The sTeX3 Package *

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

2022-04-23

Abstract

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

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

sTeX augments LaTeX with

- *semantic macros* that denote and distinguish between mathematical concepts, operators, etc. independent of their notational presentation,

- a powerful *module system* that allows for authoring and importing individual fragments containing document text and/or semantic macros, independent of – and without hard coding – directory paths relative to the current document, and

- a mechanism for exporting sTeX documents to (modular) XHTML, preserving all the semantic information for semantically informed knowledge management services.

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

- Part I is a general manual for the sTeX package and associated software. It is primarily directed at end-users who want to use sTeX to author semantically enriched documents.

- Part II documents the macros provided by the sTeX package. It is primarily directed at package authors who want to build on sTeX, but can also serve as a reference manual for end-users.

- Part III documents additional packages that build on sTeX, primarily its module system. These are not part of the sTeX package itself, but useful additions enabled by sTeX package functionality.

- Part IV is the detailled documentation of the sTeX package implementation.

---

*Version 3.0 (last revised 2022-04-23)

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 [] 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.

Chapter 2

Quickstart

2.1 Setup

There are two ways of using sTeX: as a

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

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

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

2.1.1 Minimal Setup for the PDF-only Workflow

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

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

2.1.2 GIT-based Setup for the sTeX Development Version

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

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

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

---

[1]NEW PART: MK: reorganized, we do not need the full MKM tool chain

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

2.1.3   ꚃₜₑₓ Archives (Manual Setup)

Writing semantically annotated ꚃₜₑₓ becomes much easier, if we can use well-designed libraries of already annotated content. ꚃₜₑₓ provides such libraries as ꚃₜₑₓ 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 ꚃₜₑₓ archive as an **archive path `<apath>`** and a name `<archive>`. We can clone the ꚃₜₑₓ archive by the following command-line instructions:

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

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

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

2.1.4   The ꚃₜₑₓ IDE

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

ENP:1

2.1.5   Manual Setup for Active Documents and Knowledge Management Services

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

EdN:2

- **The Mmt System** available here[2]. 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 ꚃₜₑₓ/Mmt content archives.

- **ꚃₜₑₓ 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) ꚃₜₑₓ archives are cloned as well.

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

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

---

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

2.2   A First sTeX Document

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

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

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

Compiling this document with `pdflatex` should yield the output

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

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

Theorem 0.2.   The geometric series converges towards 1.

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

Remark 2.2.1:

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

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

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

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

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

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

\importmodule Next, we *import* two modules – `series` from the sTeX archive `smglom/calculus`, and `realarith` from the sTeX 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 sTeX 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{GeometricSeries}[name=geometric-series]{\comp{S}}
```

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

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

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

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

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

```
... is the \symname{?series}
```

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

Note that the argument of \symref can be a local or 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, 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}.
```
The \definame{geometricSeries} ...
```

\definame
\definiendum
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

7

macro can have several notations – in that case, we can explicitly choose a specific notation by providing its identifier as an optional argument; e.g. `\realdivide[frac]{a}{b}` will use the explicit notation named `frac` of the semantic macro `\realdivide`, which yields $\frac{a}{b}$ instead of a/b.

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

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

2.2.1 OMDoc/xhtml Conversion

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

TODO VSCode Plugin

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

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

---

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

...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 2.2.2:**
>
> > Note that the `html` when opened in a browser will look slightly different than the `pdf` when it comes to highlighting semantic content – that is because naturally `html` allows for much more powerful features than `pdf` does. Consequently, the `html` is intended to be served by a system like MMT, which can pick up on the semantic information and offer much more powerful highlighting, linking and similar features, and being customizable by *readers* rather than being prescribed by an author.
> >
> > Additionally, not all browsers (most notably Chrome) support MATHML natively, and might require additional external JavaScript libraries such as MathJax to render mathematical formulas properly.

Chapter 3

Creating sTeX Content

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

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

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

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

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

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

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

3.1   How Knowledge is Organized in sTeX

sTeX content is organized on multiple levels:

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

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

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

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

↩M→
—M→   • sTeX archives are simultaneously Mmt archives, and the same directory
⤳T↪   structure is consequently used.
       • sTeX modules correspond to OMDoc/Mmt *theories*. `\importmodule`s (and

similar constructions) induce Mᴍᴛ `include`s and other *theory morphisms*, thus giving rise to a *theory graph* in the OMDᴏᴄ sense [RK13].

- Symbol declarations induce OMDᴏᴄ/Mᴍᴛ *constants*, with optional (formal) *type* and *definiens* components.

- Finally, sTEX expressions are converted to OMDᴏᴄ/Mᴍᴛ terms, which use the abstract syntax (and XML encoding) of OᴘᴇɴMᴀᴛʜ [Bus+04].

3.2   sTEX Archives

3.2.1   The Local MathHub-Directory

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

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

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

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

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

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

3.2.2   The Structure of sTEX Archives

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

Each such archive needs two subdirectories:

- `/source` – this is where all your tex files go.

- `/META-INF` – a directory containing a single file `MANIFEST.MF`, the content of which we will consider shortly

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

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

We recommend the following additional directory structure in the `source`-folder of an SₜₑX archive:

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

- `/source/def/` – definitions

- `/source/ex/` – examples

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

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

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

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

EdN:3
- `/source/pic/` – image files.[3]

3.2.3 MANIFEST.MF-Files

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

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

---

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

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

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

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

3.2.4 Using Files in sTeX Archives Directly

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

\mhinput

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

\inputref

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

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

\ifinput

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

\addmhbibresource

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

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

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

\libinput

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

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

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

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

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

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

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

> **Remark 3.2.1:**
>
> A good practice is to have individual STEX fragments follow basically this document frame:
>
> ```
> 1 \documentclass{stex}
> 2 \libinput{preamble}
> 3 \begin{document}
> 4     ...
> 5     \ifinputref \else \libinput{postamble} \fi
> 6 \end{document}
> ```
>
> Then the `preamble.tex` files can take care of loading the generally required packages, setting presentation customizations etc. (per archive or archive group or both), and `postamble.tex` can e.g. print the bibliography, index etc.
>
>   `\libusepackage` is particularly useful in `preamble.tex` when we want to use custom packages that are not part of TeXLive. In this case we commit the respective packages in one of the `lib` folders and use `\libusepackage` to load them.

3.3 Module, Symbol and Notation Declarations

3.3.1 The `smodule`-Environment

A new module is declared using the basic syntax

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

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

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

($\langle token\ list\rangle$) to display in customizations.

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

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

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

($\langle URI\rangle$) 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:`.

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

($\langle language\rangle$) 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 (⟨*string*⟩∗) names of the creators.

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

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

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

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

Example 1

Input:

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

Output:

```
Hello World
```

.

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

For example:

Example 2

Input:

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

Output:

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

.

3.3.2 Declaring New Symbols and Notations

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

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

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

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

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

Example 3

Input:

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

Output:

> Given a foo, we can...

.

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

Example 4

Input:

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

Output:

> this is a symbol taking two arguments.

.

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

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

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

> **Example 5**
>
> Input:
>
> ```
> 1 \notation{binarysymbol}{\text{First: }#1\text{; Second: }#2}
> 2 $\binarysymbol{a}{b}$
> ```
>
> Output:
>
> First: a; Second: b

.

> ⮠M⭢ Applications of semantic macros, such as `\binarysymbol{a}{b}` are translated to
> ⭢M⭢ Mᴍᴛ/OMDᴏᴄ as `OMA`-terms with head `<OMS name="...?binarysymbol"/>`.
> ⤳T⤳ Semantic macros with no arguments correspond to `OMS` directly.

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

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

> **Example 6**
>
> Input:
>
> ```
> 1 \notation{binarysymbol}[highlight]
> 2    {\comp{\text{First: }}#1\comp{\text{; Second: }}#2}
> 3 $\binarysymbol[highlight]{a}{b}$
> ```
>
> Output:
>
> First: a; Second: b

.

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

Note that it is required that

1. the argument markers `#n` never occur inside a `\comp`, and

2. no semantic arguments may ever occur inside a notation.

Both criteria are not just required for technical reasons, but conceptionally meaningful:

The underlying principle is that the arguments to a semantic macro represent *arguments to the mathematical operation* represented by a symbol. For example, a semantic macro `\addition{a}{b}` taking two arguments would represent *the actual addition of (mathematical objects) a and b*. It should therefore be impossible for *a* or *b* to be part of a notation component of `\addition`.

Similarly, a semantic macro can not conceptually be part of the notation of `\addition`, since a semantic macro represents a *distinct mathematical concept* with *its own semantics*, whereas notations are syntactic representations of the very symbol to which the notation belongs.

If you want an argument to a semantic macro to be a purely syntactic parameter, then you are likely somewhat confused with respect to the distinction between the precise *syntax* and *semantics* of the symbol you are trying to declare (which happens quite often even to experienced sTeX users), and might want to give those another thought - quite likely, the macro you aim to implement does not actually represent a semantically meaningful mathematical concept, and you will want to use `\def` and similar native LaTeX macro definitions rather than semantic macros.

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

Example 7

Input:

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

Output:

1.: a; 2.: b

.

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

But one man's meat is another man's poison: it is very subjective what the "default notation" of an operator should be. Different communities have different practices. For instance, the complex unit is written as i in Mathematics and as j in electrical engineering.

So to allow modular specification and facilitate re-use of document fragments sTeX allows to re-set notation defaults.

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

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

Operator Notations

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

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

Example 8

Input:

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

Output:

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

.

$\hookleftarrow$M$\rightarrow$
$\longrightarrow$M$\rightarrow$ `\symbolname!` is translated to OMDoc/Mmt as `<OMS name="...?symbolname"/>`
$\rightsquigarrow$T$\rightsquigarrow$ directly.

3.3.3   Argument Modes

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

Mode-`b` Arguments

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

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

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

Example 9

Input:

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

Output:

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

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

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.

21

Mode-B Arguments

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

Example 12

Input:

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

Output:

$\forall_{x,y,z}.P$

.

3.3.4 Type and Definiens Components

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

> ↪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 $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$

22

.

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

Example 14

Input:

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

Output:

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

.

3.3.5 Precedences and Automated Bracketing

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

Example 15

Input:

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

Output:

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

.

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

Example 16

Input:

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

Output:

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

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

Example 17

Input:

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

Output:

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

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

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

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

Example 18

Input:

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

Output:

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

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

`\infprec`
`\neginfprec`

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

> ⚠ More precisely, each notation takes
> 1. One *operator precedence* and

2. one *argument precedence* for each argument.

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

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

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

In the example above:

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

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

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

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

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

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

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

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

3.3.6 Variables

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

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

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

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

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

Example 19

Input:

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

Output:

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

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

TODO: bind=forall/exists

3.3.7   Variable Sequences

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

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

This is best shown by example:

Example 20

Input:

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

Output:

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

26

·

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

TODO: more notations for invoking sequences.

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

Example 21

Input:

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

Output:

$a_1 + \ldots + a_n$

·

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

Example 22

Input:

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

Output:

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

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

Example 23

Input:

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

Output:

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

.

3.4 Module Inheritance and Structures

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

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

3.4.1 Multilinguality and Translations

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

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

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

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

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

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

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

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

```
6 \end{smodule}
```

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

3.4.2 Simple Inheritance and Namespaces

`\importmodule`
`\usemodule`

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

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

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

> Ideally, STEX would use arbitrary URIs for modules, with no forced relationships between the *logical* namespace of a module and the *physical* location of the file declaring the module – like MMT does things.
>
> Unfortunately, TEX only provides very restricted access to the file system, so we are forced to generate namespaces systematically in such a way that they reflect the physical location of the associated files, so that STEX can resolve them accordingly. Largely, users need not concern themselves with namespaces at all, but for completenesses sake, we describe how they are constructed:
>
> - If `\begin{smodule}{Foo}` occurs in a file `/path/to/file/Foo[.`$\langle lang \rangle$`].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[.`$\langle lang \rangle$`].tex`, the namespace is `file://path/to/file/bar`.
>
> In other words: outside of archives, the namespace corresponds to the file URI with the filename dropped iff it is equal to the module name, and ignoring the (optional) language suffix.
>
> If the current file is in an archive, the procedure is the same except that the initial segment of the file path up to the archive's `source`-folder is replaced by the archive's namespace URI.

> Conversely, here is how namespaces/URIs and file paths are computed in import statements, examplary `\importmodule`:
>
> - `\importmodule{Foo}` outside of an archive refers to module `Foo` in the current namespace. Consequently, `Foo` must have been declared earlier in the same document or, if not, in a file `Foo[.`$\langle lang \rangle$`].tex` in the same directory.
> - The same statement *within* an archive refers to either the module `Foo` declared earlier in the same document, or otherwise to the module `Foo` in the archive's top-level namespace. In the latter case, is has to be declared in a

file `Foo[.⟨lang⟩].tex` directly in the archive's `source`-folder.

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

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

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

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

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

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

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

3.4.3 The `mathstructure` Environment

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

- A *monoid* is a structure $\langle M, \circ, e\rangle$ with $\circ : M \times M \to M$ and $e \in M$ such that...

- A *topological space* is a structure $\langle X, \mathcal{T}\rangle$ where X is a set and $\mathcal{T}$ is a topology on X

- A *partial order* is a structure $\langle S, \leq\rangle$ where $\leq$ is a binary relation on S such that...

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

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

How this works is again best demonstrated by example:

Example 24

Input:

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

Output:

A monoid is...

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

Example 25

Input:

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

Output:

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

.

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

Example 26

Input:

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

Output:

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

.

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

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

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

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

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

The answer is that we can also instantiate a `mathstructure` with a *variable*. The syntax of `\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  $\varM{op}!:\funtype{\varM{universe},\varM{universe}}{\varM{universe}}$
7  and...
8
9  \varinstantiate{varMb}{universe = Int}{monoid}{M_2}
10
11 \noindent Let $\varMb!:=\mathstruct{\varMb{universe},\varMb{op}!,\varMb{unit}}$
12 a \symname{monoid} on $\Int$...
```

Output:

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

.

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

3.4.4 The `copymodule` Environment

TODO: explain

Given modules:

Example 28

Input:

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

Output:

.

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

Example 29

Input:

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

Output:

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

.

TODO: explain donotclone

3.4.5 The `interpretmodule` Environment

TODO: explain

Example 30

Input:

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

Output:

.

3.5 Primitive Symbols (The sTeX Metatheory)

TODO: metatheory documentation

Chapter 4

Using sTeX Symbols

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

4.1 \symref and its variants

`\symref`
`\symname`

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

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

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

Example 31

Input:

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

Output:

A natural number is...

.

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

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

Example 32

Input:

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

Output:

Natural numbers are...

.

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

4.2 Marking Up Text and On-the-Fly Notations

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

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

Example 33

Input:

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

Output:

The sum of n and m is...

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

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

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

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

> **Example 34**
> Input:
>
> ```
> 1  \addition!{Addition} is...
> ```
>
> Output:
>
> Addition is...

.

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

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

> **Example 35**
> Input:
>
> ```
> 1  \addition{\comp{adding}
> 2    \arg[2]{$\svar{k}$}
> 3    \arg*{$\addition{\svar{n}}{\svar{m}}$}} yields...
> ```
>
> Output:
>
> adding k yields...

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

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

Example 36

Input:

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

Output:

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

.

4.3   Referencing Symbols and Statements

TODO: references documentation

Chapter 5

sTeX Statements

5.1 Definitions, Theorems, Examples, Paragraphs

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

The corresponding environments for that are:

- sdefinition for definitions,

- sassertion for assertions, i.e. propositions that are declared to be *true*, such as theorems, lemmata, axioms,

- sexample for examples, and

- sparagraph for other semantic paragraphs, such as comments, remarks, conjectures, etc.

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

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

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

Example 37

Input:

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

Output:

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

.

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

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

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

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

Example 38
Input:

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

Output:

> A **monoid** is a structure $\langle U, \circ, e \rangle$ where $\circ : U \to U$ and $e \in U$ such that
>
> **Axiom 5.1.2** (Associativity). *$\circ$ is associative*
>
> **Axiom 5.1.3** (Unit). *$x \circ e = x$ for all $x \in U$*
>
>   An example for a monoid is...

.

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

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

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

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

5.2 Proofs

TODO

---

[2]Of course, ŞTEX 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, Mмт can. TODO: should

Chapter 6

Highlighting and Presentation Customizations

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Chapter 7

Additional Packages

TODO: tikzinput documentation

7.1 Modular Document Structuring

TODO: document-structure documentation

7.2 Slides and Course Notes

TODO: notesslides documentation

7.3 Homework, Problems and Exams

TODO: problem documentation
TODO: hwexam documentation

Part II

Documentation

Chapter 8

sTeX-Basics

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

8.1 Macros and Environments

\sTeX
\stex   Both print this sTeX logo.

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

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

8.1.1 HTML Annotations

\if@latexml   LATEX2e conditional for LATEXML

\latexml_if_p: ⋆
\latexml_if:*TF* ⋆   LATEX3 conditionals for LATEXML.

\stex_if_do_html_p: ⋆
\stex_if_do_html:*TF* ⋆   Whether to currently produce any HTML annotations (can be false in some advanced structuring environments, for example)

\stex_suppress_html:n   Temporarily disables HTML annotations in its argument code

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

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

Annotates the HTML generated by ⟨*content*⟩ with

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

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

8.1.2 Babel Languages

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

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

8.1.3 Auxiliary Methods

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

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

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

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

`\ignorespacesandpars`

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

Chapter 9

sTEX-MathHub

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

9.1 Macros and Environments

\stex_kpsewhich:n

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

9.1.1 Files, Paths, URIs

\stex_path_from_string:Nn

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

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

\stex_path_to_string:NN
\stex_path_to_string:N

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

\stex_path_canonicalize:N

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

\stex_path_if_absolute_p:N ⋆
\stex_path_if_absolute:NTF ⋆

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

\c_stex_pwd_seq
\c_stex_pwd_str
\c_stex_mainfile_seq
\c_stex_mainfile_str

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

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

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

9.1.2 MathHub Archives

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

1. The `mathhub` package option, or

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

3. the `MATHHUB` system variable, or

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

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

\l_stex_current_repository_prop

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

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

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

narr: the narration namespace (for document references),

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

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

\stex_set_current_repository:n

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

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

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

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

9.1.3 Using Content in Archives

`\mhpath ⋆`

`\mhpath{⟨archive-ID⟩}{⟨filename⟩}`

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

`\inputref`
`\mhinput`

`\inputref[⟨archive-ID⟩]{⟨filename⟩}`

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

Both also set `\ifinputref` to true.

`\addmhbibresource`

`\inputref[⟨archive-ID⟩]{⟨filename⟩}`

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

`\libinput`

`\libinput{⟨filename⟩}`

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

`\libusepackage`

`\libusepackage[⟨args⟩]{⟨filename⟩}`

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

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

`\mhgraphics`
`\cmhgraphics`

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

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

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

`\lstinputmhlisting`
`\clstinputmhlisting`

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

Chapter 10

sTEX-References

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

10.1 Macros and Environments

\STEXreftitle

`\STEXreftitle{⟨some title⟩}`

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

\stex_get_document_uri:

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

\l_stex_current_docns_str

Stores its result in `\l_stex_current_docns_str`

\stex_get_document_url:

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

\l_stex_current_docurl_str

Stores its result in `\l_stex_current_docurl_str`

10.1.1 Setting Reference Targets

\stex_ref_new_doc_target:n

`\stex_ref_new_doc_target:n{⟨id⟩}`

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

\stex_ref_new_sym_target:n

`\stex_ref_new_sym_target:n{⟨uri⟩}`

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

10.1.2 Using References

\sref   \sref[⟨*opt-args*⟩]{⟨*id*⟩}

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

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

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

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

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

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

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

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

Chapter 11

sTEX-Modules

This sub package contains code related to Modules

11.1 Macros and Environments

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

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

name The *name* of the module,

ns the *namespace* in field `ns`,

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

lang the module's *language*,

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

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

meta the metatheory of the module.

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

`\c_stex_module_<URI>_constants`

The names of all constants declared in the module

`\c_stex_module_<URI>_constants`

The full URIs of all modules imported in this module

`\l_stex_current_module_str` `\l_stex_current_module_str` always contains the URI of the current module (if exis-
tent).

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

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

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

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

`\stex_add_to_current_module:n`
`\STEXexport`

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

`\stex_add_constant_to_current_module:n`

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

`\stex_add_import_to_current_module:n`

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

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

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

\stex_modules_current_namespace:

Computes the current namespace as follows:

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

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

11.1.1 The `smodule` environment

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

\STEXModule   `\STEXModule {⟨fragment⟩}`

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

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

56

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

Chapter 12

sTEX-Module Inheritance

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

12.1 Macros and Environments

12.1.1 SMS Mode

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

`\g_stex_smsmode_allowedmacros_tl`

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

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

`\g_stex_smsmode_allowedmacros_escape_tl`

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

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

`\g_stex_smsmode_allowedenvs_seq`

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

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

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

Tests whether SMS mode is currently active.

\stex_file_in_smsmode:nn

\stex_in_smsmode:nn {⟨*filename*⟩} {⟨*code*⟩}

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

\stex_smsmode_do:

Starts gobbling tokens until one is encountered that is allowed in SMS mode.

12.1.2 Imports and Inheritance

\importmodule

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

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

\usemodule

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

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

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

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

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

1. If ⟨*archive-ID*⟩ is empty:

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

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

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

2. Otherwise:

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

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

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

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

`\l_stex_import_name_str`
`\l_stex_import_archive_str`   stores the result in these four variables.
`\l_stex_import_path_str`
`\l_stex_import_ns_str`

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

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

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

Chapter 13

sTeX-Symbols

Code related to symbol declarations and notations

13.1   Macros and Environments

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

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

- **name**: An (OMDoc) name. By default equal to ⟨*macroname*⟩.

- **type**: An (ideally semantic) term, representing a *type*. Not used by sTeX, but passed on to Mmt for semantic services.

- **def**: An (ideally semantic) term, representing a *definiens*. Not used by sTeX, but passed on to Mmt for semantic services.

- **local**: A boolean (by default false). If set, this declaration will not be added to the module content, i.e. importing the current module will not make this declaration available.

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

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

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

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

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

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

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

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

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

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

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

\stex_notation_do:nn `\stex_notation_do:nn{`⟨*URI*⟩`}{`⟨*notations*[+]⟩`}`

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

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

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

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

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

Chapter 14

sTEX-Terms

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

14.1 Macros and Environments

\STEXsymbol

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

\symref

\symref{⟨symbol⟩}{⟨text⟩}

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

\stex_invoke_symbol:n

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

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

_stex_term_math_oms:nnnn
_stex_term_math_oma:nnnn
_stex_term_math_omb:nnnn

⟨URI⟩⟨fragment⟩⟨precedence⟩⟨body⟩

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

_stex_term_math_arg:nnn

\stex_term_arg:nnn⟨int⟩⟨prec⟩⟨body⟩

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

_stex_term_math_assoc_arg:nnnn

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

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

`\infprec` `\neginfprec`	Maximal and minimal notation precedences.

`\dobrackets` `\dobrackets {⟨body⟩}`

Puts ⟨*body*⟩ in parentheses; scaled if in display mode unscaled otherwise. Uses the current SₜᴇX 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 SₜᴇX for automated bracketing (by default (and)) to ⟨*left*⟩ and ⟨*right*⟩.

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

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

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

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

`\comp{⟨args⟩}`

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

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

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

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

`\ellipses` TODO

Chapter 15

sTEX-Structural Features

Code related to structural features

15.1 Macros and Environments

15.1.1 Structures

mathstructure TODO

Chapter 16

sTeX-Statements

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

16.1 Macros and Environments

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

Chapter 17

sTeX-Proofs: Structural Markup for Proofs

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

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

Contents

17.1 Introduction

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

SₜₑX is a version of TₑX/LᴬTₑX that allows to markup TₑX/LᴬTₑX documents semantically without leaving the document format, essentially turning TₑX/LᴬTₑX into a document format for mathematical knowledge management (MKM).

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

Example 1: A very explicit proof, marked up semantically

We will go over the general intuition by way of our running example (see Figure 1 for the source and Figure 2 for the formatted result).[4]

---

[4]EᴅNᴏᴛᴇ: talk a bit more about proofs and their structure,… maybe copy from OMDoc spec.

17.2 The User Interface

17.2.1 Package Options

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

17.2.2 Proofs and Proof steps

sproof
The `proof` environment is the main container for proofs. It takes an optional `KeyVal` argument that allows to specify the `id` (identifier) and `for` (for which assertion is this a proof) keys. The regular argument of the `proof` environment contains an introductory comment, that may be used to announce the proof style. The `proof` environment contains a sequence of `\step`, `proofcomment`, and `pfcases` environments that are used to markup the proof steps. The `proof` environment has a variant `Proof`, which does not use the proof end marker. This is convenient, if a proof ends in a case distinction, which brings it's own proof end marker with it. The `Proof` environment is a variant of `proof` that does not mark the end of a proof with a little box; presumably, since one of the subproofs already has one and then a box supplied by the outer proof would generate an otherwise empty line. The `\spfidea` macro allows to give a one-paragraph description of the proof idea.

sProof

\spfidea

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

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

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

17.2.3 Justifications

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

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

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

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

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

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

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

1.3. $n > 1$:

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

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

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

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

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

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

□

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

17.2.4 Proof Structure

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

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

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

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

17.2.5 Proof End Markers

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

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

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

17.2.6 Configuration of the Presentation

Finally, we provide configuration hooks in Figure 1 for the keywords in proofs. These are mainly intended for package authors building on `statements`, e.g. for multi-language
support.[5] The proof step labels can be customized via the `\pstlabelstyle` macro:

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

Figure 1: Configuration Hooks for Semantic Proof Markup

`\pstlabelstyle`

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

---

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

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

17.3   Limitations

In this section we document known limitations. If you want to help alleviate them, please feel free to contact the package author. Some of them are currently discussed in the sTeX issue tracker at [sTeX].

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

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

Chapter 18

sTEX-Metatheory

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

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

18.1 Symbols

Part III

Extensions

Chapter 19

Tikzinput

19.1 Macros and Environments

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

Chapter 20

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

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

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

20.1 Introduction

STEX is a version of TeX/LaTeX that allows to markup TeX/LaTeX documents semantically without leaving the document format, essentially turning TeX/LaTeX into a document format for mathematical knowledge management (MKM). The package supports direct translation to the OMDoc format [Koh06]

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

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

source and the formatter does the copying during document formatting/presentation.[6]

20.2 The User Interface

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

20.2.1 Package and Class Options

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

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

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

20.2.2 Document Structure

document \
\documentkeys \
id

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

sfragment

The structure of the document is given by the `omgroup` environment just like in OMDOC. In the LaTeX route, the `omgroup` environment is flexibly mapped to sectioning commands, inducing the proper sectioning level from the nesting of `omgroup` environments. Correspondingly, the `omgroup` environment takes an optional key/value argument for metadata followed by a regular argument for the (section) title of the omgroup.

id \
creators \
contributors \
short \
loadmodules

group. The optional metadata argument has the keys `id` for an identifier, `creators` and `contributors` for the Dublin Core metadata [DCM03]; see [**Kohlhase:dcm:git**] for details of the format. The `short` allows to give a short title for the generated section. If the title contains semantic macros, they need to be protected by `\protect`, and we need to give the `loadmodules` key it needs no value. For instance we would have

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

---

[6]EDNOTE: integrate with latexml's XMRef in the Math mode.

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

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

blindfragment

- The outer one groups the introductory parts of the book (which we assume to have a sectioning hierarchy topping at the part level). This `blindomgroup` makes sure that the introductory remarks become a "chapter" instead of a "part".

- Th inner one groups the frontmatter[4] and makes the preface of the book a section-level construct. Note that here the `display=flow` on the `omgroup` environment prevents numbering as is traditional for prefaces.

```
\begin{document}
\begin{blindfragment}
\begin{blindfragment}
\begin{frontmatter}
\maketitle\newpage
\begin{sfragment}[display=flow]{Preface}
... <<preface>> ...
\end{sfragment}
\clearpage\setcounter{tocdepth}{4}\tableofcontents\clearpage
\end{frontmatter}
\end{blindfragment}
... <<introductory remarks>> ...
\end{blindfragment}
\begin{sfragment}{Introduction}
... <<intro>> ...
\end{sfragment}
... <<more chapters>> ...
\bibliographystyle{alpha}\bibliography{kwarc}
\end{document}
```
Example 3: A typical Document Structure of a Book

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

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

---

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

20.2.3 Ignoring Inputs

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

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

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

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

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

20.2.4 Structure Sharing

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

The `\STRlabel` macro has a variant `\STRsemantics`, where the label argument is optional, and which takes a third argument, which is ignored in LaTeX. This allows to specify the meaning of the content (whatever that may mean) in cases, where the source document is not formatted for presentation, but is transformed into some content markup
EdN:7
format.[7]

20.2.5 Global Variables

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

---

[7]EDNOTE: document LMID und LMXREf here if we decide to keep them.

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

20.2.6 Colors

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

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

20.3   Limitations

In this section we document known limitations. If you want to help alleviate them, please feel free to contact the package author. Some of them are currently discussed in the sTeX GitHub repository [sTeX].

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

Chapter 21

NotesSlides – Slides and Course Notes

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

21.1   Introduction

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

21.2   The User Interface

The `notesslides` class takes the notion of a slide frame from Till Tantau's excellent `beamer` class and adapts its notion of frames for use in the ᴛᴇX 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.

21.2.1   Package Options

EdN:8   The `notesslides` class takes a variety of class options:[8]

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

82

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

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

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

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

21.2.2 Notes and Slides

frame
note

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

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

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

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

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

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

Example 4: A typical Course Notes File

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

\ifnotes

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

---

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

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

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

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

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

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

nparagraph
There are some environments that tend to occur at the top-level of `note` environments. We make convenience versions of these: e.g. the `nparagraph` environment is just an `sparagraph` inside a `note` environment (but looks nicer in the source, since it avoids

nfragment
one level of source indenting). Similarly, we have the `nomgroup`, `ndefinition`, `nexample`,

ndefinition

nexample
`nsproof`, and `nassertion` environments.

nsproof

nassertion
21.2.3   Header and Footer Lines of the Slides

The default logo provided by the `notesslides` package is the sTeX logo it can be cus-

\setslidelogo
tomized using `\setslidelogo{⟨logo name⟩}`.

