_set_false:N \l_stex_allow_semantic_bool
4587     \use:c{stex_var_notation_#1_cs}
4588   }{
4589     \msg_error:nnxx{stex}{error/nonotation}{variable~#1}{s}
4590   }
4591 }
4592
4593 \cs_new_protected:Nn \__stex_variables_invoke_custom:nn {
4594   \exp_args:Nnx \use:nn {
4595     \def\comp{\_varcomp}
4596     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4597     \prop_clear:N \l__stex_terms_custom_args_prop
4598     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
4599     \prop_get:cnN {
4600       l_stex_symdecl_var://#1 _prop
4601     }{ args } \l_tmpa_str
4602     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
4603     \tl_set:Nn \arg { \__stex_terms_arg: }
4604     \str_if_empty:NTF \l_tmpa_str {
4605       \_stex_term_omv:nn {var://#1}{\ignorespaces#2}
4606     }{
4607       \str_if_in:NnTF \l_tmpa_str b {
4608         \_stex_term_ombind:nnn {var://#1}{}{\ignorespaces#2}
4609       }{
4610         \str_if_in:NnTF \l_tmpa_str B {
4611           \_stex_term_ombind:nnn {var://#1}{}{\ignorespaces#2}
4612         }{
4613           \_stex_term_oma:nnn {var://#1}{}{\ignorespaces#2}
4614         }
4615       }
4616     }
4617     % TODO check that all arguments exist
4618   }{
4619     \_stex_reset:N \STEXInternalCurrentSymbolStr
4620     \_stex_reset:N \arg
4621     \_stex_reset:N \comp
4622     \_stex_reset:N \l__stex_terms_custom_args_prop
4623     %\bool_set_true:N \l_stex_allow_semantic_bool
4624   }
```

207

```
4625 }
```

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

32.5   Sequences

```
4626 ⟨@@=stex_sequences⟩
4627
4628 \cs_new_protected:Nn \stex_invoke_sequence:n {
4629   \peek_charcode_remove:NTF ! {
4630     \_stex_term_omv:nn {varseq://#1}{
4631       \exp_args:Nnx \use:nn {
4632         \def\comp{\_varcomp}
4633         \str_set:Nn \STEXInternalCurrentSymbolStr {varseq://#1}
4634         \prop_item:cn{l_stex_symdecl_varseq://#1_prop}{notation}
4635       }{
4636         \_stex_reset:N \comp
4637         \_stex_reset:N \STEXInternalCurrentSymbolStr
4638       }
4639     }
4640   }{
4641     \bool_set_false:N \l_stex_allow_semantic_bool
4642     \def\comp{\_varcomp}
4643     \str_set:Nn \STEXInternalCurrentSymbolStr {varseq://#1}
4644     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4645       \_stex_reset:N \comp
4646       \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
4647       \_stex_reset:N \STEXInternalCurrentSymbolStr
4648       \bool_set_true:N \l_stex_allow_semantic_bool
4649     }
4650     \use:c { stex_varseq_#1_cs }
4651   }
4652 }
4653 ⟨/package⟩
```

Chapter 33

ST_EX -Structural Features Implementation

```
4654  ⟨∗package⟩
4655
4656  %%%%%%%%%%%%    features.dtx    %%%%%%%%%%%%
4657
```

Warnings and error messages

```
4658  \msg_new:nnn{stex}{error/copymodule/notallowed}{
4659    Symbol~#1~can~not~be~assigned~in~copymodule~#2
4660  }
4661  \msg_new:nnn{stex}{error/interpretmodule/nodefiniens}{
4662    Symbol~#1~not~assigned~in~interpretmodule~#2
4663  }
4664
4665  \msg_new:nnn{stex}{error/unknownstructure}{
4666    No~structure~#1~found!
4667  }
4668
4669  \msg_new:nnn{stex}{error/unknownfield}{
4670    No~field~#1~in~instance~#2~found!\\#3
4671  }
4672
4673  \msg_new:nnn{stex}{error/keyval}{
4674    Invalid~key=value~pair:#1
4675  }
4676  \msg_new:nnn{stex}{error/instantiate/missing}{
4677    Assignments~missing~in~instantiate:~#1
4678  }
4679  \msg_new:nnn{stex}{error/incompatible}{
4680    Incompatible~signature:~#1~(#2)~and~#3~(#4)
4681  }
4682
```

33.1 Imports with modification

```
⟨@@=stex_copymodule⟩
\cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
  \tl_if_head_eq_catcode:nNTF { #1 } \relax {
    \tl_set:Nn \l_tmpa_tl { #1 }
    \__stex_copymodule_get_symbol_from_cs:
  }{
    % argument is a string
    % is it a command name?
    \cs_if_exist:cTF { #1 }{
      \cs_set_eq:Nc \l_tmpa_tl { #1 }
      \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
      \str_if_empty:NTF \l_tmpa_str {
        \exp_args:Nx \cs_if_eq:NNTF {
          \tl_head:N \l_tmpa_tl
        } \stex_invoke_symbol:n {
          \__stex_copymodule_get_symbol_from_cs:n{ #2 }
        }{
          \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
        }
      } {
        \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
      }
    }{
      % argument is not a command name
      \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
      % \l_stex_all_symbols_seq
    }
  }
}

\cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
  \str_set:Nn \l_tmpa_str { #1 }
  \bool_set_false:N \l_tmpa_bool
  \bool_if:NF \l_tmpa_bool {
    \tl_set:Nn \l_tmpa_tl {
    \msg_error:nnn{stex}{error/unknownsymbol}{#1}
    }
  }
  \str_set:Nn \l_tmpa_str { #1 }
  \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
  \seq_map_inline:Nn #2 {
    \str_set:Nn \l_tmpb_str { ##1 }
    \str_if_eq:eeT { \l_tmpa_str } {
      \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
    } {
      \seq_map_break:n {
        \tl_set:Nn \l_tmpa_tl {
          \str_set:Nn \l_stex_get_symbol_uri_str {
            ##1
          }
        }
      }
    }
  }
```

210

```
4735        }
4736      \l_tmpa_tl
4737    }
4738  }
4739
4740  \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
4741    \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4742      { \tl_tail:N \l_tmpa_tl }
4743    \tl_if_single:NTF \l_tmpa_tl {
4744      \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
4745        \exp_after:wN \str_set:Nn \exp_after:wN
4746          \l_stex_get_symbol_uri_str \l_tmpa_tl
4747        \__stex_copymodule_get_symbol_check:n { #1 }
4748      }{
4749        % TODO
4750        % tail is not a single group
4751      }
4752    }{
4753      % TODO
4754      % tail is not a single group
4755    }
4756  }
4757
4758  \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
4759    \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
4760      \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
4761        :~\seq_use:Nn #1 {,~}
4762      }
4763    }
4764  }
4765
4766  \cs_new_protected:Nn \stex_copymodule_start:nnnn {
4767    % import module
4768    \stex_import_module_uri:nn { #1 } { #2 }
4769    \str_set:Nx \l_stex_current_copymodule_name_str {#3}
4770    \stex_import_require_module:nnnn
4771      { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4772      { \l_stex_import_path_str } { \l_stex_import_name_str }
4773
4774    \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
4775    \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
4776
4777    % fields
4778    \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
4779    \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4780      \seq_map_inline:cn {c_stex_module_##1_constants}{
4781        \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
4782          ##1 ? ####1
4783        }
4784      }
4785    }
4786
4787    % setup prop
4788    \seq_clear:N \l_tmpa_seq
```

```
4789    \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
4790      name      = \l_stex_current_copymodule_name_str ,
4791      module    = \l_stex_current_module_str ,
4792      from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
4793      includes  = \l_tmpa_seq %,
4794    %  fields    = \l_tmpa_seq
4795    }
4796    \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
4797      as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
4798      \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_se
4799    \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,
4800
4801    \stex_if_do_html:T {
4802      \begin{stex_annotate_env} {#4} {
4803        \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4804      }
4805      \stex_annotate_invisible:nnn{domain}{\l_stex_import_ns_str ?\l_stex_import_name_str}{}
4806    }
4807  }
4808
4809  \cs_new_protected:Nn \stex_copymodule_end:n {
4810    % apply to every field
4811    \def \l_tmpa_cs ##1 ##2 {#1}
4812
4813    \tl_clear:N \__stex_copymodule_module_tl
4814    \tl_clear:N \__stex_copymodule_exec_tl
4815
4816    %\prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
4817    \seq_clear:N \__stex_copymodule_fields_seq
4818
4819    \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4820      \seq_map_inline:cn {c_stex_module_##1_constants}{
4821
4822        \tl_clear:N \__stex_copymodule_curr_symbol_tl % <- wrap in current symbol html
4823        \l_tmpa_cs{##1}{####1}
4824
4825        \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
4826          \str_set_eq:Nc \__stex_copymodule_curr_name_str {l__stex_copymodule_copymodule_##1?#
4827          \stex_if_do_html:T {
4828            \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4829              \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?###
4830            }
4831          }
4832        }{
4833          \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str /
4834        }
4835
4836        \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
4837        \prop_put:Nnx \l_tmpa_prop { name } \__stex_copymodule_curr_name_str
4838        \prop_put:Nnx \l_tmpa_prop { module } \l_stex_current_module_str
4839
4840        \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
4841          \stex_if_do_html:T {
4842            \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
```

```
4843          $\stex_annotate_invisible:nnn{definiens}{}{\exp_after:wN \exp_not:N\csname l__st
4844        }
4845      }
4846      \prop_put:Nnn \l_tmpa_prop { defined } { true }
4847    }
4848
4849    \stex_add_constant_to_current_module:n \__stex_copymodule_curr_name_str
4850    \tl_put_right:Nx \__stex_copymodule_module_tl {
4851      \seq_clear:c {l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_n
4852      \prop_set_from_keyval:cn {
4853        l_stex_symdecl_\l_stex_current_module_str ? \__stex_copymodule_curr_name_str _prop
4854      }{
4855        \prop_to_keyval:N \l_tmpa_prop
4856      }
4857    }
4858
4859    \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
4860      \stex_if_do_html:T {
4861        \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4862          \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1
4863        }
4864      }
4865      \tl_put_right:Nx \__stex_copymodule_module_tl {
4866        \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
4867          \stex_invoke_symbol:n {
4868            \l_stex_current_module_str ? \__stex_copymodule_curr_name_str
4869          }
4870        }
4871      }
4872    }
4873
4874    \seq_put_right:Nx \__stex_copymodule_fields_seq {\l_stex_current_module_str ? \__stex_
4875
4876    \tl_put_right:Nx \__stex_copymodule_exec_tl {
4877      \stex_copy_notations:nn {\l_stex_current_module_str ? \__stex_copymodule_curr_name_s
4878    }
4879
4880    \tl_put_right:Nx \__stex_copymodule_exec_tl {
4881      \stex_if_do_html:TF{
4882        \stex_annotate_invisible:nnn{assignment} {##1?####1} { \exp_after:wN \exp_not:n \e
4883      }{
4884        \exp_after:wN \exp_not:n \exp_after:wN {\__stex_copymodule_curr_symbol_tl}
4885      }
4886    }
4887  }
4888 }
4889
4890
4891 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \__stex_copymodule_fields_seq
4892 \tl_put_left:Nx \__stex_copymodule_module_tl {
4893   \prop_set_from_keyval:cn {
4894     l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _pro
4895   }{
4896     \prop_to_keyval:N \l_stex_current_copymodule_prop
```

213

```
4897        }
4898      }
4899
4900      \seq_gput_right:cx{c_stex_module_\l_stex_current_module_str _copymodules}{
4901        \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4902      }
4903
4904      \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4905      \stex_debug:nn{copymodule}{result:\meaning \__stex_copymodule_module_tl}
4906      \stex_debug:nn{copymodule}{output:\meaning \__stex_copymodule_exec_tl}
4907
4908      \__stex_copymodule_exec_tl
4909      \stex_if_do_html:T {
4910        \end{stex_annotate_env}
4911      }
4912    }
4913
4914    \NewDocumentEnvironment {copymodule} { O{} m m}{
4915      \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ copymodule }
4916      \stex_deactivate_macro:Nn \symdecl {module~environments}
4917      \stex_deactivate_macro:Nn \symdef {module~environments}
4918      \stex_deactivate_macro:Nn \notation {module~environments}
4919      \stex_reactivate_macro:N \assign
4920      \stex_reactivate_macro:N \renamedecl
4921      \stex_reactivate_macro:N \donotcopy
4922      \stex_smsmode_do:
4923    }{
4924      \stex_copymodule_end:n {}
4925    }
4926
4927    \NewDocumentEnvironment {interpretmodule} { O{} m m}{
4928      \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ interpretmodule }
4929      \stex_deactivate_macro:Nn \symdecl {module~environments}
4930      \stex_deactivate_macro:Nn \symdef {module~environments}
4931      \stex_deactivate_macro:Nn \notation {module~environments}
4932      \stex_reactivate_macro:N \assign
4933      \stex_reactivate_macro:N \renamedecl
4934      \stex_reactivate_macro:N \donotcopy
4935      \stex_smsmode_do:
4936    }{
4937      \stex_copymodule_end:n {
4938        \tl_if_exist:cF {
4939          l__stex_copymodule_copymodule_##1?##2_def_tl
4940        }{
4941          \str_if_eq:eeF {
4942            \prop_item:cn{
4943              l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
4944          }{ true }{
4945            \msg_error:nnxx{stex}{error/interpretmodule/nodefiniens}{
4946              ##1?##2
4947            }{\l_stex_current_copymodule_name_str}
4948          }
4949        }
4950      }
```

```
4951 }
4952
4953 \iffalse \begin{stex_annotate_env} \fi
4954 \NewDocumentEnvironment {realization} { O{} m}{
4955   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #2 }{ realize }
4956   \stex_deactivate_macro:Nn \symdecl {module~environments}
4957   \stex_deactivate_macro:Nn \symdef {module~environments}
4958   \stex_deactivate_macro:Nn \notation {module~environments}
4959   \stex_reactivate_macro:N \donotcopy
4960   \stex_reactivate_macro:N \assign
4961   \stex_smsmode_do:
4962 }{
4963   \stex_import_module_uri:nn { #1 } { #2 }
4964   \tl_clear:N \__stex_copymodule_exec_tl
4965   \tl_set:Nx \__stex_copymodule_module_tl {
4966     \stex_import_require_module:nnnn
4967       { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4968       { \l_stex_import_path_str } { \l_stex_import_name_str }
4969   }
4970
4971   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4972     \seq_map_inline:cn {c_stex_module_##1_constants}{
4973       \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str / #
4974       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
4975         \stex_if_do_html:T {
4976           \tl_put_right:Nx \__stex_copymodule_exec_tl {
4977             \stex_annotate_invisible:nnn{assignment} {##1?####1} {
4978               $\stex_annotate_invisible:nnn{definiens}{}{\exp_after:wN \exp_not:N\csname l__
4979             }
4980           }
4981         }
4982         \tl_put_right:Nx \__stex_copymodule_module_tl {
4983           \prop_put:cnn {l_stex_symdecl_##1?####1_prop}{ defined }{ true }
4984         }
4985       }
4986   }}
4987
4988   \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4989
4990   \__stex_copymodule_exec_tl
4991   \stex_if_do_html:T {\end{stex_annotate_env}}
4992 }
4993
4994 \NewDocumentCommand \donotcopy { m }{
4995   \str_clear:N \l_stex_import_name_str
4996   \str_set:Nn \l_tmpa_str { #1 }
4997   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4998   \seq_map_inline:Nn \l_stex_all_modules_seq {
4999     \str_set:Nn \l_tmpb_str { ##1 }
5000     \str_if_eq:eeT { \l_tmpa_str } {
5001       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
5002     } {
5003       \seq_map_break:n {
5004         \stex_if_do_html:T {
```

215

```
\stex_if_smsmode:F {
  \stex_annotate_invisible:nnn{donotcopy}{##1}{
    \stex_annotate:nnn{domain}{##1}{}
  }
}
}
\str_set_eq:NN \l_stex_import_name_str \l_tmpb_str
}
}
\seq_map_inline:cn {c_stex_module_##1_copymodules}{
  \str_set:Nn \l_tmpb_str { ####1 }
  \str_if_eq:eeT { \l_tmpa_str } {
    \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
  } {
    \seq_map_break:n {\seq_map_break:n {
      \stex_if_do_html:T {
        \stex_if_smsmode:F {
          \stex_annotate_invisible:nnn{donotcopy}{####1}{
            \stex_annotate:nnn{domain}{
              \prop_item:cn {l_stex_copymodule_ ####1 _prop}{module}
            }{}
          }
        }
      }
      \str_set:Nx \l_stex_import_name_str {
        \prop_item:cn {l_stex_copymodule_ ####1 _prop}{module}
      }
    }}
  }
}
}
\str_if_empty:NTF \l_stex_import_name_str {
  % TODO throw error
}{
  \stex_collect_imports:n {\l_stex_import_name_str }
  \seq_map_inline:Nn \l_stex_collect_imports_seq {
    \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
    \seq_map_inline:cn {c_stex_module_##1_constants}{
      \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? ####1 }
      \bool_lazy_any:nT {
        { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_name_str}}
        { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_macroname_str}}
        { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_def_tl}}
      }{
        % TODO throw error
      }
    }
  }
  \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
  \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_name_str }
  \prop_put:Nno \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
}
\stex_smsmode_do:
}
```

```
5059
5060  \NewDocumentCommand \assign { m m }{
5061    \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
5062    \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
5063    \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _def_tl}{#2}
5064    \stex_smsmode_do:
5065  }
5066
5067  \keys_define:nn { stex / renamedecl } {
5068    name          .str_set_x:N  = \l_stex_renamedecl_name_str
5069  }
5070  \cs_new_protected:Nn \__stex_copymodule_renamedecl_args:n {
5071    \str_clear:N \l_stex_renamedecl_name_str
5072    \keys_set:nn { stex / renamedecl } { #1 }
5073  }
5074
5075  \NewDocumentCommand \renamedecl { O{} m m}{
5076    \__stex_copymodule_renamedecl_args:n { #1 }
5077    \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
5078    \stex_debug:nn{renamedecl}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
5079    \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
5080    \str_if_empty:NTF \l_stex_renamedecl_name_str {
5081      \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
5082        \l_stex_get_symbol_uri_str
5083      } }
5084    } {
5085      \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex
5086      \stex_debug:nn{renamedecl}{@~\l_stex_current_module_str ? \l_stex_renamedecl_name_str}
5087      \prop_set_eq:cc {l_stex_symdecl_
5088        \l_stex_current_module_str ? \l_stex_renamedecl_name_str
5089        _prop
5090      }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
5091      \seq_set_eq:cc {l_stex_symdecl_
5092        \l_stex_current_module_str ? \l_stex_renamedecl_name_str
5093        _notations
5094      }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
5095      \prop_put:cnx {l_stex_symdecl_
5096        \l_stex_current_module_str ? \l_stex_renamedecl_name_str
5097        _prop
5098      }{ name }{ \l_stex_renamedecl_name_str }
5099      \prop_put:cnx {l_stex_symdecl_
5100        \l_stex_current_module_str ? \l_stex_renamedecl_name_str
5101        _prop
5102      }{ module }{ \l_stex_current_module_str }
5103      \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
5104        \l_stex_current_module_str ? \l_stex_renamedecl_name_str
5105      }
5106      \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
5107        \l_stex_current_module_str ? \l_stex_renamedecl_name_str
5108      } }
5109    }
5110    \stex_smsmode_do:
5111  }
5112
```

```
5113  \stex_deactivate_macro:Nn \assign {copymodules}
5114  \stex_deactivate_macro:Nn \renamedecl {copymodules}
5115  \stex_deactivate_macro:Nn \donotcopy {copymodules}
5116
5117
```

33.2   The feature environment

```
5118  ⟨@@=stex_features⟩
5119
5120  \NewDocumentEnvironment{structural_feature_module}{ m m m }{
5121    \stex_if_in_module:F {
5122      \msg_set:nnn{stex}{error/nomodule}{
5123        Structural~Feature~has~to~occur~in~a~module:\\
5124        Feature~#2~of~type~#1\\
5125        In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
5126      }
5127      \msg_error:nn{stex}{error/nomodule}
5128    }
5129
5130    \str_set_eq:NN \l_stex_feature_parent_str \l_stex_current_module_str
5131
5132    \stex_module_setup:nn{meta=NONE}{#2 - #1}
5133
5134    \stex_if_do_html:T {
5135      \begin{stex_annotate_env}{ feature:#1 }{\l_stex_feature_parent_str ? #2 - #1}
5136        \stex_annotate_invisible:nnn{header}{}{ #3 }
5137    }
5138  }{
5139    \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
5140    \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
5141    \stex_debug:nn{features}{
5142      Feature: \l_stex_last_feature_str
5143    }
5144    \stex_if_do_html:T {
5145      \end{stex_annotate_env}
5146    }
5147  }
```

33.3   Structure

```
5148  ⟨@@=stex_structures⟩
5149  \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
5150    \prop_if_exist:cF {c_stex_module_\l_stex_current_module_str _structures}{
5151      \prop_new:c {c_stex_module_\l_stex_current_module_str _structures}
5152    }
5153    \prop_gput:cxx{c_stex_module_\l_stex_current_module_str _structures}
5154      {#1}{#2}
5155  }
5156
```

218