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

\setsource
of this package. `\setsource{⟨name⟩}` can change the writer's name. For licensing, we use the Creative Commons Attribuition-ShareAlike license by default to strengthen the public domain. If package `hyperref` is loaded, then we can attach a hyperlink to the

\setlicensing
license logo. `\setlicensing[⟨url⟩]{⟨logo name⟩}` is used for customization, where ⟨url⟩ is optional.

21.2.4   Frame Images

Sometimes, we want to integrate slides as images after all – e.g. because we already have a PowerPoint presentation, to which we want to add sTeXnotes. In this case we

\frameimage
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

EdN:9
label that can be referenced like a regular `beamer` frame.[9]

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

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

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

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

---

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

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

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

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

21.2.5 Colors and Highlighting

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

21.2.6 Front Matter, Titles, etc.

21.2.7 Excursions

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

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

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

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

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

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

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

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

21.2.8 Miscellaneous

21.3 Limitations

In this section we document known limitations. If you want to help alleviate them, please feel free to contact the package author. Some of them are currently discussed in the sTEXGitHub repository [sTeX].

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

Chapter 22

`problem.sty`: An Infrastructure for formatting Problems

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

22.1 Introduction

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

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

22.2 The User Interface

22.2.1 Package Options

solutions
notes
hints
gnotes
pts
min
boxed
test
mh
showmeta

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

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

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

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

---

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

22.2.2   Problems and Solutions

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

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

Example 5: A marked up Problem

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

| **Problem 22.2.1 (Fitting Elefants)** |
| How many Elefants can you fit into a Volkswagen beetle? |
| **Hint:** Think positively, this is simple! |
| **Note:**Justify your answer |
| **Solution:** Four, two in the front seats, and two in the back. |

Example 6: The Formatted Problem from Figure 5

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

may arise in grading.

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

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

22.2.3   Multiple Choice Blocks

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

- `T` for true answers, `F` for false ones,

- `Ttext` the verdict for true answers, `Ftext` for false ones, and

- `feedback` for a short feedback text given to the student.

See Figure **??** for an example

22.2.4   Including Problems

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

22.2.5   Reporting Metadata

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

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

22.3   Limitations

In this section we document known limitations. If you want to help alleviate them,
please feel free to contact the package author. Some of them are currently discussed in
the sTeXGitHub repository [sTeX].

1. none reported yet

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

Problem 22.2.2 (Functions)

What is the keyword to introduce a function definition in python?

☐ def

☐ function

☐ fun

☐ public static void

Problem 22.2.3 (Functions)

What is the keyword to introduce a function definition in python?

☐ def
  (true)

☐ function
  (false) *(that is for C and C++)*

☐ fun
  (false) *(that is for Standard ML)*

☐ public static void
  (false) *(that is for Java)*

Example 7: A Problem with a multiple choice block

Chapter 23

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

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

Contents

23.1   Introduction

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

23.2   The User Interface

23.2.1   Package and Class Options

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

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

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

23.2.2   Assignments

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

23.2.3   Typesetting Exams

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

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

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

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

23.2.4 Including Assignments

\inputassignment

number
title
type
given
due

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

23.3 Limitations

In this section we document known limitations. If you want to help alleviate them, please feel free to contact the package author. Some of them are currently discussed in the sTeXGitHub repository [sTeX].

1. none reported yet.

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

Name: Matriculation Number:

320101 General Computer Science (Fall 2010)

2022-04-23

You have one hour (sharp) for the test;

Write the solutions to the sheet.

The estimated time for solving this exam is 58 minutes, leaving you 2 minutes for revising your exam.

You can reach 30 points if you solve all problems. You will only need 27 points for a perfect score, i.e. 3 points are bonus points.

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

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

	To be used for grading, do not write here											
prob.	22.2.1	22.2.2	22.2.3	1.1	2.1	2.2	2.3	3.1	3.2	3.3	Sum	grade
total				4	4	6	6	4	4	2	30	
reached												

good luck

Example 8: A generated test heading.

Part IV

Implementation

Chapter 24

sTEX
-Basics Implementation

24.1 The sTEXDocument Class

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

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

24.2 Preliminaries

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

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

29
30  \bool_if_exist:NF \c_stex_document_class_bool {
31    \bool_set_false:N \c_stex_document_class_bool
32    \RequirePackage{standalone}
33  }

34
35  \message{^^J
36    *****************************^^J
37    *~This~is~sTeX~version~3.1.0~*^^J
38    *****************************^^J
39  ^^J}

40
41  %\RequirePackage{morewrites}
42  %\RequirePackage{amsmath}

43
```

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

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

(End definition for `\stex` *and* `\sTeX`*. These functions are documented on page 47.)*

24.3   Messages and logging

```
60  ⟨@@=stex_log⟩
```

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

97

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

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

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

Redirecting messages:

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

24.4 HTML Annotations

```
95 ⟨@@=stex_annotate⟩
```

`\l_stex_html_arg_tl`
`\c_stex_html_emptyarg_tl`

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

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

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

`\_stex_html_checkempty:n`

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

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

`\stex_if_do_html_p:`
`\stex_if_do_html:TF`

Whether to (locally) produce HTML output

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

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

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

\stex_suppress_html:n   Whether to (locally) produce HTML output

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

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

\stex_annotate:nnn
\stex_annotate_invisible:n
\stex_annotate_invisible:nnn
We define four macros for introducing attributes in the HTML output. The definitions depend on the "backend" used (LaTeXML, R_USTEX, pdflatex).

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

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

(*End definition for* `\stex_annotate:nnn`, `\stex_annotate_invisible:n`, *and* `\stex_annotate_invisible:nnn`. *These functions are documented on page* *48.*)

24.5   Babel Languages

```
132 ⟨@@=stex_language⟩
```

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

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

99

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

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

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

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

24.6   Persistence

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

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

24.7   Auxiliary Methods

\stex_deactivate_macro:Nn

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

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

\stex_reactivate_macro:N

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

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

\ignorespacesandpars

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

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

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

\MMTrule

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

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

Chapter 25

SₜₑX
-MathHub Implementation

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

Warnings and error messages

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

25.1   Generic Path Handling

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

\stex_path_from_string:Nn

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

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

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

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

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

(End definition for `\stex_path_to_string:NN` *and* `\stex_path_to_string:N`*. These functions are documented on page 49.)*

`\c__stex_path_dot_str`
`\c__stex_path_up_str`

. and .., respectively.

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

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

`\stex_path_canonicalize:N`

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

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

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

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

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

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

25.2   PWD and kpsewhich

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

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

We determine the PWD

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

(*End definition for* \c_stex_pwd_seq *and* \c_stex_pwd_str*. These variables are documented on page* *49.*)

25.3 File Hooks and Tracking

```
379 ⟨@@=stex_files⟩
```

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

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

keeps track of file changes

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

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

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

(*End definition for* \c_stex_mainfile_seq *and* \c_stex_mainfile_str*. These variables are documented on page* *49.*)

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

(*End definition for* \g_stex_currentfile_seq*. This variable is documented on page* *50.*)

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

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

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

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

Hooks for the current file:

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

25.4 MathHub Repositories

```
412 ⟨@@=stex_mathhub⟩
```

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

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

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

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

(*End definition for* \mathhub, \c_stex_mathhub_seq, *and* \c_stex_mathhub_str. *These variables are documented on page* 50.)

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

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

(*End definition for* __stex_mathhub_do_manifest:n.)

\l__stex_mathhub_manifest_file_seq

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

(*End definition for* \l__stex_mathhub_manifest_file_seq.)

_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`:

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

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

\c_stex_mathhub_manifest_ior   File variable used for `MANIFEST`-files

```
512 \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:

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

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

(End definition for __stex_mathhub_parse_manifest:n.*)*

<div style="color:red">\stex_set_current_repository:n</div>

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

(End definition for \stex_set_current_repository:n. *This function is documented on page 50.)*

<div style="color:red">\stex_require_repository:n</div>

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

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

(*End definition for* \stex_require_repository:n. *This function is documented on page 50.*)

\l_stex_current_repository_prop Current MathHub repository

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

(*End definition for* \l_stex_current_repository_prop. *This variable is documented on page 50.*)

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

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

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

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

25.5   Using Content in Archives

\mhpath

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

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

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

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

(*End definition for* `\inputref` *and* `\mhinput`*. These functions are documented on page 51.*)

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

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

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

```
703     }
704
705     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}
706     \IfFileExists{ \l_tmpa_str }{
707       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
708     }{}
709
710     \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
711       \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
712     }{
713       \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
714         \input{ ##1 }
715       }
716     }
717 }
```

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

\libusepackage

```
718 \NewDocumentCommand \libusepackage {O{} m} {
719   \prop_if_exist:NF \l_stex_current_repository_prop {
720     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
721   }
722   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
723     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
724   }
725   \seq_clear:N \l__stex_mathhub_libinput_files_seq
726   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
727   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
728
729   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
730     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #2}
731     \IfFileExists{ \l_tmpa_str.sty }{
732       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
733     }{}
734     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
735     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
736   }
737
738   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2}
739   \IfFileExists{ \l_tmpa_str.sty }{
740     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
741   }{}
742
743   \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
744     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
745   }{
746     \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
747       \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
748         \usepackage[#1]{ ##1 }
749       }
750     }{
751       \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
752     }
```

114

```
753       }
754   }
```

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

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

(*End definition for* `\mhgraphics` *and* `\cmhgraphics`. *These functions are documented on page 51.*)

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

(*End definition for* `\lstinputmhlisting` *and* `\clstinputmhlisting`. *These functions are documented on page 51.*)

115

Chapter 26

sTEX
-References Implementation

```
774 ⟨∗package⟩
775
776 %%%%%%%%%%%%   references.dtx   %%%%%%%%%%%%
777
778 ⟨@@=stex_refs⟩
```

Warnings and error messages

```
779
```

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

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

\STEXreftitle

```
787 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
788
789 \NewDocumentCommand \STEXreftitle { m } {
790   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
791 }
```

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

26.1   Document URIs and URLs

\l_stex_current_docns_str

```
792 \str_new:N \l_stex_current_docns_str
```

(*End definition for* \l_stex_current_docns_str. *This variable is documented on page 52.*)

116

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

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

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

(End definition for \l_stex_current_docurl_str*. This variable is documented on page 52.)*

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

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

(End definition for \stex_get_document_url:. This function is documented on page 52.)

26.2   Setting Reference Targets

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

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

(*End definition for* \stex_ref_new_doc_target:n. *This function is documented on page 52.*)

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

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

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

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

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

119

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

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

26.3   Using References

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

\sref   Optional arguments:

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

The actual macro:

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

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

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

<code>\srefsym</code>

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

121

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

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

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

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

```
1071 ⟨/package⟩
```

Chapter 27

sTeX
-Modules Implementation

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

Warnings and error messages

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

`\l_stex_current_module_str` The current module:

```
1094 \str_new:N \l_stex_current_module_str
```

(*End definition for* `\l_stex_current_module_str`*. This variable is documented on page 55.*)

`\l_stex_all_modules_seq` Stores all available modules

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

(*End definition for* `\l_stex_all_modules_seq`*. This variable is documented on page 55.*)

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

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

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

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

Only allowed within modules:

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

(*End definition for* `\stex_add_to_current_module:n` *and* `\STEXexport`*. These functions are documented on page 55.*)

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

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

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

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

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

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

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

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

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

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

```
1182
```

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

\stex_modules_current_namespace: Computes the current namespace based on the current MathHub repository (if existent) and the current file.

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

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

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

27.1 The `smodule` environment

`smodule` arguments:

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

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

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

```
1265    \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1266    \str_set:Nx \l_stex_module_name_str { #2 }
1267    \__stex_modules_args:n { #1 }
```
First, we set up the name and namespace of the module.
Are we in a nested module?
```
1268    \stex_if_in_module:TF {
1269      % Nested module
1270      \prop_get:cnN {c_stex_module_\l_stex_current_module_str _prop}
1271        { ns } \l_stex_module_ns_str
1272      \str_set:Nx \l_stex_module_name_str {
1273        \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
1274          { name } / \l_stex_module_name_str
1275      }
1276      \str_if_empty:NT \l_stex_module_lang_str {
1277        \str_set:Nx \l_stex_module_lang_str {
1278          \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
1279            { lang }
1280        }
1281      }
1282    }{
1283      % not nested:
1284      \str_if_empty:NT \l_stex_module_ns_str {
1285        \stex_modules_current_namespace:
1286        \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1287          / {\l_stex_module_ns_str}
1288        \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1289        \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1290          \str_set:Nx \l_stex_module_ns_str {
1291            \stex_path_to_string:N \l_tmpa_seq
1292          }
1293        }
1294      }
1295    }
```
Next, we determine the language of the module:
```
1296    \str_if_empty:NT \l_stex_module_lang_str {
1297      \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1298      \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1299      \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1300      \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
1301        \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
1302          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
1303        }
1304      }
1305      \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1306      \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
1307        \seq_pop_right:NN \l_tmpa_seq \l_stex_module_lang_str
1308        \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1309          inferred~from~file~name}
1310      }
1311    }
1312
1313    \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
```

```
1314      \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1315        \l_tmpa_str {
1316          \ltx@ifpackageloaded{babel}{
1317            \exp_args:Nx \selectlanguage { \l_tmpa_str }
1318          }{}
1319        } {
1320          \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
1321        }
1322    }}
```

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

```
1323      \str_if_empty:NTF \l_stex_module_sig_str {
1324        \exp_args:Nnx \prop_gset_from_keyval:cn {
1325          c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1326        } {
1327          name      = \l_stex_module_name_str ,
1328          ns        = \l_stex_module_ns_str ,
1329          file      = \exp_not:o { \g_stex_currentfile_seq } ,
1330          lang      = \l_stex_module_lang_str ,
1331          sig       = \l_stex_module_sig_str ,
1332          deprecate = \l_stex_module_deprecate_str ,
1333          meta      = \l_stex_module_meta_str
1334        }
1335        \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1336        \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1337        \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _copymodules}
1338        \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1339        \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
```

We load the metatheory:

```
1340        \str_if_empty:NT \l_stex_module_meta_str {
1341          \str_set:Nx \l_stex_module_meta_str {
1342            \c_stex_metatheory_ns_str ? Metatheory
1343          }
1344        }
1345        \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1346          \bool_set_true:N \l_stex_in_meta_bool
1347          \exp_args:Nx \stex_add_to_current_module:n {
1348            \bool_set_true:N \l_stex_in_meta_bool
1349            \stex_activate_module:n {\l_stex_module_meta_str}
1350            \bool_set_false:N \l_stex_in_meta_bool
1351          }
1352          \stex_activate_module:n {\l_stex_module_meta_str}
1353          \bool_set_false:N \l_stex_in_meta_bool
1354        }
1355      }{
1356        \str_if_empty:NT \l_stex_module_lang_str {
1357          \msg_error:nnxx{stex}{error/siglanguage}{
1358            \l_stex_module_ns_str?\l_stex_module_name_str
1359          }{\l_stex_module_sig_str}
1360        }
1361        \stex_debug:nn{modules}{Signature~\l_stex_module_sig_str~for~\l_stex_module_ns_str?\l_st
1362        \stex_if_module_exists:nTF{\l_stex_module_ns_str?\l_stex_module_name_str}{
```

```
1363        \stex_debug:nn{modules}{(already exists)}
1364      }{
1365        \stex_debug:nn{modules}{(needs loading)}
1366        \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1367        \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1368        \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1369        \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1370        \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1371        \str_set:Nx \l_tmpa_str {
1372          \stex_path_to_string:N \l_tmpa_seq /
1373          \l_tmpa_str . \l_stex_module_sig_str .tex
1374        }
1375        \IfFileExists \l_tmpa_str {
1376          \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1377            \str_clear:N \l_stex_current_module_str
1378            \seq_clear:N \l_stex_all_modules_seq
1379            \stex_debug:nn{modules}{Loading~signature}
1380          }
1381        }{
1382          \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1383        }
1384      }
1385      \stex_if_smsmode:F {
1386        \stex_activate_module:n {
1387          \l_stex_module_ns_str ? \l_stex_module_name_str
1388        }
1389      }
1390      \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1391    }
1392    \str_if_empty:NF \l_stex_module_deprecate_str {
1393      \msg_warning:nnxx{stex}{warning/deprecated}{
1394        Module~\l_stex_current_module_str
1395      }{
1396        \l_stex_module_deprecate_str
1397      }
1398    }
1399    \seq_put_right:Nx \l_stex_all_modules_seq {
1400      \l_stex_module_ns_str ? \l_stex_module_name_str
1401    }
1402    \tl_clear:c{l__stex_modules_aftergroup_\l_stex_module_ns_str ? \l_stex_module_name_str _tl
1403 }
```

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

smodule   The `module` environment.

`\__stex_modules_begin_module:`   implements `\begin{smodule}`

```
1404 \cs_new_protected:Nn \__stex_modules_begin_module: {
1405    \stex_reactivate_macro:N \STEXexport
1406    \stex_reactivate_macro:N \importmodule
1407    \stex_reactivate_macro:N \symdecl
1408    \stex_reactivate_macro:N \notation
1409    \stex_reactivate_macro:N \symdef
1410
```

```
1411    \stex_debug:nn{modules}{
1412      New~module:\\
1413      Namespace:~\l_stex_module_ns_str\\
1414      Name:~\l_stex_module_name_str\\
1415      Language:~\l_stex_module_lang_str\\
1416      Signature:~\l_stex_module_sig_str\\
1417      Metatheory:~\l_stex_module_meta_str\\
1418      File:~\stex_path_to_string:N \g_stex_currentfile_seq
1419    }
1420
1421    \stex_if_do_html:T{
1422      \begin{stex_annotate_env} {theory} {
1423        \l_stex_module_ns_str ? \l_stex_module_name_str
1424      }
1425
1426      \stex_annotate_invisible:nnn{header}{} {
1427        \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1428        \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1429        \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1430          \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1431        }
1432        \str_if_empty:NF \smoduletype {
1433          \stex_annotate:nnn{type}{\smoduletype}{}
1434        }
1435      }
1436    }
1437    % TODO: Inherit metatheory for nested modules?
1438  }
1439  \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}`

```
1440  \cs_new_protected:Nn \__stex_modules_end_module: {
1441    \stex_debug:nn{modules}{Closing~module~\prop_item:cn {c_stex_module_\l_stex_current_module
1442    \_stex_reset_up_to_module:n \l_stex_current_module_str
1443    \stex_if_smsmode:T {
1444      \stex_persist:x {
1445        \prop_set_from_keyval:cn{c_stex_module_\l_stex_current_module_str _prop}{
1446          \exp_after:wN \prop_to_keyval:N \csname c_stex_module_\l_stex_current_module_str _pr
1447        }
1448        \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _constants}{
1449          \seq_use:cn{c_stex_module_\l_stex_current_module_str _constants},
1450        }
1451        \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _imports}{
1452          \seq_use:cn{c_stex_module_\l_stex_current_module_str _imports},
1453        }
1454        \tl_set:cn {c_stex_module_\l_stex_current_module_str _code}
1455      }
1456      \exp_after:wN \let \exp_after:wN \l_tmpa_tl \csname c_stex_module_\l_stex_current_module
1457      \exp_after:wN \stex_persist:n \exp_after:wN { \exp_after:wN { \l_tmpa_tl } }
1458    }
1459  }
```

(*End definition for* __stex_modules_end_module:.)

The core environment

```
1460 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1461 \NewDocumentEnvironment { smodule } { O{} m } {
1462   \stex_module_setup:nn{#1}{#2}
1463   \par
1464   \stex_if_smsmode:F{
1465     \tl_clear:N \l_tmpa_tl
1466     \clist_map_inline:Nn \smoduletype {
1467       \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1468         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1469       }
1470     }
1471     \tl_if_empty:NTF \l_tmpa_tl {
1472       \__stex_modules_smodule_start:
1473     }{
1474       \l_tmpa_tl
1475     }
1476   }
1477   \__stex_modules_begin_module:
1478   \str_if_empty:NF \smoduleid {
1479     \stex_ref_new_doc_target:n \smoduleid
1480   }
1481   \stex_smsmode_do:
1482 } {
1483   \__stex_modules_end_module:
1484   \stex_if_smsmode:F {
1485     \end{stex_annotate_env}
1486     \clist_set:No \l_tmpa_clist \smoduletype
1487     \tl_clear:N \l_tmpa_tl
1488     \clist_map_inline:Nn \l_tmpa_clist {
1489       \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1490         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1491       }
1492     }
1493     \tl_if_empty:NTF \l_tmpa_tl {
1494       \__stex_modules_smodule_end:
1495     }{
1496       \l_tmpa_tl
1497     }
1498   }
1499 }
```

\stexpatchmodule

```
1500 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1501 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1502
1503 \newcommand\stexpatchmodule[3][] {
1504   \str_set:Nx \l_tmpa_str{ #1 }
1505   \str_if_empty:NTF \l_tmpa_str {
1506     \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1507     \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1508   }{
```

```
1509        \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1510        \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1511     }
1512  }
```

(*End definition for* \stexpatchmodule. *This function is documented on page* *56*.)

27.2   Invoking modules

```
1513  \NewDocumentCommand \STEXModule { m } {
1514    \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1515    \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1516    \tl_set:Nn \l_tmpa_tl {
1517      \msg_error:nnx{stex}{error/unknownmodule}{#1}
1518    }
1519    \seq_map_inline:Nn \l_stex_all_modules_seq {
1520      \str_set:Nn \l_tmpb_str { ##1 }
1521      \str_if_eq:eeT { \l_tmpa_str } {
1522        \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1523      } {
1524        \seq_map_break:n {
1525          \tl_set:Nn \l_tmpa_tl {
1526            \stex_invoke_module:n { ##1 }
1527          }
1528        }
1529      }
1530    }
1531    \l_tmpa_tl
1532  }
1533
1534  \cs_new_protected:Nn \stex_invoke_module:n {
1535    \stex_debug:nn{modules}{Invoking~module~#1}
1536    \peek_charcode_remove:NTF ! {
1537      \__stex_modules_invoke_uri:nN { #1 }
1538    } {
1539      \peek_charcode_remove:NTF ? {
1540        \__stex_modules_invoke_symbol:nn { #1 }
1541      } {
1542        \msg_error:nnx{stex}{error/syntax}{
1543          ?~or~!~expected~after~
1544          \c_backslash_str STEXModule{#1}
1545        }
1546      }
1547    }
1548  }
1549
1550  \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1551    \str_set:Nn #2 { #1 }
1552  }
1553
1554  \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1555    \stex_invoke_symbol:n{#1?#2}
```

```
1556  }
```

(*End definition for* `\STEXModule` *and* `\stex_invoke_module:n`*. These functions are documented on page* *56.*)

\stex_activate_module:n

```
1557  \bool_new:N \l_stex_in_meta_bool
1558  \bool_set_false:N \l_stex_in_meta_bool
1559  \cs_new_protected:Nn \stex_activate_module:n {
1560    \stex_debug:nn{modules}{Activating~module~#1}
1561    \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1562      \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1563      \use:c{ c_stex_module_#1_code }
1564    }
1565  }
```

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

```
1566  ⟨/package⟩
```

Chapter 28

sTeX
-Module Inheritance
Implementation

```
1568
1569 %%%%%%%%%%%%   inheritance.dtx   %%%%%%%%%%%%
1570
```

28.1   SMS Mode

```
1571 ⟨@@=stex_smsmode⟩
```

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

```
1572 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1573 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1574 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1575
1576 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1577   \makeatletter
1578   \makeatother
1579   \ExplSyntaxOn
1580   \ExplSyntaxOff
1581   \rustexBREAK
1582 }
1583
1584 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1585   \symdef
1586   \importmodule
1587   \notation
1588   \symdecl
1589   \STEXexport
1590   \inlineass
1591   \inlinedef
1592   \inlineex
1593   \endinput
1594   \setnotation
```

```
1595     \copynotation
1596     \assign
1597     \renamedecl
1598     \donotcopy
1599     \instantiate
1600  }
1601
1602  \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1603    \tl_to_str:n {
1604      smodule,
1605      copymodule,
1606      interpretmodule,
1607      sdefinition,
1608      sexample,
1609      sassertion,
1610      sparagraph,
1611      mathstructure
1612    }
1613  }
```

(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 58.)*

```
1614  \bool_new:N \g__stex_smsmode_bool
1615  \bool_set_false:N \g__stex_smsmode_bool
1616  \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1617    \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1618  }
```

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

`\__stex_smsmode_in_smsmode:nn`

```
1619  \cs_new_protected:Nn \__stex_smsmode_in_smsmode:nn { \stex_suppress_html:n {
1620    \vbox_set:Nn \l_tmpa_box {
1621      \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1622      \bool_gset_true:N \g__stex_smsmode_bool
1623      #2
1624      \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1625    }
1626    \box_clear:N \l_tmpa_box
1627  } }
```

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

`\stex_file_in_smsmode:nn`

```
1628  \quark_new:N \q__stex_smsmode_break
1629
1630  \NewDocumentCommand \__stex_smsmode_importmodule: { O{} m } {
1631    \seq_gput_right:Nn \l__stex_smsmode_importmodules_seq {{#1}{#2}}
1632    \stex_smsmode_do:
1633  }
1634
1635  \cs_new_protected:Nn \__stex_smsmode_module:nn {
1636    \__stex_modules_args:n{#1}
```

```
1637    \stex_if_in_module:F {
1638      \str_if_empty:NF \l_stex_module_sig_str {
1639        \stex_modules_current_namespace:
1640        \str_set:Nx \l_stex_module_name_str { #2 }
1641        \stex_if_module_exists:nF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1642          \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1643          \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1644          \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1645          \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1646          \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1647          \str_set:Nx \l_tmpa_str {
1648            \stex_path_to_string:N \l_tmpa_seq /
1649            \l_tmpa_str . \l_stex_module_sig_str .tex
1650          }
1651          \IfFileExists \l_tmpa_str {
1652            \exp_args:NNx \seq_gput_right:Nn \l__stex_smsmode_sigmodules_seq \l_tmpa_str
1653          }{
1654            \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1655          }
1656        }
1657      }
1658    }
1659 }
1660
1661 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1662    \stex_filestack_push:n{#1}
1663    \seq_gclear:N \l__stex_smsmode_importmodules_seq
1664    \seq_gclear:N \l__stex_smsmode_sigmodules_seq
1665    % ----- new ---------------------------
1666    \__stex_smsmode_in_smsmode:nn{#1}{
1667      \let\importmodule\__stex_smsmode_importmodule:
1668      \let\stex_module_setup:nn\__stex_smsmode_module:nn
1669      \let\__stex_modules_begin_module:\relax
1670      \let\__stex_modules_end_module:\relax
1671      \seq_clear:N \g_stex_smsmode_allowedenvs_seq
1672      \exp_args:NNx \seq_put_right:Nn \g_stex_smsmode_allowedenvs_seq {\tl_to_str:n{smodule}}
1673      \tl_clear:N \g_stex_smsmode_allowedmacros_tl
1674      \tl_clear:N \g_stex_smsmode_allowedmacros_escape_tl
1675      \tl_put_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\importmodule}
1676      \everyeof{\q__stex_smsmode_break\noexpand}
1677      \expandafter\expandafter\expandafter
1678      \stex_smsmode_do:
1679      \csname @ @ input\endcsname "#1"\relax
1680
1681      \seq_map_inline:Nn \l__stex_smsmode_sigmodules_seq {
1682        \stex_filestack_push:n{##1}
1683        \expandafter\expandafter\expandafter
1684        \stex_smsmode_do:
1685        \csname @ @ input\endcsname "##1"\relax
1686        \stex_filestack_pop:
1687      }
1688    }
1689    % ----- new ---------------------------
1690    \__stex_smsmode_in_smsmode:nn{#1} {
```

```
1691        #2
1692        % ----- new ----------------------------
1693        \begingroup
1694        %\stex_debug:nn{smsmode}{Here:~\seq_use:Nn\l__stex_smsmode_importmodules_seq, }
1695        \seq_map_inline:Nn \l__stex_smsmode_importmodules_seq {
1696          \stex_import_module_uri:nn ##1
1697          \stex_import_require_module:nnnn
1698            \l_stex_import_ns_str
1699            \l_stex_import_archive_str
1700            \l_stex_import_path_str
1701            \l_stex_import_name_str
1702        }
1703        \endgroup
1704        \stex_debug:nn{smsmode}{Actually~loading~file~#1}
1705        % ----- new ----------------------------
1706        \everyeof{\q__stex_smsmode_break\noexpand}
1707        \expandafter\expandafter\expandafter
1708        \stex_smsmode_do:
1709        \csname @ @ input\endcsname "#1"\relax
1710      }
1711    \stex_filestack_pop:
1712 }
```

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

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

```
1713 \cs_new_protected:Npn \stex_smsmode_do: {
1714   \stex_if_smsmode:T {
1715     \__stex_smsmode_do:w
1716   }
1717 }
1718 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1719   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 }}{
1720     \expandafter\if\expandafter\relax\noexpand#1
1721       \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1722     \else\expandafter\__stex_smsmode_do:w\fi
1723   }{
1724     \__stex_smsmode_do:w %#1
1725   }
1726 }
1727 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1728   \cs_if_eq:NNF #1 \q__stex_smsmode_break {
1729     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1730       #1\__stex_smsmode_do:w
1731     }{
1732       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1733         #1
1734       }{
1735         \cs_if_eq:NNTF \begin #1 {
1736           \__stex_smsmode_check_begin:n
1737         }{
1738           \cs_if_eq:NNTF \end #1 {
1739             \__stex_smsmode_check_end:n
```

138

```
1740                    }{
1741                      \__stex_smsmode_do:w
1742                    }
1743                  }
1744                }
1745              }
1746            }
1747          }
1748
1749          \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1750            \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1751              \begin{#1}
1752            }{
1753              \__stex_smsmode_do:w
1754            }
1755          }
1756          \cs_new_protected:Nn \__stex_smsmode_check_end:n {
1757            \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1758              \end{#1}\__stex_smsmode_do:w
1759            }{
1760              \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1761            }
1762          }
```

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

28.2   Inheritance

1763   ⟨@@=stex_importmodule⟩

```
1764          \cs_new_protected:Nn \stex_import_module_uri:nn {
1765            \str_set:Nx \l_stex_import_archive_str { #1 }
1766            \str_set:Nn \l_stex_import_path_str { #2 }
1767
1768            \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1769            \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1770            \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1771
1772            \stex_modules_current_namespace:
1773            \bool_lazy_all:nTF {
1774              {\str_if_empty_p:N \l_stex_import_archive_str}
1775              {\str_if_empty_p:N \l_stex_import_path_str}
1776              {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1777            }{
1778              \str_set_eq:NN \l_stex_import_path_str \l_stex_module_subpath_str
1779              \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1780            }{
1781              \str_if_empty:NT \l_stex_import_archive_str {
1782                \prop_if_exist:NT \l_stex_current_repository_prop {
1783                  \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1784                }
1785              }
1786              \str_if_empty:NTF \l_stex_import_archive_str {
```

139

```
1787                \str_if_empty:NF \l_stex_import_path_str {
1788                  \str_set:Nx \l_stex_import_ns_str {
1789                    \l_stex_module_ns_str / \l_stex_import_path_str
1790                  }
1791                }
1792              }{
1793                \stex_require_repository:n \l_stex_import_archive_str
1794                \prop_get:cnN { c_stex_mathhub_\l_stex_import_archive_str _manifest_prop } { ns }
1795                  \l_stex_import_ns_str
1796                \str_if_empty:NF \l_stex_import_path_str {
1797                  \str_set:Nx \l_stex_import_ns_str {
1798                    \l_stex_import_ns_str / \l_stex_import_path_str
1799                  }
1800                }
1801              }
1802          }
1803      }
```

(*End definition for* `\stex_import_module_uri:nn`. *This function is documented on page 60.*)

`\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`.

```
1804 \str_new:N \l_stex_import_name_str
1805 \str_new:N \l_stex_import_archive_str
1806 \str_new:N \l_stex_import_path_str
1807 \str_new:N \l_stex_import_ns_str
```

(*End definition for* `\l_stex_import_name_str` *and others. These variables are documented on page 60.*)

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

```
1808 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1809    \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1810
1811      %\stex_debug:nn{requiremodule}{Here:\\~~1:~#1\\~~2:~#2\\~~3:~#3\\~~4:~#4}
1812
1813      \exp_args:NNxx \seq_set_split:Nnn \l_tmpa_seq {\tl_to_str:n{/}} {#4}
1814      \seq_get_left:NN \l_tmpa_seq \l_tmpc_str
1815
1816      %\stex_debug:nn{requiremodule}{Top~module:\l_tmpc_str}
1817
1818      % archive
1819      \str_set:Nx \l_tmpa_str { #2 }
1820      \str_if_empty:NTF \l_tmpa_str {
1821        \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1822      } {
1823        \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1824        \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1825        \seq_put_right:Nn \l_tmpa_seq { source }
1826      }
1827
1828      % path
1829      \str_set:Nx \l_tmpb_str { #3 }
1830      \str_if_empty:NTF \l_tmpb_str {
1831        \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / \l_tmpc_str }
1832
```

```
1833        \ltx@ifpackageloaded{babel} {
1834          \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1835            { \languagename } \l_tmpb_str {
1836              \msg_error:nnx{stex}{error/unknownlanguage}{\languagename}
1837            }
1838        } {
1839          \str_clear:N \l_tmpb_str
1840        }
1841
1842        %\stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1843        \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1844          \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1845        }{
1846          %\stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1847          \IfFileExists{ \l_tmpa_str.tex }{
1848            \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1849          }{
1850            % try english as default
1851            %\stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1852            \IfFileExists{ \l_tmpa_str.en.tex }{
1853              \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1854            }{
1855              \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1856            }
1857          }
1858        }
1859
1860      } {
1861        \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1862        \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1863
1864        \ltx@ifpackageloaded{babel} {
1865          \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1866            { \languagename } \l_tmpb_str {
1867              \msg_error:nnx{stex}{error/unknownlanguage}{\languagename}
1868            }
1869        } {
1870          \str_clear:N \l_tmpb_str
1871        }
1872
1873        \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1874
1875        %\stex_debug:nn{modules}{Checking~\l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex}
1876        \IfFileExists{ \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }{
1877          \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.te
1878        }{
1879          %\stex_debug:nn{modules}{Checking~\l_tmpa_str/\l_tmpc_str.tex}
1880          \IfFileExists{ \l_tmpa_str/\l_tmpc_str.tex }{
1881            \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.tex }
1882          }{
1883            % try english as default
1884            %\stex_debug:nn{modules}{Checking~\l_tmpa_str/\l_tmpc_str.en.tex}
1885            \IfFileExists{ \l_tmpa_str/\l_tmpc_str.en.tex }{
1886              \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.en.tex }
```

```
1887          }{
1888            %\stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1889            \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1890              \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1891            }{
1892              %\stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1893              \IfFileExists{ \l_tmpa_str.tex }{
1894                \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1895              }{
1896                % try english as default
1897                %\stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1898                \IfFileExists{ \l_tmpa_str.en.tex }{
1899                  \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1900                }{
1901                  \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1902                }
1903              }
1904            }
1905          }
1906        }
1907      }
1908    }
1909
1910    \str_if_eq:eeF{\g__stex_importmodule_file_str}{\seq_use:Nn \g_stex_currentfile_seq /}{
1911      \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
1912        \seq_clear:N \l_stex_all_modules_seq
1913        \str_clear:N \l_stex_current_module_str
1914        \str_set:Nx \l_tmpb_str { #2 }
1915        \str_if_empty:NF \l_tmpb_str {
1916          \stex_set_current_repository:n { #2 }
1917        }
1918        \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1919      }
1920
1921      \stex_if_module_exists:nF { #1 ? #4 } {
1922        \msg_error:nnx{stex}{error/unknownmodule}{
1923          #1?#4~(in~file~\g__stex_importmodule_file_str)
1924        }
1925      }
1926    }
1927
1928  }
1929  \stex_activate_module:n { #1 ? #4 }
1930 }
```

(*End definition for* \stex_import_require_module:nnnn. *This function is documented on page 60.*)

\importmodule

```
1931 \NewDocumentCommand \importmodule { O{} m } {
1932   \stex_import_module_uri:nn { #1 } { #2 }
1933   \stex_debug:nn{modules}{Importing~module:~
1934     \l_stex_import_ns_str ? \l_stex_import_name_str
1935   }
1936   \stex_import_require_module:nnnn
```

```
1937    { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1938    { \l_stex_import_path_str } { \l_stex_import_name_str }
1939    \stex_if_smsmode:F {
1940      \stex_annotate_invisible:nnn
1941        {import} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1942    }
1943    \exp_args:Nx \stex_add_to_current_module:n {
1944      \stex_import_require_module:nnnn
1945      { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1946      { \l_stex_import_path_str } { \l_stex_import_name_str }
1947    }
1948    \exp_args:Nx \stex_add_import_to_current_module:n {
1949      \l_stex_import_ns_str ? \l_stex_import_name_str
1950    }
1951    \stex_smsmode_do:
1952    \ignorespacesandpars
1953  }
1954  \stex_deactivate_macro:Nn \importmodule {module~environments}
```

(End definition for \importmodule. This function is documented on page 59.)

```
1955  \NewDocumentCommand \usemodule { O{} m } {
1956    \stex_if_smsmode:F {
1957      \stex_import_module_uri:nn { #1 } { #2 }
1958      \stex_import_require_module:nnnn
1959      { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1960      { \l_stex_import_path_str } { \l_stex_import_name_str }
1961      \stex_annotate_invisible:nnn
1962        {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1963    }
1964    \stex_smsmode_do:
1965    \ignorespacesandpars
1966  }
```

(End definition for \usemodule. This function is documented on page 59.)

```
1967  \cs_new_protected:Nn \stex_csl_to_imports:Nn {
1968    \tl_if_empty:nF{#2}{
1969      \clist_set:Nn \l_tmpa_clist {#2}
1970      \clist_map_inline:Nn \l_tmpa_clist {
1971        \tl_if_head_eq_charcode:nNTF {##1}[{
1972          #1 ##1
1973        }{
1974          #1{##1}
1975        }
1976      }
1977    }
1978  }
1979  \cs_generate_variant:Nn \stex_csl_to_imports:Nn {No}
1980
1981
1982  ⟨/package⟩
```

143

Chapter 29

sTEX
-Symbols Implementation

⟨∗package⟩

1984

1985 %%%%%%%%%%%%   symbols.dtx   %%%%%%%%%%%%

1986

Warnings and error messages

```
1987 \msg_new:nnn{stex}{error/wrongargs}{
1988   args~value~in~symbol~declaration~for~#1~
1989   needs~to~be~i,~a,~b~or~B,~but~#2~given
1990 }
1991 \msg_new:nnn{stex}{error/unknownsymbol}{
1992   No~symbol~#1~found!
1993 }
1994 \msg_new:nnn{stex}{error/seqlength}{
1995   Expected~#1~arguments;~got~#2!
1996 }
1997 \msg_new:nnn{stex}{error/unknownnotation}{
1998   Unknown~notation~#1~for~#2!
1999 }
```

29.1   Symbol Declarations

2000 ⟨@@=stex_symdecl⟩

\stex_all_symbols:n   Map over all available symbols

```
2001 \cs_new_protected:Nn \stex_all_symbols:n {
2002   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
2003   \seq_map_inline:Nn \l_stex_all_modules_seq {
2004     \seq_map_inline:cn{c_stex_module_##1_constants}{
2005       \__stex_symdecl_all_symbols_cs{##1?####1}
2006     }
2007   }
2008 }
```

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

```
2009  \NewDocumentCommand \STEXsymbol { m } {
2010    \stex_get_symbol:n { #1 }
2011    \exp_args:No
2012    \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
2013  }
```

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

symdecl arguments:

```
2014  \keys_define:nn { stex / symdecl } {
2015    name        .str_set_x:N  = \l_stex_symdecl_name_str ,
2016    local       .bool_set:N   = \l_stex_symdecl_local_bool ,
2017    args        .str_set_x:N  = \l_stex_symdecl_args_str ,
2018    type        .tl_set:N     = \l_stex_symdecl_type_tl ,
2019    deprecate   .str_set_x:N  = \l_stex_symdecl_deprecate_str ,
2020    align       .str_set:N    = \l_stex_symdecl_align_str , % TODO(?)
2021    gfc         .str_set:N    = \l_stex_symdecl_gfc_str , % TODO(?)
2022    specializes .str_set:N    = \l_stex_symdecl_specializes_str , % TODO(?)
2023    def         .tl_set:N     = \l_stex_symdecl_definiens_tl ,
2024    reorder     .str_set_x:N  = \l_stex_symdecl_reorder_str ,
2025    assoc       .choices:nn   =
2026        {bin,binl,binr,pre,conj,pwconj}
2027        {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}
2028  }
2029
2030  \bool_new:N \l_stex_symdecl_make_macro_bool
2031
2032  \cs_new_protected:Nn \__stex_symdecl_args:n {
2033    \str_clear:N \l_stex_symdecl_name_str
2034    \str_clear:N \l_stex_symdecl_args_str
2035    \str_clear:N \l_stex_symdecl_deprecate_str
2036    \str_clear:N \l_stex_symdecl_reorder_str
2037    \str_clear:N \l_stex_symdecl_assoctype_str
2038    \bool_set_false:N \l_stex_symdecl_local_bool
2039    \tl_clear:N \l_stex_symdecl_type_tl
2040    \tl_clear:N \l_stex_symdecl_definiens_tl
2041
2042    \keys_set:nn { stex / symdecl } { #1 }
2043  }
```

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

```
2044
2045  \NewDocumentCommand \symdecl { s m O{} } {
2046    \__stex_symdecl_args:n { #3 }
2047    \IfBooleanTF #1 {
2048      \bool_set_false:N \l_stex_symdecl_make_macro_bool
2049    } {
2050      \bool_set_true:N \l_stex_symdecl_make_macro_bool
2051    }
2052    \stex_symdecl_do:n { #2 }
2053    \stex_smsmode_do:
2054  }
```

```
2055
2056  \cs_new_protected:Nn \stex_symdecl_do:nn {
2057    \__stex_symdecl_args:n{#1}
2058    \bool_set_false:N \l_stex_symdecl_make_macro_bool
2059    \stex_symdecl_do:n{#2}
2060  }
2061
2062  \stex_deactivate_macro:Nn \symdecl {module~environments}
```

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

\stex_symdecl_do:n

```
2063  \cs_new_protected:Nn \stex_symdecl_do:n {
2064    \stex_if_in_module:F {
2065      % TODO throw error? some default namespace?
2066    }
2067
2068    \str_if_empty:NT \l_stex_symdecl_name_str {
2069      \str_set:Nx \l_stex_symdecl_name_str { #1 }
2070    }
2071
2072    \prop_if_exist:cT { l_stex_symdecl_
2073        \l_stex_current_module_str ?
2074        \l_stex_symdecl_name_str
2075      _prop
2076    }{
2077      % TODO throw error (beware of circular dependencies)
2078    }
2079
2080    \prop_clear:N \l_tmpa_prop
2081    \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
2082    \seq_clear:N \l_tmpa_seq
2083    \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
2084    \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
2085
2086    \str_if_empty:NT \l_stex_symdecl_deprecate_str {
2087      \str_if_empty:NF \l_stex_module_deprecate_str {
2088        \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
2089      }
2090    }
2091    \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
2092
2093    \exp_args:No \stex_add_constant_to_current_module:n {
2094      \l_stex_symdecl_name_str
2095    }
2096
2097    % arity/args
2098    \int_zero:N \l_tmpb_int
2099
2100    \bool_set_true:N \l_tmpa_bool
2101    \str_map_inline:Nn \l_stex_symdecl_args_str {
2102      \token_case_meaning:NnF ##1 {
2103        0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2104        {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
```

146

```
2105        {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2106        {\tl_to_str:n a} {
2107          \bool_set_false:N \l_tmpa_bool
2108          \int_incr:N \l_tmpb_int
2109        }
2110        {\tl_to_str:n B} {
2111          \bool_set_false:N \l_tmpa_bool
2112          \int_incr:N \l_tmpb_int
2113        }
2114      }{
2115        \msg_error:nnxx{stex}{error/wrongargs}{
2116          \l_stex_current_module_str ?
2117          \l_stex_symdecl_name_str
2118        }{##1}
2119      }
2120    }
2121    \bool_if:NTF \l_tmpa_bool {
2122      % possibly numeric
2123      \str_if_empty:NTF \l_stex_symdecl_args_str {
2124        \prop_put:Nnn \l_tmpa_prop { args } {}
2125        \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2126      }{
2127        \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
2128        \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2129        \str_clear:N \l_tmpa_str
2130        \int_step_inline:nn \l_tmpa_int {
2131          \str_put_right:Nn \l_tmpa_str i
2132        }
2133        \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
2134      }
2135    } {
2136      \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
2137      \prop_put:Nnx \l_tmpa_prop { arity }
2138        { \str_count:N \l_stex_symdecl_args_str }
2139    }
2140    \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2141
2142    \tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
2143      \prop_put:Nnx \l_tmpa_prop { defined }{ false }
2144    }{
2145      \prop_put:Nnx \l_tmpa_prop { defined }{ true }
2146    }
2147
2148    % semantic macro
2149
2150    \bool_if:NT \l_stex_symdecl_make_macro_bool {
2151      \exp_args:Nx \stex_do_up_to_module:n {
2152        \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2153          \l_stex_current_module_str ? \l_stex_symdecl_name_str
2154        }}
2155      }
2156    }
2157
2158    \stex_debug:nn{symbols}{New~symbol:~
```

```
2159    \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2160    Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2161    Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
2162    Definiens:~\exp_not:o {\l_stex_symdecl_definiens_tl}
2163    }
2164
2165    % circular dependencies require this:
2166    \stex_if_do_html:T {
2167        \stex_annotate_invisible:nnn {symdecl} {
2168            \l_stex_current_module_str ? \l_stex_symdecl_name_str
2169        } {
2170            \tl_if_empty:NF \l_stex_symdecl_type_tl {
2171                \stex_annotate_invisible:nnn{type}{}{$\l_stex_symdecl_type_tl$}
2172            }
2173            \stex_annotate_invisible:nnn{args}{}{
2174                \prop_item:Nn \l_tmpa_prop { args }
2175            }
2176            \stex_annotate_invisible:nnn{macroname}{#1}{}
2177            \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2178                \stex_annotate_invisible:nnn{definiens}{}
2179                    {$\l_stex_symdecl_definiens_tl$}
2180            }
2181            \str_if_empty:NF \l_stex_symdecl_assoctype_str {
2182                \stex_annotate_invisible:nnn{assoctype}{\l_stex_symdecl_assoctype_str}{}
2183            }
2184            \str_if_empty:NF \l_stex_symdecl_reorder_str {
2185                \stex_annotate_invisible:nnn{reorderargs}{\l_stex_symdecl_reorder_str}{}
2186            }
2187        }
2188    }
2189    \prop_if_exist:cF {
2190        l_stex_symdecl_
2191        \l_stex_current_module_str ? \l_stex_symdecl_name_str
2192        _prop
2193    } {
2194        \bool_if:NTF \l_stex_symdecl_local_bool \stex_do_up_to_module:x \stex_execute_in_module:
2195            \__stex_symdecl_restore_symbol:nnnnnnn
2196            {\l_stex_symdecl_name_str}
2197            { \prop_item:Nn \l_tmpa_prop {args} }
2198            { \prop_item:Nn \l_tmpa_prop {arity} }
2199            { \prop_item:Nn \l_tmpa_prop {assocs} }
2200            { \prop_item:Nn \l_tmpa_prop {defined} }
2201            {\bool_if:NT \l_stex_symdecl_make_macro_bool {#1} }
2202            {\l_stex_current_module_str}
2203        }
2204    }
2205 }
2206 \cs_new_protected:Nn \__stex_symdecl_restore_symbol:nnnnnnn {
2207    \prop_clear:N \l_tmpa_prop
2208    \prop_put:Nnn \l_tmpa_prop { module } { #7 }
2209    \prop_put:Nnn \l_tmpa_prop { name } { #1}
2210    \prop_put:Nnn \l_tmpa_prop { args } {#2}
2211    \prop_put:Nnn \l_tmpa_prop { arity } { #3 }
2212    \prop_put:Nnn \l_tmpa_prop { assocs } { #4 }
```

```
2213    \prop_put:Nnn \l_tmpa_prop { defined } { #5 }
2214    \tl_if_empty:nF{#6}{
2215      \tl_set:cx{#6}{\stex_invoke_symbol:n{\detokenize{#7 ? #1}}}
2216    }
2217    \prop_set_eq:cN{l_stex_symdecl_ \detokenize{#7 ? #1} _prop}\l_tmpa_prop
2218    \seq_clear:c{l_stex_symdecl_ \detokenize{#7 ? #1} _notations}
2219  }
```

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

\stex_get_symbol:n

```
2220  \str_new:N \l_stex_get_symbol_uri_str
2221
2222  \cs_new_protected:Nn \stex_get_symbol:n {
2223    \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2224      \tl_set:Nn \l_tmpa_tl { #1 }
2225      \__stex_symdecl_get_symbol_from_cs:
2226    }{
2227      % argument is a string
2228      % is it a command name?
2229      \cs_if_exist:cTF { #1 }{
2230        \cs_set_eq:Nc \l_tmpa_tl { #1 }
2231        \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2232        \str_if_empty:NTF \l_tmpa_str {
2233          \exp_args:Nx \cs_if_eq:NNTF {
2234            \tl_head:N \l_tmpa_tl
2235          } \stex_invoke_symbol:n {
2236            \__stex_symdecl_get_symbol_from_cs:
2237          }{
2238            \__stex_symdecl_get_symbol_from_string:n { #1 }
2239          }
2240        } {
2241          \__stex_symdecl_get_symbol_from_string:n { #1 }
2242        }
2243      }{
2244        % argument is not a command name
2245        \__stex_symdecl_get_symbol_from_string:n { #1 }
2246        % \l_stex_all_symbols_seq
2247      }
2248    }
2249    \str_if_eq:eeF {
2250      \prop_item:cn {
2251        l_stex_symdecl_\l_stex_get_symbol_uri_str _prop
2252      }{ deprecate }
2253    }{}{
2254      \msg_warning:nnxx{stex}{warning/deprecated}{
2255        Symbol~\l_stex_get_symbol_uri_str
2256      }{
2257        \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{ deprecate }
2258      }
2259    }
2260  }
2261
2262  \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
```

```
2263    \tl_set:Nn \l_tmpa_tl {
2264      \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2265    }
2266    \str_set:Nn \l_tmpa_str { #1 }
2267
2268    %\int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2269
2270    \str_if_in:NnTF \l_tmpa_str ? {
2271      \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
2272      \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
2273      \str_set:Nx \l_tmpb_str {\seq_use:Nn \l_tmpa_seq ?}
2274    }{
2275      \str_clear:N \l_tmpb_str
2276    }
2277    \str_if_empty:NTF \l_tmpb_str {
2278      \seq_map_inline:Nn \l_stex_all_modules_seq {
2279        \seq_map_inline:cn{c_stex_module_##1_constants}{
2280          \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2281            \seq_map_break:n{\seq_map_break:n{
2282              \tl_set:Nn \l_tmpa_tl {
2283                \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2284              }
2285            }}
2286          }
2287        }
2288      }
2289    }{
2290      \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpb_str }
2291      \seq_map_inline:Nn \l_stex_all_modules_seq {
2292        \str_if_eq:eeT{ \l_tmpb_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2293          \seq_map_inline:cn{c_stex_module_##1_constants}{
2294            \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2295              \seq_map_break:n{\seq_map_break:n{
2296                \tl_set:Nn \l_tmpa_tl {
2297                  \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2298                }
2299              }}
2300            }
2301          }
2302        }
2303      }
2304    }
2305
2306    \l_tmpa_tl
2307 }
2308
2309 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
2310    \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2311      { \tl_tail:N \l_tmpa_tl }
2312    \tl_if_single:NTF \l_tmpa_tl {
2313      \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2314        \exp_after:wN \str_set:Nn \exp_after:wN
2315          \l_stex_get_symbol_uri_str \l_tmpa_tl
2316      }{
```

```
2317        % TODO
2318        % tail is not a single group
2319      }
2320    }{
2321      % TODO
2322      % tail is not a single group
2323    }
2324 }
```

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

29.2   Notations

```
2325  ⟨@@=stex_notation⟩
```

notation arguments:
```
2326 \keys_define:nn { stex / notation } {
2327 %  lang     .tl_set_x:N  = \l__stex_notation_lang_str ,
2328    variant .tl_set_x:N  = \l__stex_notation_variant_str ,
2329    prec    .str_set_x:N = \l__stex_notation_prec_str ,
2330    op      .tl_set:N    = \l__stex_notation_op_tl ,
2331    primary .bool_set:N  = \l__stex_notation_primary_bool ,
2332    primary .default:n   = {true} ,
2333    unknown .code:n      = \str_set:Nx
2334        \l__stex_notation_variant_str \l_keys_key_str
2335 }
2336
2337 \cs_new_protected:Nn \_stex_notation_args:n {
2338 %  \str_clear:N \l__stex_notation_lang_str
2339    \str_clear:N \l__stex_notation_variant_str
2340    \str_clear:N \l__stex_notation_prec_str
2341    \tl_clear:N \l__stex_notation_op_tl
2342    \bool_set_false:N \l__stex_notation_primary_bool
2343
2344    \keys_set:nn { stex / notation } { #1 }
2345 }
```

`\notation`
```
2346 \NewDocumentCommand \notation { s m O{}} {
2347    \_stex_notation_args:n { #3 }
2348    \tl_clear:N \l_stex_symdecl_definiens_tl
2349    \stex_get_symbol:n { #2 }
2350    \tl_set:Nn \l_stex_notation_after_do_tl {
2351      \__stex_notation_final:
2352      \IfBooleanTF#1{
2353        \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2354      }{}
2355      \stex_smsmode_do:\ignorespacesandpars
2356    }
2357    \stex_notation_do:nnnnn
2358      { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } }
2359      { \prop_item:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } }
2360      { \l__stex_notation_variant_str }
2361      { \l__stex_notation_prec_str}
```

```
2362   }
2363   \stex_deactivate_macro:Nn \notation {module~environments}
```

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

\stex_notation_do:nnnnn

```
2364   \seq_new:N \l__stex_notation_precedences_seq
2365   \tl_new:N \l__stex_notation_opprec_tl
2366   \int_new:N \l__stex_notation_currarg_int
2367   \tl_new:N \stex_symbol_after_invokation_tl
2368
2369   \cs_new_protected:Nn \stex_notation_do:nnnnn {
2370     \let\l_stex_current_symbol_str\relax
2371     \seq_clear:N \l__stex_notation_precedences_seq
2372     \tl_clear:N \l__stex_notation_opprec_tl
2373     \str_set:Nx \l__stex_notation_args_str { #1 }
2374     \str_set:Nx \l__stex_notation_arity_str { #2 }
2375     \str_set:Nx \l__stex_notation_suffix_str { #3 }
2376     \str_set:Nx \l__stex_notation_prec_str { #4 }
2377
2378     % precedences
2379     \str_if_empty:NTF \l__stex_notation_prec_str {
2380       \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2381         \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2382       }{
2383         \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2384       }
2385     } {
2386       \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2387         \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2388         \int_step_inline:nn { \l__stex_notation_arity_str } {
2389           \exp_args:NNo
2390           \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2391         }
2392       }{
2393         \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2394         \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2395           \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2396           \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2397             \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2398               \l_tmpa_seq {\tl_to_str:n{x} } { \l_tmpa_str }
2399             \seq_map_inline:Nn \l_tmpa_seq {
2400               \seq_put_right:Nn \l_tmpb_seq { ##1 }
2401             }
2402           }
2403         }{
2404           \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2405             \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2406           }{
2407             \tl_set:No \l__stex_notation_opprec_tl { 0 }
2408           }
2409         }
2410       }
2411     }
```

```
2412
2413    \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2414    \int_step_inline:nn { \l__stex_notation_arity_str } {
2415      \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
2416        \exp_args:NNo
2417        \seq_put_right:No \l__stex_notation_precedences_seq {
2418          \l__stex_notation_opprec_tl
2419        }
2420      }
2421    }
2422    \tl_clear:N \l_stex_notation_dummyargs_tl
2423
2424    \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2425      \exp_args:NNe
2426      \cs_set:Npn \l_stex_notation_macrocode_cs {
2427        \_stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2428          { \l__stex_notation_suffix_str }
2429          { \l__stex_notation_opprec_tl }
2430          { \exp_not:n { #5 } }
2431      }
2432      \l_stex_notation_after_do_tl
2433    }{
2434      \str_if_in:NnTF \l__stex_notation_args_str b {
2435        \exp_args:Nne \use:nn
2436        {
2437        \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2438        \cs_set:Npn \l__stex_notation_arity_str } { {
2439          \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2440            { \l__stex_notation_suffix_str }
2441            { \l__stex_notation_opprec_tl }
2442            { \exp_not:n { #5 } }
2443      }}
2444    }{
2445      \str_if_in:NnTF \l__stex_notation_args_str B {
2446        \exp_args:Nne \use:nn
2447        {
2448        \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2449        \cs_set:Npn \l__stex_notation_arity_str } { {
2450          \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2451            { \l__stex_notation_suffix_str }
2452            { \l__stex_notation_opprec_tl }
2453            { \exp_not:n { #5 } }
2454        } }
2455      }{
2456        \exp_args:Nne \use:nn
2457        {
2458        \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2459        \cs_set:Npn \l__stex_notation_arity_str } { {
2460          \_stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2461            { \l__stex_notation_suffix_str }
2462            { \l__stex_notation_opprec_tl }
2463            { \exp_not:n { #5 } }
2464        } }
2465      }
```

```
2466        }
2467
2468        \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2469        \int_zero:N \l__stex_notation_currarg_int
2470        \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2471        \__stex_notation_arguments:
2472     }
2473 }
```

(*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

```
2474 \cs_new_protected:Nn \__stex_notation_arguments: {
2475    \int_incr:N \l__stex_notation_currarg_int
2476    \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2477       \l_stex_notation_after_do_tl
2478    }{
2479       \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2480       \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2481       \str_if_eq:VnTF \l_tmpa_str a {
2482          \__stex_notation_argument_assoc:nn{a}
2483       }{
2484          \str_if_eq:VnTF \l_tmpa_str B {
2485             \__stex_notation_argument_assoc:nn{B}
2486          }{
2487             \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpb_str
2488             \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2489                { \_stex_term_math_arg:nnn
2490                   { \l_tmpa_str\int_use:N \l__stex_notation_currarg_int }
2491                   { \l_tmpb_str }
2492                   { ####\int_use:N \l__stex_notation_currarg_int }
2493                }
2494             }
2495             \__stex_notation_arguments:
2496          }
2497       }
2498    }
2499 }
```

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

`\__stex_notation_argument_assoc:nn`

```
2500 \cs_new_protected:Nn \__stex_notation_argument_assoc:nn {
2501
2502    \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2503       {\l__stex_notation_arity_str}{
2504       #2
2505    }
2506    \int_zero:N \l_tmpa_int
2507    \tl_clear:N \l_tmpa_tl
2508    \str_map_inline:Nn \l__stex_notation_args_str {
2509       \int_incr:N \l_tmpa_int
2510       \tl_put_right:Nx \l_tmpa_tl {
2511          \str_if_eq:nnTF {##1}{a}{ {} }{
```

154

```
2512        \str_if_eq:nnTF {##1}{B}{ {} }{
2513          {\_stex_term_arg:nn{##1\int_use:N \l_tmpa_int}{############### \int_use:N \l_tmpa
2514        }
2515      }
2516    }
2517  }
2518  \exp_after:wN\exp_after:wN\exp_after:wN \def
2519  \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2520  \exp_after:wN\exp_after:wN\exp_after:wN ##
2521  \exp_after:wN\exp_after:wN\exp_after:wN 1
2522  \exp_after:wN\exp_after:wN\exp_after:wN ##
2523  \exp_after:wN\exp_after:wN\exp_after:wN 2
2524  \exp_after:wN\exp_after:wN\exp_after:wN {
2525    \exp_after:wN \exp_after:wN \exp_after:wN
2526    \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2527      \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2528    }
2529  }
2530
2531  \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2532  \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
2533    \_stex_term_math_assoc_arg:nnnn
2534      { #1\int_use:N \l__stex_notation_currarg_int }
2535      { \l_tmpa_str }
2536      { ####\int_use:N \l__stex_notation_currarg_int }
2537      { \l_tmpa_cs {####1} {####2} }
2538  } }
2539  \__stex_notation_arguments:
2540 }
```

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

`\__stex_notation_final:` Called after processing all notation arguments