```
5157  \keys_define:nn { stex / features / structure } {
5158    name          .str_set_x:N  = \l__stex_structures_name_str ,
5159  }
5160
5161  \cs_new_protected:Nn \__stex_structures_structure_args:n {
5162    \str_clear:N \l__stex_structures_name_str
5163    \keys_set:nn { stex / features / structure } { #1 }
5164  }
5165  \NewDocumentEnvironment{mathstructure}{m O{}}{
5166    \begin{mathstructure_inner}{#1}[#2]
5167      \stex_smsmode_do:
5168      \ignorespacesandpars
5169  }{\end{mathstructure_inner}}
5170  \NewDocumentEnvironment{mathstructure_inner}{m O{}}{
5171    \__stex_structures_structure_args:n { #2 }
5172    \str_if_empty:NT \l__stex_structures_name_str {
5173      \str_set:Nx \l__stex_structures_name_str { #1 }
5174    }
5175    \stex_suppress_html:n {
5176      \bool_set_true:N \l_stex_symdecl_make_macro_bool
5177      \exp_args:Nx \stex_symdecl_do:nn {
5178        name = \l__stex_structures_name_str ,
5179        def  = {\STEXsymbol{module-type}{
5180          \STEXInternalTermMathOMSiiii {
5181            \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
5182              { ns } ?
5183              \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
5184                { name } / \l__stex_structures_name_str - structure
5185          }{}{0}{}
5186        }}
5187      }{ #1 }
5188    }
5189    \exp_args:Nnnx
5190    \begin{structural_feature_module}{ structure }
5191      { \l__stex_structures_name_str }{}
5192  }{
5193    \end{structural_feature_module}
5194    \_stex_reset_up_to_module:n \l_stex_last_feature_str
5195    \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
5196    \seq_clear:N \l_tmpa_seq
5197    \seq_map_inline:Nn \l_stex_collect_imports_seq {
5198      \seq_map_inline:cn{c_stex_module_##1_constants}{
5199        \seq_put_right:Nn \l_tmpa_seq { ##1 ? ####1 }
5200      }
5201    }
5202    \exp_args:Nnno
5203    \prop_gput:cnn {c_stex_module_ \l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
5204    \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
5205    \stex_add_structure_to_current_module:nn
5206      \l__stex_structures_name_str
5207      \l_stex_last_feature_str
5208
5209    \stex_execute_in_module:x {
5210      \tl_set:cn { #1 }{
```

```
5211          \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structure
5212      }
5213    }
5214 }
5215
5216 \cs_new:Nn \stex_invoke_structure:nn {
5217    \stex_invoke_symbol:n { #1?#2 }
5218 }
5219
5220 \cs_new_protected:Nn \stex_get_structure:n {
5221    \tl_if_head_eq_catcode:nNTF { #1 } \relax {
5222      \tl_set:Nn \l_tmpa_tl { #1 }
5223      \__stex_structures_get_from_cs:
5224    }{
5225      \cs_if_exist:cTF { #1 }{
5226        \cs_set_eq:Nc \l_tmpa_cs { #1 }
5227        \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
5228        \str_if_empty:NTF \l_tmpa_str {
5229          \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs} \stex_invoke_structure:nn {
5230            \__stex_structures_get_from_cs:
5231          }{
5232            \__stex_structures_get_from_string:n { #1 }
5233          }
5234        }{
5235          \__stex_structures_get_from_string:n { #1 }
5236        }
5237      }{
5238        \__stex_structures_get_from_string:n { #1 }
5239      }
5240    }
5241 }
5242
5243 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
5244    \exp_args:NNx \tl_set:Nn \l_tmpa_tl
5245      { \tl_tail:N \l_tmpa_tl }
5246    \str_set:Nx \l_tmpa_str {
5247      \exp_after:wN \use_i:nn \l_tmpa_tl
5248    }
5249    \str_set:Nx \l_tmpb_str {
5250      \exp_after:wN \use_ii:nn \l_tmpa_tl
5251    }
5252    \str_set:Nx \l_stex_get_structure_str {
5253      \l_tmpa_str ? \l_tmpb_str
5254    }
5255    \str_set:Nx \l_stex_get_structure_module_str {
5256      \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
5257    }
5258 }
5259
5260 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
5261    \tl_set:Nn \l_tmpa_tl {
5262      \msg_error:nnn{stex}{error/unknownstructure}{#1}
5263    }
5264    \str_set:Nn \l_tmpa_str { #1 }
```

```
5265     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
5266
5267     \seq_map_inline:Nn \l_stex_all_modules_seq {
5268       \prop_if_exist:cT {c_stex_module_##1_structures} {
5269         \prop_map_inline:cn {c_stex_module_##1_structures} {
5270           \exp_args:No \str_if_eq:nnT \l_tmpa_str {####1}{
5271           %\str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
5272             \prop_map_break:n{\seq_map_break:n{
5273               \tl_set:Nn \l_tmpa_tl {
5274                 \str_set:Nn \l_stex_get_structure_str {##1?####1}
5275                 \str_set:Nn \l_stex_get_structure_module_str {####2}
5276               }
5277           }}
5278         }
5279       }
5280     }
5281   }
5282   \l_tmpa_tl
5283 }
```

\instantiate

```
5284
5285 \NewDocumentEnvironment{usestructure}{m}{
5286   \stex_get_structure:n {#1}
5287   \exp_args:Nnx \stex_debug:nn{features}{using~structure:~\l_stex_get_structure_module_str}
5288   \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5289 }{}
5290
5291 \keys_define:nn { stex / instantiate } {
5292   name          .str_set_x:N  = \l__stex_structures_name_str
5293 }
5294 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
5295   \str_clear:N \l__stex_structures_name_str
5296   \keys_set:nn { stex / instantiate } { #1 }
5297 }
5298
5299 \NewDocumentEnvironment{extstructure}{m m O{}}{
5300   \begin{mathstructure_inner}{#1}[#3]
5301     \seq_set_split:Nnn\__stex_structures_extstructure_imports_seq,{#2}
5302     \seq_map_inline:Nn\__stex_structures_extstructure_imports_seq {
5303       \stex_get_structure:n {##1}
5304       \exp_args:Nnx \stex_debug:nn{features}{importing~structure:~\l_stex_get_structure_modu
5305       \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5306       \stex_if_smsmode:F {
5307         \stex_annotate_invisible:nnn
5308           {import} {\l_stex_get_structure_module_str} {}
5309       }
5310       \exp_args:Nx \stex_add_import_to_current_module:n {
5311         \l_stex_get_structure_module_str
5312       }
5313       \exp_args:Nx \stex_add_to_current_module:n {
5314         \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5315       }
5316     }
```

```
5317        \stex_smsmode_do:
5318        \ignorespacesandpars
5319 }{
5320    \end{mathstructure_inner}
5321 }
5322
5323 \NewDocumentEnvironment{extstructure*}{m m O{}}{
5324    % TODO
5325    \begin{extstructure}{#1}{#2}[#3]
5326 }{
5327    \end{extstructure}
5328 }
5329
5330 \NewDocumentCommand \instantiate {m O{} m m O{}}{
5331    \begingroup
5332        \stex_get_structure:n {#3}
5333        \__stex_structures_instantiate_args:n { #2 }
5334        \str_if_empty:NT \l__stex_structures_name_str {
5335            \str_set:Nn \l__stex_structures_name_str { #1 }
5336        }
5337        \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5338        \seq_clear:N \l__stex_structures_fields_seq
5339        \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
5340        \seq_map_inline:Nn \l_stex_collect_imports_seq {
5341            \seq_map_inline:cn {c_stex_module_##1_constants}{
5342                \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
5343            }
5344        }
5345
5346        \tl_if_empty:nF{#5}{
5347            \seq_set_split:Nnn \l_tmpa_seq , {#5}
5348            \prop_clear:N \l_tmpa_prop
5349            \seq_map_inline:Nn \l_tmpa_seq {
5350                \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
5351                \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
5352                    \msg_error:nnn{stex}{error/keyval}{##1}
5353                }
5354                \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
5355                \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
5356                \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
5357                \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}
5358                \exp_args:Nxx \str_if_eq:nnF
5359                    {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5360                    {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
5361                    \msg_error:nnxxxx{stex}{error/incompatible}
5362                        {\l__stex_structures_dom_str}
5363                        {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5364                        {\l_stex_get_symbol_uri_str}
5365                        {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
5366                }
5367                \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str
5368            }
5369        }
5370
```

```
5371    \seq_map_inline:Nn \l__stex_structures_fields_seq {
5372      \str_set:Nx \l_tmpa_str {field:\l__stex_structures_name_str . \prop_item:cn {l_stex_sy
5373      \stex_debug:nn{instantiate}{Field~\l_tmpa_str :~##1}
5374
5375      \stex_add_constant_to_current_module:n {\l_tmpa_str}
5376      \stex_execute_in_module:x {
5377        \prop_set_from_keyval:cn { l_stex_symdecl_ \l_stex_current_module_str?\l_tmpa_str _p
5378          name  = \l_tmpa_str ,
5379          args  = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
5380          arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
5381          assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs} ,
5382          argnames = {\prop_item:cn {l_stex_symdecl_##1_prop}{argnames}}}
5383        }
5384        \seq_clear:c {l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _notations}
5385      }
5386
5387      \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
5388        \stex_find_notation:nn{##1}{}
5389        \stex_execute_in_module:x {
5390          \seq_put_right:cn {l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _notation
5391        }
5392
5393        \stex_copy_control_sequence_ii:ccN
5394          {stex_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_notation_
5395          {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5396          \l_tmpa_tl
5397        \exp_args:No \stex_execute_in_module:n \l_tmpa_tl
5398
5399
5400        \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
5401          \tl_set_eq:Nc \l_tmpa_cs {stex_op_notation_##1\c_hash_str \l_stex_notation_variant
5402          \stex_execute_in_module:x {
5403            \tl_set:cn
5404            {stex_op_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_nota
5405            { \exp_args:No \exp_not:n \l_tmpa_cs}
5406          }
5407        }
5408
5409      }
5410
5411      \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\l_stex_cur
5412    }
5413
5414    \stex_execute_in_module:x {
5415      \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
5416        domain = \l_stex_get_structure_module_str ,
5417        \prop_to_keyval:N \l_tmpa_prop
5418      }
5419      \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
5420    }
5421    \stex_debug:nn{instantiate}{
5422      Instance~\l_stex_current_module_str?\l__stex_structures_name_str \\
5423      \prop_to_keyval:N \l_tmpa_prop
5424    }
```

223

```
5425      \exp_args:Nxx \stex_symdecl_do:nn {
5426        type={\STEXsymbol{module-type}{
5427          \STEXInternalTermMathOMSiiii {
5428            \l_stex_get_structure_module_str
5429          }{}{0}{}
5430        }}
5431      }{\l__stex_structures_name_str}
5432 %    {
5433        \str_set:Nx \l_stex_get_symbol_uri_str {\l_stex_current_module_str?\l__stex_structures
5434        \tl_set:Nn \l_stex_notation_after_do_tl {\__stex_notation_final:}
5435        \stex_notation_do:nnnnn{}{0}{}{}{\comp{#4}}
5436 %    }
5437      %\exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}
5438    \endgroup
5439    \stex_smsmode_do:\ignorespacesandpars
5440 }
5441
5442 \cs_new_protected:Nn \stex_symbol_or_var:n {
5443    \cs_if_exist:cTF{#1}{
5444      \cs_set_eq:Nc \l_tmpa_tl { #1 }
5445      \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
5446      \str_if_empty:NTF \l_tmpa_str {
5447        \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
5448          \stex_invoke_variable:n {
5449            \bool_set_true:N \l_stex_symbol_or_var_bool
5450            \bool_set_false:N \l_stex_instance_or_symbol_bool
5451            \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
5452            \tl_set:Nx \l_tmpa_tl {\exp_after:wN \use:n \l_tmpa_tl}
5453            \str_set:Nx \l_stex_get_symbol_uri_str {
5454              \exp_after:wN \use:n \l_tmpa_tl
5455            }
5456          }{ % TODO \stex_invoke_varinstance:n
5457            \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl } \stex_invoke_varinstance:n {
5458              \bool_set_true:N \l_stex_symbol_or_var_bool
5459              \bool_set_true:N \l_stex_instance_or_symbol_bool
5460              \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
5461              \tl_set:Nx \l_tmpa_tl {\exp_after:wN \use:n \l_tmpa_tl}
5462              \str_set:Nx \l_stex_get_symbol_uri_str {
5463                \exp_after:wN \use:n \l_tmpa_tl
5464              }
5465            }{
5466              \bool_set_false:N \l_stex_symbol_or_var_bool
5467              \stex_get_symbol:n{#1}
5468            }
5469          }
5470      }{
5471        \__stex_structures_symbolorvar_from_string:n{ #1 }
5472      }
5473    }{
5474      \__stex_structures_symbolorvar_from_string:n{ #1 }
5475    }
5476 }
5477
5478 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
```

```
5479    \prop_if_exist:cTF {l_stex_symdecl_var://#1 _prop}{
5480      \bool_set_true:N \l_stex_symbol_or_var_bool
5481      \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
5482    }{
5483      \bool_set_false:N \l_stex_symbol_or_var_bool
5484      \stex_get_symbol:n{#1}
5485    }
5486  }
5487
5488  \keys_define:nn { stex / varinstantiate } {
5489    name          .str_set_x:N  = \l__stex_structures_name_str,
5490    bind          .choices:nn   =
5491        {forall,exists}
5492        {\str_set:Nx \l__stex_structures_bind_str {\l_keys_choice_tl}}
5493
5494  }
5495  \cs_new_protected:Nn \__stex_structures_varinstantiate_args:n {
5496    \str_clear:N \l__stex_structures_name_str
5497    \str_clear:N \l__stex_structures_bind_str
5498    \keys_set:nn { stex / varinstantiate } { #1 }
5499  }
5500
5501  \NewDocumentCommand \varinstantiate {m O{} m m O{}}{
5502    \begingroup
5503      \stex_get_structure:n {#3}
5504      \__stex_structures_varinstantiate_args:n { #2 }
5505      \str_if_empty:NT \l__stex_structures_name_str {
5506        \str_set:Nn \l__stex_structures_name_str { #1 }
5507      }
5508      \stex_if_do_html:TF{
5509        \stex_annotate:nnn{varinstance}{\l__stex_structures_name_str}
5510      }{\use:n}
5511      {
5512        \stex_if_do_html:T{
5513          \stex_annotate_invisible:nnn{domain}{\l_stex_get_structure_module_str}{}
5514        }
5515        \seq_clear:N \l__stex_structures_fields_seq
5516        \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
5517        \seq_map_inline:Nn \l_stex_collect_imports_seq {
5518          \seq_map_inline:cn {c_stex_module_##1_constants}{
5519            \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
5520          }
5521        }
5522        \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5523        \prop_clear:N \l_tmpa_prop
5524        \tl_if_empty:nF {#5} {
5525          \seq_set_split:Nnn \l_tmpa_seq , {#5}
5526          \seq_map_inline:Nn \l_tmpa_seq {
5527            \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
5528            \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
5529              \msg_error:nnn{stex}{error/keyval}{##1}
5530            }
5531            \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_stru
5532            \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
```

```
5533              \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol
5534              \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
5535              \stex_if_do_html:T{
5536                \stex_annotate:nnn{assign}{\l__stex_structures_dom_str,
5537                \bool_if:NTF\l_stex_symbol_or_var_bool{var://}{}\l_stex_get_symbol_uri_str}{}
5538              }
5539              \bool_if:NTF \l_stex_symbol_or_var_bool {
5540                \exp_args:Nxx \str_if_eq:nnF
5541                  {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5542                  {\prop_item:cn{l_stex_symdecl_var://\l_stex_get_symbol_uri_str _prop}{args}}{
5543                  \msg_error:nnxxxx{stex}{error/incompatible}
5544                    {\l__stex_structures_dom_str}
5545                    {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5546                    {\l_stex_get_symbol_uri_str}
5547                    {\prop_item:cn{l_stex_symdecl_var://\l_stex_get_symbol_uri_str _prop}{args}}
5548                }
5549                \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n
5550              }{
5551                \exp_args:Nxx \str_if_eq:nnF
5552                  {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5553                  {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
5554                  \msg_error:nnxxxx{stex}{error/incompatible}
5555                    {\l__stex_structures_dom_str}
5556                    {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5557                    {\l_stex_get_symbol_uri_str}
5558                    {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
5559                }
5560                \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n {
5561              }
5562            }
5563          }
5564          \tl_gclear:N \g__stex_structures_aftergroup_tl
5565          \seq_map_inline:Nn \l__stex_structures_fields_seq {
5566            \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdec
5567            \stex_debug:nn{varinstantiate}{Field~\l_tmpa_str :~##1}
5568            \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
5569              \stex_find_notation:nn{##1}{}
5570              \cs_gset_eq:cc{g__stex_structures_tmpa_\l_tmpa_str _cs}
5571                {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5572              \stex_debug:nn{varinstantiate}{Notation:~\cs_meaning:c{g__stex_structures_tmpa_\l_
5573              \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
5574                \cs_gset_eq:cc {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
5575                  {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5576                \stex_debug:nn{varinstantiate}{Operator~Notation:~\cs_meaning:c{g__stex_struct
5577              }
5578            }
5579
5580            \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
5581              \prop_set_from_keyval:cn { l_stex_symdecl_ var://\l_tmpa_str _prop}{
5582                name   = \l_tmpa_str ,
5583                args   = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
5584                arity  = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
5585                assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs} ,
5586                argnames = {\prop_item:cn {l_stex_symdecl_##1_prop}{argnames}} ,
```

226

```
             }
             \cs_set_eq:cc {stex_var_notation_\l_tmpa_str _cs}
               {g__stex_structures_tmpa_\l_tmpa_str _cs}
             \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str _cs}
               {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
           }
           \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\stex_inv
         }
         \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
           \prop_set_from_keyval:cn {l_stex_varinstance_\l__stex_structures_name_str _prop }{
             domain = \l_stex_get_structure_module_str ,
             \prop_to_keyval:N \l_tmpa_prop
           }
           \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
           \tl_set:cn {l_stex_varinstance_\l__stex_structures_name_str _op_tl}{
             \exp_args:Nnx \exp_not:N \use:nn {
               \str_set:Nn \exp_not:N \STEXInternalCurrentSymbolStr {var://\l__stex_structures_
               \_stex_term_omv:nn {var://\l__stex_structures_name_str}{
                 \exp_not:n{
                   \_varcomp{#4}
                 }
               }
             }{
               \exp_not:n{\_stex_reset:N \STEXInternalCurrentSymbolStr}
             }
           }
         }
       }
     \stex_debug:nn{varinstantiate}{\expandafter\detokenize\expandafter{\g__stex_structures_a
     \aftergroup\g__stex_structures_aftergroup_tl
   \endgroup
   \stex_smsmode_do:\ignorespacesandpars
}

\cs_new_protected:Nn \stex_invoke_instance:n {
   \peek_charcode_remove:NTF ! {
     \stex_invoke_symbol:n{#1}
   }{
     \_stex_invoke_instance:nn {#1}
   }
}


\cs_new_protected:Nn \stex_invoke_varinstance:n {
   \peek_charcode_remove:NTF ! {
     \exp_args:Nnx \use:nn {
       \def\comp{\_varcomp}
       \use:c{l_stex_varinstance_#1_op_tl}
     }{
       \_stex_reset:N \comp
     }
   }{
     \_stex_invoke_varinstance:nn {#1}
   }
```

```
5641 }
5642
5643 \cs_new_protected:Nn \_stex_invoke_instance:nn {
5644   \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
5645     \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
5646   }{
5647     \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
5648     \msg_error:nnxxx{stex}{error/unknownfield}{#2}{#1}{
5649       \prop_to_keyval:N \l_tmpa_prop
5650     }
5651   }
5652 }
5653
5654 \cs_new_protected:Nn \_stex_invoke_varinstance:nn {
5655   \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
5656     \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
5657     \l_tmpa_tl
5658   }{
5659     \msg_error:nnnnn{stex}{error/unknownfield}{#2}{#1}{}
5660   }
5661 }
```

(*End definition for* `\instantiate`. *This function is documented on page* *38.*)

`\stex_invoke_structure:nnn`

```
5662 % #1: URI of the instance
5663 % #2: URI of the instantiated module
5664 \cs_new_protected:Nn \stex_invoke_structure:nnn {
5665   \tl_if_empty:nTF{ #3 }{
5666     \prop_set_eq:Nc \l__stex_structures_structure_prop {
5667       c_stex_feature_ #2 _prop
5668     }
5669     \tl_clear:N \l_tmpa_tl
5670     \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
5671     \seq_map_inline:Nn \l_tmpa_seq {
5672       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
5673       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
5674       \cs_if_exist:cT {
5675         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
5676       }{
5677         \tl_if_empty:NF \l_tmpa_tl {
5678           \tl_put_right:Nn \l_tmpa_tl {,}
5679         }
5680         \tl_put_right:Nx \l_tmpa_tl {
5681           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
5682         }
5683       }
5684     }
5685     \exp_args:No \mathstruct \l_tmpa_tl
5686   }{
5687     \stex_invoke_symbol:n{#1/#3}
5688   }
5689 }
```

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

5690 ⟨/package⟩

Chapter 34

STEX -Statements Implementation

5692

5693 %%%%%%%%%%%%   features.dtx   %%%%%%%%%%%%

5694

5695 ⟨@@=stex_statements⟩

Warnings and error messages

5696

\titleemph

5697 \def\titleemph#1{\textbf{#1}}

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

34.1   Definitions

definiendum

5698 \keys_define:nn {stex / definiendum }{
5699   pre   .tl_set:N   = \l__stex_statements_definiendum_pre_tl,
5700   post   .tl_set:N   = \l__stex_statements_definiendum_post_tl,
5701   root   .str_set_x:N = \l__stex_statements_definiendum_root_str,
5702   gfa   .str_set_x:N = \l__stex_statements_definiendum_gfa_str
5703 }
5704 \cs_new_protected:Nn __stex_statements_definiendum_args:n {
5705   \str_clear:N \l__stex_statements_definiendum_root_str
5706   \tl_clear:N \l__stex_statements_definiendum_post_tl
5707   \str_clear:N \l__stex_statements_definiendum_gfa_str
5708   \keys_set:nn { stex / definiendum }{ #1 }
5709 }
5710 \NewDocumentCommand \definiendum { O{} m m} {
5711   __stex_statements_definiendum_args:n { #1 }
5712   \stex_get_symbol:n { #2 }
5713   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5714   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
5715   \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {

```
5716          \tl_set:Nn \l_tmpa_tl { #3 }
5717        } {
5718          \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
5719          \tl_set:Nn \l_tmpa_tl {
5720            \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
5721          }
5722        }
5723      } {
5724        \tl_set:Nn \l_tmpa_tl { #3 }
5725      }
5726
5727      % TODO root
5728      \stex_html_backend:TF {
5729        \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
5730      } {
5731        \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
5732      }
5733 }
5734 \stex_deactivate_macro:Nn \definiendum {definition~environments}
```

(*End definition for* `definiendum`. *This function is documented on page* *48*.)

definame

```
5735
5736 \NewDocumentCommand \definame { O{} m } {
5737   \__stex_statements_definiendum_args:n { #1 }
5738   % TODO: root
5739   \stex_get_symbol:n { #2 }
5740   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5741   \str_set:Nx \l_tmpa_str {
5742     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5743   }
5744   \str_replace_all:Nnn \l_tmpa_str {-} {~}
5745   \stex_html_backend:TF {
5746     \stex_if_do_html:T {
5747       \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5748         \l_tmpa_str\l__stex_statements_definiendum_post_tl
5749       }
5750     }
5751   } {
5752     \exp_args:Nnx \defemph@uri {
5753       \l_tmpa_str\l__stex_statements_definiendum_post_tl
5754     } { \l_stex_get_symbol_uri_str }
5755   }
5756 }
5757 \stex_deactivate_macro:Nn \definame {definition~environments}
5758
5759 \NewDocumentCommand \Definame { O{} m } {
5760   \__stex_statements_definiendum_args:n { #1 }
5761   \stex_get_symbol:n { #2 }
5762   \str_set:Nx \l_tmpa_str {
5763     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5764   }
5765   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
```

```
5766    \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5767    \stex_html_backend:TF {
5768      \stex_if_do_html:T {
5769        \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5770          \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5771        }
5772      }
5773    } {
5774      \exp_args:Nnx \defemph@uri {
5775        \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5776      } { \l_stex_get_symbol_uri_str }
5777    }
5778  }
5779  \stex_deactivate_macro:Nn \Definame {definition~environments}
5780
5781  \NewDocumentCommand \premise { m }{
5782    \noindent\stex_annotate:nnn{ premise }{}{\ignorespaces #1 }
5783  }
5784  \NewDocumentCommand \conclusion { m }{
5785    \noindent\stex_annotate:nnn{ conclusion }{}{\ignorespaces #1 }
5786  }
5787  \NewDocumentCommand \definiens { O{} m }{
5788    \str_clear:N \l_stex_get_symbol_uri_str
5789    \tl_if_empty:nF {#1} {
5790      \stex_get_symbol:n { #1 }
5791    }
5792    \str_if_empty:NT \l_stex_get_symbol_uri_str {
5793      \int_compare:nNnTF {\clist_count:N \l__stex_statements_sdefinition_for_clist} = 1 {
5794        \str_set:Nx \l_stex_get_symbol_uri_str {\clist_item:Nn \l__stex_statements_sdefinition
5795      }{
5796        % TODO throw error
5797      }
5798    }
5799    \str_if_eq:eeT {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{module}}
5800      {\l_stex_current_module_str}{
5801        \str_if_eq:eeF {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defin
5802        {true}{
5803          \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5804          \exp_args:Nx \stex_add_to_current_module:n {
5805            \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5806          }
5807        }
5808    }
5809    \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
5810  }
5811
5812  \NewDocumentCommand \varbindforall {m}{
5813    \stex_symbol_or_var:n {#1}
5814    \bool_if:NTF\l_stex_symbol_or_var_bool{
5815      \stex_if_do_html:T {
5816        \stex_annotate_invisible:nnn {bindtype}{forall,\l_stex_get_symbol_uri_str}{}
5817      }
5818    }{
5819      % todo throw error
```

232

```
5820        }
5821  }
5822
5823  \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
5824  \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
5825  \stex_deactivate_macro:Nn \definiens {definition~environments}
5826  \stex_deactivate_macro:Nn \varbindforall {definition~or~assertion~environments}
5827
```

(*End definition for* `definame`*. This function is documented on page* *48.*)

sdefinition (*env.*)

```
5828
5829  \keys_define:nn {stex / sdefinition }{
5830    type     .str_set_x:N  = \sdefinitiontype,
5831    id       .str_set_x:N  = \sdefinitionid,
5832    name     .str_set_x:N  = \sdefinitionname,
5833    for      .clist_set:N   = \l__stex_statements_sdefinition_for_clist ,
5834    title    .tl_set:N       = \sdefinitiontitle
5835  }
5836  \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
5837    \str_clear:N \sdefinitiontype
5838    \str_clear:N \sdefinitionid
5839    \str_clear:N \sdefinitionname
5840    \clist_clear:N \l__stex_statements_sdefinition_for_clist
5841    \tl_clear:N \sdefinitiontitle
5842    \keys_set:nn { stex / sdefinition }{ #1 }
5843  }
5844
5845  \NewDocumentEnvironment{sdefinition}{O{}}{
5846    \__stex_statements_sdefinition_args:n{ #1 }
5847    \stex_reactivate_macro:N \definiendum
5848    \stex_reactivate_macro:N \definame
5849    \stex_reactivate_macro:N \Definame
5850    \stex_reactivate_macro:N \premise
5851    \stex_reactivate_macro:N \definiens
5852    \stex_reactivate_macro:N \varbindforall
5853    \stex_if_smsmode:F{
5854      \seq_clear:N \l_tmpb_seq
5855      \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5856        \tl_if_empty:nF{ ##1 }{
5857          \stex_get_symbol:n { ##1 }
5858          \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5859            \l_stex_get_symbol_uri_str
5860          }
5861        }
5862      }
5863      \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5864      \exp_args:Nnnx
5865      \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpb_seq {,}}
5866      \str_if_empty:NF \sdefinitiontype {
5867        \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5868      }
5869      \str_if_empty:NF \sdefinitionname {
```

233

```
5870          \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5871        }
5872        \clist_set:No \l_tmpa_clist \sdefinitiontype
5873        \tl_clear:N \l_tmpa_tl
5874        \clist_map_inline:Nn \l_tmpa_clist {
5875          \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
5876            \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
5877          }
5878        }
5879        \tl_if_empty:NTF \l_tmpa_tl {
5880          \__stex_statements_sdefinition_start:
5881        }{
5882          \l_tmpa_tl
5883        }
5884      }
5885      \stex_ref_new_doc_target:n \sdefinitionid
5886      \stex_smsmode_do:
5887  }{
5888      \stex_suppress_html:n {
5889        \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5890      }
5891      \stex_if_smsmode:F {
5892        \clist_set:No \l_tmpa_clist \sdefinitiontype
5893        \tl_clear:N \l_tmpa_tl
5894        \clist_map_inline:Nn \l_tmpa_clist {
5895          \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
5896            \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
5897          }
5898        }
5899        \tl_if_empty:NTF \l_tmpa_tl {
5900          \__stex_statements_sdefinition_end:
5901        }{
5902          \l_tmpa_tl
5903        }
5904        \end{stex_annotate_env}
5905      }
5906  }
```

```
5907  \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
5908    \stex_par:\noindent\titleemph{Definition\tl_if_empty:NF \sdefinitiontitle {
5909      ~(\sdefinitiontitle)
5910    }~}
5911  }
5912  \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\stex_par:\medskip}
5913
5914  \newcommand\stexpatchdefinition[3][] {
5915      \str_set:Nx \l_tmpa_str{ #1 }
5916      \str_if_empty:NTF \l_tmpa_str {
5917        \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
5918        \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
5919      }{
5920        \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2
5921        \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
```

```
5922        }
5923    }
```

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

`\inlinedef`   inline:

```
5924  \keys_define:nn {stex / inlinedef }{
5925    type     .str_set_x:N  = \sdefinitiontype,
5926    id       .str_set_x:N  = \sdefinitionid,
5927    for      .clist_set:N  = \l__stex_statements_sdefinition_for_clist ,
5928    name     .str_set_x:N  = \sdefinitionname
5929  }
5930  \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
5931    \str_clear:N \sdefinitiontype
5932    \str_clear:N \sdefinitionid
5933    \str_clear:N \sdefinitionname
5934    \clist_clear:N \l__stex_statements_sdefinition_for_clist
5935    \keys_set:nn { stex / inlinedef }{ #1 }
5936  }
5937  \NewDocumentCommand \inlinedef { O{} m } {
5938    \begingroup
5939    \__stex_statements_inlinedef_args:n{ #1 }
5940    \stex_reactivate_macro:N \definiendum
5941    \stex_reactivate_macro:N \definame
5942    \stex_reactivate_macro:N \Definame
5943    \stex_reactivate_macro:N \premise
5944    \stex_reactivate_macro:N \definiens
5945    \stex_reactivate_macro:N \varbindforall
5946    \stex_ref_new_doc_target:n \sdefinitionid
5947    \stex_if_smsmode:TF{\stex_suppress_html:n {
5948      \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5949    }}{
5950      \seq_clear:N \l_tmpb_seq
5951      \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5952        \tl_if_empty:nF{ ##1 }{
5953          \stex_get_symbol:n { ##1 }
5954          \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5955            \l_stex_get_symbol_uri_str
5956          }
5957        }
5958      }
5959      \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5960      \ifvmode\noindent\fi
5961      \exp_args:Nnx
5962      \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpb_seq {,}}{
5963        \str_if_empty:NF \sdefinitiontype {
5964          \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5965        }
5966        #2
5967        \str_if_empty:NF \sdefinitionname {
5968          \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sdefinitionname}}
5969          \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5970        }
5971      }
```

235

```
5972      }
5973    \endgroup
5974    \stex_smsmode_do:
5975 }
```

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

34.2 Assertions

sassertion (*env.*)

```
5976
5977 \keys_define:nn {stex / sassertion }{
5978   type    .str_set_x:N  = \sassertiontype,
5979   id      .str_set_x:N  = \sassertionid,
5980   title   .tl_set:N     = \sassertiontitle ,
5981   for     .clist_set:N  = \l__stex_statements_sassertion_for_clist ,
5982   name    .str_set_x:N  = \sassertionname
5983 }
5984 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
5985   \str_clear:N \sassertiontype
5986   \str_clear:N \sassertionid
5987   \str_clear:N \sassertionname
5988   \clist_clear:N \l__stex_statements_sassertion_for_clist
5989   \tl_clear:N \sassertiontitle
5990   \keys_set:nn { stex / sassertion }{ #1 }
5991 }
5992
5993 %\tl_new:N \g__stex_statements_aftergroup_tl
5994
5995 \NewDocumentEnvironment{sassertion}{O{}}{
5996   \__stex_statements_sassertion_args:n{ #1 }
5997   \stex_reactivate_macro:N \premise
5998   \stex_reactivate_macro:N \conclusion
5999   \stex_reactivate_macro:N \varbindforall
6000   \stex_if_smsmode:F {
6001     \seq_clear:N \l_tmpb_seq
6002     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
6003       \tl_if_empty:nF{ ##1 }{
6004         \stex_get_symbol:n { ##1 }
6005         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6006           \l_stex_get_symbol_uri_str
6007         }
6008       }
6009     }
6010     \exp_args:Nnnx
6011     \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
6012     \str_if_empty:NF \sassertiontype {
6013       \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
6014     }
6015     \str_if_empty:NF \sassertionname {
6016       \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
6017     }
6018     \clist_set:No \l_tmpa_clist \sassertiontype
```

236

```
6019        \tl_clear:N \l_tmpa_tl
6020        \clist_map_inline:Nn \l_tmpa_clist {
6021          \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
6022            \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
6023          }
6024        }
6025        \tl_if_empty:NTF \l_tmpa_tl {
6026          \__stex_statements_sassertion_start:
6027        }{
6028          \l_tmpa_tl
6029        }
6030      }
6031      \str_if_empty:NTF \sassertionid {
6032        \str_if_empty:NF \sassertionname {
6033          \stex_ref_new_doc_target:n {}
6034        }
6035      } {
6036        \stex_ref_new_doc_target:n \sassertionid
6037      }
6038      \stex_smsmode_do:
6039    }{
6040      \str_if_empty:NF \sassertionname {
6041        \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
6042        \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
6043      }
6044      \stex_if_smsmode:F {
6045        \clist_set:No \l_tmpa_clist \sassertiontype
6046        \tl_clear:N \l_tmpa_tl
6047        \clist_map_inline:Nn \l_tmpa_clist {
6048          \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
6049            \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
6050          }
6051        }
6052        \tl_if_empty:NTF \l_tmpa_tl {
6053          \__stex_statements_sassertion_end:
6054        }{
6055          \l_tmpa_tl
6056        }
6057        \end{stex_annotate_env}
6058      }
6059    }
```

\stexpatchassertion

```
6060
6061  \cs_new_protected:Nn \__stex_statements_sassertion_start: {
6062    \stex_par:\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {
6063      (\sassertiontitle)
6064    }~}
6065  }
6066  \cs_new_protected:Nn \__stex_statements_sassertion_end: {\stex_par:\medskip}
6067
6068  \newcommand\stexpatchassertion[3][] {
6069      \str_set:Nx \l_tmpa_str{ #1 }
6070      \str_if_empty:NTF \l_tmpa_str {
```

```
6071        \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
6072        \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
6073      }{
6074        \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
6075        \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3
6076      }
6077  }
```

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

\inlineass   inline:

```
6078  \keys_define:nn {stex / inlineass }{
6079    type     .str_set_x:N  = \sassertiontype,
6080    id       .str_set_x:N  = \sassertionid,
6081    for      .clist_set:N  = \l__stex_statements_sassertion_for_clist ,
6082    name     .str_set_x:N  = \sassertionname
6083  }
6084  \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
6085    \str_clear:N \sassertiontype
6086    \str_clear:N \sassertionid
6087    \str_clear:N \sassertionname
6088    \clist_clear:N \l__stex_statements_sassertion_for_clist
6089    \keys_set:nn { stex / inlineass }{ #1 }
6090  }
6091  \NewDocumentCommand \inlineass { O{} m } {
6092    \begingroup
6093    \stex_reactivate_macro:N \premise
6094    \stex_reactivate_macro:N \conclusion
6095    \stex_reactivate_macro:N \varbindforall
6096    \__stex_statements_inlineass_args:n{ #1 }
6097    \str_if_empty:NTF \sassertionid {
6098      \str_if_empty:NF \sassertionname {
6099        \stex_ref_new_doc_target:n {}
6100      }
6101    } {
6102      \stex_ref_new_doc_target:n \sassertionid
6103    }
6104
6105    \stex_if_smsmode:TF{
6106      \str_if_empty:NF \sassertionname {
6107        \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
6108        \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
6109      }
6110    }{
6111      \seq_clear:N \l_tmpb_seq
6112      \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
6113        \tl_if_empty:nF{ ##1 }{
6114          \stex_get_symbol:n { ##1 }
6115          \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6116            \l_stex_get_symbol_uri_str
6117          }
6118        }
6119      }
6120      \ifvmode\noindent\fi
```

```
6121       \exp_args:Nnx
6122       \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpb_seq {,}}{
6123         \str_if_empty:NF \sassertiontype {
6124           \stex_annotate_invisible:nnn{typestrings}{\sassertiontype}{}
6125         }
6126         #2
6127         \str_if_empty:NF \sassertionname {
6128           \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
6129           \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
6130           \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
6131         }
6132       }
6133     }
6134     \endgroup
6135     \stex_smsmode_do:
6136 }
```

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

34.3   Examples

sexample *(env.)*

```
6137
6138 \keys_define:nn {stex / sexample }{
6139   type      .str_set_x:N  = \exampletype,
6140   id        .str_set_x:N  = \sexampleid,
6141   title     .tl_set:N     = \sexampletitle,
6142   name      .str_set_x:N  = \sexamplename ,
6143   for       .clist_set:N  = \l__stex_statements_sexample_for_clist,
6144 }
6145 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
6146   \str_clear:N \sexampletype
6147   \str_clear:N \sexampleid
6148   \str_clear:N \sexamplename
6149   \tl_clear:N \sexampletitle
6150   \clist_clear:N \l__stex_statements_sexample_for_clist
6151   \keys_set:nn { stex / sexample }{ #1 }
6152 }
6153
6154 \NewDocumentEnvironment{sexample}{O{}}{
6155   \__stex_statements_sexample_args:n{ #1 }
6156   \stex_reactivate_macro:N \premise
6157   \stex_reactivate_macro:N \conclusion
6158   \stex_if_smsmode:F {
6159     \seq_clear:N \l_tmpb_seq
6160     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
6161       \tl_if_empty:nF{ ##1 }{
6162         \stex_get_symbol:n { ##1 }
6163         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6164           \l_stex_get_symbol_uri_str
6165         }
6166       }
6167     }
```

239

```
6168        \exp_args:Nnnx
6169        \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpb_seq {,}}
6170        \str_if_empty:NF \sexampletype {
6171          \stex_annotate_invisible:nnn{typestrings}{\sexampletype}{}
6172        }
6173        \str_if_empty:NF \sexamplename {
6174          \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}
6175        }
6176        \clist_set:No \l_tmpa_clist \sexampletype
6177        \tl_clear:N \l_tmpa_tl
6178        \clist_map_inline:Nn \l_tmpa_clist {
6179          \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
6180            \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
6181          }
6182        }
6183        \tl_if_empty:NTF \l_tmpa_tl {
6184          \__stex_statements_sexample_start:
6185        }{
6186          \l_tmpa_tl
6187        }
6188      }
6189      \str_if_empty:NF \sexampleid {
6190        \stex_ref_new_doc_target:n \sexampleid
6191      }
6192      \stex_smsmode_do:
6193    }{
6194      \str_if_empty:NF \sexamplename {
6195        \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
6196      }
6197      \stex_if_smsmode:F {
6198        \clist_set:No \l_tmpa_clist \sexampletype
6199        \tl_clear:N \l_tmpa_tl
6200        \clist_map_inline:Nn \l_tmpa_clist {
6201          \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
6202            \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
6203          }
6204        }
6205        \tl_if_empty:NTF \l_tmpa_tl {
6206          \__stex_statements_sexample_end:
6207        }{
6208          \l_tmpa_tl
6209        }
6210        \end{stex_annotate_env}
6211      }
6212    }
```

\stexpatchexample

```
6213
6214 \cs_new_protected:Nn \__stex_statements_sexample_start: {
6215   \stex_par:\noindent\titleemph{Example~\tl_if_empty:NF \sexampletitle {
6216     (\sexampletitle)
6217   }~}
6218 }
6219 \cs_new_protected:Nn \__stex_statements_sexample_end: {\stex_par:\medskip}
```

```
6220
6221 \newcommand\stexpatchexample[3][] {
6222     \str_set:Nx \l_tmpa_str{ #1 }
6223     \str_if_empty:NTF \l_tmpa_str {
6224         \tl_set:Nn \__stex_statements_sexample_start: { #2 }
6225         \tl_set:Nn \__stex_statements_sexample_end: { #3 }
6226     }{
6227         \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
6228         \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
6229     }
6230 }
```

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

\inlineex   inline:

```
6231 \keys_define:nn {stex / inlineex }{
6232   type      .str_set_x:N  = \sexampletype,
6233   id        .str_set_x:N  = \sexampleid,
6234   for       .clist_set:N   = \l__stex_statements_sexample_for_clist ,
6235   name      .str_set_x:N  = \sexamplename
6236 }
6237 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
6238   \str_clear:N \sexampletype
6239   \str_clear:N \sexampleid
6240   \str_clear:N \sexamplename
6241   \clist_clear:N \l__stex_statements_sexample_for_clist
6242   \keys_set:nn { stex / inlineex }{ #1 }
6243 }
6244 \NewDocumentCommand \inlineex { O{} m } {
6245   \begingroup
6246   \stex_reactivate_macro:N \premise
6247   \stex_reactivate_macro:N \conclusion
6248   \__stex_statements_inlineex_args:n{ #1 }
6249   \str_if_empty:NF \sexampleid {
6250     \stex_ref_new_doc_target:n \sexampleid
6251   }
6252   \stex_if_smsmode:TF{
6253     \str_if_empty:NF \sexamplename {
6254       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\examplename}}
6255     }
6256   }{
6257     \seq_clear:N \l_tmpb_seq
6258     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
6259       \tl_if_empty:nF{ ##1 }{
6260         \stex_get_symbol:n { ##1 }
6261         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6262           \l_stex_get_symbol_uri_str
6263         }
6264       }
6265     }
6266     \ifvmode\noindent\fi
6267     \exp_args:Nnx
6268     \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpb_seq {,}}{
6269       \str_if_empty:NF \sexampletype {
```

```
6270            \stex_annotate_invisible:nnn{typestrings}{\sexampletype}{}
6271          }
6272          #2
6273        \str_if_empty:NF \sexamplename {
6274          \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
6275          \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}
6276        }
6277      }
6278    }
6279    \endgroup
6280    \stex_smsmode_do:
6281 }
```

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

34.4   Logical Paragraphs

sparagraph (*env.*)
```
6282 \keys_define:nn { stex / sparagraph} {
6283    id      .str_set_x:N  = \sparagraphid ,
6284    title   .tl_set:N     = \l_stex_sparagraph_title_tl ,
6285    type    .str_set_x:N  = \sparagraphtype ,
6286    for     .clist_set:N  = \l__stex_statements_sparagraph_for_clist ,
6287    from    .tl_set:N     = \sparagraphfrom ,
6288    to      .tl_set:N     = \sparagraphto ,
6289    start   .tl_set:N     = \l_stex_sparagraph_start_tl ,
6290    name    .str_set:N    = \sparagraphname ,
6291    imports .tl_set:N     = \l__stex_statements_sparagraph_imports_tl
6292 }
6293
6294 \cs_new_protected:Nn \stex_sparagraph_args:n {
6295    \tl_clear:N \l_stex_sparagraph_title_tl
6296    \tl_clear:N \sparagraphfrom
6297    \tl_clear:N \sparagraphto
6298    \tl_clear:N \l_stex_sparagraph_start_tl
6299    \tl_clear:N \l__stex_statements_sparagraph_imports_tl
6300    \str_clear:N \sparagraphid
6301    \str_clear:N \sparagraphtype
6302    \clist_clear:N \l__stex_statements_sparagraph_for_clist
6303    \str_clear:N \sparagraphname
6304    \keys_set:nn { stex / sparagraph }{ #1 }
6305 }
6306 \newif\if@in@omtext\@in@omtextfalse
6307
6308 \NewDocumentEnvironment {sparagraph} { O{} } {
6309    \stex_sparagraph_args:n { #1 }
6310    \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
6311      \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
6312    }{
6313      \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
6314    }
6315    \@in@omtexttrue
6316    \stex_if_smsmode:F {
```