```
2541 \cs_new_protected:Nn \__stex_notation_restore_notation:nnnnn {
2542   \cs_generate_from_arg_count:cNnn{stex_notation_\detokenize{#1} \c_hash_str \detokenize{#2}
2543   \cs_set_nopar:Npn {#3}{#4}
2544   \tl_if_empty:nF {#5}{
2545     \tl_set:cn{stex_op_notation_\detokenize{#1} \c_hash_str \detokenize{#2}_cs}{ \comp{ #5 }
2546   }
2547   \seq_if_exist:cT { l_stex_symdecl_\detokenize{#1} _notations }{
2548     \seq_put_right:cx { l_stex_symdecl_\detokenize{#1} _notations } { \detokenize{#2} }
2549   }
2550 }
2551
2552 \cs_new_protected:Nn \__stex_notation_final: {
2553
2554   \stex_execute_in_module:x {
2555     \__stex_notation_restore_notation:nnnnn
2556       {\l_stex_get_symbol_uri_str}
2557       {\l__stex_notation_suffix_str}
2558       {\l__stex_notation_arity_str}
2559       {
2560         \exp_after:wN \exp_after:wN \exp_after:wN
2561         \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
```

```
2562          { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2563        }
2564        {\exp_args:No \exp_not:n \l__stex_notation_op_tl }
2565    }
2566
2567    \stex_debug:nn{symbols}{
2568      Notation~\l__stex_notation_suffix_str
2569      ~for~\l_stex_get_symbol_uri_str^^J
2570      Operator~precedence:~\l__stex_notation_opprec_tl^^J
2571      Argument~precedences:~
2572        \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2573      Notation: \cs_meaning:c {
2574        stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2575        \l__stex_notation_suffix_str
2576        _cs
2577      }
2578    }
2579      % HTML annotations
2580    \stex_if_do_html:T {
2581      \stex_annotate_invisible:nnn { notation }
2582      { \l_stex_get_symbol_uri_str } {
2583        \stex_annotate_invisible:nnn { notationfragment }
2584          { \l__stex_notation_suffix_str }{}
2585        \stex_annotate_invisible:nnn { precedence }
2586          { \l__stex_notation_prec_str }{}
2587
2588        \int_zero:N \l_tmpa_int
2589        \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2590        \tl_clear:N \l_tmpa_tl
2591        \int_step_inline:nn { \l__stex_notation_arity_str }{
2592          \int_incr:N \l_tmpa_int
2593          \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2594          \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_rem
2595          \str_if_eq:VnTF \l_tmpb_str a {
2596            \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2597              \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2598              \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2599            } }
2600          }{
2601            \str_if_eq:VnTF \l_tmpb_str B {
2602              \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2603                \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2604                \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2605              } }
2606            }{
2607              \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2608                \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
2609              } }
2610            }
2611          }
2612        }
2613        \stex_annotate_invisible:nnn { notationcomp }{}{
2614          \str_set:Nx \l_stex_current_symbol_str {\l_stex_get_symbol_uri_str }
2615          $ \exp_args:Nno \use:nn { \use:c {
```

```
2616              stex_notation_ \l_stex_current_symbol_str
2617              \c_hash_str \l__stex_notation_suffix_str _cs
2618          } } { \l_tmpa_tl } $
2619        }
2620      }
2621    }
2622 }
```

(*End definition for* __stex_notation_final:.)

\setnotation

```
2623 \keys_define:nn { stex / setnotation } {
2624 % lang    .tl_set_x:N  = \l__stex_notation_lang_str ,
2625   variant .tl_set_x:N  = \l__stex_notation_variant_str ,
2626   unknown .code:n       = \str_set:Nx
2627        \l__stex_notation_variant_str \l_keys_key_str
2628 }
2629
2630 \cs_new_protected:Nn \_stex_setnotation_args:n {
2631 % \str_clear:N \l__stex_notation_lang_str
2632   \str_clear:N \l__stex_notation_variant_str
2633   \keys_set:nn { stex / setnotation } { #1 }
2634 }
2635
2636 \cs_new_protected:Nn \__stex_notation_setnotation:nn {
2637   \seq_if_exist:cT{l_stex_symdecl_#1_notations}{
2638      \seq_remove_all:cn { l_stex_symdecl_#1 _notations }{ #2 }
2639      \seq_put_left:cn { l_stex_symdecl_#1 _notations }{ #2 }
2640   }
2641 }
2642
2643 \cs_new_protected:Nn \stex_setnotation:n {
2644   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1 _notations }
2645      { \l__stex_notation_variant_str }{
2646        \stex_execute_in_module:x{ \__stex_notation_setnotation:nn {#1}{\l__stex_notation_vari
2647        \stex_debug:nn {notations}{
2648          Setting~default~notation~
2649          {\l__stex_notation_variant_str }~for~
2650          #1 \\
2651          \expandafter\meaning\csname
2652          l_stex_symdecl_#1 _notations\endcsname
2653        }
2654      }{
2655        \msg_error:nnxx{stex}{unknownnotation}{\l__stex_notation_variant_str}{#1}
2656      }
2657 }
2658
2659 \NewDocumentCommand \setnotation {m m} {
2660   \stex_get_symbol:n { #1 }
2661   \_stex_setnotation_args:n { #2 }
2662   \stex_setnotation:n{\l_stex_get_symbol_uri_str}
2663   \stex_smsmode_do:\ignorespacesandpars
2664 }
2665
```

```
2666 \cs_new_protected:Nn \stex_copy_notations:nn {
2667   \stex_debug:nn {notations}{
2668     Copying~notations~from~#2~to~#1\\
2669     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
2670   }
2671   \tl_clear:N \l_tmpa_tl
2672   \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2673     \tl_put_right:Nn \l_tmpa_tl { {## ##1} }
2674   }
2675   \seq_map_inline:cn {l_stex_symdecl_#2_notations}{
2676     \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2677     \edef \l_tmpa_tl {
2678       \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2679       \exp_after:wN\exp_after:wN\exp_after:wN {
2680         \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2681       }
2682     }
2683
2684     \stex_execute_in_module:x {
2685       \__stex_notation_restore_notation:nnnnn
2686         {#1}{##1}
2687         { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }
2688         { \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl} }
2689         {
2690           \cs_if_exist:cT{stex_op_notation_ #2\c_hash_str ##1 _cs}{
2691             \exp_args:NNo\exp_args:No\exp_not:n{\csname stex_op_notation_ #2\c_hash_str ##1
2692           }
2693         }
2694     }
2695   }
2696 }
2697
2698 \NewDocumentCommand \copynotation {m m} {
2699   \stex_get_symbol:n { #1 }
2700   \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2701   \stex_get_symbol:n { #2 }
2702   \exp_args:Noo
2703   \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2704   \stex_smsmode_do:\ignorespacesandpars
2705 }
2706
```

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

\symdef

```
2707 \keys_define:nn { stex / symdef } {
2708   name    .str_set_x:N = \l_stex_symdecl_name_str ,
2709   local   .bool_set:N  = \l_stex_symdecl_local_bool ,
2710   args    .str_set_x:N = \l_stex_symdecl_args_str ,
2711   type    .tl_set:N    = \l_stex_symdecl_type_tl ,
2712   def     .tl_set:N    = \l_stex_symdecl_definiens_tl ,
2713   reorder .str_set_x:N = \l_stex_symdecl_reorder_str ,
2714   op      .tl_set:N    = \l__stex_notation_op_tl ,
2715 % lang    .str_set_x:N = \l__stex_notation_lang_str ,
```

```
2716    variant .str_set_x:N = \l__stex_notation_variant_str ,
2717    prec    .str_set_x:N = \l__stex_notation_prec_str ,
2718    assoc   .choices:nn =
2719        {bin,binl,binr,pre,conj,pwconj}
2720        {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
2721    unknown .code:n       = \str_set:Nx
2722        \l__stex_notation_variant_str \l_keys_key_str
2723 }
2724
2725 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2726    \str_clear:N \l_stex_symdecl_name_str
2727    \str_clear:N \l_stex_symdecl_args_str
2728    \str_clear:N \l_stex_symdecl_assoctype_str
2729    \str_clear:N \l_stex_symdecl_reorder_str
2730    \bool_set_false:N \l_stex_symdecl_local_bool
2731    \tl_clear:N \l_stex_symdecl_type_tl
2732    \tl_clear:N \l_stex_symdecl_definiens_tl
2733  % \str_clear:N \l__stex_notation_lang_str
2734    \str_clear:N \l__stex_notation_variant_str
2735    \str_clear:N \l__stex_notation_prec_str
2736    \tl_clear:N \l__stex_notation_op_tl
2737
2738    \keys_set:nn { stex / symdef } { #1 }
2739 }
2740
2741 \NewDocumentCommand \symdef { m O{} } {
2742    \__stex_notation_symdef_args:n { #2 }
2743    \bool_set_true:N \l_stex_symdecl_make_macro_bool
2744    \stex_symdecl_do:n { #1 }
2745    \tl_set:Nn \l_stex_notation_after_do_tl {
2746      \__stex_notation_final:
2747      \stex_smsmode_do:\ignorespacesandpars
2748    }
2749    \str_set:Nx \l_stex_get_symbol_uri_str {
2750      \l_stex_current_module_str ? \l_stex_symdecl_name_str
2751    }
2752    \exp_args:Nx \stex_notation_do:nnnnn
2753      { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } } }
2754      { \prop_item:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } } }
2755      { \l__stex_notation_variant_str }
2756      { \l__stex_notation_prec_str}
2757 }
2758 \stex_deactivate_macro:Nn \symdef {module~environments}
```

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

29.3   Variables

```
2759 ⟨@@=stex_variables⟩
2760
2761 \keys_define:nn { stex / vardef } {
2762    name    .str_set_x:N  = \l__stex_variables_name_str ,
2763    args    .str_set_x:N  = \l__stex_variables_args_str ,
2764    type    .tl_set:N     = \l__stex_variables_type_tl ,
```

159

```
2765  def      .tl_set:N     = \l__stex_variables_def_tl ,
2766  op       .tl_set:N     = \l__stex_variables_op_tl ,
2767  prec     .str_set_x:N  = \l__stex_variables_prec_str ,
2768  assoc    .choices:nn   =
2769      {bin,binl,binr,pre,conj,pwconj}
2770      {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
2771  bind     .choices:nn   =
2772      {forall,exists}
2773      {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2774 }
2775
2776 \cs_new_protected:Nn \__stex_variables_args:n {
2777   \str_clear:N \l__stex_variables_name_str
2778   \str_clear:N \l__stex_variables_args_str
2779   \str_clear:N \l__stex_variables_prec_str
2780   \str_clear:N \l__stex_variables_assoctype_str
2781   \str_clear:N \l__stex_variables_bind_str
2782   \tl_clear:N \l__stex_variables_type_tl
2783   \tl_clear:N \l__stex_variables_def_tl
2784   \tl_clear:N \l__stex_variables_op_tl
2785
2786   \keys_set:nn { stex / vardef } { #1 }
2787 }
2788
2789 \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{}} {
2790   \__stex_variables_args:n {#2}
2791   \str_if_empty:NT \l__stex_variables_name_str {
2792     \str_set:Nx \l__stex_variables_name_str { #1 }
2793   }
2794   \prop_clear:N \l_tmpa_prop
2795   \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
2796
2797   \int_zero:N \l_tmpb_int
2798   \bool_set_true:N \l_tmpa_bool
2799   \str_map_inline:Nn \l__stex_variables_args_str {
2800     \token_case_meaning:NnF ##1 {
2801       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2802       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2803       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2804       {\tl_to_str:n a} {
2805         \bool_set_false:N \l_tmpa_bool
2806         \int_incr:N \l_tmpb_int
2807       }
2808       {\tl_to_str:n B} {
2809         \bool_set_false:N \l_tmpa_bool
2810         \int_incr:N \l_tmpb_int
2811       }
2812     }{
2813       \msg_error:nnxx{stex}{error/wrongargs}{
2814         variable~\l__stex_variables_name_str
2815       }{##1}
2816     }
2817   }
2818   \bool_if:NTF \l_tmpa_bool {
```

```
2819        % possibly numeric
2820        \str_if_empty:NTF \l__stex_variables_args_str {
2821          \prop_put:Nnn \l_tmpa_prop { args } {}
2822          \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2823        }{
2824          \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
2825          \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2826          \str_clear:N \l_tmpa_str
2827          \int_step_inline:nn \l_tmpa_int {
2828            \str_put_right:Nn \l_tmpa_str i
2829          }
2830          \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
2831          \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2832        }
2833      } {
2834        \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2835        \prop_put:Nnx \l_tmpa_prop { arity }
2836          { \str_count:N \l__stex_variables_args_str }
2837      }
2838      \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2839      \tl_set:cx { #1 }{ \stex_invoke_variable:n { \l__stex_variables_name_str } }
2840
2841      \prop_set_eq:cN { l_stex_variable_\l__stex_variables_name_str _prop} \l_tmpa_prop
2842
2843      \tl_if_empty:NF \l__stex_variables_op_tl {
2844        \cs_set:cpx {
2845          stex_var_op_notation_ \l__stex_variables_name_str _cs
2846        } { \exp_not:N\comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } } }
2847      }
2848
2849      \tl_set:Nn \l_stex_notation_after_do_tl {
2850        \exp_args:Nne \use:nn {
2851          \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str _cs }
2852            \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
2853        } {{
2854          \exp_after:wN \exp_after:wN \exp_after:wN
2855          \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2856          { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_symb
2857        }}
2858        \stex_if_do_html:T {
2859          \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
2860            \stex_annotate_invisible:nnn { precedence }
2861              { \l__stex_variables_prec_str }{}
2862            \tl_if_empty:NF \l__stex_variables_type_tl {\stex_annotate_invisible:nnn{type}{}{$\l
2863            \stex_annotate_invisible:nnn{args}{}{ \l__stex_variables_args_str }
2864            \stex_annotate_invisible:nnn{macroname}{#1}{}
2865            \tl_if_empty:NF \l__stex_variables_def_tl {
2866              \stex_annotate_invisible:nnn{definiens}{}
2867                {$\l__stex_variables_def_tl$}
2868            }
2869            \str_if_empty:NF \l__stex_variables_assoctype_str {
2870              \stex_annotate_invisible:nnn{assoctype}{\l__stex_variables_assoctype_str}{}
2871            }
2872            \str_if_empty:NF \l__stex_variables_bind_str {
```

161

```
2873                \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{}
2874              }
2875            \int_zero:N \l_tmpa_int
2876            \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
2877            \tl_clear:N \l_tmpa_tl
2878            \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
2879              \int_incr:N \l_tmpa_int
2880              \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
2881              \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables
2882              \str_if_eq:VnTF \l_tmpb_str a {
2883                \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2884                  \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2885                  \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2886                } }
2887              }{
2888                \str_if_eq:VnTF \l_tmpb_str B {
2889                  \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2890                    \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2891                    \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2892                  } }
2893                }{
2894                  \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2895                    \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
2896                  } }
2897                }
2898              }
2899            }
2900            \stex_annotate_invisible:nnn { notationcomp }{}{
2901              \str_set:Nx \l_stex_current_symbol_str {var://\l__stex_variables_name_str }
2902              $ \exp_args:Nno \use:nn { \use:c {
2903                stex_var_notation_\l__stex_variables_name_str _cs
2904              } } { \l_tmpa_tl } $
2905            }
2906          }
2907      }\ignorespacesandpars
2908    }
2909
2910    \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
2911  }
2912
2913  \cs_new:Nn \_stex_reset:N {
2914    \tl_if_exist:NTF #1 {
2915      \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
2916    }{
2917      \let \exp_not:N #1 \exp_not:N \undefined
2918    }
2919  }
2920
2921  \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
2922    \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
2923    \exp_args:Nnx \use:nn {
2924      % TODO
2925      \stex_annotate_invisible:nnn {vardecl}{\clist_use:Nn\l__stex_variables_names,}{
2926        #2
```

162

```
2927       }
2928     }{
2929       \_stex_reset:N \varnot
2930       \_stex_reset:N \vartype
2931       \_stex_reset:N \vardefi
2932     }
2933   }
2934
2935   \NewDocumentCommand \vardef { s } {
2936     \IfBooleanTF#1 {
2937       \__stex_variables_do_complex:nn
2938     }{
2939       \__stex_variables_do_simple:nnn
2940     }
2941   }
2942
2943   \NewDocumentCommand \svar { O{} m }{
2944     \tl_if_empty:nTF {#1}{
2945       \str_set:Nn \l_tmpa_str { #2 }
2946     }{
2947       \str_set:Nn \l_tmpa_str { #1 }
2948     }
2949     \_stex_term_omv:nn {
2950       var://\l_tmpa_str
2951     }{
2952       \exp_args:Nnx \use:nn {
2953         \def\comp{\_varcomp}
2954         \str_set:Nx \l_stex_current_symbol_str { var://\l_tmpa_str }
2955         \comp{ #2 }
2956       }{
2957         \_stex_reset:N \comp
2958         \_stex_reset:N \l_stex_current_symbol_str
2959       }
2960     }
2961   }
2962
2963
2964
2965   \keys_define:nn { stex / varseq } {
2966     name     .str_set_x:N  = \l__stex_variables_name_str ,
2967     args     .int_set:N    = \l__stex_variables_args_int ,
2968     type     .tl_set:N     = \l__stex_variables_type_tl  ,
2969     mid      .tl_set:N     = \l__stex_variables_mid_tl   ,
2970     bind     .choices:nn   =
2971         {forall,exists}
2972         {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2973   }
2974
2975   \cs_new_protected:Nn \__stex_variables_seq_args:n {
2976     \str_clear:N \l__stex_variables_name_str
2977     \int_set:Nn \l__stex_variables_args_int 1
2978     \tl_clear:N \l__stex_variables_type_tl
2979     \str_clear:N \l__stex_variables_bind_str
2980
```

```
2981    \keys_set:nn { stex / varseq } { #1 }
2982 }
2983
2984 \NewDocumentCommand \varseq {m O{} m m m}{
2985    \__stex_variables_seq_args:n { #2 }
2986    \str_if_empty:NT \l__stex_variables_name_str {
2987      \str_set:Nx \l__stex_variables_name_str { #1 }
2988    }
2989    \prop_clear:N \l_tmpa_prop
2990    \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
2991
2992    \seq_set_from_clist:Nn \l_tmpa_seq {#3}
2993    \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
2994      \msg_error:nnxx{stex}{error/seqlength}
2995        {\int_use:N \l__stex_variables_args_int}
2996        {\seq_count:N \l_tmpa_seq}
2997    }
2998    \seq_set_from_clist:Nn \l_tmpb_seq {#4}
2999    \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
3000      \msg_error:nnxx{stex}{error/seqlength}
3001        {\int_use:N \l__stex_variables_args_int}
3002        {\seq_count:N \l_tmpb_seq}
3003    }
3004    \prop_put:Nnn \l_tmpa_prop {starts} {#3}
3005    \prop_put:Nnn \l_tmpa_prop {ends} {#4}
3006
3007    \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
3008      \cs_set:Npn  {\int_use:N \l__stex_variables_args_int} { #5 }
3009
3010    \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3011    \int_step_inline:nn \l__stex_variables_args_int {
3012      \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
3013    }
3014    \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
3015    \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
3016    \tl_if_empty:NF \l__stex_variables_mid_tl {
3017      \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
3018      \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
3019    }
3020    \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3021    \int_step_inline:nn \l__stex_variables_args_int {
3022      \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
3023    }
3024    \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
3025    \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
3026
3027
3028    \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl
3029
3030    \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}
3031
3032    \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3033
3034    \int_step_inline:nn \l__stex_variables_args_int {
```

```
3035    \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
3036      \_stex_term_math_arg:nnn{i##1}{0}{\exp_not:n{####}##1}
3037    }}
3038  }
3039
3040  \tl_set:Nx \l_tmpa_tl {
3041    \_stex_term_math_oma:nnnn { varseq://\l__stex_variables_name_str}{}{0}{
3042      \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
3043    }
3044  }
3045
3046  \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \stex_symbol_after_invokation_tl} }
3047
3048  \exp_args:Nno \use:nn {
3049  \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
3050    \cs_set:Npn {\int_use:N \l_stex_variables_args_int}}{\l_tmpa_tl}
3051
3052  \stex_debug:nn{sequences}{New~Sequence:~
3053    \expandafter\meaning\csname stex_varseq_\l__stex_variables_name_str _cs\endcsname\\~\\
3054    \prop_to_keyval:N \l_tmpa_prop
3055  }
3056  \stex_if_do_html:T{\stex_annotate_invisible:nnn{varseq}{\l__stex_variables_name_str}{
3057    \tl_if_empty:NF \l__stex_variables_type_tl {
3058      \stex_annotate:nnn {type}{}{$\seqtype\l__stex_variables_type_tl$}
3059    }
3060    \stex_annotate:nnn {args}{\int_use:N \l_stex_variables_args_int}{}
3061    \str_if_empty:NF \l__stex_variables_bind_str {
3062      \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{}
3063    }
3064  }}
3065
3066  \prop_set_eq:cN {stex_varseq_\l__stex_variables_name_str _prop}\l_tmpa_prop
3067  \ignorespacesandpars
3068 }
3069
3070 ⟨/package⟩
```

Chapter 30

STEX
-Terms Implementation

```
3071 ⟨∗package⟩
3072
3073 %%%%%%%%%%%%    terms.dtx    %%%%%%%%%%%%
3074
3075 ⟨@@=stex_terms⟩
```

Warnings and error messages

```
3076 \msg_new:nnn{stex}{error/nonotation}{
3077   Symbol~#1~invoked,~but~has~no~notation#2!
3078 }
3079 \msg_new:nnn{stex}{error/notationarg}{
3080   Error~in~parsing~notation~#1
3081 }
3082 \msg_new:nnn{stex}{error/noop}{
3083   Symbol~#1~has~no~operator~notation~for~notation~#2
3084 }
3085 \msg_new:nnn{stex}{error/notallowed}{
3086   Symbol~invokation~#1~not~allowed~in~notation~component~of~#2
3087 }
3088 \msg_new:nnn{stex}{error/doubleargument}{
3089   Argument~#1~of~symbol~#2~already~assigned
3090 }
3091 \msg_new:nnn{stex}{error/overarity}{
3092   Argument~#1~invalid~for~symbol~#2~with~arity~#3
3093 }
3094
```

30.1   Symbol Invocations

`\stex_invoke_symbol:n`   Invokes a semantic macro

```
3095
3096
3097 \bool_new:N \l_stex_allow_semantic_bool
3098 \bool_set_true:N \l_stex_allow_semantic_bool
3099
```

```
3100  \cs_new_protected:Nn \stex_invoke_symbol:n {
3101    \bool_if:NTF \l_stex_allow_semantic_bool {
3102      \str_if_eq:eeF {
3103        \prop_item:cn {
3104          l_stex_symdecl_#1_prop
3105        }{ deprecate }
3106      }{}{
3107        \msg_warning:nnxx{stex}{warning/deprecated}{
3108          Symbol~#1
3109        }{
3110          \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
3111        }
3112      }
3113      \if_mode_math:
3114        \exp_after:wN \__stex_terms_invoke_math:n
3115      \else:
3116        \exp_after:wN \__stex_terms_invoke_text:n
3117      \fi: { #1 }
3118    }{
3119      \msg_error:nnxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
3120    }
3121  }
3122
3123  \cs_new_protected:Nn \__stex_terms_invoke_text:n {
3124    \peek_charcode_remove:NTF ! {
3125      \__stex_terms_invoke_op_custom:nn {#1}
3126    }{
3127      \__stex_terms_invoke_custom:nn {#1}
3128    }
3129  }
3130
3131  \cs_new_protected:Nn \__stex_terms_invoke_math:n {
3132    \peek_charcode_remove:NTF ! {
3133      % operator
3134      \peek_charcode_remove:NTF * {
3135        % custom op
3136        \__stex_terms_invoke_op_custom:nn {#1}
3137      }{
3138        % op notation
3139        \peek_charcode:NTF [ {
3140          \__stex_terms_invoke_op_notation:nw {#1}
3141        }{
3142          \__stex_terms_invoke_op_notation:nw {#1}[]
3143        }
3144      }
3145    }{
3146      \peek_charcode_remove:NTF * {
3147        \__stex_terms_invoke_custom:nn {#1}
3148        % custom
3149      }{
3150        % normal
3151        \peek_charcode:NTF [ {
3152          \__stex_terms_invoke_notation:nw {#1}
3153        }{
```

```
3154        \__stex_terms_invoke_notation:nw {#1}[]
3155      }
3156    }
3157  }
3158 }
3159
3160
3161 \cs_new_protected:Nn \__stex_terms_invoke_op_custom:nn {
3162   \exp_args:Nnx \use:nn {
3163     \def\comp{\_comp}
3164     \str_set:Nn \l_stex_current_symbol_str { #1 }
3165     \bool_set_false:N \l_stex_allow_semantic_bool
3166     \_stex_term_oms:nnn {#1}{#1 \c_hash_str CUSTOM-}{
3167       \comp{ #2 }
3168     }
3169   }{
3170     \_stex_reset:N \comp
3171     \_stex_reset:N \l_stex_current_symbol_str
3172     \bool_set_true:N \l_stex_allow_semantic_bool
3173   }
3174 }
3175
3176 \keys_define:nn { stex / terms } {
3177 %  lang     .tl_set_x:N = \l_stex_notation_lang_str ,
3178   variant .tl_set_x:N = \l_stex_notation_variant_str ,
3179   unknown .code:n     = \str_set:Nx
3180       \l_stex_notation_variant_str \l_keys_key_str
3181 }
3182
3183 \cs_new_protected:Nn \__stex_terms_args:n {
3184  % \str_clear:N \l_stex_notation_lang_str
3185   \str_clear:N \l_stex_notation_variant_str
3186
3187   \keys_set:nn { stex / terms } { #1 }
3188 }
3189
3190 \cs_new_protected:Nn \stex_find_notation:nn {
3191   \__stex_terms_args:n { #2 }
3192   \seq_if_empty:cTF {
3193     l_stex_symdecl_ #1 _notations
3194   } {
3195     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
3196   } {
3197     \str_if_empty:NTF \l_stex_notation_variant_str {
3198       \seq_get_left:cN {l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
3199     }{
3200       \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
3201         \l_stex_notation_variant_str
3202       }{
3203       % \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3204       }{
3205         \msg_error:nnxx{stex}{error/nonotation}{#1}{
3206           ~\l_stex_notation_variant_str
3207         }
```

```
3208          }
3209        }
3210      }
3211  }
3212
3213  \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3214    \exp_args:Nnx \use:nn {
3215      \def\comp{\_comp}
3216      \str_set:Nn \l_stex_current_symbol_str { #1 }
3217      \stex_find_notation:nn { #1 }{ #2 }
3218      \bool_set_false:N \l_stex_allow_semantic_bool
3219      \cs_if_exist:cTF {
3220        stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3221      }{
3222        \_stex_term_oms:nnn { #1 }{
3223          #1 \c_hash_str \l_stex_notation_variant_str
3224        }{
3225          \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3226        }
3227      }{
3228        \int_compare:nNnTF {\prop_item:cn {l_stex_symdecl_#1_prop}{arity}} = 0{
3229          \cs_if_exist:cTF {
3230            stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3231          }{
3232            \tl_set:Nx \stex_symbol_after_invokation_tl {
3233              \_stex_reset:N \comp
3234              \_stex_reset:N \stex_symbol_after_invokation_tl
3235              \_stex_reset:N \l_stex_current_symbol_str
3236              \bool_set_true:N \l_stex_allow_semantic_bool
3237            }
3238            \def\comp{\_comp}
3239            \str_set:Nn \l_stex_current_symbol_str { #1 }
3240            \bool_set_false:N \l_stex_allow_semantic_bool
3241            \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3242          }{
3243            \msg_error:nnxx{stex}{error/nonotation}{#1}{
3244              ~\l_stex_notation_variant_str
3245            }
3246          }
3247        }{
3248          \msg_error:nnxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3249        }
3250      }
3251    }{
3252      \_stex_reset:N \comp
3253      \_stex_reset:N \l_stex_current_symbol_str
3254      \bool_set_true:N \l_stex_allow_semantic_bool
3255    }
3256  }
3257
3258  \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {
3259    \stex_find_notation:nn { #1 }{ #2 }
3260    \cs_if_exist:cTF {
3261      stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
```

```
3262    }{
3263      \tl_set:Nx \stex_symbol_after_invokation_tl {
3264        \_stex_reset:N \comp
3265        \_stex_reset:N \stex_symbol_after_invokation_tl
3266        \_stex_reset:N \l_stex_current_symbol_str
3267        \bool_set_true:N \l_stex_allow_semantic_bool
3268      }
3269      \def\comp{\_comp}
3270      \str_set:Nn \l_stex_current_symbol_str { #1 }
3271      \bool_set_false:N \l_stex_allow_semantic_bool
3272      \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3273    }{
3274      \msg_error:nnxx{stex}{error/nonotation}{#1}{
3275        ~\l_stex_notation_variant_str
3276      }
3277    }
3278 }
3279
3280 \prop_new:N \l__stex_terms_custom_args_prop
3281
3282 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3283    \exp_args:Nnx \use:nn {
3284      \bool_set_false:N \l_stex_allow_semantic_bool
3285      \def\comp{\_comp}
3286      \str_set:Nn \l_stex_current_symbol_str { # 1 }
3287      \prop_clear:N \l__stex_terms_custom_args_prop
3288      \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3289      \prop_get:cnN {
3290        l_stex_symdecl_#1 _prop
3291      }{ args } \l_tmpa_str
3292      \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3293      \tl_set:Nn \arg { \__stex_terms_arg: }
3294      \str_if_empty:NTF \l_tmpa_str {
3295        \_stex_term_oms:nnn {#1}{#1\c_hash_str CUSTOM-}{#2}
3296      }{
3297        \str_if_in:NnTF \l_tmpa_str b {
3298          \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3299        }{
3300          \str_if_in:NnTF \l_tmpa_str B {
3301            \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3302          }{
3303            \_stex_term_oma:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3304          }
3305        }
3306      }
3307      % TODO check that all arguments exist
3308    }{
3309      \_stex_reset:N \l_stex_current_symbol_str
3310      \_stex_reset:N \arg
3311      \_stex_reset:N \comp
3312      \_stex_reset:N \l__stex_terms_custom_args_prop
3313      \bool_set_true:N \l_stex_allow_semantic_bool
3314    }
3315 }
```

```
3316
3317  \NewDocumentCommand \__stex_terms_arg: { s O{} m }{
3318    \tl_if_empty:nTF {#2}{
3319      \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
3320      \bool_set_true:N \l_tmpa_bool
3321      \bool_do_while:Nn \l_tmpa_bool {
3322        \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int
3323          \int_incr:N \l_tmpa_int
3324        }{
3325          \bool_set_false:N \l_tmpa_bool
3326        }
3327      }
3328    }{
3329      \int_set:Nn \l_tmpa_int { #2 }
3330    }
3331    \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
3332    \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
3333      \msg_error:nnxxx{stex}{error/overarity}
3334        {\int_use:N \l_tmpa_int}
3335        {\l_stex_current_symbol_str}
3336        {\str_count:N \l_tmpa_str}
3337    }
3338    \str_set:Nx \l_tmpa_str {\str_item:Nn \l_tmpa_str \l_tmpa_int}
3339    \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
3340      \bool_lazy_any:nF {
3341        {\str_if_eq_p:Vn \l_tmpa_str {a}}
3342        {\str_if_eq_p:Vn \l_tmpa_str {B}}
3343      }{
3344        \msg_error:nnxx{stex}{error/doubleargument}
3345          {\int_use:N \l_tmpa_int}
3346          {\l_stex_current_symbol_str}
3347      }
3348    }
3349    \exp_args:NNx \prop_put:Nnn \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {#3}
3350    \bool_set_true:N \l_stex_allow_semantic_bool
3351    \IfBooleanTF#1{
3352      \stex_annotate_invisible:n { %TODO
3353        \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{#3}
3354      }
3355    }{ %TODO
3356      \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{#3}
3357    }
3358    \bool_set_false:N \l_stex_allow_semantic_bool
3359  }
3360
3361
3362  \cs_new_protected:Nn \_stex_term_arg:nn {
3363    \bool_set_true:N \l_stex_allow_semantic_bool
3364    \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3365    \bool_set_false:N \l_stex_allow_semantic_bool
3366  }
3367
3368  \cs_new_protected:Nn \_stex_term_math_arg:nnn {
3369    \exp_args:Nnx \use:nn
```

```
3370        { \int_set:Nn \l__stex_terms_downprec { #2 }
3371          \_stex_term_arg:nn { #1 }{ #3 }
3372        }
3373        { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3374  }
```

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

```
3375  \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
3376    \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3377    \tl_set:Nn \l_tmpb_tl {\_stex_term_math_arg:nnn{#1}{#2}}
3378    \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 }}{
3379      \expandafter\if\expandafter\relax\noexpand#3
3380        \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_maybe_sequence:Nn#3{#1}}
3381      \else
3382        \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_simple:nn{#1}{#3}}
3383      \fi
3384      \l_tmpa_tl
3385    }{
3386      \__stex_terms_math_assoc_arg_simple:nn{#1}{#3}
3387    }
3388  }
3389
3390  \cs_new_protected:Nn \__stex_terms_math_assoc_arg_maybe_sequence:Nn {
3391    \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
3392    \str_if_empty:NTF \l_tmpa_str {
3393      \exp_args:Nx \cs_if_eq:NNTF {
3394        \tl_head:N #1
3395      } \stex_invoke_sequence:n {
3396        \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
3397        \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
3398        \tl_set:Nx \l_tmpa_tl {\prop_item:cn {stex_varseq_\l_tmpa_str _prop}{notation}}
3399        \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
3400        \tl_set:Nx \l_tmpa_tl {{\exp_not:N \exp_not:n{
3401          \exp_not:n{\exp_args:Nnx \use:nn} {
3402            \exp_not:n {
3403              \def\comp{\_varcomp}
3404              \str_set:Nn \l_stex_current_symbol_str
3405            } {varseq://\l_tmpa_str}
3406            \exp_not:n{ ##1 }
3407          }{
3408            \exp_not:n {
3409              \_stex_reset:N \comp
3410              \_stex_reset:N \l_stex_current_symbol_str
3411            }
3412          }
3413        }}}
3414        \exp_args:Nno \use:nn {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
3415        \seq_reverse:N \l_tmpa_seq
3416        \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
3417        \seq_map_inline:Nn \l_tmpa_seq {
3418          \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3419            \exp_args:Nno
```

```
3420              \l_tmpa_cs { ##1 } \l_tmpa_tl
3421            }
3422          }
3423        \tl_set:Nx \l_tmpa_tl {
3424          \_stex_term_omv:nn {varseq://\l_tmpa_str}{
3425            \exp_args:No \exp_not:n \l_tmpa_tl
3426          }
3427        }
3428        \exp_args:No\l_tmpb_tl\l_tmpa_tl
3429      }{
3430        \__stex_terms_math_assoc_arg_simple:nn{#2} { #1 }
3431      }
3432    } {
3433      \__stex_terms_math_assoc_arg_simple:nn{#2} { #1 }
3434    }
3435
3436  }
3437
3438  \cs_new_protected:Nn \__stex_terms_math_assoc_arg_simple:nn {
3439    \clist_set:Nn \l_tmpa_clist{ #2 }
3440    \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3441      \tl_set:Nn \l_tmpa_tl { \_stex_term_arg:nn{A#1}{ #2 } }
3442    }{
3443      \clist_reverse:N \l_tmpa_clist
3444      \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3445      \tl_set:Nx \l_tmpa_tl { \_stex_term_arg:nn{A#1}{
3446        \exp_args:No \exp_not:n \l_tmpa_tl
3447      }}
3448      \clist_map_inline:Nn \l_tmpa_clist {
3449        \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3450          \exp_args:Nno
3451          \l_tmpa_cs { \_stex_term_arg:nn{A#1}{##1} } \l_tmpa_tl
3452        }
3453      }
3454    }
3455    \exp_args:No\l_tmpb_tl\l_tmpa_tl
3456  }
```

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

30.2 Terms

Precedences:

```
3457  \tl_const:Nx \infprec {\int_use:N \c_max_int}
3458  \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
3459  \int_new:N \l__stex_terms_downprec
3460  \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 64.*)

Bracketing:

```
3461 \tl_set:Nn \l__stex_terms_left_bracket_str (
3462 \tl_set:Nn \l__stex_terms_right_bracket_str )
```

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

Compares precedences and insert brackets accordingly

```
3463 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
3464   \bool_if:NTF \l__stex_terms_brackets_done_bool {
3465     \bool_set_false:N \l__stex_terms_brackets_done_bool
3466     #2
3467   } {
3468     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
3469       \bool_if:NTF \l_stex_inparray_bool { #2 }{
3470         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
3471         \dobrackets { #2 }
3472       }
3473     }{ #2 }
3474   }
3475 }
```

(*End definition for* __stex_terms_maybe_brackets:nn.)

```
3476 \bool_new:N \l__stex_terms_brackets_done_bool
3477 %\RequirePackage{scalerel}
3478 \cs_new_protected:Npn \dobrackets #1 {
3479   %\ThisStyle{\if D\m@switch
3480 %    \exp_args:Nnx \use:nn
3481 %    { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
3482 %    { \exp_not:N\right\l__stex_terms_right_bracket_str }
3483 %  \else
3484     \exp_args:Nnx \use:nn
3485     {
3486       \bool_set_true:N \l__stex_terms_brackets_done_bool
3487       \int_set:Nn \l__stex_terms_downprec \infprec
3488       \l__stex_terms_left_bracket_str
3489       #1
3490     }
3491     {
3492       \bool_set_false:N \l__stex_terms_brackets_done_bool
3493       \l__stex_terms_right_bracket_str
3494       \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3495     }
3496   %\fi}
3497 }
```

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

```
3498 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
3499   \exp_args:Nnx \use:nn
3500   {
3501     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
```

174

```
3502        \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
3503          #3
3504      }
3505      {
3506        \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
3507          {\l__stex_terms_left_bracket_str}
3508        \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
3509          {\l__stex_terms_right_bracket_str}
3510      }
3511 }
```

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

\STEXinvisible

```
3512 \cs_new_protected:Npn \STEXinvisible #1 {
3513    \stex_annotate_invisible:n { #1 }
3514 }
```

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

OMDoc terms:

_stex_term_math_oms:nnnn

```
3515 \cs_new_protected:Nn \_stex_term_oms:nnn {
3516    \stex_annotate:nnn{ OMID }{ #2 }{
3517      #3
3518    }
3519 }
3520
3521 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
3522    \__stex_terms_maybe_brackets:nn { #3 }{
3523      \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3524    }
3525 }
```

(*End definition for* _stex_term_math_oms:nnnn*. This function is documented on page* *63.*)

_stex_term_math_omv:nn

```
3526 \cs_new_protected:Nn \_stex_term_omv:nn {
3527    \stex_annotate:nnn{ OMV }{ #1 }{
3528      #2
3529    }
3530 }
```

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

_stex_term_math_oma:nnnn

```
3531 \cs_new_protected:Nn \_stex_term_oma:nnn {
3532    \stex_annotate:nnn{ OMA }{ #2 }{
3533      #3
3534    }
3535 }
3536
3537 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
3538    \__stex_terms_maybe_brackets:nn { #3 }{
3539      \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
```

175

```
3540        }
3541      }
```

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

_stex_term_math_omb:nnnn

```
3542  \cs_new_protected:Nn \_stex_term_ombind:nnn {
3543    \stex_annotate:nnn{ OMBIND }{ #2 }{
3544        #3
3545    }
3546  }
3547
3548  \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
3549    \__stex_terms_maybe_brackets:nn { #3 }{
3550      \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3551    }
3552  }
```

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

\symref
\symname

```
3553  \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3554
3555  \keys_define:nn { stex / symname } {
3556    pre      .tl_set_x:N    = \l__stex_terms_pre_tl ,
3557    post     .tl_set_x:N    = \l__stex_terms_post_tl ,
3558    root     .tl_set_x:N    = \l__stex_terms_root_tl
3559  }
3560
3561  \cs_new_protected:Nn \stex_symname_args:n {
3562    \tl_clear:N \l__stex_terms_post_tl
3563    \tl_clear:N \l__stex_terms_pre_tl
3564    \tl_clear:N \l__stex_terms_root_str
3565    \keys_set:nn { stex / symname } { #1 }
3566  }
3567
3568  \NewDocumentCommand \symref { m m }{
3569    \let\compemph_uri_prev:\compemph@uri
3570    \let\compemph@uri\symrefemph@uri
3571    \STEXsymbol{#1}!{ #2 }
3572    \let\compemph@uri\compemph_uri_prev:
3573  }
3574
3575  \NewDocumentCommand \synonym { O{} m m}{
3576    \stex_symname_args:n { #1 }
3577    \let\compemph_uri_prev:\compemph@uri
3578    \let\compemph@uri\symrefemph@uri
3579    % TODO
3580    \STEXsymbol{#2}!{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
3581    \let\compemph@uri\compemph_uri_prev:
3582  }
3583
3584  \NewDocumentCommand \symname { O{} m }{
3585    \stex_symname_args:n { #1 }
3586    \stex_get_symbol:n { #2 }
```

```
3587    \str_set:Nx \l_tmpa_str {
3588        \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3589    }
3590    \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3591
3592    \let\compemph_uri_prev:\compemph@uri
3593    \let\compemph@uri\symrefemph@uri
3594    \exp_args:NNx \use:nn
3595    \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmmode*\fi{
3596        \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
3597    } }
3598    \let\compemph@uri\compemph_uri_prev:
3599 }
3600
3601 \NewDocumentCommand \Symname { O{} m }{
3602    \stex_symname_args:n { #1 }
3603    \stex_get_symbol:n { #2 }
3604    \str_set:Nx \l_tmpa_str {
3605        \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3606    }
3607    \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3608    \let\compemph_uri_prev:\compemph@uri
3609    \let\compemph@uri\symrefemph@uri
3610    \exp_args:NNx \use:nn
3611    \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmmode*\fi{
3612        \exp_after:wN \stex_capitalize:n \l_tmpa_str
3613            \l__stex_terms_post_tl
3614    } }
3615    \let\compemph@uri\compemph_uri_prev:
3616 }
```

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

30.3   Notation Components

```
3617 ⟨@@=stex_notationcomps⟩
```

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

```
3618 \cs_new_protected:Npn \_comp #1 {
3619    \str_if_empty:NF \l_stex_current_symbol_str {
3620        \stex_html_backend:TF {
3621            \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
3622        }{
3623            \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3624        }
3625    }
3626 }
3627
3628 \cs_new_protected:Npn \_varcomp #1 {
3629    \str_if_empty:NF \l_stex_current_symbol_str {
3630        \stex_html_backend:TF {
3631            \stex_annotate:nnn { varcomp }{ \l_stex_current_symbol_str }{ #1 }
3632        }{
3633            \exp_args:Nnx \varemph@uri { #1 } { \l_stex_current_symbol_str }
```

177

```
3634        }
3635      }
3636  }
3637
3638  \def\comp{\_comp}
3639
3640  \cs_new_protected:Npn \compemph@uri #1 #2 {
3641      \compemph{ #1 }
3642  }
3643
3644
3645  \cs_new_protected:Npn \compemph #1 {
3646      #1
3647  }
3648
3649  \cs_new_protected:Npn \defemph@uri #1 #2 {
3650      \defemph{#1}
3651  }
3652
3653  \cs_new_protected:Npn \defemph #1 {
3654      \textbf{#1}
3655  }
3656
3657  \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3658      \symrefemph{#1}
3659  }
3660
3661  \cs_new_protected:Npn \symrefemph #1 {
3662      \emph{#1}
3663  }
3664
3665  \cs_new_protected:Npn \varemph@uri #1 #2 {
3666      \varemph{#1}
3667  }
3668
3669  \cs_new_protected:Npn \varemph #1 {
3670      #1
3671  }
```

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

\ellipses

```
3672  \NewDocumentCommand \ellipses {} { \ldots }
```

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

\parray
\prmatrix
\parrayline
\parraylineh
\parraycell

```
3673  \bool_new:N \l_stex_inparray_bool
3674  \bool_set_false:N \l_stex_inparray_bool
3675  \NewDocumentCommand \parray { m m } {
3676      \begingroup
3677      \bool_set_true:N \l_stex_inparray_bool
3678      \begin{array}{#1}
3679          #2
3680      \end{array}
```

```
3681      \endgroup
3682 }
3683
3684 \NewDocumentCommand \prmatrix { m } {
3685    \begingroup
3686    \bool_set_true:N \l_stex_inparray_bool
3687    \begin{matrix}
3688      #1
3689    \end{matrix}
3690    \endgroup
3691 }
3692
3693 \def \maybephline {
3694    \bool_if:NT \l_stex_inparray_bool {\hline}
3695 }
3696
3697 \def \parrayline #1 #2 {
3698    #1 #2 \bool_if:NT \l_stex_inparray_bool {\\}
3699 }
3700
3701 \def \pmrow #1 { \parrayline{}{ #1 } }
3702
3703 \def \parraylineh #1 #2 {
3704    #1 #2 \bool_if:NT \l_stex_inparray_bool {\\\hline}
3705 }
3706
3707 \def \parraycell #1 {
3708    #1 \bool_if:NT \l_stex_inparray_bool {&}
3709 }
```

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

30.4   Variables

```
3710 ⟨@@=stex_variables⟩
```

\stex_invoke_variable:n   Invokes a variable

```
3711 \cs_new_protected:Nn \stex_invoke_variable:n {
3712    \if_mode_math:
3713      \exp_after:wN \__stex_variables_invoke_math:n
3714    \else:
3715      \exp_after:wN \__stex_variables_invoke_text:n
3716    \fi: {#1}
3717 }
3718
3719 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
3720    %TODO
3721 }
3722
3723
3724 \cs_new_protected:Nn \__stex_variables_invoke_math:n {
3725    \peek_charcode_remove:NTF ! {
3726      \peek_charcode_remove:NTF ! {
3727        \peek_charcode:NTF [ {
```

```
3728        \__stex_variables_invoke_op_custom:nw
3729      }{
3730        % TODO throw error
3731      }
3732    }{
3733      \__stex_variables_invoke_op:n { #1 }
3734    }
3735  }{
3736    \peek_charcode_remove:NTF * {
3737      \__stex_variables_invoke_text:n { #1 }
3738    }{
3739      \__stex_variables_invoke_math_ii:n { #1 }
3740    }
3741  }
3742 }
3743
3744 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
3745   \cs_if_exist:cTF {
3746     stex_var_op_notation_ #1 _cs
3747   }{
3748     \exp_args:Nnx \use:nn {
3749       \def\comp{\_varcomp}
3750       \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3751       \_stex_term_omv:nn { var://#1 }{
3752         \use:c{stex_var_op_notation_ #1 _cs }
3753       }
3754     }{
3755       \_stex_reset:N \comp
3756       \_stex_reset:N \l_stex_current_symbol_str
3757     }
3758   }{
3759     \int_compare:nNnTF {\prop_item:cn {l_stex_variable_#1_prop}{arity}}} = 0{
3760       \__stex_variables_invoke_math_ii:n {#1}
3761     }{
3762       \msg_error:nnxx{stex}{error/noop}{variable~#1}{}
3763     }
3764   }
3765 }
3766
3767 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n  #1 {
3768   \cs_if_exist:cTF {
3769     stex_var_notation_#1cs
3770   }{
3771     \tl_set:Nx \stex_symbol_after_invokation_tl {
3772       \_stex_reset:N \comp
3773       \_stex_reset:N \stex_symbol_after_invokation_tl
3774       \_stex_reset:N \l_stex_current_symbol_str
3775       \bool_set_true:N \l_stex_allow_semantic_bool
3776     }
3777     \def\comp{\_varcomp}
3778     \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3779     \bool_set_false:N \l_stex_allow_semantic_bool
3780     \use:c{stex_var_notation_#1_cs}
3781   }{
```

```
3782        \msg_error:nnxx{stex}{error/nonotation}{variable~#1}{s}
3783    }
3784 }
```

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

30.5   Sequences

```
3785 ⟨@@=stex_sequences⟩
3786
3787 \cs_new_protected:Nn \stex_invoke_sequence:n {
3788    \peek_charcode_remove:NTF ! {
3789       \_stex_term_omv:nn {varseq://#1}{
3790          \exp_args:Nnx \use:nn {
3791             \def\comp{\_varcomp}
3792             \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3793             \prop_item:cn{stex_varseq_#1_prop}{notation}
3794          }{
3795             \_stex_reset:N \comp
3796             \_stex_reset:N \l_stex_current_symbol_str
3797          }
3798       }
3799    }{
3800       \bool_set_false:N \l_stex_allow_semantic_bool
3801       \def\comp{\_varcomp}
3802       \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3803       \tl_set:Nx \stex_symbol_after_invokation_tl {
3804          \_stex_reset:N \comp
3805          \_stex_reset:N \stex_symbol_after_invokation_tl
3806          \_stex_reset:N \l_stex_current_symbol_str
3807          \bool_set_true:N \l_stex_allow_semantic_bool
3808       }
3809       \use:c { stex_varseq_#1_cs }
3810    }
3811 }
3812 ⟨/package⟩
```

Chapter 31

sTeX
-Structural Features
Implementation

```
3813 ⟨∗package⟩
3814
3815 %%%%%%%%%%%%%   features.dtx   %%%%%%%%%%%%%
3816
```

Warnings and error messages

```
3817 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3818   Symbol~#1~can~not~be~assigned~in~copymodule~#2
3819 }
3820 \msg_new:nnn{stex}{error/interpretmodule/nodefiniens}{
3821   Symbol~#1~not~assigned~in~interpretmodule~#2
3822 }
3823
3824 \msg_new:nnn{stex}{error/unknownstructure}{
3825   No~structure~#1~found!
3826 }
3827
3828 \msg_new:nnn{stex}{error/unknownfield}{
3829   No~field~#1~in~instance~#2~found!\\#3
3830 }
3831
3832 \msg_new:nnn{stex}{error/keyval}{
3833   Invalid~key=value~pair:#1
3834 }
3835 \msg_new:nnn{stex}{error/instantiate/missing}{
3836   Assignments~missing~in~instantiate:~#1
3837 }
3838 \msg_new:nnn{stex}{error/incompatible}{
3839   Incompatible~signature:~#1~(#2)~and~#3~(#4)
3840 }
3841
```

31.1 Imports with modification

```
3842 ⟨@@=stex_copymodule⟩
3843 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
3844   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3845     \tl_set:Nn \l_tmpa_tl { #1 }
3846     \__stex_copymodule_get_symbol_from_cs:
3847   }{
3848     % argument is a string
3849     % is it a command name?
3850     \cs_if_exist:cTF { #1 }{
3851       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3852       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3853       \str_if_empty:NTF \l_tmpa_str {
3854         \exp_args:Nx \cs_if_eq:NNTF {
3855           \tl_head:N \l_tmpa_tl
3856         } \stex_invoke_symbol:n {
3857           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
3858         }{
3859           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3860         }
3861       } {
3862         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3863       }
3864     }{
3865       % argument is not a command name
3866       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3867       % \l_stex_all_symbols_seq
3868     }
3869   }
3870 }
3871
3872 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
3873   \str_set:Nn \l_tmpa_str { #1 }
3874   \bool_set_false:N \l_tmpa_bool
3875   \bool_if:NF \l_tmpa_bool {
3876     \tl_set:Nn \l_tmpa_tl {
3877       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
3878     }
3879   \str_set:Nn \l_tmpa_str { #1 }
3880   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3881   \seq_map_inline:Nn #2 {
3882     \str_set:Nn \l_tmpb_str { ##1 }
3883     \str_if_eq:eeT { \l_tmpa_str } {
3884       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3885     } {
3886       \seq_map_break:n {
3887         \tl_set:Nn \l_tmpa_tl {
3888           \str_set:Nn \l_stex_get_symbol_uri_str {
3889             ##1
3890           }
3891         }
3892       }
3893     }
```

```
3894        }
3895        \l_tmpa_tl
3896    }
3897 }
3898
3899 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
3900    \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3901      { \tl_tail:N \l_tmpa_tl }
3902    \tl_if_single:NTF \l_tmpa_tl {
3903      \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3904        \exp_after:wN \str_set:Nn \exp_after:wN
3905          \l_stex_get_symbol_uri_str \l_tmpa_tl
3906        \__stex_copymodule_get_symbol_check:n { #1 }
3907      }{
3908        % TODO
3909        % tail is not a single group
3910      }
3911    }{
3912      % TODO
3913      % tail is not a single group
3914    }
3915 }
3916
3917 \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
3918    \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
3919      \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3920        :~\seq_use:Nn #1 {,~}
3921      }
3922    }
3923 }
3924
3925 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
3926    % import module
3927    \stex_import_module_uri:nn { #1 } { #2 }
3928    \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3929    \stex_import_require_module:nnnn
3930      { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3931      { \l_stex_import_path_str } { \l_stex_import_name_str }
3932
3933    \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3934    \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
3935
3936    % fields
3937    \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
3938    \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
3939      \seq_map_inline:cn {c_stex_module_##1_constants}{
3940        \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
3941          ##1 ? ####1
3942        }
3943      }
3944    }
3945
3946    % setup prop
3947    \seq_clear:N \l_tmpa_seq
```

```
3948    \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3949      name     = \l_stex_current_copymodule_name_str ,
3950      module   = \l_stex_current_module_str ,
3951      from     = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3952      includes = \l_tmpa_seq %,
3953    %  fields   = \l_tmpa_seq
3954    }
3955    \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3956      as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3957    \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_se
3958    \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,
3959
3960    \stex_if_do_html:T {
3961      \begin{stex_annotate_env} {#4} {
3962        \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3963      }
3964      \stex_annotate_invisible:nnn{domain}{\l_stex_import_ns_str ?\l_stex_import_name_str}{}
3965    }
3966  }
3967
3968  \cs_new_protected:Nn \stex_copymodule_end:n {
3969    % apply to every field
3970    \def \l_tmpa_cs ##1 ##2 {#1}
3971
3972    \tl_clear:N \__stex_copymodule_module_tl
3973    \tl_clear:N \__stex_copymodule_exec_tl
3974
3975    %\prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3976    \seq_clear:N \__stex_copymodule_fields_seq
3977
3978    \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
3979      \seq_map_inline:cn {c_stex_module_##1_constants}{
3980
3981        \tl_clear:N \__stex_copymodule_curr_symbol_tl % <- wrap in current symbol html
3982        \l_tmpa_cs{##1}{####1}
3983
3984        \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
3985          \str_set_eq:Nc \__stex_copymodule_curr_name_str {l__stex_copymodule_copymodule_##1?#
3986          \stex_if_do_html:T {
3987            \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
3988              \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?###
3989            }
3990          }
3991        }{
3992          \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str /
3993        }
3994
3995        \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
3996        \prop_put:Nnx \l_tmpa_prop { name } \__stex_copymodule_curr_name_str
3997        \prop_put:Nnx \l_tmpa_prop { module } \l_stex_current_module_str
3998
3999        \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
4000          \stex_if_do_html:T {
4001            \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
```

```
4002              $\stex_annotate_invisible:nnn{definiens}{}{\exp_after:wN \exp_not:N\csname l__st
4003            }
4004          }
4005          \prop_put:Nnn \l_tmpa_prop { defined } { true }
4006        }
4007
4008        \stex_add_constant_to_current_module:n \__stex_copymodule_curr_name_str
4009        \tl_put_right:Nx \__stex_copymodule_module_tl {
4010          \seq_clear:c {l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_n
4011          \prop_set_from_keyval:cn {
4012            l_stex_symdecl_\l_stex_current_module_str ? \__stex_copymodule_curr_name_str _prop
4013          }{
4014            \prop_to_keyval:N \l_tmpa_prop
4015          }
4016        }
4017
4018        \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
4019          \stex_if_do_html:T {
4020            \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4021              \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1
4022            }
4023          }
4024          \tl_put_right:Nx \__stex_copymodule_module_tl {
4025            \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
4026              \stex_invoke_symbol:n {
4027                \l_stex_current_module_str ? \__stex_copymodule_curr_name_str
4028              }
4029            }
4030          }
4031        }
4032
4033        \seq_put_right:Nx \__stex_copymodule_fields_seq {\l_stex_current_module_str ? \__stex_
4034
4035        \tl_put_right:Nx \__stex_copymodule_exec_tl {
4036          \stex_copy_notations:nn {\l_stex_current_module_str ? \__stex_copymodule_curr_name_s
4037        }
4038
4039        \tl_put_right:Nx \__stex_copymodule_exec_tl {
4040          \stex_if_do_html:TF{
4041            \stex_annotate_invisible:nnn{assignment} {##1?####1} { \exp_after:wN \exp_not:n \e
4042          }{
4043            \exp_after:wN \exp_not:n \exp_after:wN {\__stex_copymodule_curr_symbol_tl}
4044          }
4045        }
4046      }
4047    }
4048
4049
4050    \prop_put:Nno \l_stex_current_copymodule_prop {fields} \__stex_copymodule_fields_seq
4051    \tl_put_left:Nx \__stex_copymodule_module_tl {
4052      \prop_set_from_keyval:cn {
4053        l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _pro
4054      }{
4055        \prop_to_keyval:N \l_stex_current_copymodule_prop
```

```
4056        }
4057    }
4058
4059    \seq_gput_right:cx{c_stex_module_\l_stex_current_module_str _copymodules}{
4060      \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4061    }
4062
4063    \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4064    \stex_debug:nn{copymodule}{result:\meaning \__stex_copymodule_module_tl}
4065    \stex_debug:nn{copymodule}{output:\meaning \__stex_copymodule_exec_tl}
4066
4067    \__stex_copymodule_exec_tl
4068    \stex_if_do_html:T {
4069      \end{stex_annotate_env}
4070    }
4071 }
4072
4073 \NewDocumentEnvironment {copymodule} { O{} m m}{
4074    \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ copymodule }
4075    \stex_deactivate_macro:Nn \symdecl {module~environments}
4076    \stex_deactivate_macro:Nn \symdef {module~environments}
4077    \stex_deactivate_macro:Nn \notation {module~environments}
4078    \stex_reactivate_macro:N \assign
4079    \stex_reactivate_macro:N \renamedecl
4080    \stex_reactivate_macro:N \donotcopy
4081    \stex_smsmode_do:
4082 }{
4083    \stex_copymodule_end:n {}
4084 }
4085
4086 \NewDocumentEnvironment {interpretmodule} { O{} m m}{
4087    \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ interpretmodule }
4088    \stex_deactivate_macro:Nn \symdecl {module~environments}
4089    \stex_deactivate_macro:Nn \symdef {module~environments}
4090    \stex_deactivate_macro:Nn \notation {module~environments}
4091    \stex_reactivate_macro:N \assign
4092    \stex_reactivate_macro:N \renamedecl
4093    \stex_reactivate_macro:N \donotcopy
4094    \stex_smsmode_do:
4095 }{
4096    \stex_copymodule_end:n {
4097      \tl_if_exist:cF {
4098        l__stex_copymodule_copymodule_##1?##2_def_tl
4099      }{
4100        \str_if_eq:eeF {
4101          \prop_item:cn{
4102            l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
4103        }{ true }{
4104          \msg_error:nnxx{stex}{error/interpretmodule/nodefiniens}{
4105            ##1?##2
4106          }{\l_stex_current_copymodule_name_str}
4107        }
4108      }
4109    }
```

```
4110 }
4111
4112 \iffalse \begin{stex_annotate_env} \fi
4113 \NewDocumentEnvironment {realization} { O{} m}{
4114   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #2 }{ realize }
4115   \stex_deactivate_macro:Nn \symdecl {module~environments}
4116   \stex_deactivate_macro:Nn \symdef {module~environments}
4117   \stex_deactivate_macro:Nn \notation {module~environments}
4118   \stex_reactivate_macro:N \donotcopy
4119   \stex_reactivate_macro:N \assign
4120   \stex_smsmode_do:
4121 }{
4122   \stex_import_module_uri:nn { #1 } { #2 }
4123   \tl_clear:N \__stex_copymodule_exec_tl
4124   \tl_set:Nx \__stex_copymodule_module_tl {
4125     \stex_import_require_module:nnnn
4126       { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4127       { \l_stex_import_path_str } { \l_stex_import_name_str }
4128   }
4129
4130   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4131     \seq_map_inline:cn {c_stex_module_##1_constants}{
4132       \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str / # 
4133       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
4134         \stex_if_do_html:T {
4135           \tl_put_right:Nx \__stex_copymodule_exec_tl {
4136             \stex_annotate_invisible:nnn{assignment} {##1?####1} {
4137               $\stex_annotate_invisible:nnn{definiens}{}{\exp_after:wN \exp_not:N\csname l__
4138             }
4139           }
4140         }
4141         \tl_put_right:Nx \__stex_copymodule_module_tl {
4142           \prop_put:cnn {l_stex_symdecl_##1?####1_prop}{ defined }{ true }
4143         }
4144       }
4145   }}
4146
4147   \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4148
4149   \__stex_copymodule_exec_tl
4150   \stex_if_do_html:T {\end{stex_annotate_env}}
4151 }
4152
4153 \NewDocumentCommand \donotcopy { m }{
4154   \str_clear:N \l_stex_import_name_str
4155   \str_set:Nn \l_tmpa_str { #1 }
4156   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4157   \seq_map_inline:Nn \l_stex_all_modules_seq {
4158     \str_set:Nn \l_tmpb_str { ##1 }
4159     \str_if_eq:eeT { \l_tmpa_str } {
4160       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4161     } {
4162       \seq_map_break:n {
4163         \stex_if_do_html:T {
```

```
4164              \stex_if_smsmode:F {
4165                \stex_annotate_invisible:nnn{donotcopy}{##1}{
4166                  \stex_annotate:nnn{domain}{##1}{}
4167                }
4168              }
4169            }
4170            \str_set_eq:NN \l_stex_import_name_str \l_tmpb_str
4171          }
4172        }
4173        \seq_map_inline:cn {c_stex_module_##1_copymodules}{
4174          \str_set:Nn \l_tmpb_str { ####1 }
4175          \str_if_eq:eeT { \l_tmpa_str } {
4176            \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4177          } {
4178            \seq_map_break:n {\seq_map_break:n {
4179              \stex_if_do_html:T {
4180                \stex_if_smsmode:F {
4181                  \stex_annotate_invisible:nnn{donotcopy}{####1}{
4182                    \stex_annotate:nnn{domain}{
4183                      \prop_item:cn {l_stex_copymodule_ ####1 _prop}{module}
4184                    }{}
4185                  }
4186                }
4187              }
4188              \str_set:Nx \l_stex_import_name_str {
4189                \prop_item:cn {l_stex_copymodule_ ####1 _prop}{module}
4190              }
4191            }}
4192          }
4193        }
4194      }
4195      \str_if_empty:NTF \l_stex_import_name_str {
4196        % TODO throw error
4197      }{
4198        \stex_collect_imports:n {\l_stex_import_name_str }
4199        \seq_map_inline:Nn \l_stex_collect_imports_seq {
4200          \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
4201          \seq_map_inline:cn {c_stex_module_##1_constants}{
4202            \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? ####1 }
4203            \bool_lazy_any:nT {
4204              { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_name_str}}
4205              { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_macroname_str}}
4206              { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_def_tl}}
4207            }{
4208              % TODO throw error
4209            }
4210          }
4211        }
4212        \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
4213        \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_name_str }
4214        \prop_put:Nno \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
4215      }
4216      \stex_smsmode_do:
4217  }
```

```
4218
4219  \NewDocumentCommand \assign { m m }{
4220    \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
4221    \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
4222    \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _def_tl}{#2}
4223    \stex_smsmode_do:
4224  }
4225
4226  \keys_define:nn { stex / renamedecl } {
4227    name          .str_set_x:N  = \l_stex_renamedecl_name_str
4228  }
4229  \cs_new_protected:Nn \__stex_copymodule_renamedecl_args:n {
4230    \str_clear:N \l_stex_renamedecl_name_str
4231    \keys_set:nn { stex / renamedecl } { #1 }
4232  }
4233
4234  \NewDocumentCommand \renamedecl { O{} m m}{
4235    \__stex_copymodule_renamedecl_args:n { #1 }
4236    \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
4237    \stex_debug:nn{renamedecl}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
4238    \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
4239    \str_if_empty:NTF \l_stex_renamedecl_name_str {
4240      \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4241        \l_stex_get_symbol_uri_str
4242      } }
4243    } {
4244      \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex
4245      \stex_debug:nn{renamedecl}{@~\l_stex_current_module_str ? \l_stex_renamedecl_name_str}
4246      \prop_set_eq:cc {l_stex_symdecl_
4247        \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4248        _prop
4249      }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
4250      \seq_set_eq:cc {l_stex_symdecl_
4251        \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4252        _notations
4253      }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
4254      \prop_put:cnx {l_stex_symdecl_
4255        \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4256        _prop
4257      }{ name }{ \l_stex_renamedecl_name_str }
4258      \prop_put:cnx {l_stex_symdecl_
4259        \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4260        _prop
4261      }{ module }{ \l_stex_current_module_str }
4262      \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
4263        \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4264      }
4265      \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4266        \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4267      } }
4268    }
4269    \stex_smsmode_do:
4270  }
4271
```

```
4272  \stex_deactivate_macro:Nn \assign {copymodules}
4273  \stex_deactivate_macro:Nn \renamedecl {copymodules}
4274  \stex_deactivate_macro:Nn \donotcopy {copymodules}
4275
4276
```