```
6317    \seq_clear:N \l_tmpb_seq
6318    \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
6319      \tl_if_empty:nF{ ##1 }{
6320        \stex_get_symbol:n { ##1 }
6321        \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6322          \l_stex_get_symbol_uri_str
6323        }
6324      }
6325    }
6326    \exp_args:Nnnx
6327    \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}
6328    \str_if_empty:NF \sparagraphtype {
6329      \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
6330    }
6331    \str_if_empty:NF \sparagraphfrom {
6332      \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
6333    }
6334    \str_if_empty:NF \sparagraphto {
6335      \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
6336    }
6337    \str_if_empty:NF \sparagraphname {
6338      \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
6339    }
6340    \clist_set:No \l_tmpa_clist \sparagraphtype
6341    \tl_clear:N \l_tmpa_tl
6342    \clist_map_inline:Nn \sparagraphtype {
6343      \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
6344        \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
6345      }
6346    }
6347    \stex_csl_to_imports:No \usemodule \l__stex_statements_sparagraph_imports_tl
6348    \tl_if_empty:NTF \l_tmpa_tl {
6349      \__stex_statements_sparagraph_start:
6350    }{
6351      \l_tmpa_tl
6352    }
6353  }
6354  \clist_set:No \l_tmpa_clist \sparagraphtype
6355  \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}
6356  {
6357    \stex_reactivate_macro:N \definiendum
6358    \stex_reactivate_macro:N \definame
6359    \stex_reactivate_macro:N \Definame
6360    \stex_reactivate_macro:N \premise
6361    \stex_reactivate_macro:N \definiens
6362  }
6363  \str_if_empty:NTF \sparagraphid {
6364    \str_if_empty:NTF \sparagraphname {
6365      \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
6366        \stex_ref_new_doc_target:n {}
6367      }
6368    } {
6369      \stex_ref_new_doc_target:n {}
6370    }
```

```
6371      } {
6372        \stex_ref_new_doc_target:n \sparagraphid
6373      }
6374      \exp_args:NNx
6375      \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
6376        \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
6377          \tl_if_empty:nF{ ##1 }{
6378            \stex_get_symbol:n { ##1 }
6379            \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
6380          }
6381        }
6382      }
6383      \stex_smsmode_do:
6384      \ignorespacesandpars
6385 }{
6386      \str_if_empty:NF \sparagraphname {
6387        \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
6388        \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
6389      }
6390      \stex_if_smsmode:F {
6391        \clist_set:No \l_tmpa_clist \sparagraphtype
6392        \tl_clear:N \l_tmpa_tl
6393        \clist_map_inline:Nn \l_tmpa_clist {
6394          \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
6395            \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
6396          }
6397        }
6398        \tl_if_empty:NTF \l_tmpa_tl {
6399          \__stex_statements_sparagraph_end:
6400        }{
6401          \l_tmpa_tl
6402        }
6403        \end{stex_annotate_env}
6404      }
6405 }
```

```
6406
6407 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
6408   \stex_par:\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
6409     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
6410       \titleemph{\l_stex_sparagraph_title_tl}:~
6411     }
6412   }{
6413     \titleemph{\l_stex_sparagraph_start_tl}~
6414   }
6415 }
6416 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\stex_par:\medskip}
6417
6418 \newcommand\stexpatchparagraph[3][] {
6419     \str_set:Nx \l_tmpa_str{ #1 }
6420     \str_if_empty:NTF \l_tmpa_str {
6421       \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
6422       \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
```

```
6423      }{
6424        \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
6425        \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3
6426      }
6427  }
6428
6429  \keys_define:nn { stex / inlinepara} {
6430    id      .str_set_x:N  = \sparagraphid ,
6431    type    .str_set_x:N  = \sparagraphtype ,
6432    for     .clist_set:N   = \l__stex_statements_sparagraph_for_clist ,
6433    from    .tl_set:N      = \sparagraphfrom ,
6434    to      .tl_set:N      = \sparagraphto ,
6435    name    .str_set:N     = \sparagraphname
6436  }
6437  \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
6438    \tl_clear:N \sparagraphfrom
6439    \tl_clear:N \sparagraphto
6440    \str_clear:N \sparagraphid
6441    \str_clear:N \sparagraphtype
6442    \clist_clear:N \l__stex_statements_sparagraph_for_clist
6443    \str_clear:N \sparagraphname
6444    \keys_set:nn { stex / inlinepara }{ #1 }
6445  }
6446  \NewDocumentCommand \inlinepara { O{} m } {
6447    \begingroup
6448    \__stex_statements_inlinepara_args:n{ #1 }
6449    \clist_set:No \l_tmpa_clist \sparagraphtype
6450    \str_if_empty:NTF \sparagraphid {
6451      \str_if_empty:NTF \sparagraphname {
6452        \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
6453          \stex_ref_new_doc_target:n {}
6454        }
6455      } {
6456        \stex_ref_new_doc_target:n {}
6457      }
6458    } {
6459      \stex_ref_new_doc_target:n \sparagraphid
6460    }
6461    \stex_if_smsmode:TF{
6462      \str_if_empty:NF \sparagraphname {
6463        \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
6464        \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
6465      }
6466    }{
6467      \seq_clear:N \l_tmpb_seq
6468      \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
6469        \tl_if_empty:nF{ ##1 }{
6470          \stex_get_symbol:n { ##1 }
6471          \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6472            \l_stex_get_symbol_uri_str
6473          }
6474        }
6475      }
6476      \ifvmode\noindent\fi
```

245

```
6477      \exp_args:Nnx
6478      \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}{
6479        \str_if_empty:NF \sparagraphtype {
6480          \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
6481        }
6482        \str_if_empty:NF \sparagraphfrom {
6483          \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
6484        }
6485        \str_if_empty:NF \sparagraphto {
6486          \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
6487        }
6488        \str_if_empty:NF \sparagraphname {
6489          \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
6490          \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
6491          \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
6492        }
6493        \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
6494          \clist_map_inline:Nn \l_tmpb_seq {
6495            \stex_ref_new_sym_target:n {##1}
6496          }
6497        }
6498        #2
6499      }
6500    }
6501    \endgroup
6502    \stex_smsmode_do:
6503 }
6504
```

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

```
6505 ⟨/package⟩
```

Chapter 35

The Implementation

```
6506  ⟨∗package⟩
6507  ⟨@@=stex_sproof⟩
6508
6509  %%%%%%%%%%%%   sproof.dtx   %%%%%%%%%%%%
6510
```

35.1   Proofs

We first define some keys for the proof environment.

```
6511  \keys_define:nn { stex / spf } {
6512    id          .str_set_x:N  = \spfid,
6513    for         .clist_set:N  = \l__stex_sproof_spf_for_clist ,
6514    from        .tl_set:N     = \l__stex_sproof_spf_from_tl ,
6515    proofend    .tl_set:N     = \l__stex_sproof_spf_proofend_tl,
6516    type        .str_set_x:N  = \spftype,
6517    title       .tl_set:N     = \spftitle,
6518    continues   .tl_set:N     = \l__stex_sproof_spf_continues_tl,
6519    functions   .tl_set:N     = \l__stex_sproof_spf_functions_tl,
6520    term        .tl_set:N     = \l__stex_sproof_spf_term_tl,
6521    method      .tl_set:N     = \l__stex_sproof_spf_method_tl,
6522    hide        .bool_set:N   = \l__stex_sproof_spf_hide_bool
6523  }
6524  \cs_new_protected:Nn \__stex_sproof_spf_args:n {
6525  \str_clear:N \spfid
6526  \tl_clear:N \l__stex_sproof_spf_for_tl
6527  \tl_clear:N \l__stex_sproof_spf_from_tl
6528  \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
6529  \str_clear:N \spftype
6530  \tl_clear:N \spftitle
6531  \tl_clear:N \l__stex_sproof_spf_continues_tl
6532  \tl_clear:N \l__stex_sproof_spf_term_tl
6533  \tl_clear:N \l__stex_sproof_spf_functions_tl
6534  \tl_clear:N \l__stex_sproof_spf_method_tl
6535    \bool_set_false:N \l__stex_sproof_spf_hide_bool
6536  \keys_set:nn { stex / spf }{ #1 }
6537  }
6538  \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`

```
6539 \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.

```
6540 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
6541 \cs_new_protected:Npn \sproofnumber {
6542   \int_set:Nn \l_tmpa_int {1}
6543   \bool_while_do:nn {
6544     \int_compare_p:nNn {
6545       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6546     } > 0
6547   }{
6548     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
6549     \int_incr:N \l_tmpa_int
6550   }
6551 }
6552 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
6553   \int_set:Nn \l_tmpa_int {1}
6554   \bool_while_do:nn {
6555     \int_compare_p:nNn {
6556       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6557     } > 0
6558   }{
6559     \int_incr:N \l_tmpa_int
6560   }
6561   \int_compare:nNnF \l_tmpa_int = 1 {
6562     \int_decr:N \l_tmpa_int
6563   }
6564   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
6565     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1
6566   }
6567 }
6568
6569 \cs_new_protected:Npn \__stex_sproof_add_counter: {
6570   \int_set:Nn \l_tmpa_int {1}
6571   \bool_while_do:nn {
6572     \int_compare_p:nNn {
6573       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6574     } > 0
6575   }{
6576     \int_incr:N \l_tmpa_int
6577   }
6578   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 1 }
6579 }
6580
```

```
6581 \cs_new_protected:Npn \__stex_sproof_remove_counter: {
6582   \int_set:Nn \l_tmpa_int {1}
6583   \bool_while_do:nn {
6584     \int_compare_p:nNn {
6585       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6586     } > 0
6587   }{
6588     \int_incr:N \l_tmpa_int
6589   }
6590   \int_decr:N \l_tmpa_int
6591   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 0 }
6592 }
```

\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

```
6593 \def\sproof@box{
6594   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
6595 }
6596 \def\sproofend{
6597   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
6598     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
6599   }
6600 }
```

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

spf@*@kw

```
6601 \def\spf@proofsketch@kw{Proof~Sketch}
6602 \def\spf@proof@kw{Proof}
6603 \def\spf@step@kw{Step}
```

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

For the other languages, we set up triggers

```
6604 \AddToHook{begindocument}{
6605   \ltx@ifpackageloaded{babel}{
6606     \makeatletter
6607     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6608     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6609       \input{sproof-ngerman.ldf}
6610     }
6611     \clist_if_in:NnT \l_tmpa_clist {finnish}{
6612       \input{sproof-finnish.ldf}
6613     }
6614     \clist_if_in:NnT \l_tmpa_clist {french}{
6615       \input{sproof-french.ldf}
6616     }
6617     \clist_if_in:NnT \l_tmpa_clist {russian}{
6618       \input{sproof-russian.ldf}
6619     }
6620     \makeatother
6621   }{}
6622 }
```

```
6623 \newcommand\spfsketch[2][]{
6624   \begingroup
6625   \let \premise \stex_proof_premise:
6626   \__stex_sproof_spf_args:n{#1}
6627   \stex_if_smsmode:TF {
6628     \str_if_empty:NF \spfid {
6629       \stex_ref_new_doc_target:n \spfid
6630     }
6631   }{
6632     \seq_clear:N \l_tmpa_seq
6633     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6634       \tl_if_empty:nF{ ##1 }{
6635         \stex_get_symbol:n { ##1 }
6636         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6637           \l_stex_get_symbol_uri_str
6638         }
6639       }
6640     }
6641     \exp_args:Nnx
6642     \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
6643       \str_if_empty:NF \spftype {
6644         \stex_annotate_invisible:nnn{type}{\spftype}{}
6645       }
6646       \clist_set:No \l_tmpa_clist \spftype
6647       \tl_set:Nn \l_tmpa_tl {
6648         \titleemph{
6649           \tl_if_empty:NTF \spftitle {
6650             \spf@proofsketch@kw
6651           }{
6652             \spftitle
6653           }
6654         }:~
6655       }
6656       \clist_map_inline:Nn \l_tmpa_clist {
6657         \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6658           \tl_clear:N \l_tmpa_tl
6659         }
6660       }
6661       \str_if_empty:NF \spfid {
6662         \stex_ref_new_doc_target:n \spfid
6663       }
6664       \l_tmpa_tl #2 \sproofend
6665     }
6666   }
6667   \endgroup
6668   \stex_smsmode_do:
6669 }
6670
```

(*End definition for* spfsketch*. This function is documented on page 54.*)

```
6671 \bool_set_false:N \l__stex_sproof_in_spfblock_bool
```

250

```
6672
6673 \cs_new_protected:Nn \__stex_sproof_maybe_comment: {
6674   \bool_if:NF \l__stex_sproof_in_spfblock_bool {
6675     \par \setbox \l_tmpa_box \vbox \bgroup \everypar{\__stex_sproof_start_comment:}
6676   }
6677 }
6678 \cs_new_protected:Nn \__stex_sproof_maybe_comment_end: {
6679   \bool_if:NF \l__stex_sproof_in_spfblock_bool { \egroup }
6680 }
6681 \cs_new_protected:Nn \__stex_sproof_start_comment: {
6682   \csname @ @ par\endcsname\egroup\item[]\bgroup\stexcommentfont
6683 }
6684
```

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

\stexcommentfont

```
6685 \cs_new_protected:Npn \stexcommentfont {
6686   \small\itshape
6687 }
```

(*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.

```
6688 \cs_new_protected:Nn \__stex_sproof_start_env:nnn {
6689   \seq_clear:N \l_tmpa_seq
6690   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6691     \tl_if_empty:nF{ ##1 }{
6692       \stex_get_symbol:n { ##1 }
6693       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6694         \l_stex_get_symbol_uri_str
6695       }
6696     }
6697   }
6698   \exp_args:Nnnx
6699   \begin{stex_annotate_env}{#1}{\seq_use:Nn \l_tmpa_seq {,}}
6700   \str_if_empty:NF \spftype {
6701     \stex_annotate_invisible:nnn{type}{\spftype}{}
6702   }
6703   #3 {~\stex_annotate:nnn{spftitle}{}{#2}}
6704   \str_if_empty:NF \spfid {
6705     \stex_ref_new_doc_target:n \spfid
6706   }
6707   \begin{stex_annotate_env}{spfbody}{\bool_if:NTF \l__stex_sproof_spf_hide_bool {false}{true
6708   \bool_if:NT \l__stex_sproof_spf_hide_bool{
6709     \stex_html_backend:F{\setbox\l_tmpa_box\vbox\bgroup}
6710   }
6711   \begin{list}{}{
6712     \setlength\topsep{0pt}
6713     \setlength\parsep{0pt}
6714     \setlength\rightmargin{0pt}
```

```
6715
6716    }\__stex_sproof_maybe_comment:
6717  }
6718  \cs_new_protected:Nn \__stex_sproof_end_env:n {
6719    \stex_if_smsmode:F{
6720      \__stex_sproof_maybe_comment_end:
6721      \end{list}
6722      \bool_if:NT \l__stex_sproof_spf_hide_bool{
6723        \stex_html_backend:F{\egroup}
6724      }
6725      \clist_set:No \l_tmpa_clist \spftype
6726      #1
6727      \end{stex_annotate_env}
6728      \end{stex_annotate_env}
6729    }
6730  }
6731  \NewDocumentEnvironment{sproof}{s O{} m}{
6732    \intarray_gzero:N \l__stex_sproof_counter_intarray
6733    \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
6734    \stex_reactivate_macro:N \yield
6735    \stex_reactivate_macro:N \eqstep
6736    \stex_reactivate_macro:N \assumption
6737    \stex_reactivate_macro:N \conclude
6738    \stex_reactivate_macro:N \spfstep
6739    \__stex_sproof_spf_args:n{#2}
6740    \stex_if_smsmode:TF {
6741      \str_if_empty:NF \spfid {
6742        \stex_ref_new_doc_target:n \spfid
6743      }
6744    }{
6745      \__stex_sproof_start_env:nnn{sproof}{#3}{
6746        \clist_set:No \l_tmpa_clist \spftype
6747        \tl_clear:N \l_tmpa_tl
6748        \clist_map_inline:Nn \l_tmpa_clist {
6749          \tl_if_exist:cT {__stex_sproof_sproof_##1_start:}{
6750            \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_start:}}
6751          }
6752          \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6753            \tl_set:Nn \l_tmpa_tl {\use:n{}}
6754          }
6755        }
6756        \tl_if_empty:NTF \l_tmpa_tl {
6757          \__stex_sproof_sproof_start:
6758        }{
6759          \l_tmpa_tl
6760        }
6761      }
6762    }
6763    \stex_smsmode_do:
6764  }{\__stex_sproof_end_env:n{
6765    \tl_clear:N \l_tmpa_tl
6766    \clist_map_inline:Nn \l_tmpa_clist {
6767      \tl_if_exist:cT {__stex_sproof_sproof_##1_end:}{
6768        \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_end:}}
```

```
6769          }
6770      }
6771      \tl_if_empty:NTF \l_tmpa_tl {
6772          \__stex_sproof_sproof_end:
6773      }{
6774          \l_tmpa_tl
6775      }
6776 }}
6777 \NewDocumentEnvironment{subproof}{s O{} m}{
6778      \__stex_sproof_spf_args:n{#2}
6779      \stex_if_smsmode:TF {
6780          \str_if_empty:NF \spfid {
6781              \stex_ref_new_doc_target:n \spfid
6782          }
6783      }{
6784          \__stex_sproof_start_env:nnn{subproof}{\item[\sproofnumber]\ignorespacesandpars #3}{}
6785      }
6786      \__stex_sproof_add_counter:
6787      \stex_smsmode_do:
6788 }{\__stex_sproof_remove_counter:\__stex_sproof_end_env:n{}
6789      \bool_if:NT \l__stex_sproof_inc_counter_bool {
6790          \__stex_sproof_inc_counter:
6791      }
6792      \aftergroup\__stex_sproof_maybe_comment:
6793 }
6794 \AddToHook{env/subproof/before}{\__stex_sproof_maybe_comment_end:}
6795
6796 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
6797      \par\noindent\titleemph{
6798          \tl_if_empty:NTF \spftype {
6799              \spf@proof@kw
6800          }{
6801              \spftype
6802          }
6803      }:
6804 }
6805 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
6806
6807 \newcommand\stexpatchproof[3][] {
6808      \str_set:Nx \l_tmpa_str{ #1 }
6809      \str_if_empty:NTF \l_tmpa_str {
6810          \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
6811          \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
6812      }{
6813          \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
6814          \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
6815      }
6816 }
```

\pstep
\conclude
\assumption
\have
\eqstep

```
6817
6818 \keys_define:nn { stex / spfsteps } {
6819      id            .str_set_x:N  = \spfstepid,
6820      for           .clist_set:N  = \l__stex_sproof_spf_for_clist ,
```

```
6821   type        .str_set_x:N  = \spftype,
6822   title       .tl_set:N     = \spftitle,
6823   method      .tl_set:N     = \l__stex_sproof_spf_method_tl,
6824   term        .tl_set:N     = \l__stex_sproof_spf_term_tl
6825 }
6826 \cs_new_protected:Nn \__stex_sproof_spfstep_args:n {
6827 \str_clear:N \spfstepid
6828 \clist_clear:N \l__stex_sproof_spf_for_clist
6829 \str_clear:N \spftype
6830 \tl_clear:N \l__stex_sproof_spf_method_tl
6831 \tl_clear:N \l__stex_sproof_spf_term_tl
6832   %\bool_set_false:N \l__stex_sproof_inc_counter_bool
6833 \keys_set:nn { stex / spfsteps }{ #1 }
6834 }
6835
6836 \cs_new_protected:Nn \__stex_sproof_make_step_macro:Nnnnn {
6837   \NewDocumentCommand #1 {s O{} +m} {
6838     \__stex_sproof_maybe_comment_end:
6839
6840     \__stex_sproof_spfstep_args:n{##2}
6841     \stex_annotate:nnn{spfstep}{#2}{
6842       \tl_if_empty:NF \l__stex_sproof_spf_term_tl {
6843         \stex_annotate_invisible:nnn{spfyield}{}{$\l__stex_sproof_spf_term_tl$}
6844       }
6845       \bool_if:NTF \l__stex_sproof_in_spfblock_bool {
6846         #4
6847       }{
6848         \item[\IfBooleanTF ##1 {}{#3}]
6849       }
6850       \ignorespacesandpars ##3
6851     }
6852     \bool_if:NF \l__stex_sproof_in_spfblock_bool { \IfBooleanTF ##1 {}{ #5 } }
6853     \__stex_sproof_maybe_comment:
6854   }
6855   \stex_deactivate_macro:Nn #1 {sproof~environments}
6856 }
6857
6858 \__stex_sproof_make_step_macro:Nnnnn \assumption {assumption} \sproofnumber {} \__stex_sproo
6859 \__stex_sproof_make_step_macro:Nnnnn \conclude {conclusion} {$\Rightarrow$} {} {}
6860 \__stex_sproof_make_step_macro:Nnnnn \spfstep {} \sproofnumber {} \__stex_sproof_inc_counter
6861
6862 \NewDocumentCommand \eqstep {s m}{
6863   \__stex_sproof_maybe_comment_end:
6864   \bool_if:NTF \l__stex_sproof_in_spfblock_bool {
6865     $=$
6866   }{
6867     \item[$=$]
6868   }
6869   $\stex_annotate:nnn{spfstep}{eq}{ #2 }$
6870   \__stex_sproof_maybe_comment:
6871 }
6872 \stex_deactivate_macro:Nn \eqstep {sproof~environments}
6873
6874 \NewDocumentCommand \yield {+m}{
```

254