31.2   The feature environment

```
4277  ⟨@@=stex_features⟩
4278
4279  \NewDocumentEnvironment{structural_feature_module}{ m m m }{
4280    \stex_if_in_module:F {
4281      \msg_set:nnn{stex}{error/nomodule}{
4282        Structural~Feature~has~to~occur~in~a~module:\\
4283        Feature~#2~of~type~#1\\
4284        In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
4285      }
4286      \msg_error:nn{stex}{error/nomodule}
4287    }
4288
4289    \str_set_eq:NN \l_stex_feature_parent_str \l_stex_current_module_str
4290
4291    \stex_module_setup:nn{meta=NONE}{#2 - #1}
4292
4293    \stex_if_do_html:T {
4294      \begin{stex_annotate_env}{ feature:#1 }{\l_stex_feature_parent_str ? #2 - #1}
4295        \stex_annotate_invisible:nnn{header}{}{ #3 }
4296    }
4297  }{
4298    \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
4299    \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
4300    \stex_debug:nn{features}{
4301      Feature: \l_stex_last_feature_str
4302    }
4303    \stex_if_do_html:T {
4304      \end{stex_annotate_env}
4305    }
4306  }
```

31.3   Structure

```
4307  ⟨@@=stex_structures⟩
4308  \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
4309    \prop_if_exist:cF {c_stex_module_\l_stex_current_module_str _structures}{
4310      \prop_new:c {c_stex_module_\l_stex_current_module_str _structures}
4311    }
4312    \prop_gput:cxx{c_stex_module_\l_stex_current_module_str _structures}
4313      {#1}{#2}
4314  }
4315
```

```
4316 \keys_define:nn { stex / features / structure } {
4317   name           .str_set_x:N  = \l__stex_structures_name_str ,
4318 }
4319
4320 \cs_new_protected:Nn \__stex_structures_structure_args:n {
4321   \str_clear:N \l__stex_structures_name_str
4322   \keys_set:nn { stex / features / structure } { #1 }
4323 }
4324
4325 \NewDocumentEnvironment{mathstructure}{m O{}}{
4326   \__stex_structures_structure_args:n { #2 }
4327   \str_if_empty:NT \l__stex_structures_name_str {
4328     \str_set:Nx \l__stex_structures_name_str { #1 }
4329   }
4330   \stex_suppress_html:n {
4331     \exp_args:Nx \stex_symdecl_do:nn {
4332       name = \l__stex_structures_name_str ,
4333       def  = {\STEXsymbol{module-type}{
4334         \_stex_term_math_oms:nnnn {
4335           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4336             { ns } ?
4337             \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4338               { name } / \l__stex_structures_name_str - structure
4339         }{}{0}{}
4340       }}
4341     }{ #1 }
4342   }
4343   \exp_args:Nnnx
4344   \begin{structural_feature_module}{ structure }
4345     { \l__stex_structures_name_str }{}
4346   \stex_smsmode_do:
4347 }{
4348   \end{structural_feature_module}
4349   \_stex_reset_up_to_module:n \l_stex_last_feature_str
4350   \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
4351   \seq_clear:N \l_tmpa_seq
4352   \seq_map_inline:Nn \l_stex_collect_imports_seq {
4353     \seq_map_inline:cn{c_stex_module_##1_constants}{
4354       \seq_put_right:Nn \l_tmpa_seq { ##1 ? ####1 }
4355     }
4356   }
4357   \exp_args:Nnno
4358   \prop_gput:cnn {c_stex_module_ \l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
4359   \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
4360   \stex_add_structure_to_current_module:nn
4361     \l__stex_structures_name_str
4362     \l_stex_last_feature_str
4363
4364   \stex_execute_in_module:x {
4365     \tl_set:cn { #1 }{
4366       \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structure
4367     }
4368   }
4369 }
```

192

```
4370
4371  \cs_new:Nn \stex_invoke_structure:nn {
4372    \stex_invoke_symbol:n { #1?#2 }
4373  }
4374
4375  \cs_new_protected:Nn \stex_get_structure:n {
4376    \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4377      \tl_set:Nn \l_tmpa_tl { #1 }
4378      \__stex_structures_get_from_cs:
4379    }{
4380      \cs_if_exist:cTF { #1 }{
4381        \cs_set_eq:Nc \l_tmpa_cs { #1 }
4382        \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
4383        \str_if_empty:NTF \l_tmpa_str {
4384          \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs} \stex_invoke_structure:nn {
4385            \__stex_structures_get_from_cs:
4386          }{
4387            \__stex_structures_get_from_string:n { #1 }
4388          }
4389        }{
4390          \__stex_structures_get_from_string:n { #1 }
4391        }
4392      }{
4393        \__stex_structures_get_from_string:n { #1 }
4394      }
4395    }
4396  }
4397
4398  \cs_new_protected:Nn \__stex_structures_get_from_cs: {
4399    \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4400      { \tl_tail:N \l_tmpa_tl }
4401    \str_set:Nx \l_tmpa_str {
4402      \exp_after:wN \use_i:nn \l_tmpa_tl
4403    }
4404    \str_set:Nx \l_tmpb_str {
4405      \exp_after:wN \use_ii:nn \l_tmpa_tl
4406    }
4407    \str_set:Nx \l_stex_get_structure_str {
4408      \l_tmpa_str ? \l_tmpb_str
4409    }
4410    \str_set:Nx \l_stex_get_structure_module_str {
4411      \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
4412    }
4413  }
4414
4415  \cs_new_protected:Nn \__stex_structures_get_from_string:n {
4416    \tl_set:Nn \l_tmpa_tl {
4417      \msg_error:nnn{stex}{error/unknownstructure}{#1}
4418    }
4419    \str_set:Nn \l_tmpa_str { #1 }
4420    \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4421
4422    \seq_map_inline:Nn \l_stex_all_modules_seq {
4423      \prop_if_exist:cT {c_stex_module_##1_structures} {
```

```
4424        \prop_map_inline:cn {c_stex_module_##1_structures} {
4425          \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
4426            \prop_map_break:n{\seq_map_break:n{
4427              \tl_set:Nn \l_tmpa_tl {
4428                \str_set:Nn \l_stex_get_structure_str {##1?####1}
4429                \str_set:Nn \l_stex_get_structure_module_str {####2}
4430              }
4431            }}
4432          }
4433        }
4434      }
4435    }
4436    \l_tmpa_tl
4437 }
```

```
4438
4439 \keys_define:nn { stex / instantiate } {
4440    name          .str_set_x:N  = \l__stex_structures_name_str
4441 }
4442 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
4443    \str_clear:N \l__stex_structures_name_str
4444    \keys_set:nn { stex / instantiate } { #1 }
4445 }
4446
4447 \NewDocumentCommand \instantiate {m O{} m m m}{
4448    \begingroup
4449      \stex_get_structure:n {#4}
4450      \__stex_structures_instantiate_args:n { #2 }
4451      \str_if_empty:NT \l__stex_structures_name_str {
4452        \str_set:Nn \l__stex_structures_name_str { #1 }
4453      }
4454      \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4455      \seq_clear:N \l__stex_structures_fields_seq
4456      \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4457      \seq_map_inline:Nn \l_stex_collect_imports_seq {
4458        \seq_map_inline:cn {c_stex_module_##1_constants}{
4459          \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4460        }
4461      }
4462
4463      \tl_if_empty:nF{#3}{
4464        \seq_set_split:Nnn \l_tmpa_seq , {#3}
4465        \prop_clear:N \l_tmpa_prop
4466        \seq_map_inline:Nn \l_tmpa_seq {
4467          \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4468          \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4469            \msg_error:nnn{stex}{error/keyval}{##1}
4470          }
4471          \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
4472          \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4473          \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
4474          \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}
4475          \exp_args:Nxx \str_if_eq:nnF
```

194

```
4476            {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4477            {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
4478          \msg_error:nnxxxx{stex}{error/incompatible}
4479            {\l__stex_structures_dom_str}
4480            {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4481            {\l_stex_get_symbol_uri_str}
4482            {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
4483        }
4484        \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str
4485      }
4486    }
4487
4488    \seq_map_inline:Nn \l__stex_structures_fields_seq {
4489      \str_set:Nx \l_tmpa_str {field:\l__stex_structures_name_str . \prop_item:cn {l_stex_sy
4490      \stex_debug:nn{instantiate}{Field~\l_tmpa_str :~##1}
4491
4492      \stex_add_constant_to_current_module:n {\l_tmpa_str}
4493      \stex_execute_in_module:x {
4494        \prop_set_from_keyval:cn { l_stex_symdecl_ \l_stex_current_module_str?\l_tmpa_str _p
4495          name  = \l_tmpa_str ,
4496          args  = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4497          arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4498          assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4499        }
4500        \seq_clear:c {l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _notations}
4501      }
4502
4503      \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4504        \stex_find_notation:nn{##1}{}
4505        \stex_execute_in_module:x {
4506          \seq_put_right:cn {l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _notation
4507        }
4508
4509        \stex_copy_control_sequence:ccN
4510          {stex_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_notation_
4511          {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4512          \l_tmpa_tl
4513        \exp_args:No \stex_execute_in_module:n \l_tmpa_tl
4514
4515
4516        \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
4517          \tl_set_eq:Nc \l_tmpa_cs {stex_op_notation_##1\c_hash_str \l_stex_notation_variant
4518          \stex_execute_in_module:x {
4519            \tl_set:cn
4520            {stex_op_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_nota
4521            { \exp_args:No \exp_not:n \l_tmpa_cs}
4522          }
4523        }
4524
4525      }
4526
4527      \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\l_stex_cur
4528    }
4529
```

```
4530      \stex_execute_in_module:x {
4531        \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
4532          domain = \l_stex_get_structure_module_str ,
4533          \prop_to_keyval:N \l_tmpa_prop
4534        }
4535        \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
4536      }
4537      \stex_debug:nn{instantiate}{
4538        Instance~\l_stex_current_module_str?\l__stex_structures_name_str \\
4539        \prop_to_keyval:N \l_tmpa_prop
4540      }
4541      \exp_args:Nxx \stex_symdecl_do:nn {
4542        type={\STEXsymbol{module-type}{
4543          \_stex_term_math_oms:nnnn {
4544            \l_stex_get_structure_module_str
4545          }{}{0}{}
4546        }}
4547      }{\l__stex_structures_name_str}
4548 %    {
4549        \str_set:Nx \l_stex_get_symbol_uri_str {\l_stex_current_module_str?\l__stex_structures
4550        \tl_set:Nn \l_stex_notation_after_do_tl {\__stex_notation_final:}
4551        \stex_notation_do:nnnnn{}{0}{}{}{\comp{#5}}
4552 %  }
4553      %\exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}
4554    \endgroup
4555    \stex_smsmode_do:\ignorespacesandpars
4556 }
4557
4558 \cs_new_protected:Nn \stex_symbol_or_var:n {
4559    \cs_if_exist:cTF{#1}{
4560      \cs_set_eq:Nc \l_tmpa_tl { #1 }
4561      \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4562      \str_if_empty:NTF \l_tmpa_str {
4563        \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
4564          \stex_invoke_variable:n {
4565            \bool_set_true:N \l_stex_symbol_or_var_bool
4566            \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
4567            \str_set:Nx \l_stex_get_symbol_uri_str {
4568              \exp_after:wN \use:n \l_tmpa_tl
4569            }
4570          }{
4571            \bool_set_false:N \l_stex_symbol_or_var_bool
4572            \stex_get_symbol:n{#1}
4573          }
4574      }{
4575        \__stex_structures_symbolorvar_from_string:n{ #1 }
4576      }
4577    }{
4578      \__stex_structures_symbolorvar_from_string:n{ #1 }
4579    }
4580 }
4581
4582 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
4583    \prop_if_exist:cTF {l_stex_variable_#1 _prop}{
```

196

```
4584        \bool_set_true:N \l_stex_symbol_or_var_bool
4585        \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
4586      }{
4587        \bool_set_false:N \l_stex_symbol_or_var_bool
4588        \stex_get_symbol:n{#1}
4589      }
4590    }
4591
4592    \keys_define:nn { stex / varinstantiate } {
4593      name          .str_set_x:N  = \l__stex_structures_name_str,
4594      bind          .choices:nn   =
4595        {forall,exists}
4596        {\str_set:Nx \l__stex_structures_bind_str {\l_keys_choice_tl}}
4597
4598    }
4599    \cs_new_protected:Nn \__stex_structures_varinstantiate_args:n {
4600      \str_clear:N \l__stex_structures_name_str
4601      \str_clear:N \l__stex_structures_bind_str
4602      \keys_set:nn { stex / varinstantiate } { #1 }
4603    }
4604
4605    \NewDocumentCommand \varinstantiate {m O{} m m m}{
4606      \begingroup
4607        \stex_get_structure:n {#4}
4608        \__stex_structures_varinstantiate_args:n { #2 }
4609        \str_if_empty:NT \l__stex_structures_name_str {
4610          \str_set:Nn \l__stex_structures_name_str { #1 }
4611        }
4612        \stex_if_do_html:TF{
4613          \stex_annotate:nnn{varinstance}{\l__stex_structures_name_str}
4614        }{\use:n}
4615        {
4616          \stex_if_do_html:T{
4617            \stex_annotate_invisible:nnn{domain}{\l_stex_get_structure_module_str}{}
4618          }
4619          \seq_clear:N \l__stex_structures_fields_seq
4620          \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4621          \seq_map_inline:Nn \l_stex_collect_imports_seq {
4622            \seq_map_inline:cn {c_stex_module_##1_constants}{
4623              \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4624            }
4625          }
4626          \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4627          \prop_clear:N \l_tmpa_prop
4628          \tl_if_empty:nF {#3} {
4629            \seq_set_split:Nnn \l_tmpa_seq , {#3}
4630            \seq_map_inline:Nn \l_tmpa_seq {
4631              \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4632              \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4633                \msg_error:nnn{stex}{error/keyval}{##1}
4634              }
4635              \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_stru
4636              \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4637              \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol
```

197

```
4638                \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
4639                \stex_if_do_html:T{
4640                  \stex_annotate:nnn{assign}{\l__stex_structures_dom_str,\l_stex_get_symbol_uri_st
4641                }
4642                \bool_if:NTF \l_stex_symbol_or_var_bool {
4643                  \exp_args:Nxx \str_if_eq:nnF
4644                    {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4645                    {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}{
4646                    \msg_error:nnxxxx{stex}{error/incompatible}
4647                      {\l__stex_structures_dom_str}
4648                      {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4649                      {\l_stex_get_symbol_uri_str}
4650                      {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}
4651                  }
4652                  \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n
4653                }{
4654                  \exp_args:Nxx \str_if_eq:nnF
4655                    {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4656                    {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
4657                    \msg_error:nnxxxx{stex}{error/incompatible}
4658                      {\l__stex_structures_dom_str}
4659                      {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4660                      {\l_stex_get_symbol_uri_str}
4661                      {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
4662                  }
4663                  \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n {
4664                }
4665              }
4666            }
4667          \tl_gclear:N \g__stex_structures_aftergroup_tl
4668          \seq_map_inline:Nn \l__stex_structures_fields_seq {
4669            \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdec
4670            \stex_debug:nn{varinstantiate}{Field~\l_tmpa_str :~##1}
4671            \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4672              \stex_find_notation:nn{##1}{}
4673              \cs_gset_eq:cc{g__stex_structures_tmpa_\l_tmpa_str _cs}
4674                {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4675              \stex_debug:nn{varinstantiate}{Notation:~\cs_meaning:c{g__stex_structures_tmpa_\l_
4676              \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
4677                \cs_gset_eq:cc {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
4678                  {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4679                  \stex_debug:nn{varinstantiate}{Operator~Notation:~\cs_meaning:c{g__stex_struct
4680            }
4681          }
4682
4683          \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4684            \prop_set_from_keyval:cn { l_stex_variable_ \l_tmpa_str _prop}{
4685              name  = \l_tmpa_str ,
4686              args  = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4687              arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4688              assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4689            }
4690            \cs_set_eq:cc {stex_var_notation_\l_tmpa_str _cs}
4691              {g__stex_structures_tmpa_\l_tmpa_str _cs}
```

198

```
4692                 \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str _cs}
4693                   {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
4694               }
4695             \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\stex_inv
4696           }
4697         \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4698           \prop_set_from_keyval:cn {l_stex_varinstance_\l__stex_structures_name_str _prop }{
4699             domain = \l_stex_get_structure_module_str ,
4700             \prop_to_keyval:N \l_tmpa_prop
4701           }
4702         \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
4703         \tl_set:cn {l_stex_varinstance_\l__stex_structures_name_str _op_tl}{
4704           \exp_args:Nnx \exp_not:N \use:nn {
4705             \str_set:Nn \exp_not:N \l_stex_current_symbol_str {var://\l__stex_structures_nam
4706             \_stex_term_omv:nn {var://\l__stex_structures_name_str}{
4707               \exp_not:n{
4708                 \_varcomp{#5}
4709               }
4710             }
4711           }{
4712             \exp_not:n{\_stex_reset:N \l_stex_current_symbol_str}
4713           }
4714         }
4715       }
4716     }
4717     \stex_debug:nn{varinstantiate}{\expandafter\detokenize\expandafter{\g__stex_structures_a
4718     \aftergroup\g__stex_structures_aftergroup_tl
4719   \endgroup
4720   \stex_smsmode_do:\ignorespacesandpars
4721 }
4722
4723 \cs_new_protected:Nn \stex_invoke_instance:n {
4724   \peek_charcode_remove:NTF ! {
4725     \stex_invoke_symbol:n{#1}
4726   }{
4727     \_stex_invoke_instance:nn {#1}
4728   }
4729 }
4730
4731
4732 \cs_new_protected:Nn \stex_invoke_varinstance:n {
4733   \peek_charcode_remove:NTF ! {
4734     \exp_args:Nnx \use:nn {
4735       \def\comp{\_varcomp}
4736       \use:c{l_stex_varinstance_#1_op_tl}
4737     }{
4738       \_stex_reset:N \comp
4739     }
4740   }{
4741     \_stex_invoke_varinstance:nn {#1}
4742   }
4743 }
4744
4745 \cs_new_protected:Nn \_stex_invoke_instance:nn {
```

199

```
4746        \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
4747          \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
4748        }{
4749          \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
4750          \msg_error:nnxxx{stex}{error/unknownfield}{#2}{#1}{
4751            \prop_to_keyval:N \l_tmpa_prop
4752          }
4753        }
4754    }
4755
4756    \cs_new_protected:Nn \_stex_invoke_varinstance:nn {
4757      \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
4758        \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
4759        \l_tmpa_tl
4760      }{
4761        \msg_error:nnnnn{stex}{error/unknownfield}{#2}{#1}{}
4762      }
4763    }
```

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

`\stex_invoke_structure:nnn`

```
4764    % #1: URI of the instance
4765    % #2: URI of the instantiated module
4766    \cs_new_protected:Nn \stex_invoke_structure:nnn {
4767      \tl_if_empty:nTF{ #3 }{
4768        \prop_set_eq:Nc \l__stex_structures_structure_prop {
4769          c_stex_feature_ #2 _prop
4770        }
4771        \tl_clear:N \l_tmpa_tl
4772        \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
4773        \seq_map_inline:Nn \l_tmpa_seq {
4774          \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
4775          \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
4776          \cs_if_exist:cT {
4777            stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
4778          }{
4779            \tl_if_empty:NF \l_tmpa_tl {
4780              \tl_put_right:Nn \l_tmpa_tl {,}
4781            }
4782            \tl_put_right:Nx \l_tmpa_tl {
4783              \stex_invoke_symbol:n {#1/\l_tmpa_str}!
4784            }
4785          }
4786        }
4787        \exp_args:No \mathstruct \l_tmpa_tl
4788      }{
4789        \stex_invoke_symbol:n{#1/#3}
4790      }
4791    }
```

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

```
4792    ⟨/package⟩
```

Chapter 32

STEX
-Statements Implementation

4793 ⟨∗package⟩
4794
4795 %%%%%%%%%%%%   features.dtx   %%%%%%%%%%%%
4796
4797 ⟨@@=stex_statements⟩

Warnings and error messages

4798

\titleemph

4799 \def\titleemph#1{\textbf{#1}}

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

32.1   Definitions

definiendum

```
4800  \keys_define:nn {stex / definiendum }{
4801    pre      .tl_set:N     = \l__stex_statements_definiendum_pre_tl,
4802    post     .tl_set:N     = \l__stex_statements_definiendum_post_tl,
4803    root     .str_set_x:N  = \l__stex_statements_definiendum_root_str,
4804    gfa      .str_set_x:N  = \l__stex_statements_definiendum_gfa_str
4805  }
4806  \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
4807    \str_clear:N \l__stex_statements_definiendum_root_str
4808    \tl_clear:N \l__stex_statements_definiendum_post_tl
4809    \str_clear:N \l__stex_statements_definiendum_gfa_str
4810    \keys_set:nn { stex / definiendum }{ #1 }
4811  }
4812  \NewDocumentCommand \definiendum { O{} m m} {
4813    \__stex_statements_definiendum_args:n { #1 }
4814    \stex_get_symbol:n { #2 }
4815    \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4816    \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
4817      \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

```
4818          \tl_set:Nn \l_tmpa_tl { #3 }
4819       } {
4820          \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
4821          \tl_set:Nn \l_tmpa_tl {
4822             \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
4823          }
4824       }
4825    } {
4826       \tl_set:Nn \l_tmpa_tl { #3 }
4827    }
4828
4829    % TODO root
4830    \stex_html_backend:TF {
4831       \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
4832    } {
4833       \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
4834    }
4835 }
4836 \stex_deactivate_macro:Nn \definiendum {definition~environments}
```

(*End definition for* `definiendum`. *This function is documented on page* *41*.)

definame

```
4837
4838 \NewDocumentCommand \definame { O{} m } {
4839    \__stex_statements_definiendum_args:n { #1 }
4840    % TODO: root
4841    \stex_get_symbol:n { #2 }
4842    \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4843    \str_set:Nx \l_tmpa_str {
4844       \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4845    }
4846    \str_replace_all:Nnn \l_tmpa_str {-} {~}
4847    \stex_html_backend:TF {
4848       \stex_if_do_html:T {
4849          \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4850             \l_tmpa_str\l__stex_statements_definiendum_post_tl
4851          }
4852       }
4853    } {
4854       \exp_args:Nnx \defemph@uri {
4855          \l_tmpa_str\l__stex_statements_definiendum_post_tl
4856       } { \l_stex_get_symbol_uri_str }
4857    }
4858 }
4859 \stex_deactivate_macro:Nn \definame {definition~environments}
4860
4861 \NewDocumentCommand \Definame { O{} m } {
4862    \__stex_statements_definiendum_args:n { #1 }
4863    \stex_get_symbol:n { #2 }
4864    \str_set:Nx \l_tmpa_str {
4865       \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4866    }
4867    \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
```

```
4868    \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4869    \stex_html_backend:TF {
4870      \stex_if_do_html:T {
4871        \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4872          \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
4873        }
4874      }
4875    } {
4876      \exp_args:Nnx \defemph@uri {
4877        \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
4878      } { \l_stex_get_symbol_uri_str }
4879    }
4880  }
4881  \stex_deactivate_macro:Nn \Definame {definition~environments}
4882
4883  \NewDocumentCommand \premise { m }{
4884    \stex_annotate:nnn{ premise }{}{ #1 }
4885  }
4886  \NewDocumentCommand \conclusion { m }{
4887    \stex_annotate:nnn{ conclusion }{}{ #1 }
4888  }
4889  \NewDocumentCommand \definiens { O{} m }{
4890    \str_clear:N \l_stex_get_symbol_uri_str
4891    \tl_if_empty:nF {#1} {
4892      \stex_get_symbol:n { #1 }
4893    }
4894    \str_if_empty:NT \l_stex_get_symbol_uri_str {
4895      \int_compare:nNnTF {\clist_count:N \l__stex_statements_sdefinition_for_clist} = 1 {
4896        \str_set:Nx \l_stex_get_symbol_uri_str {\clist_item:Nn \l__stex_statements_sdefinition
4897      }{
4898        % TODO throw error
4899      }
4900    }
4901    \str_if_eq:eeT {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{module}}
4902      {\l_stex_current_module_str}{
4903        \str_if_eq:eeF {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defin
4904        {true}{
4905          \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
4906          \exp_args:Nx \stex_add_to_current_module:n {
4907            \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
4908          }
4909        }
4910    }
4911    \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
4912  }
4913
4914  \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
4915  \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
4916  \stex_deactivate_macro:Nn \definiens {definition~environments}
4917
```

(*End definition for* `definame`*. This function is documented on page* *41.*)

sdefinition

```
4918
4919  \keys_define:nn {stex / sdefinition }{
4920    type    .str_set_x:N  = \sdefinitiontype,
4921    id      .str_set_x:N  = \sdefinitionid,
4922    name    .str_set_x:N  = \sdefinitionname,
4923    for     .clist_set:N  = \l__stex_statements_sdefinition_for_clist ,
4924    title   .tl_set:N     = \sdefinitiontitle
4925  }
4926  \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
4927    \str_clear:N \sdefinitiontype
4928    \str_clear:N \sdefinitionid
4929    \str_clear:N \sdefinitionname
4930    \clist_clear:N \l__stex_statements_sdefinition_for_clist
4931    \tl_clear:N \sdefinitiontitle
4932    \keys_set:nn { stex / sdefinition }{ #1 }
4933  }
4934
4935  \NewDocumentEnvironment{sdefinition}{O{}}{
4936    \__stex_statements_sdefinition_args:n{ #1 }
4937    \stex_reactivate_macro:N \definiendum
4938    \stex_reactivate_macro:N \definame
4939    \stex_reactivate_macro:N \Definame
4940    \stex_reactivate_macro:N \premise
4941    \stex_reactivate_macro:N \definiens
4942    \stex_if_smsmode:F{
4943      \seq_clear:N \l_tmpa_seq
4944      \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4945        \tl_if_empty:nF{ ##1 }{
4946          \stex_get_symbol:n { ##1 }
4947          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4948            \l_stex_get_symbol_uri_str
4949          }
4950        }
4951      }
4952      \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpa_seq
4953      \exp_args:Nnnx
4954      \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {,}}
4955      \str_if_empty:NF \sdefinitiontype {
4956        \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
4957      }
4958      \str_if_empty:NF \sdefinitionname {
4959        \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
4960      }
4961      \clist_set:No \l_tmpa_clist \sdefinitiontype
4962      \tl_clear:N \l_tmpa_tl
4963      \clist_map_inline:Nn \l_tmpa_clist {
4964        \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
4965          \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
4966        }
4967      }
4968      \tl_if_empty:NTF \l_tmpa_tl {
4969        \__stex_statements_sdefinition_start:
4970      }{
4971        \l_tmpa_tl
```