```
6875        \stex_annotate:nnn{spfyield}{}{ #1 }
6876   }
6877   \stex_deactivate_macro:Nn \yield {sproof~environments}
6878
6879   \NewDocumentEnvironment{spfblock}{}{
6880     \item[]
6881     \bool_set_true:N \l__stex_sproof_in_spfblock_bool
6882   }{
6883     \aftergroup\__stex_sproof_maybe_comment:
6884   }
6885   \AddToHook{env/spfblock/before}{\__stex_sproof_maybe_comment_end:}
6886
```

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

```
6887   \NewDocumentCommand\spfidea{O{} +m}{
6888     \__stex_sproof_spf_args:n{#1}
6889     \titleemph{
6890       \tl_if_empty:NTF \spftype {Proof~Idea}{
6891         \spftype
6892       }:
6893     }~#2
6894     \sproofend
6895   }
```

(*End definition for* `\spfidea`*. This function is documented on page* *54.*)

```
6896   \newcommand\spfjust[1]{
6897     #1
6898   }
6899   ⟨/package⟩
```

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

255

Chapter 36

sTEX
-Others Implementation

```
6900 ⟨∗package⟩
6901
6902 %%%%%%%%%%%%   others.dtx   %%%%%%%%%%%%
6903
6904 ⟨@@=stex_others⟩
```

Warnings and error messages
```
6905   % None
```

\MSC   Math subject classifier
```
6906 \NewDocumentCommand \MSC {m} {
6907   % TODO
6908 }
```

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

Patching tikzinput, if loaded
```
6909 \@ifpackageloaded{tikzinput}{
6910   \RequirePackage{stex-tikzinput}
6911 }{}
6912
6913 \bool_if:NT \c_stex_persist_mode_bool {
6914   \let\__stex_notation_restore_notation_old:nnnnn
6915     \__stex_notation_restore_notation:nnnnn
6916   \def\__stex_notation_restore_notation_new:nnnnn#1#2#3#4#5{
6917     \__stex_notation_restore_notation_old:nnnnn{#1}{#2}{#3}{#4}{#5}
6918     \ExplSyntaxOn
6919   }
6920   \def\__stex_notation_restore_notation:nnnnn{
6921     \ExplSyntaxOff
6922     \catcode`~10
6923     \__stex_notation_restore_notation_new:nnnnn
6924   }
6925   \input{\jobname.sms}
6926   \let\__stex_notation_restore_notation:nnnnn
6927     \__stex_notation_restore_notation_old:nnnnn
6928   \prop_if_exist:NT\c_stex_mathhub_main_manifest_prop{
```

```
6929      \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
6930        \l_tmpa_str
6931      \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str _manifest_prop }
6932        \c_stex_mathhub_main_manifest_prop
6933      \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
6934    }
6935 }
6936
6937 \stex_get_document_uri:
6938 ⟨/package⟩
```

Chapter 37

STEX
-Metatheory Implementation

```
6939  ⟨∗package⟩
6940  ⟨@@=stex_modules⟩
6941
6942  %%%%%%%%%%%%  metatheory.dtx  %%%%%%%%%%%%
6943
6944  \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX/meta}
6945  \begingroup
6946  \stex_module_setup:nn{
6947    ns=\c_stex_metatheory_ns_str,
6948    meta=NONE
6949  }{Metatheory}
6950  \stex_reactivate_macro:N \symdecl
6951  \stex_reactivate_macro:N \notation
6952  \stex_reactivate_macro:N \symdef
6953  \ExplSyntaxOff
6954  \csname stex_suppress_html:n\endcsname{
6955    % is-a (a:A, a \in A, a is an A, etc.)
6956    \symdecl{isa}[args=ai]
6957    \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
6958    \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
6959    \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
6960
6961    % bind (\forall, \Pi, \lambda etc.)
6962    \symdecl{bind}[args=Bi,assoc=pre]
6963    \notation{bind}[depfun,prec=nobrackets,op={(\cdot)\;\to\;\cdot}]{\comp( #1 \comp{)\;\to\;}
6964    \notation{bind}[forall]{\comp\forall #1.\;#2}{##1 \comp, ##2}
6965    \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6966
6967    % implicit bind
6968    \symdecl{implicitbind}[args=Bi,assoc=pre]
6969    \notation{implicitbind}[braces,prec=nobrackets,op={\{\cdot\}_I\;\cdot}]{\comp\{ #1 \comp{\
6970    \notation{implicitbind}[depfun,prec=nobrackets]{\comp( #1 \comp{)\;\to_I\;} #2}{##1 \comp,
6971    \notation{implicitbind}[Pi]{\comp\prod^I_{#1}#2}{##1\comp,##2}
6972
6973    % dummy variable
```

```
6974    \symdecl{dummyvar}
6975    \notation{dummyvar}[underscore]{\comp\_}
6976    \notation{dummyvar}[dot]{\comp\cdot}
6977    \notation{dummyvar}[dash]{\comp{{\rm --}}}}
6978
6979    %fromto (function space, Hom-set, implication etc.)
6980    \symdecl{fromto}[args=ai]
6981    \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
6982    \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
6983
6984    % mapto (lambda etc.)
6985    %\symdecl{mapto}[args=Bi]
6986    %\notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
6987    %\notation{mapto}[lambda]{\comp\lambda #1 \comp.\; #2}{#1 \comp, #2}
6988    %\notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.\; #2}{#1 \comp, #2}
6989
6990    % function/operator application
6991    \symdecl{apply}[args=ia]
6992    \notation{apply}[prec=0;0x\infprec,parens,op=\cdot(\cdot)]{#1 \comp( #2 \comp)}{##1 \comp,
6993    \notation{apply}[prec=0;0x\infprec,lambda]{#1 \; #2 }{##1 \; ##2}
6994
6995    % collection of propositions/booleans/truth values
6996    \symdecl{prop}[name=proposition]
6997    \notation{prop}[prop]{\comp{{\rm prop}}}
6998    \notation{prop}[BOOL]{\comp{{\rm BOOL}}}
6999
7000    \symdecl{judgmentholds}[args=1]
7001    \notation{judgmentholds}[vdash,op=\vdash]{\comp\vdash\; #1}
7002
7003    % sequences
7004    \symdecl{seqtype}[args=1]
7005    \notation{seqtype}[kleene]{#1^{\comp\ast}}
7006
7007    \symdecl{seqexpr}[args=a]
7008    \notation{seqexpr}[angle,prec=nobrackets]{\comp\langle #1\comp\rangle}{##1\comp,##2}
7009
7010    \symdef{seqmap}[args=abi,setlike]{\comp\{#3 \comp| #2\comp\in \dobrackets{#1} \comp\}}{##1
7011    \symdef{seqprepend}[args=ia]{#1 \comp{::} #2}{##1 \comp, ##2}
7012    \symdef{seqappend}[args=ai]{#1 \comp{::} #2}{##1 \comp, ##2}
7013    \symdef{seqfoldleft}[args=iabbi]{ \comp{foldl}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\com
7014    \symdef{seqfoldright}[args=iabbi,op=foldr]{ \comp{foldr}\dobrackets{#1,#2}\dobrackets{#3\c
7015    \symdef{seqhead}[args=a]{\comp{head}\dobrackets{#1}}{##1 \comp, ##2}
7016    \symdef{seqtail}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
7017    \symdef{seqlast}[args=a]{\comp{last}\dobrackets{#1}}{##1 \comp, ##2}
7018    \symdef{seqinit}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
7019
7020    \symdef{sequence-index}[args=2,li,prec=nobrackets]{{#1}_{#2}}
7021    \notation{sequence-index}[ui,prec=nobrackets]{{#1}^{#2}}
7022
7023    \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{,\ellipses}}{##1\comp,##2}
7024    \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{,\ellipses,}#2}{##1\comp,##2}
7025    \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{,\ellipses,}#2\comp{,\ellipses,}#
7026
7027    % nat literals
```

```
7028    \symdef{natliteral}{\comp{\mathtt{Ord}}}

7029

7030    % letin (``let'', local definitions, variable substitution)
7031    \symdecl{letin}[args=bii]
7032    \notation{letin}[let]{\comp{{\rm let}}\;#1\comp{=}#2\;\comp{{\rm in}}\;#3}
7033    \notation{letin}[subst]{#3 \comp[ #1 \comp/ #2 \comp]}
7034    \notation{letin}[frac]{#3 \comp[ \frac{#2}{#1} \comp]}

7035

7036    % structures
7037    \symdecl*{module-type}[args=1]
7038    \notation{module-type}{\comp{\mathtt{MOD}} #1}
7039    \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
7040    \notation{mathstruct}[angle,prec=nobrackets]{\comp\langle #1 \comp\rangle}{##1 \comp, ##2}

7041

7042    % objects
7043    \symdecl{object}
7044    \notation{object}{\comp{\mathtt{OBJECT}}}

7045

7046 }

7047

7048 % The following are abbreviations in the sTeX corpus that are left over from earlier
7049 % developments. They will eventually be phased out.

7050

7051    \ExplSyntaxOn
7052    \stex_add_to_current_module:n{
7053      \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
7054      \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
7055      \def\livar{\csname sequence-index\endcsname[li]}
7056      \def\uivar{\csname sequence-index\endcsname[ui]}
7057      \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
7058      \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
7059    }
7060 \__stex_modules_end_module:
7061 \endgroup

7062 ⟨/package⟩
```

Chapter 38

Tikzinput Implementation

```
7063  ⟨@@=tikzinput⟩
7064  ⟨*package⟩
7065
7066  %%%%%%%%%%%%   tikzinput.dtx   %%%%%%%%%%%%
7067
7068  \ProvidesExplPackage{tikzinput}{2022/08/08}{3.2.0}{tikzinput package}
7069  \RequirePackage{l3keys2e}
7070
7071  \keys_define:nn { tikzinput } {
7072    image    .bool_set:N   = \c_tikzinput_image_bool,
7073    image    .default:n    = false ,
7074    unknown    .code:n       = {}
7075  }
7076
7077  \ProcessKeysOptions { tikzinput }
7078
7079  \bool_if:NTF \c_tikzinput_image_bool {
7080    \RequirePackage{graphicx}
7081
7082    \providecommand\usetikzlibrary[]{}
7083    \newcommand\tikzinput[2][]{\includegraphics[#1]{#2}}
7084  }{
7085    \RequirePackage{tikz}
7086    \RequirePackage{standalone}
7087
7088    \newcommand \tikzinput [2] [] {
7089      \setkeys{Gin}{#1}
7090      \ifx \Gin@ewidth \Gin@exclamation
7091        \ifx \Gin@eheight \Gin@exclamation
7092          \input { #2 }
7093        \else
7094          \resizebox{!}{ \Gin@eheight }{
7095            \input { #2 }
7096          }
7097        \fi
7098      \else
7099        \ifx \Gin@eheight \Gin@exclamation
7100          \resizebox{ \Gin@ewidth }{!}{
```

```
7101        \input { #2 }
7102      }
7103    \else
7104      \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
7105        \input { #2 }
7106      }
7107    \fi
7108  \fi
7109  }
7110 }

7111
7112 \newcommand \ctikzinput [2] [] {
7113   \begin{center}
7114     \tikzinput [#1] {#2}
7115   \end{center}
7116 }

7117
7118 \@ifpackageloaded{stex}{
7119   \RequirePackage{stex-tikzinput}
7120 }{}

7121
7122 ⟨/package⟩
7123 ⟨∗stex⟩

7124 \ProvidesExplPackage{stex-tikzinput}{2022/08/08}{3.2.0}{stex-tikzinput}
7125 \RequirePackage{stex}
7126 \RequirePackage{tikzinput}

7127
7128 \newcommand\mhtikzinput[2][]{%
7129   \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
7130   \stex_in_repository:nn\Gin@mhrepos{
7131     \tikzinput[#1]{\mhpath{##1}{#2}}
7132   }
7133 }
7134 \newcommand\cmhtikzinput[2][]{\begin{center}\mhtikzinput[#1]{#2}\end{center}}

7135
7136 \cs_new_protected:Nn \__tikzinput_usetikzlibrary:nn {
7137   \pgfkeys@spdef\pgf@temp{#1}
7138   \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
7139   \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgfut
7140   \expandafter\edef\csname tikz@library@#1atcode\endcsname{\the\catcode`\@}
7141   \expandafter\edef\csname tikz@library@#1barcode\endcsname{\the\catcode`\|}
7142   \expandafter\edef\csname tikz@library@#1dollarcode\endcsname{\the\catcode`\$}
7143   \catcode`\@=11
7144   \catcode`\|=12
7145   \catcode`\$=3
7146   \pgfutil@InputIfFileExists{#2}{}{}
7147   \catcode`\@=\csname tikz@library@#1atcode\endcsname
7148   \catcode`\|=\csname tikz@library@#1barcode\endcsname
7149   \catcode`\$=\csname tikz@library@#1dollarcode\endcsname
7150 }

7151

7152
7153 \newcommand\libusetikzlibrary[1]{
```

```
7154    \prop_if_exist:NF \l_stex_current_repository_prop {
7155      \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
7156    }
7157    \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
7158      \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
7159    }
7160    \seq_clear:N \l__tikzinput_libinput_files_seq
7161    \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
7162    \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
7163
7164    \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
7165      \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / tikzlibra
7166      \IfFileExists{ \l_tmpa_str }{
7167        \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
7168      }{}
7169      \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
7170      \seq_put_right:No \l_tmpa_seq \l_tmpa_str
7171    }
7172
7173    \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / tikzlibrary #1 .code.t
7174    \IfFileExists{ \l_tmpa_str }{
7175      \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
7176    }{}
7177
7178    \seq_if_empty:NTF \l__tikzinput_libinput_files_seq {
7179      \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .code.t
7180    }{
7181      \int_compare:nNnTF {\seq_count:N \l__tikzinput_libinput_files_seq} = 1 {
7182        \seq_map_inline:Nn \l__tikzinput_libinput_files_seq {
7183          \__tikzinput_usetikzlibrary:nn{#1}{ ##1 }
7184        }
7185      }{
7186        \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .co
7187      }
7188    }
7189 }
7190 ⟨/stex⟩
```

263

Chapter 39

document-structure.sty Implementation

₇₁₉₁ ⟨∗package⟩
₇₁₉₂ ⟨@@=document_structure⟩
₇₁₉₃ \ProvidesExplPackage{document-structure}{2022/08/08}{3.2.0}{Modular Document Structure}
₇₁₉₄ \RequirePackage{l3keys2e}

39.1   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).

```
7195
7196 \keys_define:nn{ document-structure }{
7197   class      .str_set_x:N  = \c_document_structure_class_str,
7198   topsect    .str_set_x:N  = \c_document_structure_topsect_str,
7199   unknown    .code:n       = {
7200     \PassOptionsToClass{\CurrentOption}{stex}
7201     \PassOptionsToClass{\CurrentOption}{tikzinput}
7202   }
7203 %  showignores .bool_set:N  = \c_document_structure_showignores_bool,
7204 }
7205 \ProcessKeysOptions{ document-structure }
7206 \str_if_empty:NT \c_document_structure_class_str {
7207   \str_set:Nn \c_document_structure_class_str {article}
7208 }
7209 \str_if_empty:NT \c_document_structure_topsect_str {
7210   \str_set:Nn \c_document_structure_topsect_str {section}
7211 }
```

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

```
7212 \RequirePackage{xspace}
7213 \RequirePackage{comment}
7214 \RequirePackage{stex}
7215 \AddToHook{begindocument}{
```

```
7216   \ltx@ifpackageloaded{babel}{
7217       \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7218       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
7219           \makeatletter\input{document-structure-ngerman.ldf}\makeatother
7220       }
7221   }{}
7222   }
```

\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.

```
7223   \int_new:N \l_document_structure_section_level_int
7224   \str_case:VnF \c_document_structure_topsect_str {
7225     {part}{
7226         \int_set:Nn \l_document_structure_section_level_int {0}
7227     }
7228     {chapter}{
7229         \int_set:Nn \l_document_structure_section_level_int {1}
7230     }
7231   }{
7232     \str_case:VnF \c_document_structure_class_str {
7233       {book}{
7234           \int_set:Nn \l_document_structure_section_level_int {0}
7235       }
7236       {report}{
7237           \int_set:Nn \l_document_structure_section_level_int {0}
7238       }
7239     }{
7240         \int_set:Nn \l_document_structure_section_level_int {2}
7241     }
7242   }
```

39.2   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 ini-
tialize 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]

```
7243   \def\current@section@level{document}%
7244   \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
7245   \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

\skipfragment

```
7246   \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.

```
7247    \ifcase\l_document_structure_section_level_int
7248      \or\stepcounter{part}
7249      \or\stepcounter{chapter}
7250      \or\stepcounter{section}
7251      \or\stepcounter{subsection}
7252      \or\stepcounter{subsubsection}
7253      \or\stepcounter{paragraph}
7254      \or\stepcounter{subparagraph}
7255    \fi
7256 }
```

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

blindfragment (*env.*)

```
7257 \newcommand\at@begin@blindsfragment[1]{}
7258 \newenvironment{blindfragment}
7259 {
7260    \int_incr:N\l_document_structure_section_level_int
7261    \at@begin@blindsfragment\l_document_structure_section_level_int
7262 }{}
```

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

```
7263 \newcommand\sfragment@nonum[2]{
7264    \ifx\hyper@anchor\@undefined\else\phantomsection\fi
7265    \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
7266 }
```

(*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.

```
7267 \newcommand\sfragment@num[2]{
7268    \tl_if_empty:NTF \l__document_structure_sfragment_short_tl {
7269      \@nameuse{#1}{#2}
7270    }{
7271      \cs_if_exist:NTF\rdfmeta@sectioning{
7272        \@nameuse{rdfmeta@#1@old}[\l__document_structure_sfragment_short_tl]{#2}
7273      }{
7274        \@nameuse{#1}[\l__document_structure_sfragment_short_tl]{#2}
7275      }
7276    }
7277 %\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\sfragment@id
7278 }
```

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

sfragment (*env.*)

```
7279 \keys_define:nn { document-structure / sfragment }{
7280    id            .str_set_x:N = \l__document_structure_sfragment_id_str,
7281    date          .str_set_x:N = \l__document_structure_sfragment_date_str,
```

266

```
7282    creators      .clist_set:N = \l__document_structure_sfragment_creators_clist,
7283    contributors  .clist_set:N = \l__document_structure_sfragment_contributors_clist,
7284    srccite       .tl_set:N    = \l__document_structure_sfragment_srccite_tl,
7285    type          .tl_set:N    = \l__document_structure_sfragment_type_tl,
7286    short         .tl_set:N    = \l__document_structure_sfragment_short_tl,
7287    intro         .tl_set:N    = \l__document_structure_sfragment_intro_tl,
7288    imports       .tl_set:N    = \l__document_structure_sfragment_imports_tl,
7289    loadmodules   .bool_set:N  = \l__document_structure_sfragment_loadmodules_bool
7290 }
7291 \cs_new_protected:Nn \__document_structure_sfragment_args:n {
7292    \str_clear:N \l__document_structure_sfragment_id_str
7293    \str_clear:N \l__document_structure_sfragment_date_str
7294    \clist_clear:N \l__document_structure_sfragment_creators_clist
7295    \clist_clear:N \l__document_structure_sfragment_contributors_clist
7296    \tl_clear:N \l__document_structure_sfragment_srccite_tl
7297    \tl_clear:N \l__document_structure_sfragment_type_tl
7298    \tl_clear:N \l__document_structure_sfragment_short_tl
7299    \tl_clear:N \l__document_structure_sfragment_imports_tl
7300    \tl_clear:N \l__document_structure_sfragment_intro_tl
7301    \bool_set_false:N \l__document_structure_sfragment_loadmodules_bool
7302    \keys_set:nn { document-structure / sfragment } { #1 }
7303 }
```

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.

```
7304 \newif\if@mainmatter\@mainmattertrue
7305 \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.

```
7306 \keys_define:nn { document-structure / sectioning }{
7307    name    .str_set_x:N = \l__document_structure_sect_name_str   ,
7308    ref     .str_set_x:N = \l__document_structure_sect_ref_str    ,
7309    clear   .bool_set:N  = \l__document_structure_sect_clear_bool ,
7310    clear   .default:n   = {true}                                 ,
7311    num     .bool_set:N  = \l__document_structure_sect_num_bool   ,
7312    num     .default:n   = {true}
7313 }
7314 \cs_new_protected:Nn \__document_structure_sect_args:n {
7315    \str_clear:N \l__document_structure_sect_name_str
7316    \str_clear:N \l__document_structure_sect_ref_str
7317    \bool_set_false:N \l__document_structure_sect_clear_bool
7318    \bool_set_false:N \l__document_structure_sect_num_bool
7319    \keys_set:nn { document-structure / sectioning } { #1 }
7320 }
7321 \newcommand\omdoc@sectioning[3][]{
7322    \__document_structure_sect_args:n {#1 }
7323    \let\omdoc@sect@name\l__document_structure_sect_name_str
7324    \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
7325    \if@mainmatter% numbering not overridden by frontmatter, etc.
7326      \bool_if:NTF \l__document_structure_sect_num_bool {
7327        \sfragment@num{#2}{#3}
7328      }{
```

267

```
7329        \sfragment@nonum{#2}{#3}
7330      }
7331      \def\current@section@level{\omdoc@sect@name}
7332    \else
7333      \sfragment@nonum{#2}{#3}
7334    \fi
7335 }% if@mainmatter
```

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

```
7336 \newcommand\sfragment@redefine@addtocontents[1]{%
7337 %\edef\__document_structureimport{#1}%
7338 %\@for\@I:=\__document_structureimport\do{%
7339 %\edef\@path{\csname module@\@I  @path\endcsname}%
7340 %\@ifundefined{tf@toc}\relax%
7341 %      {\protected@write\tf@toc{}{\string\@requiremodules{\@path}}}}
7342 %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
7343 %\def\addcontentsline##1##2##3{%
7344 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}
7345 %\else% hyperref.sty not loaded
7346 %\def\addcontentsline##1##2##3{%
7347 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}{
7348 %\fi
7349 }% 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.

```
7350 \newenvironment{sfragment}[2][]% keys, title
7351 {
7352    \__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.

```
7353    \stex_csl_to_imports:No \usemodule \l__document_structure_sfragment_imports_tl
7354
7355    \bool_if:NT \l__document_structure_sfragment_loadmodules_bool {
7356      \sfragment@redefine@addtocontents{
7357        %\@ifundefined{module@id}\used@modules%
7358        %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}
7359      }
7360    }
```

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

```
7361
7362    \stex_document_title:n { #2 }
7363
7364    \int_incr:N\l_document_structure_section_level_int
7365    \ifcase\l_document_structure_section_level_int
7366      \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
7367      \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
7368      \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
7369      \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
```

```
7370        \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
7371        \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#
7372        \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{paragr
7373      \fi
7374      \at@begin@sfragment[#1]\l_document_structure_section_level_int{#2}
7375      \str_if_empty:NF \l__document_structure_sfragment_id_str {
7376        \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
7377      }
7378  }% for customization
7379  {}
```

and finally, we localize the sections

```
7380  \newcommand\omdoc@part@kw{Part}
7381  \newcommand\omdoc@chapter@kw{Chapter}
7382  \newcommand\omdoc@section@kw{Section}
7383  \newcommand\omdoc@subsection@kw{Subsection}
7384  \newcommand\omdoc@subsubsection@kw{Subsubsection}
7385  \newcommand\omdoc@paragraph@kw{paragraph}
7386  \newcommand\omdoc@subparagraph@kw{subparagraph}
```

39.3   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

```
7387  \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).

```
7388  \cs_if_exist:NTF\frontmatter{
7389    \let\__document_structure_orig_frontmatter\frontmatter
7390    \let\frontmatter\relax
7391  }{
7392    \tl_set:Nn\__document_structure_orig_frontmatter{
7393      \clearpage
7394      \@mainmatterfalse
7395      \pagenumbering{roman}
7396    }
7397  }
7398  \cs_if_exist:NTF\backmatter{
7399    \let\__document_structure_orig_backmatter\backmatter
7400    \let\backmatter\relax
7401  }{
7402    \tl_set:Nn\__document_structure_orig_backmatter{
7403      \clearpage
7404      \@mainmatterfalse
7405      \pagenumbering{roman}
7406    }
```

```
7407 }
```

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.

```
7408 \newenvironment{frontmatter}{
7409   \__document_structure_orig_frontmatter
7410 }{
7411   \cs_if_exist:NTF\mainmatter{
7412     \mainmatter
7413   }{
7414     \clearpage
7415     \@mainmattertrue
7416     \pagenumbering{arabic}
7417   }
7418 }
```

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

```
7419 \newenvironment{backmatter}{
7420   \__document_structure_orig_backmatter
7421 }{
7422   \cs_if_exist:NTF\mainmatter{
7423     \mainmatter
7424   }{
7425     \clearpage
7426     \@mainmattertrue
7427     \pagenumbering{arabic}
7428   }
7429 }
```

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

```
7430 \@mainmattertrue\pagenumbering{arabic}
```

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

```
7431 \def \c__document_structure_document_str{document}
7432 \newcommand\afterprematurestop{}
7433 \def\prematurestop@endsfragment{
7434   \unless\ifx\@currenvir\c__document_structure_document_str
7435     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafte
7436     \expandafter\prematurestop@endsfragment
7437   \fi
7438 }
7439 \providecommand\prematurestop{
7440   \message{Stopping~sTeX~processing~prematurely}
7441   \prematurestop@endsfragment
7442   \afterprematurestop
7443   \end{document}
7444 }
```

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

39.4 Global Variables

\setSGvar set a global variable

```
7445 \RequirePackage{etoolbox}
7446 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}
```

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

\useSGvar use a global variable

```
7447 \newrobustcmd\useSGvar[1]{%
7448   \@ifundefined{sTeX@Gvar@#1}
7449   {\PackageError{document-structure}
7450     {The sTeX Global variable #1 is undefined}
7451     {set it with \protect\setSGvar}}
7452 \@nameuse{sTeX@Gvar@#1}}
```

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

\ifSGvar execute something conditionally based on the state of the global variable.

```
7453 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
7454   \@ifundefined{sTeX@Gvar@#1}
7455   {\PackageError{document-structure}
7456     {The sTeX Global variable #1 is undefined}
7457     {set it with \protect\setSGvar}}
7458   {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}
```

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

Chapter 40

NotesSlides – Implementation

40.1 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.

```
7459  ⟨∗cls⟩
7460  ⟨@@=notesslides⟩
7461  \ProvidesExplClass{notesslides}{2022/08/08}{3.2.0}{notesslides Class}
7462  \RequirePackage{l3keys2e}
7463
7464  \keys_define:nn{notesslides / cls}{
7465    class   .str_set_x:N = \c__notesslides_class_str,
7466    notes   .bool_set:N  = \c__notesslides_notes_bool ,
7467    slides  .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
7468    docopt  .str_set_x:N = \c__notesslides_docopt_str,
7469    unknown .code:n      = {
7470      \PassOptionsToPackage{\CurrentOption}{document-structure}
7471      \PassOptionsToClass{\CurrentOption}{beamer}
7472      \PassOptionsToPackage{\CurrentOption}{notesslides}
7473      \PassOptionsToPackage{\CurrentOption}{stex}
7474    }
7475  }
7476  \ProcessKeysOptions{ notesslides / cls }
7477
7478  \str_if_empty:NF \c__notesslides_class_str {
7479    \PassOptionsToPackage{class=\c__notesslides_class_str}{document-structure}
7480  }
7481
7482  \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{book}{
7483    \PassOptionsToPackage{defaulttopsect=part}{notesslides}
7484  }
7485  \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{report}{
7486    \PassOptionsToPackage{defaulttopsect=part}{notesslides}
7487  }
7488
7489  \RequirePackage{stex}
```

```
7490  \stex_html_backend:T {
7491    \bool_set_true:N\c__notesslides_notes_bool
7492  }
7493
7494  \bool_if:NTF \c__notesslides_notes_bool {
7495    \PassOptionsToPackage{notes=true}{notesslides}
7496    \message{notesslides.cls:~Formatting~course~materials~in~notes~mode}
7497  }{
7498    \PassOptionsToPackage{notes=false}{notesslides}
7499    \message{notesslides.cls:~Formatting~course~materials~in~slides~mode}
7500  }
7501  ⟨/cls⟩
```

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

```
7502  ⟨*package⟩
7503  \ProvidesExplPackage{notesslides}{2022/08/08}{3.2.0}{notesslides Package}
7504  \RequirePackage{l3keys2e}
7505
7506  \keys_define:nn{notesslides / pkg}{
7507    topsect         .str_set_x:N  = \c__notesslides_topsect_str,
7508    defaulttopsect  .str_set_x:N  = \c__notesslides_defaulttopsec_str,
7509    notes           .bool_set:N   = \c__notesslides_notes_bool ,
7510    slides          .code:n       = { \bool_set_false:N \c__notesslides_notes_bool },
7511    sectocframes    .bool_set:N   = \c__notesslides_sectocframes_bool ,
7512    frameimages     .bool_set:N   = \c__notesslides_frameimages_bool ,
7513    fiboxed         .bool_set:N   = \c__notesslides_fiboxed_bool ,
7514    noproblems      .bool_set:N   = \c__notesslides_noproblems_bool,
7515    unknown         .code:n       = {
7516      \PassOptionsToClass{\CurrentOption}{stex}
7517      \PassOptionsToClass{\CurrentOption}{tikzinput}
7518    }
7519  }
7520  \ProcessKeysOptions{ notesslides / pkg }
7521
7522  \RequirePackage{stex}
7523  \stex_html_backend:T {
7524    \bool_set_true:N\c__notesslides_notes_bool
7525  }
7526
7527  \newif\ifnotes
7528  \bool_if:NTF \c__notesslides_notes_bool {
7529    \notestrue
7530  }{
7531    \notesfalse
7532  }
7533
```

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

```
7534  \str_if_empty:NTF \c__notesslides_topsect_str {
7535    \str_set_eq:NN \__notesslidestopsect \c__notesslides_defaulttopsec_str
7536  }{
7537    \str_set_eq:NN \__notesslidestopsect \c__notesslides_topsect_str
7538  }
7539  \PassOptionsToPackage{topsect=\__notesslidestopsect}{document-structure}
```

7540 ⟨/package⟩

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

7541 ⟨*cls⟩
7542 `\bool_if:NTF \c__notesslides_notes_bool {`
7543 `  \str_if_empty:NT \c__notesslides_class_str {`
7544 `    \str_set:Nn \c__notesslides_class_str {article}`
7545 `  }`
7546 `  \exp_after:wN\LoadClass\exp_after:wN[\c__notesslides_docopt_str]`
7547 `    {\c__notesslides_class_str}`
7548 `}{`
7549 `  \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}`
7550 `  \newcounter{Item}`
7551 `  \newcounter{paragraph}`
7552 `  \newcounter{subparagraph}`
7553 `  \newcounter{Hfootnote}`
7554 `}`
7555 `\RequirePackage{document-structure}`

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

7556 `\RequirePackage{notesslides}`
7557 ⟨/cls⟩

In `notes` mode, we also have to make the `beamer`-specific things available to `article` via the `beamerarticle` package. We use options to avoid loading theorem-like environments, since we want to use our own from the SₜEX packages. The first batch of packages we want are loaded on `notesslides.sty`. These are the general ones, we will load the SₜEX-specific ones after we have done some work (e.g. defined the counters `m*`). Only the `stex-logo` package is already needed now for the default theme.

7558 ⟨*package⟩
7559 `\bool_if:NT \c__notesslides_notes_bool {`
7560 `  \RequirePackage{a4wide}`
7561 `  \RequirePackage{marginnote}`
7562 `  \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}`
7563 `  \RequirePackage{mdframed}`
7564 `  \RequirePackage[noxcolor,noamsthm]{beamerarticle}`
7565 `  \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}`
7566 `}`
7567 `\RequirePackage{stex-tikzinput}`
7568 `\RequirePackage{comment}`
7569 `\RequirePackage{url}`
7570 `\RequirePackage{graphicx}`
7571 `\RequirePackage{pgf}`
7572 `\RequirePackage{bookmark}`

40.2 Notes and Slides

For the lecture notes cases, we also provide the `\usetheme` macro that would otherwise come from the the `beamer` class.

7573 `\bool_if:NT \c__notesslides_notes_bool {`
7574 `  \renewcommand\usetheme[2][]{\usepackage[#1]{beamertheme#2}}`
7575 `}`

```
7576  \NewDocumentCommand \libusetheme {O{} m} {
7577    \libusepackage[#1]{beamertheme#2}
7578  }
7579
```

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

```
7580  \newcounter{slide}
7581  \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
7582  \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.

```
7583  \bool_if:NTF \c__notesslides_notes_bool {
7584    \renewenvironment{note}{\ignorespaces}{}
7585  }{
7586    \excludecomment{note}
7587  }
```

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.

```
7588  \bool_if:NT \c__notesslides_notes_bool {
7589    \newlength{\slideframewidth}
7590    \setlength{\slideframewidth}{1.5pt}
```

frame (*env.*) We first define the keys.

```
7591  \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
7592    \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
7593      \bool_set_true:N #1
7594    }{
7595      \bool_set_false:N #1
7596    }
7597  }
7598  \keys_define:nn{notesslides / frame}{
7599    label                .str_set_x:N  = \l__notesslides_frame_label_str,
7600    allowframebreaks    .code:n       = {
7601      \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
7602    },
7603    allowdisplaybreaks  .code:n       = {
7604      \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
7605    },
7606    fragile             .code:n       = {
7607      \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
7608    },
7609    shrink              .code:n       = {
7610      \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
7611    },
7612    squeeze             .code:n       = {
7613      \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
7614    },
7615    t                   .code:n       = {
```

```
7616        \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
7617      },
7618      unknown   .code:n      = {}
7619    }
7620    \cs_new_protected:Nn \__notesslides_frame_args:n {
7621      \str_clear:N \l__notesslides_frame_label_str
7622      \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
7623      \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
7624      \bool_set_true:N \l__notesslides_frame_fragile_bool
7625      \bool_set_true:N \l__notesslides_frame_shrink_bool
7626      \bool_set_true:N \l__notesslides_frame_squeeze_bool
7627      \bool_set_true:N \l__notesslides_frame_t_bool
7628      \keys_set:nn { notesslides / frame }{ #1 }
7629    }
```

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

```
7630    \renewenvironment{frame}[1][]{
7631      \__notesslides_frame_args:n{#1}
7632      \sffamily
7633      \stepcounter{slide}
7634      \def\@currentlabel{\theslide}
7635      \str_if_empty:NF \l__notesslides_frame_label_str {
7636        \label{\l__notesslides_frame_label_str}
7637      }
```

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

```
7638      \def\itemize@level{outer}
7639      \def\itemize@outer{outer}
7640      \def\itemize@inner{inner}
7641      \renewcommand\newpage{\addtocounter{framenumber}{1}}
7642      %\newcommand\metakeys@show@keys[2]{\marginnote{{\scriptsize ##2}}}
7643      \renewenvironment{itemize}{
7644        \ifx\itemize@level\itemize@outer
7645          \def\itemize@label{$\rhd$}
7646        \fi
7647        \ifx\itemize@level\itemize@inner
7648          \def\itemize@label{$\scriptstyle\rhd$}
7649        \fi
7650        \begin{list}
7651        {\itemize@label}
7652        {\setlength{\labelsep}{.3em}
7653         \setlength{\labelwidth}{.5em}
7654         \setlength{\leftmargin}{1.5em}
7655        }
7656        \edef\itemize@level{\itemize@inner}
7657      }{
7658        \end{list}
7659      }
```

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

```
7660      \stex_html_backend:TF {
7661        \begin{stex_annotate_env}{frame}{}\vbox\bgroup
7662          \mdf@patchamsthm
7663      }{
7664        \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwid
```

276

```
7665        }
7666    }{
7667      \stex_html_backend:TF {
7668        \miko@slidelabel\egroup\end{stex_annotate_env}
7669      }{\medskip\miko@slidelabel\end{mdframed}}
7670    }
```

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

\frametitle

```
7671    \renewcommand{\frametitle}[1]{
7672      \stex_document_title:n { #1 }
7673      {\Large\bf\sf\color{blue}{#1}}\medskip
7674    }
7675 }
```

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

EdN:10 \pause ¹⁰

```
7676 \bool_if:NT \c__notesslides_notes_bool {
7677    \newcommand\pause{}
7678 }
```

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

nparagraph (*env.*)

```
7679 \bool_if:NTF \c__notesslides_notes_bool {
7680    \newenvironment{nparagraph}[1][]{\begin{sparagraph}[#1]}{\end{sparagraph}}
7681 }{
7682    \excludecomment{nparagraph}
7683 }
```

nfragment (*env.*)

```
7684 \bool_if:NTF \c__notesslides_notes_bool {
7685    \newenvironment{nfragment}[2][]{\begin{sfragment}[#1]{#2}}{\end{sfragment}}
7686 }{
7687    \excludecomment{nfragment}
7688 }
```

ndefinition (*env.*)

```
7689 \bool_if:NTF \c__notesslides_notes_bool {
7690    \newenvironment{ndefinition}[1][]{\begin{sdefinition}[#1]}{\end{sdefinition}}
7691 }{
7692    \excludecomment{ndefinition}
7693 }
```

nassertion (*env.*)

```
7694 \bool_if:NTF \c__notesslides_notes_bool {
7695    \newenvironment{nassertion}[1][]{\begin{sassertion}[#1]}{\end{sassertion}}
7696 }{
7697    \excludecomment{nassertion}
7698 }
```

---

¹⁰EDNOTE: MK: fake it in notes mode for now

```
7699  \bool_if:NTF \c__notesslides_notes_bool {
7700    \newenvironment{nproof}[2][]{\begin{sproof}[#1]{#2}}{\end{sproof}}
7701  }{
7702    \excludecomment{nproof}
7703  }
```

```
7704  \bool_if:NTF \c__notesslides_notes_bool {
7705    \newenvironment{nexample}[1][]{\begin{sexample}[#1]}{\end{sexample}}
7706  }{
7707    \excludecomment{nexample}
7708  }
```

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

```
7709  \def\inputref@preskip{\smallskip}
7710  \def\inputref@postskip{\medskip}
```

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

\inputref*

```
7711  \let\orig@inputref\inputref
7712  \def\inputref{\@ifstar\ninputref\orig@inputref}
7713  \newcommand\ninputref[2][]{
7714    \bool_if:NT \c__notesslides_notes_bool {
7715      \orig@inputref[#1]{#2}
7716    }
7717  }
```

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

40.3   Header and Footer Lines

Now, we set up the infrastructure for the footer line of the slides, we use boxes for the logos, so that they are only loaded once, that considerably speeds up processing.

\setslidelogo   The default logo is the STEX logo. Customization can be done by `\setslidelogo{⟨logo name⟩}`.

```
7718  \newlength{\slidelogoheight}
7719
7720  \RequirePackage{graphicx}
7721
7722  \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7723  \providecommand\mhgraphics[2][]{
7724    \def\Gin@mhrepos{}\setkeys{Gin}{#1}
7725    \includegraphics[#1]{\mhpath\Gin@mhrepos{#2}}
7726  }
7727
7728  \bool_if:NTF \c__notesslides_notes_bool {
7729    \setlength{\slidelogoheight}{.4cm}
7730  }{
7731    \setlength{\slidelogoheight}{.25cm}
7732  }
```

278

```
7733  \ifcsname slidelogo\endcsname\else
7734    \newsavebox{\slidelogo}
7735    \sbox{\slidelogo}{\sTeX}
7736  \fi
7737  \newrobustcmd{\setslidelogo}[2][]{
7738    \tl_if_empty:nTF{#1}{
7739      \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#2}}
7740    }{
7741      \sbox{\slidelogo}{\mhgraphics[height=\slidelogoheight,mhrepos=#1]{#2}}
7742    }
7743  }
```

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

\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.

```
7744  \bool_if:NT \c__notesslides_notes_bool {
7745    \def\author{\@dblarg\ns@author}
7746    \long\def\ns@author[#1]#2{%
7747      \def\c__notesslides_shortauthor{#1}%
7748      \def\@author{#2}
7749    }
7750  }
```

(*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.

```
7751  \newrobustcmd{\setsource}[1]{\def\source{#1}}
```

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

\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.

```
7752  \def\copyrightnotice{%
7753    \footnotesize\copyright :\hspace{.3ex}%
7754    \ifcsname source\endcsname\source\else%
7755    \ifcsname c__notesslides_shortauthor\endcsname\c__notesslides_shortauthor\else%
7756    \PackageWarning{notesslides}{Author/Source~undefined~in~copyright~notice}%
7757    ?source/author?\fi%
7758    \fi}
7759  \newsavebox{\cclogo}
7760  \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{stex-cc_somerights}}
7761  \newif\ifcchref\cchreffalse
7762  \AtBeginDocument{
7763    \@ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
7764  }
7765  \def\licensing{
7766    \ifcchref
7767      \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
7768    \else
7769      {\usebox{\cclogo}}
```

```
7770    \fi
7771  }
7772  \newrobustcmd{\setlicensing}[2][]{
7773    \def\@url{#1}
7774    \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
7775    \ifx\@url\@empty
7776      \def\licensing{{\usebox{\cclogo}}}
7777    \else
7778      \def\licensing{
7779        \ifcchref
7780        \href{#1}{\usebox{\cclogo}}
7781        \else
7782        {\usebox{\cclogo}}
7783        \fi
7784      }
7785    \fi
7786  }
```

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

EdN:11   \slidelabel   Now, we set up the slide label for the `article` mode.[11]

```
7787  \newrobustcmd\miko@slidelabel{
7788    \vbox to \slidelogoheight{
7789      \vss\hbox to \slidewidth
7790      {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
7791    }
7792  }
```

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

40.4   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.

```
7793  \def\Gin@mhrepos{}
7794  \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7795  \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
7796  \newrobustcmd\frameimage[2][]{
7797    \stepcounter{slide}
7798    \bool_if:NT \c__notesslides_frameimages_bool {
7799      \def\Gin@ewidth{}\setkeys{Gin}{#1}
7800      \bool_if:NF \c__notesslides_notes_bool { \vfill }
7801      \begin{center}
7802        \bool_if:NTF \c__notesslides_fiboxed_bool {
7803          \fbox{
7804            \ifx\Gin@ewidth\@empty
7805              \ifx\Gin@mhrepos\@empty
7806                \mhgraphics[width=\slidewidth,#1]{#2}
7807              \else
7808                \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7809              \fi
7810            \else% Gin@ewidth empty
```

---

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

```
7811          \ifx\Gin@mhrepos\@empty
7812            \mhgraphics[#1]{#2}
7813          \else
7814            \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7815          \fi
7816        \fi% Gin@ewidth empty
7817      }
7818    }{
7819      \ifx\Gin@ewidth\@empty
7820        \ifx\Gin@mhrepos\@empty
7821          \mhgraphics[width=\slidewidth,#1]{#2}
7822        \else
7823          \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7824        \fi
7825        \ifx\Gin@mhrepos\@empty
7826          \mhgraphics[#1]{#2}
7827        \else
7828          \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7829        \fi
7830      \fi% Gin@ewidth empty
7831    }
7832    \end{center}
7833    \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
7834    \bool_if:NF \c__notesslides_notes_bool { \vfill }
7835  }
7836 } % ifmks@sty@frameimages
```

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

40.5 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`.

```
7837 \stex_html_backend:F {
7838   \bool_if:NT \c__notesslides_sectocframes_bool {
7839     \str_if_eq:VnTF \__notesslidestopsect{part}{
7840       \newcounter{chapter}\counterwithin*{section}{chapter}
7841     }{
7842       \str_if_eq:VnT\__notesslidestopsect{chapter}{
7843         \newcounter{chapter}\counterwithin*{section}{chapter}
7844       }
7845     }
7846   }
7847 }
```