```
4972        }
4973      }
4974      \stex_ref_new_doc_target:n \sdefinitionid
4975      \stex_smsmode_do:
4976 }{
4977      \stex_suppress_html:n {
4978        \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
4979      }
4980      \stex_if_smsmode:F {
4981        \clist_set:No \l_tmpa_clist \sdefinitiontype
4982        \tl_clear:N \l_tmpa_tl
4983        \clist_map_inline:Nn \l_tmpa_clist {
4984          \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
4985            \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
4986          }
4987        }
4988        \tl_if_empty:NTF \l_tmpa_tl {
4989          \__stex_statements_sdefinition_end:
4990        }{
4991          \l_tmpa_tl
4992        }
4993        \end{stex_annotate_env}
4994      }
4995 }
```

```
4996 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
4997      \par\noindent\titleemph{Definition\tl_if_empty:NF \sdefinitiontitle {
4998        ~(\sdefinitiontitle)
4999      }~}
5000 }
5001 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\par\medskip}
5002
5003 \newcommand\stexpatchdefinition[3][] {
5004      \str_set:Nx \l_tmpa_str{ #1 }
5005      \str_if_empty:NTF \l_tmpa_str {
5006        \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
5007        \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
5008      }{
5009        \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2
5010        \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
5011      }
5012 }
```

(*End definition for* \stexpatchdefinition*. This function is documented on page 43.*)

\inlinedef inline:

```
5013 \keys_define:nn {stex / inlinedef }{
5014    type     .str_set_x:N  = \sdefinitiontype,
5015    id       .str_set_x:N  = \sdefinitionid,
5016    for      .clist_set:N   = \l__stex_statements_sdefinition_for_clist ,
5017    name     .str_set_x:N  = \sdefinitionname
5018 }
5019 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
```

205

```
5020    \str_clear:N \sdefinitiontype
5021    \str_clear:N \sdefinitionid
5022    \str_clear:N \sdefinitionname
5023    \clist_clear:N \l__stex_statements_sdefinition_for_clist
5024    \keys_set:nn { stex / inlinedef }{ #1 }
5025 }
5026 \NewDocumentCommand \inlinedef { O{} m } {
5027    \begingroup
5028    \__stex_statements_inlinedef_args:n{ #1 }
5029    \stex_reactivate_macro:N \definiendum
5030    \stex_reactivate_macro:N \definame
5031    \stex_reactivate_macro:N \Definame
5032    \stex_reactivate_macro:N \premise
5033    \stex_reactivate_macro:N \definiens
5034    \stex_ref_new_doc_target:n \sdefinitionid
5035    \stex_if_smsmode:TF{\stex_suppress_html:n {
5036       \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5037    }}{
5038       \seq_clear:N \l_tmpa_seq
5039       \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5040          \tl_if_empty:nF{ ##1 }{
5041             \stex_get_symbol:n { ##1 }
5042             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5043                \l_stex_get_symbol_uri_str
5044             }
5045          }
5046       }
5047       \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpa_seq
5048       \exp_args:Nnx
5049       \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
5050          \str_if_empty:NF \sdefinitiontype {
5051             \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5052          }
5053          #2
5054          \str_if_empty:NF \sdefinitionname {
5055             \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sdefinitionname}}
5056             \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5057          }
5058       }
5059    }
5060    \endgroup
5061    \stex_smsmode_do:
5062 }
```

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

32.2 Assertions

sassertion

```
5063
5064 \keys_define:nn {stex / sassertion }{
5065    type    .str_set_x:N  = \sassertiontype,
5066    id      .str_set_x:N  = \sassertionid,
```

```
5067    title    .tl_set:N      = \sassertiontitle ,
5068    for      .clist_set:N  = \l__stex_statements_sassertion_for_clist ,
5069    name     .str_set_x:N  = \sassertionname
5070  }
5071  \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
5072    \str_clear:N \sassertiontype
5073    \str_clear:N \sassertionid
5074    \str_clear:N \sassertionname
5075    \clist_clear:N \l__stex_statements_sassertion_for_clist
5076    \tl_clear:N \sassertiontitle
5077    \keys_set:nn { stex / sassertion }{ #1 }
5078  }
5079
5080  %\tl_new:N \g__stex_statements_aftergroup_tl
5081
5082  \NewDocumentEnvironment{sassertion}{O{}}{
5083    \__stex_statements_sassertion_args:n{ #1 }
5084    \stex_reactivate_macro:N \premise
5085    \stex_reactivate_macro:N \conclusion
5086    \stex_if_smsmode:F {
5087      \seq_clear:N \l_tmpa_seq
5088      \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5089        \tl_if_empty:nF{ ##1 }{
5090          \stex_get_symbol:n { ##1 }
5091          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5092            \l_stex_get_symbol_uri_str
5093          }
5094        }
5095      }
5096      \exp_args:Nnnx
5097      \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}
5098      \str_if_empty:NF \sassertiontype {
5099        \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
5100      }
5101      \str_if_empty:NF \sassertionname {
5102        \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
5103      }
5104      \clist_set:No \l_tmpa_clist \sassertiontype
5105      \tl_clear:N \l_tmpa_tl
5106      \clist_map_inline:Nn \l_tmpa_clist {
5107        \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
5108          \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
5109        }
5110      }
5111      \tl_if_empty:NTF \l_tmpa_tl {
5112        \__stex_statements_sassertion_start:
5113      }{
5114        \l_tmpa_tl
5115      }
5116    }
5117    \str_if_empty:NTF \sassertionid {
5118      \str_if_empty:NF \sassertionname {
5119        \stex_ref_new_doc_target:n {}
5120      }
```

```
5121        } {
5122          \stex_ref_new_doc_target:n \sassertionid
5123        }
5124      \stex_smsmode_do:
5125    }{
5126      \str_if_empty:NF \sassertionname {
5127        \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5128        \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5129      }
5130      \stex_if_smsmode:F {
5131        \clist_set:No \l_tmpa_clist \sassertiontype
5132        \tl_clear:N \l_tmpa_tl
5133        \clist_map_inline:Nn \l_tmpa_clist {
5134          \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
5135            \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
5136          }
5137        }
5138        \tl_if_empty:NTF \l_tmpa_tl {
5139          \__stex_statements_sassertion_end:
5140        }{
5141          \l_tmpa_tl
5142        }
5143        \end{stex_annotate_env}
5144      }
5145    }
```

\stexpatchassertion

```
5146
5147  \cs_new_protected:Nn \__stex_statements_sassertion_start: {
5148    \par\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {
5149      (\sassertiontitle)
5150    }~}
5151  }
5152  \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
5153
5154  \newcommand\stexpatchassertion[3][] {
5155    \str_set:Nx \l_tmpa_str{ #1 }
5156    \str_if_empty:NTF \l_tmpa_str {
5157      \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
5158      \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
5159    }{
5160      \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
5161      \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3
5162    }
5163  }
```

(*End definition for* \stexpatchassertion. *This function is documented on page 43.*)

\inlineass   inline:

```
5164  \keys_define:nn {stex / inlineass }{
5165    type     .str_set_x:N  = \sassertiontype,
5166    id       .str_set_x:N  = \sassertionid,
5167    for      .clist_set:N  = \l__stex_statements_sassertion_for_clist ,
5168    name     .str_set_x:N  = \sassertionname
```

```
5169 }
5170 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
5171   \str_clear:N \sassertiontype
5172   \str_clear:N \sassertionid
5173   \str_clear:N \sassertionname
5174   \clist_clear:N \l__stex_statements_sassertion_for_clist
5175   \keys_set:nn { stex / inlineass }{ #1 }
5176 }
5177 \NewDocumentCommand \inlineass { O{} m } {
5178   \begingroup
5179   \stex_reactivate_macro:N \premise
5180   \stex_reactivate_macro:N \conclusion
5181   \__stex_statements_inlineass_args:n{ #1 }
5182   \str_if_empty:NTF \sassertionid {
5183     \str_if_empty:NF \sassertionname {
5184       \stex_ref_new_doc_target:n {}
5185     }
5186   } {
5187     \stex_ref_new_doc_target:n \sassertionid
5188   }
5189
5190   \stex_if_smsmode:TF{
5191     \str_if_empty:NF \sassertionname {
5192       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5193       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5194     }
5195   }{
5196     \seq_clear:N \l_tmpa_seq
5197     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5198       \tl_if_empty:nF{ ##1 }{
5199         \stex_get_symbol:n { ##1 }
5200         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5201           \l_stex_get_symbol_uri_str
5202         }
5203       }
5204     }
5205     \exp_args:Nnx
5206     \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{
5207       \str_if_empty:NF \sassertiontype {
5208         \stex_annotate_invisible:nnn{typestrings}{\sassertiontype}{}
5209       }
5210       #2
5211       \str_if_empty:NF \sassertionname {
5212         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5213         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5214         \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
5215       }
5216     }
5217   }
5218   \endgroup
5219   \stex_smsmode_do:
5220 }
```

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

32.3   Examples

```
5221
5222 \keys_define:nn {stex / sexample }{
5223   type    .str_set_x:N  = \exampletype,
5224   id      .str_set_x:N  = \sexampleid,
5225   title   .tl_set:N     = \sexampletitle,
5226   name    .str_set_x:N  = \sexamplename ,
5227   for     .clist_set:N  = \l__stex_statements_sexample_for_clist,
5228 }
5229 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
5230   \str_clear:N \sexampletype
5231   \str_clear:N \sexampleid
5232   \str_clear:N \sexamplename
5233   \tl_clear:N \sexampletitle
5234   \clist_clear:N \l__stex_statements_sexample_for_clist
5235   \keys_set:nn { stex / sexample }{ #1 }
5236 }
5237
5238 \NewDocumentEnvironment{sexample}{O{}}{
5239   \__stex_statements_sexample_args:n{ #1 }
5240   \stex_reactivate_macro:N \premise
5241   \stex_reactivate_macro:N \conclusion
5242   \stex_if_smsmode:F {
5243     \seq_clear:N \l_tmpa_seq
5244     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5245       \tl_if_empty:nF{ ##1 }{
5246         \stex_get_symbol:n { ##1 }
5247         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5248           \l_stex_get_symbol_uri_str
5249         }
5250       }
5251     }
5252     \exp_args:Nnnx
5253     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
5254     \str_if_empty:NF \sexampletype {
5255       \stex_annotate_invisible:nnn{typestrings}{\sexampletype}{}
5256     }
5257     \str_if_empty:NF \sexamplename {
5258       \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}
5259     }
5260     \clist_set:No \l_tmpa_clist \sexampletype
5261     \tl_clear:N \l_tmpa_tl
5262     \clist_map_inline:Nn \l_tmpa_clist {
5263       \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
5264         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
5265       }
5266     }
5267     \tl_if_empty:NTF \l_tmpa_tl {
5268       \__stex_statements_sexample_start:
5269     }{
5270       \l_tmpa_tl
5271     }
```

210

```
5272        }
5273      \str_if_empty:NF \sexampleid {
5274        \stex_ref_new_doc_target:n \sexampleid
5275      }
5276      \stex_smsmode_do:
5277  }{
5278      \str_if_empty:NF \sexamplename {
5279        \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5280      }
5281      \stex_if_smsmode:F {
5282        \clist_set:No \l_tmpa_clist \sexampletype
5283        \tl_clear:N \l_tmpa_tl
5284        \clist_map_inline:Nn \l_tmpa_clist {
5285          \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
5286            \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
5287          }
5288        }
5289        \tl_if_empty:NTF \l_tmpa_tl {
5290          \__stex_statements_sexample_end:
5291        }{
5292          \l_tmpa_tl
5293        }
5294        \end{stex_annotate_env}
5295      }
5296  }
```

\stexpatchexample

```
5297
5298  \cs_new_protected:Nn \__stex_statements_sexample_start: {
5299      \par\noindent\titleemph{Example~\tl_if_empty:NF \sexampletitle {
5300        (\sexampletitle)
5301      }~}
5302  }
5303  \cs_new_protected:Nn \__stex_statements_sexample_end: {\par\medskip}
5304
5305  \newcommand\stexpatchexample[3][] {
5306      \str_set:Nx \l_tmpa_str{ #1 }
5307      \str_if_empty:NTF \l_tmpa_str {
5308        \tl_set:Nn \__stex_statements_sexample_start: { #2 }
5309        \tl_set:Nn \__stex_statements_sexample_end: { #3 }
5310      }{
5311        \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
5312        \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
5313      }
5314  }
```

(*End definition for* \stexpatchexample. *This function is documented on page 43.*)

\inlineex   inline:

```
5315  \keys_define:nn {stex / inlineex }{
5316    type     .str_set_x:N  = \sexampletype,
5317    id       .str_set_x:N  = \sexampleid,
5318    for      .clist_set:N   = \l__stex_statements_sexample_for_clist ,
5319    name     .str_set_x:N  = \sexamplename
```

```
5320 }
5321 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
5322   \str_clear:N \sexampletype
5323   \str_clear:N \sexampleid
5324   \str_clear:N \sexamplename
5325   \clist_clear:N \l__stex_statements_sexample_for_clist
5326   \keys_set:nn { stex / inlineex }{ #1 }
5327 }
5328 \NewDocumentCommand \inlineex { O{} m } {
5329   \begingroup
5330   \stex_reactivate_macro:N \premise
5331   \stex_reactivate_macro:N \conclusion
5332   \__stex_statements_inlineex_args:n{ #1 }
5333   \str_if_empty:NF \sexampleid {
5334     \stex_ref_new_doc_target:n \sexampleid
5335   }
5336   \stex_if_smsmode:TF{
5337     \str_if_empty:NF \sexamplename {
5338       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\examplename}}
5339     }
5340   }{
5341     \seq_clear:N \l_tmpa_seq
5342     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5343       \tl_if_empty:nF{ ##1 }{
5344         \stex_get_symbol:n { ##1 }
5345         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5346           \l_stex_get_symbol_uri_str
5347         }
5348       }
5349     }
5350     \exp_args:Nnx
5351     \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpa_seq {,}}{
5352       \str_if_empty:NF \sexampletype {
5353         \stex_annotate_invisible:nnn{typestrings}{\sexampletype}{}
5354       }
5355       #2
5356       \str_if_empty:NF \sexamplename {
5357         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5358         \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}
5359       }
5360     }
5361   }
5362   \endgroup
5363   \stex_smsmode_do:
5364 }
```

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

32.4   Logical Paragraphs

sparagraph

```
5365 \keys_define:nn { stex / sparagraph} {
5366   id       .str_set_x:N  = \sparagraphid ,
```

```
5367   title    .tl_set:N      = \l_stex_sparagraph_title_tl ,
5368   type     .str_set_x:N   = \sparagraphtype ,
5369   for      .clist_set:N   = \l__stex_statements_sparagraph_for_clist ,
5370   from     .tl_set:N      = \sparagraphfrom ,
5371   to       .tl_set:N      = \sparagraphto ,
5372   start    .tl_set:N      = \l_stex_sparagraph_start_tl ,
5373   name     .str_set:N     = \sparagraphname ,
5374   imports .tl_set:N       = \l__stex_statements_sparagraph_imports_tl
5375 }
5376
5377 \cs_new_protected:Nn \stex_sparagraph_args:n {
5378   \tl_clear:N \l_stex_sparagraph_title_tl
5379   \tl_clear:N \sparagraphfrom
5380   \tl_clear:N \sparagraphto
5381   \tl_clear:N \l_stex_sparagraph_start_tl
5382   \tl_clear:N \l__stex_statements_sparagraph_imports_tl
5383   \str_clear:N \sparagraphid
5384   \str_clear:N \sparagraphtype
5385   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5386   \str_clear:N \sparagraphname
5387   \keys_set:nn { stex / sparagraph }{ #1 }
5388 }
5389 \newif\if@in@omtext\@in@omtextfalse
5390
5391 \NewDocumentEnvironment {sparagraph} { O{} } {
5392   \stex_sparagraph_args:n { #1 }
5393   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5394     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
5395   }{
5396     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
5397   }
5398   \@in@omtexttrue
5399   \stex_if_smsmode:F {
5400     \seq_clear:N \l_tmpa_seq
5401     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5402       \tl_if_empty:nF{ ##1 }{
5403         \stex_get_symbol:n { ##1 }
5404         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5405           \l_stex_get_symbol_uri_str
5406         }
5407       }
5408     }
5409     \exp_args:Nnnx
5410     \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
5411     \str_if_empty:NF \sparagraphtype {
5412       \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
5413     }
5414     \str_if_empty:NF \sparagraphfrom {
5415       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5416     }
5417     \str_if_empty:NF \sparagraphto {
5418       \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5419     }
5420     \str_if_empty:NF \sparagraphname {
```

```
5421        \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
5422      }
5423      \clist_set:No \l_tmpa_clist \sparagraphtype
5424      \tl_clear:N \l_tmpa_tl
5425      \clist_map_inline:Nn \sparagraphtype {
5426        \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
5427          \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
5428        }
5429      }
5430      \stex_csl_to_imports:No \usemodule \l__stex_statements_sparagraph_imports_tl
5431      \tl_if_empty:NTF \l_tmpa_tl {
5432        \__stex_statements_sparagraph_start:
5433      }{
5434        \l_tmpa_tl
5435      }
5436    }
5437    \clist_set:No \l_tmpa_clist \sparagraphtype
5438    \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}
5439    {
5440      \stex_reactivate_macro:N \definiendum
5441      \stex_reactivate_macro:N \definame
5442      \stex_reactivate_macro:N \Definame
5443      \stex_reactivate_macro:N \premise
5444      \stex_reactivate_macro:N \definiens
5445    }
5446    \str_if_empty:NTF \sparagraphid {
5447      \str_if_empty:NTF \sparagraphname {
5448        \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5449          \stex_ref_new_doc_target:n {}
5450        }
5451      } {
5452        \stex_ref_new_doc_target:n {}
5453      }
5454    } {
5455      \stex_ref_new_doc_target:n \sparagraphid
5456    }
5457    \exp_args:NNx
5458    \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5459      \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5460        \tl_if_empty:nF{ ##1 }{
5461          \stex_get_symbol:n { ##1 }
5462          \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5463        }
5464      }
5465    }
5466    \stex_smsmode_do:
5467    \ignorespacesandpars
5468 }{
5469    \str_if_empty:NF \sparagraphname {
5470      \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5471      \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5472    }
5473    \stex_if_smsmode:F {
5474      \clist_set:No \l_tmpa_clist \sparagraphtype
```

```
5475      \tl_clear:N \l_tmpa_tl
5476      \clist_map_inline:Nn \l_tmpa_clist {
5477        \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
5478          \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}}
5479        }
5480      }
5481      \tl_if_empty:NTF \l_tmpa_tl {
5482        \__stex_statements_sparagraph_end:
5483      }{
5484        \l_tmpa_tl
5485      }
5486      \end{stex_annotate_env}
5487    }
5488  }
```

\stexpatchparagraph

```
5489
5490  \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
5491    \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5492      \tl_if_empty:NF \l_stex_sparagraph_title_tl {
5493        \titleemph{\l_stex_sparagraph_title_tl}:~
5494      }
5495    }{
5496      \titleemph{\l_stex_sparagraph_start_tl}~
5497    }
5498  }
5499  \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
5500
5501  \newcommand\stexpatchparagraph[3][] {
5502      \str_set:Nx \l_tmpa_str{ #1 }
5503      \str_if_empty:NTF \l_tmpa_str {
5504        \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
5505        \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
5506      }{
5507        \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
5508        \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
5509      }
5510  }
5511
5512  \keys_define:nn { stex / inlinepara} {
5513    id      .str_set_x:N  = \sparagraphid ,
5514    type    .str_set_x:N  = \sparagraphtype ,
5515    for     .clist_set:N   = \l__stex_statements_sparagraph_for_clist ,
5516    from    .tl_set:N      = \sparagraphfrom ,
5517    to      .tl_set:N      = \sparagraphto ,
5518    name    .str_set:N     = \sparagraphname
5519  }
5520  \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
5521    \tl_clear:N \sparagraphfrom
5522    \tl_clear:N \sparagraphto
5523    \str_clear:N \sparagraphid
5524    \str_clear:N \sparagraphtype
5525    \clist_clear:N \l__stex_statements_sparagraph_for_clist
5526    \str_clear:N \sparagraphname
```

```
5527    \keys_set:nn { stex / inlinepara }{ #1 }
5528  }
5529  \NewDocumentCommand \inlinepara { O{} m } {
5530    \begingroup
5531    \__stex_statements_inlinepara_args:n{ #1 }
5532    \clist_set:No \l_tmpa_clist \sparagraphtype
5533    \str_if_empty:NTF \sparagraphid {
5534      \str_if_empty:NTF \sparagraphname {
5535        \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5536          \stex_ref_new_doc_target:n {}
5537        }
5538      } {
5539        \stex_ref_new_doc_target:n {}
5540      }
5541    } {
5542      \stex_ref_new_doc_target:n \sparagraphid
5543    }
5544    \stex_if_smsmode:TF{
5545      \str_if_empty:NF \sparagraphname {
5546        \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5547        \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5548      }
5549    }{
5550      \seq_clear:N \l_tmpa_seq
5551      \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5552        \tl_if_empty:nF{ ##1 }{
5553          \stex_get_symbol:n { ##1 }
5554          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5555            \l_stex_get_symbol_uri_str
5556          }
5557        }
5558      }
5559      \exp_args:Nnx
5560      \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}{
5561        \str_if_empty:NF \sparagraphtype {
5562          \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
5563        }
5564        \str_if_empty:NF \sparagraphfrom {
5565          \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5566        }
5567        \str_if_empty:NF \sparagraphto {
5568          \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5569        }
5570        \str_if_empty:NF \sparagraphname {
5571          \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5572          \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
5573          \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5574        }
5575        \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5576          \clist_map_inline:Nn \l_tmpa_seq {
5577            \stex_ref_new_sym_target:n {##1}
5578          }
5579        }
5580        #2
```

216

```
5581        }
5582    }
5583    \endgroup
5584    \stex_smsmode_do:
5585 }
5586
```

(*End definition for* `\stexpatchparagraph`. *This function is documented on page* *43*.)

```
5587 ⟨/package⟩
```

Chapter 33

The Implementation

33.1 Package Options

We declare some switches which will modify the behavior according to the package options. Generally, an option **xxx** will just set the appropriate switches to true (otherwise they stay false).[10]

```
5588 ⟨∗package⟩
5589 ⟨@@=stex_sproof⟩
5590
5591 %%%%%%%%%%%%    sproof.dtx    %%%%%%%%%%%%
5592
```

33.2 Proofs

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

```
5593 \keys_define:nn { stex / spf } {
5594    id           .str_set_x:N  = \spfid,
5595    for          .clist_set:N  = \l__stex_sproof_spf_for_clist ,
5596    from         .tl_set:N     = \l__stex_sproof_spf_from_tl ,
5597    proofend     .tl_set:N     = \l__stex_sproof_spf_proofend_tl,
5598    type         .str_set_x:N  = \spftype,
5599    title        .tl_set:N     = \spftitle,
5600    continues    .tl_set:N     = \l__stex_sproof_spf_continues_tl,
5601    functions    .tl_set:N     = \l__stex_sproof_spf_functions_tl,
5602    method       .tl_set:N     = \l__stex_sproof_spf_method_tl
5603 }
5604 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
5605 \str_clear:N \spfid
5606 \tl_clear:N \l__stex_sproof_spf_for_tl
5607 \tl_clear:N \l__stex_sproof_spf_from_tl
5608 \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
5609 \str_clear:N \spftype
5610 \tl_clear:N \spftitle
5611 \tl_clear:N \l__stex_sproof_spf_continues_tl
5612 \tl_clear:N \l__stex_sproof_spf_functions_tl
```

---

[10]EDNOTE: need an implementation for LaTeXML

218

```
5613 \tl_clear:N \l__stex_sproof_spf_method_tl
5614    \bool_set_false:N \l__stex_sproof_inc_counter_bool
5615 \keys_set:nn { stex / spf }{ #1 }
5616 }
```

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

```
5617 \str_set:Nn\c__stex_sproof_flow_str{inline}
```

(*End definition for* \c__stex_sproof_flow_str*.*)

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

pst@with@label   This environment manages[7] the path labeling of the proof steps in the description environment of the outermost `proof` environment. The argument is the label prefix up to now; which we cache in `\pst@label` (we need evaluate it first, since are in the right place now!). Then we increment the proof depth which is stored in `\count10` (lower counters are used by TEX for page numbering) and initialize the next level counter `\count\count10` with 1. In the end call for this environment, we just decrease the proof depth counter by 1 again.

```
5618 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
5619 \cs_new_protected:Npn \sproofnumber {
5620    \int_set:Nn \l_tmpa_int {1}
5621    \bool_while_do:nn {
5622       \int_compare_p:nNn {
5623          \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5624       } > 0
5625    }{
5626       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
5627       \int_incr:N \l_tmpa_int
5628    }
5629 }
5630 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
5631    \int_set:Nn \l_tmpa_int {1}
5632    \bool_while_do:nn {
5633       \int_compare_p:nNn {
5634          \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5635       } > 0
5636    }{
5637       \int_incr:N \l_tmpa_int
5638    }
5639    \int_compare:nNnF \l_tmpa_int = 1 {
5640       \int_decr:N \l_tmpa_int
5641    }
5642    \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
5643       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1
```

---

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

```
5644        }
5645    }
5646
5647    \cs_new_protected:Npn \__stex_sproof_add_counter: {
5648      \int_set:Nn \l_tmpa_int {1}
5649      \bool_while_do:nn {
5650        \int_compare_p:nNn {
5651          \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5652        } > 0
5653      }{
5654        \int_incr:N \l_tmpa_int
5655      }
5656      \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 1 }
5657    }
5658
5659    \cs_new_protected:Npn \__stex_sproof_remove_counter: {
5660      \int_set:Nn \l_tmpa_int {1}
5661      \bool_while_do:nn {
5662        \int_compare_p:nNn {
5663          \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5664        } > 0
5665      }{
5666        \int_incr:N \l_tmpa_int
5667      }
5668      \int_decr:N \l_tmpa_int
5669      \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 0 }
5670    }
```

\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

```
5671    \def\sproof@box{
5672      \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
5673    }
5674    \def\sproofend{
5675      \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
5676        \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
5677      }
5678    }
```

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

spf@*@kw

```
5679    \def\spf@proofsketch@kw{Proof~Sketch}
5680    \def\spf@proof@kw{Proof}
5681    \def\spf@step@kw{Step}
```

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

For the other languages, we set up triggers

```
5682    \AddToHook{begindocument}{
5683      \ltx@ifpackageloaded{babel}{
5684        \makeatletter
5685        \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5686        \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5687          \input{sproof-ngerman.ldf}
```

```
5688        }
5689      \clist_if_in:NnT \l_tmpa_clist {finnish}{
5690        \input{sproof-finnish.ldf}
5691      }
5692      \clist_if_in:NnT \l_tmpa_clist {french}{
5693        \input{sproof-french.ldf}
5694      }
5695      \clist_if_in:NnT \l_tmpa_clist {russian}{
5696        \input{sproof-russian.ldf}
5697      }
5698      \makeatother
5699    }{}
5700 }
```

spfsketch

```
5701 \newcommand\spfsketch[2][]{
5702    \begingroup
5703    \let \premise \stex_proof_premise:
5704    \__stex_sproof_spf_args:n{#1}
5705    \stex_if_smsmode:TF {
5706      \str_if_empty:NF \spfid {
5707        \stex_ref_new_doc_target:n \spfid
5708      }
5709    }{
5710      \seq_clear:N \l_tmpa_seq
5711      \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5712        \tl_if_empty:nF{ ##1 }{
5713          \stex_get_symbol:n { ##1 }
5714          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5715            \l_stex_get_symbol_uri_str
5716          }
5717        }
5718      }
5719      \exp_args:Nnx
5720      \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
5721        \str_if_empty:NF \spftype {
5722          \stex_annotate_invisible:nnn{type}{\spftype}{}
5723        }
5724        \clist_set:No \l_tmpa_clist \spftype
5725        \tl_set:Nn \l_tmpa_tl {
5726          \titleemph{
5727            \tl_if_empty:NTF \spftitle {
5728              \spf@proofsketch@kw
5729            }{
5730              \spftitle
5731            }
5732          }:~
5733        }
5734        \clist_map_inline:Nn \l_tmpa_clist {
5735          \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5736            \tl_clear:N \l_tmpa_tl
5737          }
5738        }
5739        \str_if_empty:NF \spfid {
```

221

```
5740          \stex_ref_new_doc_target:n \spfid
5741        }
5742      \l_tmpa_tl #2 \sproofend
5743    }
5744  }
5745  \endgroup
5746  \stex_smsmode_do:
5747 }
5748
```

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

spfeq   This is very similar to \spfsketch, but uses a computation array[11][12]

```
5749 \newenvironment{spfeq}[2][]{
5750    \__stex_sproof_spf_args:n{#1}
5751    \let \premise \stex_proof_premise:
5752    \stex_if_smsmode:TF {
5753      \str_if_empty:NF \spfid {
5754        \stex_ref_new_doc_target:n \spfid
5755      }
5756    }{
5757      \seq_clear:N \l_tmpa_seq
5758      \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5759        \tl_if_empty:nF{ ##1 }{
5760          \stex_get_symbol:n { ##1 }
5761          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5762            \l_stex_get_symbol_uri_str
5763          }
5764        }
5765      }
5766      \exp_args:Nnnx
5767      \begin{stex_annotate_env}{spfeq}{\seq_use:Nn \l_tmpa_seq {,}}
5768      \str_if_empty:NF \spftype {
5769        \stex_annotate_invisible:nnn{type}{\spftype}{}
5770      }
5771
5772      \clist_set:No \l_tmpa_clist \spftype
5773      \tl_clear:N \l_tmpa_tl
5774      \clist_map_inline:Nn \l_tmpa_clist {
5775        \tl_if_exist:cT {__stex_sproof_spfeq_##1_start:}{
5776          \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_start:}}
5777        }
5778        \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5779          \tl_set:Nn \l_tmpa_tl {\use:n{}}
5780        }
5781      }
5782      \tl_if_empty:NTF \l_tmpa_tl {
5783        \__stex_sproof_spfeq_start:
5784      }{
5785        \l_tmpa_tl
5786      }{~#2}
```

---

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

[12]EDNOTE: document above

```
5787        \str_if_empty:NF \spfid {
5788          \stex_ref_new_doc_target:n \spfid
5789        }
5790        \begin{displaymath}\begin{array}{rcll}
5791      }
5792      \stex_smsmode_do:
5793    }{
5794      \stex_if_smsmode:F {
5795        \end{array}\end{displaymath}
5796        \clist_set:No \l_tmpa_clist \spftype
5797        \tl_clear:N \l_tmpa_tl
5798        \clist_map_inline:Nn \l_tmpa_clist {
5799          \tl_if_exist:cT {__stex_sproof_spfeq_##1_end:}{
5800            \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_end:}}
5801          }
5802        }
5803        \tl_if_empty:NTF \l_tmpa_tl {
5804          \__stex_sproof_spfeq_end:
5805        }{
5806          \l_tmpa_tl
5807        }
5808        \end{stex_annotate_env}
5809      }
5810    }
5811
5812    \cs_new_protected:Nn \__stex_sproof_spfeq_start: {
5813      \titleemph{
5814        \tl_if_empty:NTF \spftitle {
5815          \spf@proof@kw
5816        }{
5817          \spftitle
5818        }
5819      }:
5820    }
5821    \cs_new_protected:Nn \__stex_sproof_spfeq_end: {\sproofend}
5822
5823    \newcommand\stexpatchspfeq[3][] {
5824        \str_set:Nx \l_tmpa_str{ #1 }
5825        \str_if_empty:NTF \l_tmpa_str {
5826          \tl_set:Nn \__stex_sproof_spfeq_start: { #2 }
5827          \tl_set:Nn \__stex_sproof_spfeq_end: { #3 }
5828        }{
5829          \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_start:\endcsname{ #2 }
5830          \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_end:\endcsname{ #3 }
5831        }
5832    }
5833
```

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

sproof   In this environment, we initialize the proof depth counter `\count10` to 10, and set up
the description environment that will take the proof steps. At the end of the proof, we
position the proof end into the last line.