\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

```
7848 \def\part@prefix{}
7849 \@ifpackageloaded{document-structure}{}{
7850   \str_case:VnF \__notesslidestopsect {
```

```
7851     {part}{
7852       \int_set:Nn \l_document_structure_section_level_int {0}
7853       \def\thesection{\arabic{chapter}.\arabic{section}}
7854       \def\part@prefix{\arabic{chapter}.}
7855     }
7856     {chapter}{
7857       \int_set:Nn \l_document_structure_section_level_int {1}
7858       \def\thesection{\arabic{chapter}.\arabic{section}}
7859       \def\part@prefix{\arabic{chapter}.}
7860     }
7861   }{
7862     \int_set:Nn \l_document_structure_section_level_int {2}
7863     \def\part@prefix{}
7864   }
7865 }
7866
7867 \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.*)

```
7868   \renewenvironment{sfragment}[2][]{
7869     \__document_structure_sfragment_args:n { #1 }
7870     \int_incr:N \l_document_structure_section_level_int
7871     \bool_if:NT \c__notesslides_sectocframes_bool {
7872       \stepcounter{slide}
7873       \begin{frame}[noframenumbering]
7874       \vfill\Large\centering
7875       \red{
7876         \ifcase\l_document_structure_section_level_int\or
7877           \stepcounter{part}
7878           \def\__notesslideslabel{{\omdoc@part@kw}~\Roman{part}}
7879           \addcontentsline{toc}{part}{\protect\numberline{\thepart}#2}
7880           \pdfbookmark[0]{\thepart\ #2}{part.\thepart}
7881           \def\currentsectionlevel{\omdoc@part@kw}
7882         \or
7883           \stepcounter{chapter}
7884           \def\__notesslideslabel{{\omdoc@chapter@kw}~\arabic{chapter}}
7885           \addcontentsline{toc}{chapter}{\protect\numberline{\thechapter}#2}
7886           \pdfbookmark[1]{\thechapter\ #2}{chapter.\cs_if_exist:cT{thepart}\thepart.\thechap
7887           \def\currentsectionlevel{\omdoc@chapter@kw}
7888         \or
7889           \stepcounter{section}
7890           \def\__notesslideslabel{\part@prefix\arabic{section}}
7891           \addcontentsline{toc}{section}{\protect\numberline{\thesection}#2}
7892           \pdfbookmark[2]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection\ #2}
7893           {section.\cs_if_exist:cT{thepart}{\thepart.}\cs_if_exist:cT{thechapter}{\thechapte
7894           \def\currentsectionlevel{\omdoc@section@kw}
7895         \or
7896           \stepcounter{subsection}
7897           \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
7898           \addcontentsline{toc}{subsection}{\protect\numberline{\thesubsection}#2}
```

```
7899          \pdfbookmark[3]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsectio
7900          {subsection.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thecha
7901          \def\currentsectionlevel{\omdoc@subsection@kw}
7902        \or
7903          \stepcounter{subsubsection}
7904          \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
7905          \addcontentsline{toc}{subsubsection}{\protect\numberline{\thesubsubsection}#2}
7906          \pdfbookmark[4]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsectio
7907          {subsubsection.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\the
7908          \def\currentsectionlevel{\omdoc@subsubsection@kw}
7909        \or
7910          \stepcounter{paragraph}
7911          \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
7912          \addcontentsline{toc}{paragraph}{\protect\numberline{\theparagraph}#2}
7913          \pdfbookmark[5]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsectio
7914          {paragraph.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechap
7915          \def\currentsectionlevel{\omdoc@paragraph@kw}
7916        \else
7917          \def\__notesslideslabel{}
7918          \def\currentsectionlevel{\omdoc@paragraph@kw}
7919        \fi% end ifcase
7920        \__notesslideslabel\quad #2%
7921      }%
7922      \vfill%
7923      \end{frame}%
7924    }
7925    \str_if_empty:NF \l__document_structure_sfragment_id_str {
7926      \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
7927    }
7928  }{}
7929 }
```

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

```
7930 \def\inserttheorembodyfont{\normalfont}
7931 %\bool_if:NF \c__notesslides_notes_bool {
7932 %  \defbeamertemplate{theorem begin}{miko}
7933 %  {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
7934 %    \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
7935 %    \inserttheorempunctuation\inserttheorembodyfont\xspace}
7936 %  \defbeamertemplate{theorem end}{miko}{}
```

and we set it as the default one.

```
7937 %  \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.

```
7938 %  \expandafter\def\csname Parent2\endcsname{}
7939 %}
7940
7941 \AddToHook{begindocument}{ % this does not work for some reasone
7942   \setbeamertemplate{theorems}[ams style]
7943 }
7944 \bool_if:NT \c__notesslides_notes_bool {
7945   \renewenvironment{columns}[1][]{%
```

```
7946        \par\noindent%
7947        \begin{minipage}%
7948        \slidewidth\centering\leavevmode%
7949    }{%
7950        \end{minipage}\par\noindent%
7951    }%
7952    \newsavebox\columnbox%
7953    \renewenvironment<>{column}[2][]{%
7954        \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
7955    }{%
7956        \end{minipage}\end{lrbox}\usebox\columnbox%
7957    }%
7958 }

7959 \bool_if:NTF \c__notesslides_noproblems_bool {
7960    \newenvironment{problems}{}{}
7961 }{
7962    \excludecomment{problems}
7963 }
```

40.6 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.

```
7964 \gdef\printexcursions{}
7965 \newcommand\excursionref[2]{% label, text
7966    \bool_if:NT \c__notesslides_notes_bool {
7967        \begin{sparagraph}[title=Excursion]
7968            #2 \sref[fallback=the appendix]{#1}.
7969        \end{sparagraph}
7970    }
7971 }
7972 \newcommand\activate@excursion[2][]{
7973    \gappto\printexcursions{\inputref[#1]{#2}}
7974 }
7975 \newcommand\excursion[4][]{% repos, label, path, text
7976    \bool_if:NT \c__notesslides_notes_bool {
7977        \activate@excursion[#1]{#3}\excursionref{#2}{#4}
7978    }
7979 }
```

(*End definition for* \excursion. *This function is documented on page 66.*)

```
7980 \keys_define:nn{notesslides / excursiongroup }{
7981    id       .str_set_x:N  = \l__notesslides_excursion_id_str,
7982    intro    .tl_set:N     = \l__notesslides_excursion_intro_tl,
7983    mhrepos  .str_set_x:N  = \l__notesslides_excursion_mhrepos_str
7984 }
7985 \cs_new_protected:Nn \__notesslides_excursion_args:n {
7986    \tl_clear:N \l__notesslides_excursion_intro_tl
7987    \str_clear:N \l__notesslides_excursion_id_str
```

```
7988    \str_clear:N \l__notesslides_excursion_mhrepos_str
7989    \keys_set:nn {notesslides / excursiongroup }{ #1 }
7990  }
7991  \newcommand\excursiongroup[1][]{
7992    \__notesslides_excursion_args:n{ #1 }
7993    \ifdefempty\printexcursions{}% only if there are excursions
7994    {\begin{note}
7995      \begin{sfragment}[#1]{Excursions}%
7996        \ifdefempty\l__notesslides_excursion_intro_tl{}{
7997          \inputref[\l__notesslides_excursion_mhrepos_str]{
7998            \l__notesslides_excursion_intro_tl
7999          }
8000        }
8001      \printexcursions%
8002    \end{sfragment}
8003  \end{note}}
8004  }
8005  \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
8006  ⟨/package⟩
```

(*End definition for* \excursiongroup. *This function is documented on page* 66.)

Chapter 41

The Implementation

41.1 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.

```
8007 ⟨∗package⟩
8008 ⟨@@=problems⟩
8009 \ProvidesExplPackage{problem}{2022/08/08}{3.2.0}{Semantic Markup for Problems}
8010 \RequirePackage{l3keys2e}
8011 \RequirePackage{amssymb}% for \Box
8012
8013 \keys_define:nn { problem / pkg }{
8014   notes      .default:n   = { true },
8015   notes      .bool_set:N  = \c__problems_notes_bool,
8016   gnotes     .default:n   = { true },
8017   gnotes     .bool_set:N  = \c__problems_gnotes_bool,
8018   hints      .default:n   = { true },
8019   hints      .bool_set:N  = \c__problems_hints_bool,
8020   solutions .default:n    = { true },
8021   solutions .bool_set:N   = \c__problems_solutions_bool,
8022   pts        .default:n   = { true },
8023   pts        .bool_set:N  = \c__problems_pts_bool,
8024   min        .default:n   = { true },
8025   min        .bool_set:N  = \c__problems_min_bool,
8026   boxed      .default:n   = { true },
8027   boxed      .bool_set:N  = \c__problems_boxed_bool,
8028   test      .default:n   = { true },
8029   test      .bool_set:N  = \c__problems_test_bool,
8030   unknown      .code:n       = {
8031     \PassOptionsToPackage{\CurrentOption}{stex}
8032   }
8033 }
8034 \newif\ifsolutions
8035
8036 \ProcessKeysOptions{ problem / pkg }
8037 \bool_if:NTF \c__problems_solutions_bool {
8038   \solutionstrue
```

```
8039  }{
8040    \solutionsfalse
8041  }
8042  \RequirePackage{stex}
```

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

```
8043  \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.

```
8044  \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.

```
8045  \def\prob@problem@kw{Problem}
8046  \def\prob@solution@kw{Solution}
8047  \def\prob@hint@kw{Hint}
8048  \def\prob@note@kw{Note}
8049  \def\prob@gnote@kw{Grading}
8050  \def\prob@pt@kw{pt}
8051  \def\prob@min@kw{min}
8052  \def\prob@correct@kw{Correct}
8053  \def\prob@wrong@kw{Wrong}
```

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

For the other languages, we set up triggers

```
8054  \AddToHook{begindocument}{
8055    \ltx@ifpackageloaded{babel}{
8056        \makeatletter
8057        \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
8058        \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
8059          \input{problem-ngerman.ldf}
8060        }
8061        \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
8062          \input{problem-finnish.ldf}
8063        }
8064        \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
8065          \input{problem-french.ldf}
8066        }
8067        \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
8068          \input{problem-russian.ldf}
8069        }
8070        \makeatother
8071    }{}
8072  }
```

41.2 Problems and Solutions

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

```
8073  \keys_define:nn{ problem / problem }{
8074    id      .str_set_x:N  = \l__problems_prob_id_str,
```

```
8075    pts     .tl_set:N     = \l__problems_prob_pts_tl,
8076    min     .tl_set:N     = \l__problems_prob_min_tl,
8077    title   .tl_set:N     = \l__problems_prob_title_tl,
8078    type    .tl_set:N     = \l__problems_prob_type_tl,
8079    imports .tl_set:N     = \l__problems_prob_imports_tl,
8080    name    .str_set_x:N  = \l__problems_prob_name_str,
8081    refnum  .int_set:N    = \l__problems_prob_refnum_int
8082 }
8083 \cs_new_protected:Nn \__problems_prob_args:n {
8084   \str_clear:N \l__problems_prob_id_str
8085   \str_clear:N \l__problems_prob_name_str
8086   \tl_clear:N \l__problems_prob_pts_tl
8087   \tl_clear:N \l__problems_prob_min_tl
8088   \tl_clear:N \l__problems_prob_title_tl
8089   \tl_clear:N \l__problems_prob_type_tl
8090   \tl_clear:N \l__problems_prob_imports_tl
8091   \int_zero_new:N \l__problems_prob_refnum_int
8092   \keys_set:nn { problem / problem }{ #1 }
8093   \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
8094     \let\l__problems_prob_refnum_int\undefined
8095   }
8096 }
```

Then we set up a counter for problems.

\numberproblemsin

```
8097 \newcounter{sproblem}[section]
8098 \newcommand\numberproblemsin[1]{\@addtoreset{sproblem}{#1}}
8099 \def\theplainsproblem{\arabic{sproblem}}
8100 \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.

```
8101 \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

```
8102 \newcommand\prob@number{
8103   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
8104     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
8105   }{
8106     \int_if_exist:NTF \l__problems_prob_refnum_int {
8107       \prob@label{\int_use:N \l__problems_prob_refnum_int }
8108     }{
8109         \prob@label\theplainsproblem
8110     }
8111   }
8112 }
8113 \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.

```
8114 \newcommand\prob@title[3]{%
8115   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
8116     #2 \l__problems_inclprob_title_tl #3
8117   }{
8118     \tl_if_empty:NTF \l__problems_prob_title_tl {
8119       #1
8120     }{
8121       #2 \l__problems_prob_title_tl #3
8122     }
8123   }
8124 }
```

(*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.

```
8125 \def\prob@heading{
8126   {\prob@problem@kw}\ \prob@number\prob@title{~}{~(}{)\strut}
8127   %\sref@label@id{\prob@problem@kw~\prob@number}{}
8128 }
```

(*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.*)

```
8129 \newenvironment{sproblem}[1][]{
8130   \__problems_prob_args:n{#1}%\sref@target%
8131   \@in@omtexttrue% we are in a statement (for inline definitions)
8132   \refstepcounter{sproblem}\record@problem
8133   \def\current@section@level{\prob@problem@kw}
8134
8135   \str_if_empty:NT \l__problems_prob_name_str {
8136     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
8137     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
8138     \seq_get_left:NN \l_tmpa_seq \l__problems_prob_name_str
8139   }
8140
8141   \stex_if_do_html:T{
8142     \tl_if_empty:NF \l__problems_prob_title_tl {
8143       \exp_args:No \stex_document_title:n \l__problems_prob_title_tl
8144     }
8145   }
8146
8147   \exp_args:Nno\stex_module_setup:nn{type=problem}\l__problems_prob_name_str
8148
8149   \stex_reactivate_macro:N \STEXexport
8150   \stex_reactivate_macro:N \importmodule
```

```
8151    \stex_reactivate_macro:N \symdecl
8152    \stex_reactivate_macro:N \notation
8153    \stex_reactivate_macro:N \symdef
8154
8155    \stex_if_do_html:T{
8156      \begin{stex_annotate_env} {problem} {
8157        \l_stex_module_ns_str ? \l_stex_module_name_str
8158      }
8159
8160      \stex_annotate_invisible:nnn{header}{} {
8161        \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
8162        \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
8163        \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
8164          \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
8165        }
8166      }
8167    }
8168
8169    \stex_csl_to_imports:No \importmodule \l__problems_prob_imports_tl
8170
8171
8172    \tl_if_exist:NTF \l__problems_inclprob_type_tl {
8173      \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
8174    }{
8175      \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
8176    }
8177    \str_if_exist:NTF \l__problems_inclprob_id_str {
8178      \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
8179    }{
8180      \str_set_eq:NN \sproblemid \l__problems_prob_id_str
8181    }
8182
8183
8184    \stex_if_smsmode:F {
8185      \clist_set:No \l_tmpa_clist \sproblemtype
8186      \tl_clear:N \l_tmpa_tl
8187      \clist_map_inline:Nn \l_tmpa_clist {
8188        \tl_if_exist:cT {__problems_sproblem_##1_start:}{
8189          \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_start:}}
8190        }
8191      }
8192      \tl_if_empty:NTF \l_tmpa_tl {
8193        \__problems_sproblem_start:
8194      }{
8195        \l_tmpa_tl
8196      }
8197    }
8198    \stex_ref_new_doc_target:n \sproblemid
8199    \stex_if_smsmode:TF \stex_smsmode_do: \ignorespacesandpars
8200 }{
8201    \__stex_modules_end_module:
8202    \stex_if_smsmode:F{
8203      \clist_set:No \l_tmpa_clist \sproblemtype
8204      \tl_clear:N \l_tmpa_tl
```

```
8205    \clist_map_inline:Nn \l_tmpa_clist {
8206      \tl_if_exist:cT {__problems_sproblem_##1_end:}{
8207        \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_end:}}
8208      }
8209    }
8210    \tl_if_empty:NTF \l_tmpa_tl {
8211      \__problems_sproblem_end:
8212    }{
8213      \l_tmpa_tl
8214    }
8215  }
8216  \stex_if_do_html:T{
8217    \end{stex_annotate_env}
8218  }
8219
8220  \smallskip
8221 }
8222
8223 \seq_put_right:Nx\g_stex_smsmode_allowedenvs_seq{\tl_to_str:n{sproblem}}
8224
8225
8226
8227 \cs_new_protected:Nn \__problems_sproblem_start: {
8228   \par\noindent\textbf\prob@heading\show@pts\show@min\\\ignorespacesandpars
8229 }
8230 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
8231
8232 \newcommand\stexpatchproblem[3][] {
8233     \str_set:Nx \l_tmpa_str{ #1 }
8234     \str_if_empty:NTF \l_tmpa_str {
8235       \tl_set:Nn \__problems_sproblem_start: { #2 }
8236       \tl_set:Nn \__problems_sproblem_end: { #3 }
8237     }{
8238       \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
8239       \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
8240     }
8241 }
8242
8243
8244 \bool_if:NT \c__problems_boxed_bool {
8245   \surroundwithmdframed{problem}
8246 }
```

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

```
8247 \def\record@problem{
8248   \protected@write\@auxout{}
8249   {
8250     \string\@problem{\prob@number}
8251     {
8252       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
8253         \l__problems_inclprob_pts_tl
8254       }{
8255         \l__problems_prob_pts_tl
8256       }
```

```
8257        }%
8258        {
8259          \tl_if_exist:NTF \l__problems_inclprob_min_tl {
8260            \l__problems_inclprob_min_tl
8261          }{
8262            \l__problems_prob_min_tl
8263          }
8264        }
8265      }
8266    }
```

(*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).

```
8267  \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.

```
8268  \keys_define:nn { problem / solution }{
8269    id              .str_set_x:N  = \l__problems_solution_id_str ,
8270    for             .str_set_x:N  = \l__problems_solution_for_str ,
8271    type            .str_set_x:N  = \l__problems_solution_type_str ,
8272    title           .tl_set:N     = \l__problems_solution_title_tl
8273  }
8274  \cs_new_protected:Nn \__problems_solution_args:n {
8275    \str_clear:N \l__problems_solution_id_str
8276    \str_clear:N \l__problems_solution_type_str
8277    \str_clear:N \l__problems_solution_for_str
8278    \tl_clear:N \l__problems_solution_title_tl
8279    \keys_set:nn { problem / solution }{ #1 }
8280  }
```

`\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.

```
8281  \box_new:N \l__problems_solution_box
8282  \newenvironment{solution}[1][]{
8283    \__problems_solution_args:n{#1}
8284    \stex_html_backend:TF{
8285      \stex_if_do_html:T{
8286        \begin{stex_annotate_env}{solution}{}
8287          \str_if_empty:NF \l__problems_solution_type_str {
8288            \par\noindent
8289            \stex_annotate_invisible:nnn{typestrings}{\sexampletype}{}
8290          }
8291          \noindent\textbf{Solution\tl_if_empty:NF\l__problems_solution_title_tl{~(\l__problem
8292      }
8293    }{
8294      \setbox\l__problems_solution_box\vbox\bgroup
8295        \par\smallskip\hrule\smallskip
8296        \noindent\textbf{Solution\tl_if_empty:NF\l__problems_solution_title_tl{~(\l__problems_
8297    }
```

```
8298 }{
8299     \stex_html_backend:TF{
8300       \stex_if_do_html:T{
8301         \end{stex_annotate_env}
8302       }
8303     }{
8304       \smallskip\hrule
8305       \egroup
8306       \bool_if:NT \c__problems_solutions_bool {
8307         \strut\par\noindent
8308         \box\l__problems_solution_box
8309       }
8310     }
8311 }

8312
8313 \newcommand\startsolutions{
8314   \bool_set_true:N \c__problems_solutions_bool
8315   \solutionstrue
8316 %   \specialcomment{solution}{\@startsolution}{
8317 %     \bool_if:NF \c__problems_boxed_bool {
8318 %       \hrule\medskip
8319 %     }
8320 %     \end{small}%
8321 %   }
8322 %   \bool_if:NT \c__problems_boxed_bool {
8323 %     \surroundwithmdframed{solution}
8324 %   }
8325 }
```

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

```
8326 \newcommand\stopsolutions{\bool_set_false:N \c__problems_solutions_bool \solutionsfalse}%\ex
```

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

exnote (*env.*)

```
8327 \bool_if:NTF \c__problems_notes_bool {
8328   \newenvironment{exnote}[1][]{
8329     \par\smallskip\hrule\smallskip
8330     \noindent\textbf{\prob@note@kw :~ }\small
8331   }{
8332     \smallskip\hrule
8333   }
8334 }{
8335   \excludecomment{exnote}
8336 }
```

hint (*env.*)

```
8337 \bool_if:NTF \c__problems_notes_bool {
8338   \newenvironment{hint}[1][]{
8339     \par\smallskip\hrule\smallskip
8340     \noindent\textbf{\prob@hint@kw :~ }\small
8341   }{
```

gnote (*env.*)

```
8342      \smallskip\hrule
8343    }
8344    \newenvironment{exhint}[1][]{
8345      \par\smallskip\hrule\smallskip
8346      \noindent\textbf{\prob@hint@kw :~ }\small
8347    }{
8348      \smallskip\hrule
8349    }
8350  }{
8351    \excludecomment{hint}
8352    \excludecomment{exhint}
8353  }
```

gnote (*env.*)

```
8354  \bool_if:NTF \c__problems_notes_bool {
8355    \newenvironment{gnote}[1][]{
8356      \par\smallskip\hrule\smallskip
8357      \noindent\textbf{\prob@gnote@kw :~ }\small
8358    }{
8359      \smallskip\hrule
8360    }
8361  }{
8362    \excludecomment{gnote}
8363  }
```

41.3 Markup for Added Value Services

41.4 Multiple Choice Blocks

EdN:12

mcb (*env.*) [12]

```
8364  \newenvironment{mcb}{
8365    \begin{enumerate}
8366  }{
8367    \end{enumerate}
8368  }
```

we define the keys for the mcc macro

```
8369  \cs_new_protected:Nn \__problems_do_yes_param:Nn {
8370    \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
8371      \bool_set_true:N #1
8372    }{
8373      \bool_set_false:N #1
8374    }
8375  }
8376  \keys_define:nn { problem / mcc }{
8377    id        .str_set_x:N  = \l__problems_mcc_id_str ,
8378    feedback  .tl_set:N     = \l__problems_mcc_feedback_tl ,
8379    T         .default:n    = { false } ,
8380    T         .bool_set:N   = \l__problems_mcc_t_bool ,
8381    F         .default:n    = { false } ,
```

---

[12]EdNote: MK: maybe import something better here from a dedicated MC package

294

```
8382    F        .bool_set:N   = \l__problems_mcc_f_bool ,
8383    Ttext    .tl_set:N     = \l__problems_mcc_Ttext_tl ,
8384    Ftext    .tl_set:N     = \l__problems_mcc_Ftext_tl
8385  }
8386  \cs_new_protected:Nn \l__problems_mcc_args:n {
8387    \str_clear:N \l__problems_mcc_id_str
8388    \tl_clear:N \l__problems_mcc_feedback_tl
8389    \bool_set_false:N \l__problems_mcc_t_bool
8390    \bool_set_false:N \l__problems_mcc_f_bool
8391    \tl_clear:N \l__problems_mcc_Ttext_tl
8392    \tl_clear:N \l__problems_mcc_Ftext_tl
8393    \str_clear:N \l__problems_mcc_id_str
8394    \keys_set:nn { problem / mcc }{ #1 }
8395  }
```

\mcc

```
8396  \def\mccTrueText{\textbf{\prob@correct@kw!~}}
8397  \def\mccFalseText{\textbf{\prob@wrong@kw!~}}
8398  \newcommand\mcc[2][]{
8399    \l__problems_mcc_args:n{ #1 }
8400    \item[$\Box$] #2
8401    \bool_if:NT \c__problems_solutions_bool{
8402      \\
8403      \bool_if:NT \l__problems_mcc_t_bool {
8404        \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccTrueText\l__problems_mcc_Ttext_tl
8405      }
8406      \bool_if:NT \l__problems_mcc_f_bool {
8407        \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccFalseText\l__problems_mcc_Ftext_tl
8408      }
8409      \tl_if_empty:NF \l__problems_mcc_feedback_tl {
8410        \emph{\l__problems_mcc_feedback_tl}
8411      }
8412    }
8413  } %solutions
```

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

41.5   Filling in Concrete Solutions

\includeproblem   This is embarrasingly simple, but can grow over time.

```
8414  \newcommand\fillinsol[2][]{%
8415    \def\@test{#1}
8416    \quad%
8417    \ifsolutions\textcolor{red}{#1!}\else%
8418    \fbox{\ifx\@test\@empty\phantom{\huge{21}}\else\hspace{#1}\fi}%
8419    \fi}
```

(*End definition for* \includeproblem. *This function is documented on page* *71.*)

41.6 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.

```
8420
8421 \keys_define:nn{ problem / inclproblem }{
8422   id      .str_set_x:N  = \l__problems_inclprob_id_str,
8423   pts     .tl_set:N     = \l__problems_inclprob_pts_tl,
8424   min     .tl_set:N     = \l__problems_inclprob_min_tl,
8425   title   .tl_set:N     = \l__problems_inclprob_title_tl,
8426   refnum  .int_set:N    = \l__problems_inclprob_refnum_int,
8427   type    .tl_set:N     = \l__problems_inclprob_type_tl,
8428   mhrepos .str_set_x:N  = \l__problems_inclprob_mhrepos_str
8429 }
8430 \cs_new_protected:Nn \__problems_inclprob_args:n {
8431   \str_clear:N \l__problems_prob_id_str
8432   \tl_clear:N \l__problems_inclprob_pts_tl
8433   \tl_clear:N \l__problems_inclprob_min_tl
8434   \tl_clear:N \l__problems_inclprob_title_tl
8435   \tl_clear:N \l__problems_inclprob_type_tl
8436   \int_zero_new:N \l__problems_inclprob_refnum_int
8437   \str_clear:N \l__problems_inclprob_mhrepos_str
8438   \keys_set:nn { problem / inclproblem }{ #1 }
8439   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
8440     \let\l__problems_inclprob_pts_tl\undefined
8441   }
8442   \tl_if_empty:NT \l__problems_inclprob_min_tl {
8443     \let\l__problems_inclprob_min_tl\undefined
8444   }
8445   \tl_if_empty:NT \l__problems_inclprob_title_tl {
8446     \let\l__problems_inclprob_title_tl\undefined
8447   }
8448   \tl_if_empty:NT \l__problems_inclprob_type_tl {
8449     \let\l__problems_inclprob_type_tl\undefined
8450   }
8451   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
8452     \let\l__problems_inclprob_refnum_int\undefined
8453   }
8454 }
8455
8456 \cs_new_protected:Nn \__problems_inclprob_clear: {
8457   \let\l__problems_inclprob_id_str\undefined
8458   \let\l__problems_inclprob_pts_tl\undefined
8459   \let\l__problems_inclprob_min_tl\undefined
8460   \let\l__problems_inclprob_title_tl\undefined
8461   \let\l__problems_inclprob_type_tl\undefined
8462   \let\l__problems_inclprob_refnum_int\undefined
8463   \let\l__problems_inclprob_mhrepos_str\undefined
8464 }
8465 \__problems_inclprob_clear:
8466
8467 \newcommand\includeproblem[2][]{
8468   \__problems_inclprob_args:n{ #1 }
```

```
8469    \exp_args:No \stex_in_repository:nn\l__problems_inclprob_mhrepos_str{
8470      \stex_html_backend:TF {
8471        \str_clear:N \l_tmpa_str
8472        \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
8473          \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
8474        }
8475        \stex_annotate_invisible:nnn{includeproblem}{
8476          \l_tmpa_str / #2
8477        }{}
8478      }{
8479        \begingroup
8480          \inputreftrue
8481          \tl_if_empty:nTF{ ##1 }{
8482            \input{#2}
8483          }{
8484            \input{ \c_stex_mathhub_str / ##1 / source / #2 }
8485          }
8486        \endgroup
8487      }
8488    }
8489    \__problems_inclprob_clear:
8490 }
```

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

41.7 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.

```
8491 \AddToHook{enddocument}{
8492   \bool_if:NT \c__problems_pts_bool {
8493     \message{Total:~\arabic{pts}~points}
8494   }
8495   \bool_if:NT \c__problems_min_bool {
8496     \message{Total:~\arabic{min}~minutes}
8497   }
8498 }
```

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

```
8499 \def\pts#1{
8500   \bool_if:NT \c__problems_pts_bool {
8501     \marginpar{#1~\prob@pt@kw}
8502   }
8503 }
8504 \def\min#1{
8505   \bool_if:NT \c__problems_min_bool {
8506     \marginpar{#1~\prob@min@kw}
8507   }
8508 }
```

\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.

```
8509 \newcounter{pts}
8510 \def\show@pts{
8511   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
8512     \bool_if:NT \c__problems_pts_bool {
8513       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
8514       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
8515     }
8516   }{
8517     \tl_if_exist:NT \l__problems_prob_pts_tl {
8518       \bool_if:NT \c__problems_pts_bool {
8519         \tl_if_empty:NT\l__problems_prob_pts_tl{
8520           \tl_set:Nn \l__problems_prob_pts_tl {0}
8521         }
8522         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
8523         \addtocounter{pts}{\l__problems_prob_pts_tl}
8524       }
8525     }
8526   }
8527 }
```

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

and now the same for the minutes

```
8528 \newcounter{min}
8529 \def\show@min{
8530   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
8531     \bool_if:NT \c__problems_min_bool {
8532       \marginpar{\l__problems_inclprob_pts_tl\ min}
8533       \addtocounter{min}{\l__problems_inclprob_min_tl}
8534     }
8535   }{
8536     \tl_if_exist:NT \l__problems_prob_min_tl {
8537       \bool_if:NT \c__problems_min_bool {
8538         \tl_if_empty:NT\l__problems_prob_min_tl{
8539           \tl_set:Nn \l__problems_prob_min_tl {0}
8540         }
8541         \marginpar{\l__problems_prob_min_tl\ min}
8542         \addtocounter{min}{\l__problems_prob_min_tl}
8543       }
8544     }
8545   }
8546 }
8547 ⟨/package⟩
```

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

41.8   Testing and Spacing

\testspace

```
8548 \newcommand\testspace[1]{\bool_if:NT \c__problems_boxed_bool {\vspace*{#1}}}
```

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

298

\testnewpage

8549 `\newcommand\testnewpage{\bool_if:NT \c__problems_boxed_bool  {\newpage}}`

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

\testemptypage

8550 `\newcommand\testemptypage[1][]{%`
8551 `\bool_if:NT \c__problems_boxed_bool {\begin{center}\hwexam@testemptypage@kw\end{center}\vfil`

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

\test*space

8552 `\newcommand\testsmallspace{\testspace{1cm}}`
8553 `\newcommand\testmedspace{\testspace{2cm}}`
8554 `\newcommand\testbigspace{\testspace{3cm}}`

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

Chapter 42

Implementation: The hwexam Package

42.1 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.

```
8555 ⟨∗package⟩
8556 \ProvidesExplPackage{hwexam}{2022/08/08}{3.2.0}{homework assignments and exams}
8557 \RequirePackage{l3keys2e}
8558
8559 \newif\iftest\testfalse
8560 \DeclareOption{test}{\testtrue\PassOptionsToPackage{\CurrentOption}{problem}}
8561 \newif\ifmultiple\multiplefalse
8562 \DeclareOption{multiple}{\multipletrue}
8563 \DeclareOption{lang}{\PassOptionsToPackage{\CurrentOption}{problem}}
8564 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
8565 \ProcessOptions
```

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

```
8566 \RequirePackage{keyval}[1997/11/10]
8567 \RequirePackage{problem}
```

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

```
8568 \newcommand\hwexam@assignment@kw{Assignment}
8569 \newcommand\hwexam@given@kw{Given}
8570 \newcommand\hwexam@due@kw{Due}
8571 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~blank~for~extra~space}
8572 \newcommand\hwexam@minutes@kw{minutes}
8573 \newcommand\correction@probs@kw{prob.}
8574 \newcommand\correction@pts@kw{total}
8575 \newcommand\correction@reached@kw{reached}
8576 \newcommand\correction@sum@kw{Sum}
8577 \newcommand\correction@grade@kw{grade}
8578 \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

```
8579  \AddToHook{begindocument}{
8580  \ltx@ifpackageloaded{babel}{
8581  \makeatletter
8582  \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
8583  \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
8584    \input{hwexam-ngerman.ldf}
8585  }
8586  \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
8587    \input{hwexam-finnish.ldf}
8588  }
8589  \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
8590    \input{hwexam-french.ldf}
8591  }
8592  \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
8593    \input{hwexam-russian.ldf}
8594  }
8595  \makeatother
8596  }{}
8597  }
8598
```

42.2   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.

```
8599  \newcounter{assignment}
8600  %\numberproblemsin{assignment}
```

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

```
8601  \keys_define:nn { hwexam / assignment } {
8602  id   .str_set_x:N = \l_@@_assign_id_str,
8603  number   .int_set:N  = \l_@@_assign_number_int,
8604  title   .tl_set:N  = \l_@@_assign_title_tl,
8605  type   .tl_set:N  = \l_@@_assign_type_tl,
8606  given .tl_set:N  = \l_@@_assign_given_tl,
8607  due .tl_set:N  = \l_@@_assign_due_tl,
8608  loadmodules .code:n  = {
8609  \bool_set_true:N \l_@@_assign_loadmodules_bool
8610  }
8611  }
8612  \cs_new_protected:Nn \_@@_assignment_args:n {
8613  \str_clear:N \l_@@_assign_id_str
8614  \int_set:Nn \l_@@_assign_number_int {-1}
8615  \tl_clear:N \l_@@_assign_title_tl
8616  \tl_clear:N \l_@@_assign_type_tl
8617  \tl_clear:N \l_@@_assign_given_tl
8618  \tl_clear:N \l_@@_assign_due_tl
8619  \bool_set_false:N \l_@@_assign_loadmodules_bool
8620  \keys_set:nn { hwexam / assignment }{ #1 }
8621  }
```

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.

```
8622  \newcommand\given@due[2]{
8623  \bool_lazy_all:nF {
8624  {\tl_if_empty_p:V \l_@@_inclassign_given_tl}
8625  {\tl_if_empty_p:V \l_@@_assign_given_tl}
8626  {\tl_if_empty_p:V \l_@@_inclassign_due_tl}
8627  {\tl_if_empty_p:V \l_@@_assign_due_tl}
8628  }{ #1 }
8629
8630  \tl_if_empty:NTF \l_@@_inclassign_given_tl {
8631  \tl_if_empty:NF \l_@@_assign_given_tl {
8632  \hwexam@given@kw\xspace\l_@@_assign_given_tl
8633  }
8634  }{
8635  \hwexam@given@kw\xspace\l_@@_inclassign_given_tl
8636  }
8637
8638  \bool_lazy_or:nnF {
8639  \bool_lazy_and_p:nn {
8640  \tl_if_empty_p:V \l_@@_inclassign_due_tl
8641  }{
8642  \tl_if_empty_p:V \l_@@_assign_due_tl
8643  }
8644  }{
8645  \bool_lazy_and_p:nn {
8646  \tl_if_empty_p:V \l_@@_inclassign_due_tl
8647  }{
8648  \tl_if_empty_p:V \l_@@_assign_due_tl
8649  }
8650  }{ ,~ }
8651
8652  \tl_if_empty:NTF \l_@@_inclassign_due_tl {
8653  \tl_if_empty:NF \l_@@_assign_due_tl {
8654  \hwexam@due@kw\xspace \l_@@_assign_due_tl
8655  }
8656  }{
8657  \hwexam@due@kw\xspace \l_@@_inclassign_due_tl
8658  }
8659
8660  \bool_lazy_all:nF {
8661  { \tl_if_empty_p:V \l_@@_inclassign_given_tl }
8662  { \tl_if_empty_p:V \l_@@_assign_given_tl }
8663  { \tl_if_empty_p:V \l_@@_inclassign_due_tl }
8664  { \tl_if_empty_p:V \l_@@_assign_due_tl }
8665  }{ #2 }
8666  }
```

`\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.

```
8667 \newcommand\assignment@title[3]{
8668 \tl_if_empty:NTF \l_@@_inclassign_title_tl {
8669 \tl_if_empty:NTF \l_@@_assign_title_tl {
8670 #1
8671 }{
8672 #2\l_@@_assign_title_tl#3
8673 }
8674 }{
8675 #2\l_@@_inclassign_title_tl#3
8676 }
8677 }
```

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

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

```
8678 \newcommand\assignment@number{
8679 \int_compare:nNnTF \l_@@_inclassign_number_int = {-1} {
8680 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8681 \arabic{assignment}
8682 } {
8683 \int_use:N \l_@@_assign_number_int
8684 }
8685 }{
8686 \int_use:N \l_@@_inclassign_number_int
8687 }
8688 }
```

(*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.

```
8689 \newenvironment{assignment}[1][]{
8690 \_@@_assignment_args:n { #1 }
8691 %\sref@target
8692 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8693 \global\stepcounter{assignment}
8694 }{
8695 \global\setcounter{assignment}{\int_use:N\l_@@_assign_number_int}
8696 }
8697 \setcounter{sproblem}{0}
8698 \renewcommand\prob@label[1]{\assignment@number.##1}
8699 \def\current@section@level{\document@hwexamtype}
8700 %\sref@label@id{\document@hwexamtype \thesection}
8701 \begin{@assignment}
8702 }{
8703 \end{@assignment}
8704 }
```

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

```
8705  \def\ass@title{
8706  {\protect\document@hwexamtype}~\arabic{assignment}
8707  \assignment@title{}{\;(}{)\;} -- \given@due{}{}
8708  }
8709  \ifmultiple
8710  \newenvironment{@assignment}{
8711  \bool_if:NTF \l_@@_assign_loadmodules_bool {
8712  \begin{sfragment}[loadmodules]{\ass@title}
8713  }{
8714  \begin{sfragment}{\ass@title}
8715  }
8716  }{
8717  \end{sfragment}
8718  }
```

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

```
8719  \else
8720  \newenvironment{@assignment}{
8721  \begin{center}\bf
8722  \Large\@title\strut\\
8723  \document@hwexamtype~\arabic{assignment}\assignment@title{\;}{:\;}{\\}
8724  \large\given@due{--\;}{\;--}
8725  \end{center}
8726  }{}
8727  \fi% multiple
```

42.3   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.

```
8728  \keys_define:nn { hwexam / inclassignment } {
8729  %id   .str_set_x:N = \l_@@_assign_id_str,
8730  number   .int_set:N = \l_@@_inclassign_number_int,
8731  title   .tl_set:N  = \l_@@_inclassign_title_tl,
8732  type   .tl_set:N  = \l_@@_inclassign_type_tl,
8733  given .tl_set:N  = \l_@@_inclassign_given_tl,
8734  due .tl_set:N  = \l_@@_inclassign_due_tl,
8735  mhrepos   .str_set_x:N = \l_@@_inclassign_mhrepos_str
8736  }
8737  \cs_new_protected:Nn \_@@_inclassignment_args:n {
8738  \int_set:Nn \l_@@_inclassign_number_int {-1}
8739  \tl_clear:N \l_@@_inclassign_title_tl
8740  \tl_clear:N \l_@@_inclassign_type_tl
8741  \tl_clear:N \l_@@_inclassign_given_tl
8742  \tl_clear:N \l_@@_inclassign_due_tl
8743  \str_clear:N \l_@@_inclassign_mhrepos_str
8744  \keys_set:nn { hwexam / inclassignment }{ #1 }
8745  }
8746  \_@@_inclassignment_args:n {}
8747
8748  \newcommand\inputassignment[2][]{
```

304

```
8749  \_@@_inclassignment_args:n { #1 }
8750  \str_if_empty:NTF \l_@@_inclassign_mhrepos_str {
8751  \input{#2}
8752  }{
8753  \stex_in_repository:nn{\l_@@_inclassign_mhrepos_str}{
8754  \input{\mhpath{\l_@@_inclassign_mhrepos_str}{#2}}}
8755  }
8756  }
8757  \_@@_inclassignment_args:n {}
8758  }
8759  \newcommand\includeassignment[2][]{
8760  \newpage
8761  \inputassignment[#1]{#2}
8762  }
```

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

42.4 Typesetting Exams

\quizheading

```
8763  \ExplSyntaxOff
8764  \newcommand\quizheading[1]{%
8765  \def\@tas{#1}%
8766  \large\noindent NAME: \hspace{8cm}  MAILBOX:\\[2ex]%
8767  \ifx\@tas\@empty\else%
8768  \noindent TA:~\@for\@I:=\@tas\do{{\Large$\Box$}\@I\hspace*{1em}}\\[2ex]%
8769  \fi%
8770  }
8771  \ExplSyntaxOn
```

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

\testheading

```
8772
8773  \def\hwexamheader{\input{hwexam-default.header}}
8774
8775  \def\hwexamminutes{
8776  \tl_if_empty:NTF \testheading@duration {
8777  {\testheading@min}~\hwexam@minutes@kw
8778  }{
8779  \testheading@duration
8780  }
8781  }
8782
8783  \keys_define:nn { hwexam / testheading } {
8784  min   .tl_set:N  = \testheading@min,
8785  duration .tl_set:N  = \testheading@duration,
8786  reqpts .tl_set:N  = \testheading@reqpts,
8787  tools .tl_set:N  = \testheading@tools
8788  }
8789  \cs_new_protected:Nn \_@@_testheading_args:n {
8790  \tl_clear:N \testheading@min
8791  \tl_clear:N \testheading@duration
```

305

```
8792   \tl_clear:N \testheading@reqpts
8793   \tl_clear:N \testheading@tools
8794   \keys_set:nn { hwexam / testheading }{ #1 }
8795 }
8796 \newenvironment{testheading}[1][]{
8797   \_@@_testheading_args:n{ #1 }
8798   \newcount\check@time\check@time=\testheading@min
8799   \advance\check@time by -\theassignment@totalmin
8800   \newif\if@bonuspoints
8801   \tl_if_empty:NTF \testheading@reqpts {
8802   \@bonuspointsfalse
8803   }{
8804   \newcount\bonus@pts
8805   \bonus@pts=\theassignment@totalpts
8806   \advance\bonus@pts by -\testheading@reqpts
8807   \edef\bonus@pts{\the\bonus@pts}
8808   \@bonuspointstrue
8809   }
8810   \edef\check@time{\the\check@time}
8811
8812   \makeatletter\hwexamheader\makeatother
8813 }{
8814   \newpage
8815 }
```

(*End definition for* \testheading. *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.

```
8816 ⟨@@=problems⟩
8817 \renewcommand\@problem[3]{
8818   \stepcounter{assignment@probs}
8819   \def\__problemspts{#2}
8820   \ifx\__problemspts\@empty\else
8821   \addtocounter{assignment@totalpts}{#2}
8822   \fi
8823   \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\f
8824   \xdef\correction@probs{\correction@probs & #1}%
8825   \xdef\correction@pts{\correction@pts & #2}
8826   \xdef\correction@reached{\correction@reached &}
8827 }
8828 ⟨@@=hwexam⟩
```

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

\correction@table   This macro generates the correction table

```
8829 \newcounter{assignment@probs}
8830 \newcounter{assignment@totalpts}
8831 \newcounter{assignment@totalmin}
8832 \def\correction@probs{\correction@probs@kw}
8833 \def\correction@pts{\correction@pts@kw}
8834 \def\correction@reached{\correction@reached@kw}
8835 \stepcounter{assignment@probs}
8836 \newcommand\correction@table{
```

```
8837  \resizebox{\textwidth}{!}{%
8838  \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
8839  &\multicolumn{\theassignment@probs}{c||}%|
8840  {\footnotesize\correction@forgrading@kw} &\\\hline
8841  \correction@probs & \correction@sum@kw & \correction@grade@kw\\\hline
8842  \correction@pts &\theassignment@totalpts & \\\hline
8843  \correction@reached & & \\[.7cm]\hline
8844  \end{tabular}}}
8845  ⟨/package⟩
```

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

42.5   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 43

References

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).

[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).