```
5834    \newenvironment{sproof}[2][]{
```

```
5835    \let \premise \stex_proof_premise:
5836    \intarray_gzero:N \l__stex_sproof_counter_intarray
5837    \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
5838    \__stex_sproof_spf_args:n{#1}
5839    \stex_if_smsmode:TF {
5840      \str_if_empty:NF \spfid {
5841        \stex_ref_new_doc_target:n \spfid
5842      }
5843    }{
5844      \seq_clear:N \l_tmpa_seq
5845      \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5846        \tl_if_empty:nF{ ##1 }{
5847          \stex_get_symbol:n { ##1 }
5848          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5849            \l_stex_get_symbol_uri_str
5850          }
5851        }
5852      }
5853      \exp_args:Nnnx
5854      \begin{stex_annotate_env}{sproof}{\seq_use:Nn \l_tmpa_seq {,}}
5855      \str_if_empty:NF \spftype {
5856        \stex_annotate_invisible:nnn{type}{\spftype}{}
5857      }
5858
5859      \clist_set:No \l_tmpa_clist \spftype
5860      \tl_clear:N \l_tmpa_tl
5861      \clist_map_inline:Nn \l_tmpa_clist {
5862        \tl_if_exist:cT {__stex_sproof_sproof_##1_start:}{
5863          \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_start:}}
5864        }
5865        \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5866          \tl_set:Nn \l_tmpa_tl {\use:n{}}
5867        }
5868      }
5869      \tl_if_empty:NTF \l_tmpa_tl {
5870        \__stex_sproof_sproof_start:
5871      }{
5872        \l_tmpa_tl
5873      }{~#2}
5874      \str_if_empty:NF \spfid {
5875        \stex_ref_new_doc_target:n \spfid
5876      }
5877      \begin{description}
5878    }
5879    \stex_smsmode_do:
5880  }{
5881    \stex_if_smsmode:F{
5882      \end{description}
5883      \clist_set:No \l_tmpa_clist \spftype
5884      \tl_clear:N \l_tmpa_tl
5885      \clist_map_inline:Nn \l_tmpa_clist {
5886        \tl_if_exist:cT {__stex_sproof_sproof_##1_end:}{
5887          \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_end:}}
5888        }
```

```
5889        }
5890      \tl_if_empty:NTF \l_tmpa_tl {
5891        \__stex_sproof_sproof_end:
5892      }{
5893        \l_tmpa_tl
5894      }
5895      \end{stex_annotate_env}
5896    }
5897  }
5898
5899  \cs_new_protected:Nn \__stex_sproof_sproof_start: {
5900    \par\noindent\titleemph{
5901      \tl_if_empty:NTF \spftype {
5902        \spf@proof@kw
5903      }{
5904        \spftype
5905      }
5906    }:
5907  }
5908  \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
5909
5910  \newcommand\stexpatchproof[3][] {
5911    \str_set:Nx \l_tmpa_str{ #1 }
5912    \str_if_empty:NTF \l_tmpa_str {
5913      \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
5914      \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
5915    }{
5916      \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
5917      \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
5918    }
5919  }
```

```
5920  \newcommand\spfidea[2][]{
5921    \__stex_sproof_spf_args:n{#1}
5922    \titleemph{
5923      \tl_if_empty:NTF \spftype {Proof~Idea}{
5924        \spftype
5925      }:
5926    }~#2
5927    \sproofend
5928  }
```

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

The next two environments (proof steps) and comments, are mostly semantical, they take KeyVal arguments that specify their semantic role. In draft mode, they read these values and show them. If the surrounding proof had display=flow, then no new \item is generated, otherwise it is. In any case, the proof step number (at the current level) is incremented.

```
5929  \newenvironment{spfstep}[1][]{
5930    \__stex_sproof_spf_args:n{#1}
5931    \stex_if_smsmode:TF {
```

```
5932        \str_if_empty:NF \spfid {
5933          \stex_ref_new_doc_target:n \spfid
5934        }
5935      }{
5936        \@in@omtexttrue
5937        \seq_clear:N \l_tmpa_seq
5938        \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5939          \tl_if_empty:nF{ ##1 }{
5940            \stex_get_symbol:n { ##1 }
5941            \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5942              \l_stex_get_symbol_uri_str
5943            }
5944          }
5945        }
5946        \exp_args:Nnnx
5947        \begin{stex_annotate_env}{spfstep}{\seq_use:Nn \l_tmpa_seq {,}}
5948        \str_if_empty:NF \spftype {
5949          \stex_annotate_invisible:nnn{type}{\spftype}{}
5950        }
5951        \clist_set:No \l_tmpa_clist \spftype
5952        \tl_set:Nn \l_tmpa_tl {
5953          \item[\sproofnumber]
5954          \bool_set_true:N \l__stex_sproof_inc_counter_bool
5955        }
5956        \clist_map_inline:Nn \l_tmpa_clist {
5957          \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5958            \tl_clear:N \l_tmpa_tl
5959          }
5960        }
5961        \l_tmpa_tl
5962        \tl_if_empty:NF \spftitle {
5963          {(\titleemph{\spftitle})\enspace}
5964        }
5965        \str_if_empty:NF \spfid {
5966          \stex_ref_new_doc_target:n \spfid
5967        }
5968      }
5969      \stex_smsmode_do:
5970      \ignorespacesandpars
5971    }{
5972      \bool_if:NT \l__stex_sproof_inc_counter_bool {
5973        \__stex_sproof_inc_counter:
5974      }
5975      \stex_if_smsmode:F {
5976        \end{stex_annotate_env}
5977      }
5978 }
```

sproofcomment

```
5979 \newenvironment{sproofcomment}[1][]{
5980    \__stex_sproof_spf_args:n{#1}
5981    \clist_set:No \l_tmpa_clist \spftype
5982    \tl_set:Nn \l_tmpa_tl {
5983      \item[\sproofnumber]
```

```
5984        \bool_set_true:N \l__stex_sproof_inc_counter_bool
5985      }
5986    \clist_map_inline:Nn \l_tmpa_clist {
5987      \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5988        \tl_clear:N \l_tmpa_tl
5989      }
5990    }
5991    \l_tmpa_tl
5992  }{
5993    \bool_if:NT \l__stex_sproof_inc_counter_bool {
5994      \__stex_sproof_inc_counter:
5995    }
5996  }
```

The next two environments also take a `KeyVal` argument, but also a regular one, which contains a start text. Both environments start a new numbered proof level.

subproof In the `subproof` environment, a new (lower-level) proproofof environment is started.

```
5997  \newenvironment{subproof}[2][]{
5998    \__stex_sproof_spf_args:n{#1}
5999    \stex_if_smsmode:TF{
6000      \str_if_empty:NF \spfid {
6001        \stex_ref_new_doc_target:n \spfid
6002      }
6003    }{
6004      \seq_clear:N \l_tmpa_seq
6005      \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6006        \tl_if_empty:nF{ ##1 }{
6007          \stex_get_symbol:n { ##1 }
6008          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6009            \l_stex_get_symbol_uri_str
6010          }
6011        }
6012      }
6013      \exp_args:Nnnx
6014      \begin{stex_annotate_env}{subproof}{\seq_use:Nn \l_tmpa_seq {,}}
6015      \str_if_empty:NF \spftype {
6016        \stex_annotate_invisible:nnn{type}{\spftype}{}
6017      }
6018
6019      \clist_set:No \l_tmpa_clist \spftype
6020      \tl_set:Nn \l_tmpa_tl {
6021        \item[\sproofnumber]
6022        \bool_set_true:N \l__stex_sproof_inc_counter_bool
6023      }
6024      \clist_map_inline:Nn \l_tmpa_clist {
6025        \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6026          \tl_clear:N \l_tmpa_tl
6027        }
6028      }
6029      \l_tmpa_tl
6030      \tl_if_empty:NF \spftitle {
6031        {(\titleemph{\spftitle})\enspace}
6032      }
```

```
6033        {~#2}
6034        \str_if_empty:NF \spfid {
6035          \stex_ref_new_doc_target:n \spfid
6036        }
6037      }
6038      \__stex_sproof_add_counter:
6039      \stex_smsmode_do:
6040 }{
6041      \__stex_sproof_remove_counter:
6042      \bool_if:NT \l__stex_sproof_inc_counter_bool {
6043        \__stex_sproof_inc_counter:
6044      }
6045      \stex_if_smsmode:F{
6046        \end{stex_annotate_env}
6047      }
6048 }
```

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

```
6049 \newenvironment{spfcases}[2][]{
6050    \tl_if_empty:nTF{#1}{
6051      \begin{subproof}[method=by-cases]{#2}
6052    }{
6053      \begin{subproof}[#1,method=by-cases]{#2}
6054    }
6055 }{
6056    \end{subproof}
6057 }
```

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

```
6058 \newenvironment{spfcase}[2][]{
6059    \__stex_sproof_spf_args:n{#1}
6060    \stex_if_smsmode:TF {
6061      \str_if_empty:NF \spfid {
6062        \stex_ref_new_doc_target:n \spfid
6063      }
6064    }{
6065      \seq_clear:N \l_tmpa_seq
6066      \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6067        \tl_if_empty:nF{ ##1 }{
6068          \stex_get_symbol:n { ##1 }
6069          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6070            \l_stex_get_symbol_uri_str
6071          }
6072        }
6073      }
6074      \exp_args:Nnnx
6075      \begin{stex_annotate_env}{spfcase}{\seq_use:Nn \l_tmpa_seq {,}}
6076      \str_if_empty:NF \spftype {
6077        \stex_annotate_invisible:nnn{type}{\spftype}{}
6078      }
6079      \clist_set:No \l_tmpa_clist \spftype
6080      \tl_set:Nn \l_tmpa_tl {
6081        \item[\sproofnumber]
```

228

```
6082        \bool_set_true:N \l__stex_sproof_inc_counter_bool
6083      }
6084      \clist_map_inline:Nn \l_tmpa_clist {
6085        \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6086          \tl_clear:N \l_tmpa_tl
6087        }
6088      }
6089      \l_tmpa_tl
6090      \tl_if_empty:nF{#2}{
6091        \titleemph{#2}:~
6092      }
6093    }
6094    \__stex_sproof_add_counter:
6095    \stex_smsmode_do:
6096  }{
6097    \__stex_sproof_remove_counter:
6098    \bool_if:NT \l__stex_sproof_inc_counter_bool {
6099      \__stex_sproof_inc_counter:
6100    }
6101    \stex_if_smsmode:F{
6102      \clist_set:No \l_tmpa_clist \spftype
6103      \tl_set:Nn \l_tmpa_tl{\sproofend}
6104      \clist_map_inline:Nn \l_tmpa_clist {
6105        \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6106          \tl_clear:N \l_tmpa_tl
6107        }
6108      }
6109      \l_tmpa_tl
6110      \end{stex_annotate_env}
6111    }
6112 }
```

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

```
6113 \newcommand\spfcasesketch[3][]{
6114    \begin{spfcase}[#1]{#2}#3\end{spfcase}
6115 }
```

33.3 Justifications

We define the actions that are undertaken, when the keys for justifications are encountered. Here this is very simple, we just define an internal macro with the value, so that we can use it later.

```
6116 \keys_define:nn { stex / just }{
6117    id         .str_set_x:N  = \l__stex_sproof_just_id_str,
6118    method     .tl_set:N     = \l__stex_sproof_just_method_tl,
6119    premises   .tl_set:N     = \l__stex_sproof_just_premises_tl,
6120    args       .tl_set:N     = \l__stex_sproof_just_args_tl
6121 }
```

The next three environments and macros are purely semantic, so we ignore the keyval
EdN:13   arguments for now and only display the content.[13]

---

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

justification

```
6122 \newenvironment{justification}[1][]{}{}
```

\premise

```
6123 \newcommand\stex_proof_premise:[2][]{#2}
```

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

\justarg   the \justarg macro is purely semantic, so we ignore the keyval arguments for now and only display the content.

```
6124 \newcommand\justarg[2][]{#2}
6125 ⟨/package⟩
```

(*End definition for* \justarg*. This function is documented on page* **??**.)

Some auxiliary code, and clean up to be executed at the end of the package.

Chapter 34

sTEX -Others Implementation

```
6126 ⟨*package⟩
6127
6128 %%%%%%%%%%%    others.dtx    %%%%%%%%%%%%
6129
6130 ⟨@@=stex_others⟩
```

Warnings and error messages
```
6131    % None
```

\MSC Math subject classifier
```
6132 \NewDocumentCommand \MSC {m} {
6133    % TODO
6134 }
```

(*End definition for* \MSC. *This function is documented on page* **??**.)

Patching tikzinput, if loaded

```
6135 \@ifpackageloaded{tikzinput}{
6136    \RequirePackage{stex-tikzinput}
6137 }{}
6138
6139 \bool_if:NT \c_stex_persist_mode_bool {
6140    \input{\jobname.sms}
6141    \prop_if_exist:NT\c_stex_mathhub_main_manifest_prop{
6142        \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
6143            \l_tmpa_str
6144        \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str _manifest_prop }
6145            \c_stex_mathhub_main_manifest_prop
6146        \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
6147    }
6148 }
6149 ⟨/package⟩
```

Chapter 35

sTeX -Metatheory Implementation

```
6150 ⟨*package⟩
6151 ⟨@@=stex_modules⟩
6152
6153 %%%%%%%%%%%%   metatheory.dtx   %%%%%%%%%%%%
6154
6155 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX/meta}
6156 \begingroup
6157 \stex_module_setup:nn{
6158   ns=\c_stex_metatheory_ns_str,
6159   meta=NONE
6160 }{Metatheory}
6161 \stex_reactivate_macro:N \symdecl
6162 \stex_reactivate_macro:N \notation
6163 \stex_reactivate_macro:N \symdef
6164 \ExplSyntaxOff
6165 \csname stex_suppress_html:n\endcsname{
6166   % is-a (a:A, a \in A, a is an A, etc.)
6167   \symdecl{isa}[args=ai]
6168   \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
6169   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
6170   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
6171
6172   % bind (\forall, \Pi, \lambda etc.)
6173   \symdecl{bind}[args=Bi]
6174   \notation{bind}[forall]{\comp\forall #1.\;#2}{##1 \comp, ##2}
6175   \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6176   \notation{bind}[depfun]{\comp( #1 \comp()\;\to\;} #2}{##1 \comp, ##2}
6177
6178   % implicit bind
6179   \symdef{implicitbind}[args=Bi]{\comp\prod_{#1}#2}{##1\comp,##2}
6180
6181   % dummy variable
6182   \symdecl{dummyvar}
6183   \notation{dummyvar}[underscore]{\comp\_}
6184   \notation{dummyvar}[dot]{\comp\cdot}
```

```
6185  \notation{dummyvar}[dash]{\comp{{\rm --}}}

6186

6187  %fromto (function space, Hom-set, implication etc.)
6188  \symdecl{fromto}[args=ai]
6189  \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
6190  \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}

6191

6192  % mapto (lambda etc.)
6193  %\symdecl{mapto}[args=Bi]
6194  %\notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
6195  %\notation{mapto}[lambda]{\comp\lambda #1 \comp.\; #2}{#1 \comp, #2}
6196  %\notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.\; #2}{#1 \comp, #2}

6197

6198  % function/operator application
6199  \symdecl{apply}[args=ia]
6200  \notation{apply}[prec=0;0x\infprec,parens]{#1 \comp( #2 \comp)}{##1 \comp, ##2}
6201  \notation{apply}[prec=0;0x\infprec,lambda]{#1 \; #2 }{##1 \; ##2}

6202

6203  % collection of propositions/booleans/truth values
6204  \symdecl{prop}[name=proposition]
6205  \notation{prop}[prop]{\comp{{\rm prop}}}
6206  \notation{prop}[BOOL]{\comp{{\rm BOOL}}}

6207

6208  \symdecl{judgmentholds}[args=1]
6209  \notation{judgmentholds}[vdash,op=\vdash]{\comp\vdash\; #1}

6210

6211  % sequences
6212  \symdecl{seqtype}[args=1]
6213  \notation{seqtype}[kleene]{#1^{\comp\ast}}

6214

6215  \symdecl{seqexpr}[args=a]
6216  \notation{seqexpr}[angle,prec=nobrackets]{\comp\langle #1\comp\rangle}{##1\comp,##2}

6217

6218  \symdef{sequence-index}[args=2,li,prec=nobrackets]{{#1}_{#2}}
6219  \notation{sequence-index}[ui,prec=nobrackets]{{#1}^{#2}}

6220

6221  \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{,\ellipses}}{##1\comp,##2}
6222  \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{,\ellipses,}#2}{##1\comp,##2}
6223  \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{,\ellipses,}#2\comp{,\ellipses,}#

6224

6225  % letin (``let'', local definitions, variable substitution)
6226  \symdecl{letin}[args=bii]
6227  \notation{letin}[let]{\comp{{\rm let}}\;#1\comp{=}#2\;\comp{{\rm in}}\;#3}
6228  \notation{letin}[subst]{#3 \comp[ #1 \comp/ #2 \comp]}
6229  \notation{letin}[frac]{#3 \comp[ \frac{#2}{#1} \comp]}

6230

6231  % structures
6232  \symdecl*{module-type}[args=1]
6233  \notation{module-type}{\comp{\mathtt{MOD}} #1}
6234  \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
6235  \notation{mathstruct}[angle,prec=nobrackets]{\comp\langle #1 \comp\rangle}{##1 \comp, ##2}

6236

6237  % objects
6238  \symdecl{object}
```

233

```
6239    \notation{object}{\comp{\mathtt{OBJECT}}}

6240

6241  }

6242    \ExplSyntaxOn

6243    \stex_add_to_current_module:n{

6244      \let\nappa\apply

6245      \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}

6246      \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}

6247      \def\livar{\csname sequence-index\endcsname[li]}

6248      \def\uivar{\csname sequence-index\endcsname[ui]}

6249      \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}

6250      \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}

6251      \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}

6252    }

6253  \__stex_modules_end_module:

6254  \endgroup

6255  ⟨/package⟩
```

Chapter 36

Tikzinput Implementation

```
6256 ⟨@@=tikzinput⟩
6257 ⟨*package⟩
6258
6259 %%%%%%%%%%%%   tikzinput.dtx   %%%%%%%%%%%%
6260
6261 \ProvidesExplPackage{tikzinput}{2022/02/26}{3.0.1}{tikzinput package}
6262 \RequirePackage{l3keys2e}
6263
6264 \keys_define:nn { tikzinput } {
6265   image    .bool_set:N   = \c_tikzinput_image_bool,
6266   image    .default:n    = false ,
6267   unknown    .code:n        = {}
6268 }
6269
6270 \ProcessKeysOptions { tikzinput }
6271
6272 \bool_if:NTF \c_tikzinput_image_bool {
6273   \RequirePackage{graphicx}
6274
6275   \providecommand\usetikzlibrary[]{}
6276   \newcommand\tikzinput[2][]{\includegraphics[#1]{#2}}
6277 }{
6278   \RequirePackage{tikz}
6279   \RequirePackage{standalone}
6280
6281   \newcommand \tikzinput [2] [] {
6282     \setkeys{Gin}{#1}
6283     \ifx \Gin@ewidth \Gin@exclamation
6284       \ifx \Gin@eheight \Gin@exclamation
6285         \input { #2 }
6286       \else
6287         \resizebox{!}{ \Gin@eheight }{
6288           \input { #2 }
6289         }
6290       \fi
6291     \else
6292       \ifx \Gin@eheight \Gin@exclamation
6293         \resizebox{ \Gin@ewidth }{!}{
```

```
6294              \input { #2 }
6295            }
6296        \else
6297          \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
6298              \input { #2 }
6299          }
6300        \fi
6301      \fi
6302    }
6303  }

6304
6305  \newcommand \ctikzinput [2] [] {
6306    \begin{center}
6307      \tikzinput [#1] {#2}
6308    \end{center}
6309  }

6310
6311  \@ifpackageloaded{stex}{
6312    \RequirePackage{stex-tikzinput}
6313  }{}

6314
6315  ⟨/package⟩
6316  ⟨∗stex⟩

6317  \ProvidesExplPackage{stex-tikzinput}{2022/02/26}{3.0.1}{stex-tikzinput}
6318  \RequirePackage{stex}
6319  \RequirePackage{tikzinput}

6320
6321  \newcommand\mhtikzinput[2][]{%
6322    \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
6323    \stex_in_repository:nn\Gin@mhrepos{
6324      \tikzinput[#1]{\mhpath{##1}{#2}}
6325    }
6326  }
6327  \newcommand\cmhtikzinput[2][]{\begin{center}\mhtikzinput[#1]{#2}\end{center}}

6328
6329  \cs_new_protected:Nn \__tikzinput_usetikzlibrary:nn {
6330    \pgfkeys@spdef\pgf@temp{#1}
6331    \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
6332    \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgfut
6333    \expandafter\edef\csname tikz@library@#1atcode\endcsname{\the\catcode`\@}
6334    \expandafter\edef\csname tikz@library@#1barcode\endcsname{\the\catcode`\|}
6335    \expandafter\edef\csname tikz@library@#1dollarcode\endcsname{\the\catcode`\$}
6336    \catcode`\@=11
6337    \catcode`\|=12
6338    \catcode`\$=3
6339    \pgfutil@InputIfFileExists{#2}{}{}
6340    \catcode`\@=\csname tikz@library@#1atcode\endcsname
6341    \catcode`\|=\csname tikz@library@#1barcode\endcsname
6342    \catcode`\$=\csname tikz@library@#1dollarcode\endcsname
6343  }

6344

6345
6346  \newcommand\libusetikzlibrary[1]{
```

```
6347    \prop_if_exist:NF \l_stex_current_repository_prop {
6348      \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6349    }
6350    \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
6351      \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6352    }
6353    \seq_clear:N \l__tikzinput_libinput_files_seq
6354    \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
6355    \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
6356
6357    \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
6358      \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / tikzlibra
6359      \IfFileExists{ \l_tmpa_str }{
6360        \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6361      }{}
6362      \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
6363      \seq_put_right:No \l_tmpa_seq \l_tmpa_str
6364    }
6365
6366    \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / tikzlibrary #1 .code.t
6367    \IfFileExists{ \l_tmpa_str }{
6368      \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6369    }{}
6370
6371    \seq_if_empty:NTF \l__tikzinput_libinput_files_seq {
6372      \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .code.t
6373    }{
6374      \int_compare:nNnTF {\seq_count:N \l__tikzinput_libinput_files_seq} = 1 {
6375        \seq_map_inline:Nn \l__tikzinput_libinput_files_seq {
6376          \__tikzinput_usetikzlibrary:nn{#1}{ ##1 }
6377        }
6378      }{
6379        \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .co
6380      }
6381    }
6382 }
6383 ⟨/stex⟩
```

LocalWords: bibfolder jobname.dtx tikzinput.dtx usetikzlibrary Gin@ewidth Gin@eheight
LocalWords: resizebox ctikzinput mhtikzinput Gin@mhrepos mhpath

Chapter 37

document-structure.sty Implementation

6384 ⟨∗package⟩
6385 ⟨@@=document_structure⟩
6386 \ProvidesExplPackage{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure}
6387 \RequirePackage{l3keys2e}

37.1 Package Options

We declare some switches which will modify the behavior according to the package options. Generally, an option xxx will just set the appropriate switches to true (otherwise they stay false).

```
6388
6389 \keys_define:nn{ document-structure }{
6390   class        .str_set_x:N  = \c_document_structure_class_str,
6391   topsect      .str_set_x:N  = \c_document_structure_topsect_str,,
6392   unknown      .code:n       = {
6393     \PassOptionsToClass{\CurrentOption}{stex}
6394     \PassOptionsToClass{\CurrentOption}{tikzinput}
6395   }
6396 %  showignores .bool_set:N   = \c_document_structure_showignores_bool,
6397 }
6398 \ProcessKeysOptions{ document-structure }
6399 \str_if_empty:NT \c_document_structure_class_str {
6400   \str_set:Nn \c_document_structure_class_str {article}
6401 }
6402 \str_if_empty:NT \c_document_structure_topsect_str {
6403   \str_set:Nn \c_document_structure_topsect_str {section}
6404 }
```

Then we need to set up the packages by requiring the sref package to be loaded, and set up triggers for other languages

```
6405 \RequirePackage{xspace}
6406 \RequirePackage{comment}
6407 \RequirePackage{stex}
6408 \AddToHook{begindocument}{
```

```
6409  \ltx@ifpackageloaded{babel}{
6410    \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6411    \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6412      \makeatletter\input{document-structure-ngerman.ldf}\makeatother
6413    }
6414  }{}
6415  }
```

\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.

```
6416  \int_new:N \l_document_structure_section_level_int
6417  \str_case:VnF \c_document_structure_topsect_str {
6418    {part}{
6419      \int_set:Nn \l_document_structure_section_level_int {0}
6420    }
6421    {chapter}{
6422      \int_set:Nn \l_document_structure_section_level_int {1}
6423    }
6424  }{
6425    \str_case:VnF \c_document_structure_class_str {
6426      {book}{
6427        \int_set:Nn \l_document_structure_section_level_int {0}
6428      }
6429      {report}{
6430        \int_set:Nn \l_document_structure_section_level_int {0}
6431      }
6432    }{
6433      \int_set:Nn \l_document_structure_section_level_int {2}
6434    }
6435  }
```

37.2 Document Structure

The structure of the document is given by the omgroup environment just like in OMDoc. The hierarchy is adjusted automatically according to the LaTeX class in effect.

\currentsectionlevel   For the \currentsectionlevel and \Currentsectionlevel macros we use an internal macro \current@section@level that only contains the keyword (no markup). We initialize it with "document" as a default. In the generated OMDoc, we only generate a text element of class omdoc_currentsectionlevel, wich will be instantiated by CSS later.[14]

EdN:14

```
6436  \def\current@section@level{document}%
6437  \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
6438  \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

(*End definition for* \currentsectionlevel. *This function is documented on page* **??**.)

\skipomgroup

```
6439  \cs_new_protected:Npn \skipomgroup {
```

---

[14]EDNOTE: MK: we may have to experiment with the more powerful uppercasing macro from mfirstuc.sty once we internationalize.

```
6440     \ifcase\l_document_structure_section_level_int
6441     \or\stepcounter{part}
6442     \or\stepcounter{chapter}
6443     \or\stepcounter{section}
6444     \or\stepcounter{subsection}
6445     \or\stepcounter{subsubsection}
6446     \or\stepcounter{paragraph}
6447     \or\stepcounter{subparagraph}
6448     \fi
6449 }
```

(*End definition for* `\skipomgroup`*. This function is documented on page* **??***.*)

blindfragment

```
6450 \newcommand\at@begin@blindomgroup[1]{}
6451 \newenvironment{blindfragment}
6452 {
6453     \int_incr:N\l_document_structure_section_level_int
6454     \at@begin@blindomgroup\l_document_structure_section_level_int
6455 }{}
```

`\omgroup@nonum`   convenience macro: `\omgroup@nonum{⟨level⟩}{⟨title⟩}` makes an unnumbered sectioning with title ⟨*title*⟩ at level ⟨*level*⟩.

```
6456 \newcommand\omgroup@nonum[2]{
6457     \ifx\hyper@anchor\@undefined\else\phantomsection\fi
6458     \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
6459 }
```

(*End definition for* `\omgroup@nonum`*. This function is documented on page* **??***.*)

`\omgroup@num`   convenience macro: `\omgroup@nonum{⟨level⟩}{⟨title⟩}` makes numbered sectioning with title ⟨*title*⟩ at level ⟨*level*⟩. We have to check the `short` key was given in the `omgroup` environment and – if it is use it. But how to do that depends on whether the `rdfmeta` package has been loaded. In the end we call `\sref@label@id` to enable crossreferencing.

```
6460 \newcommand\omgroup@num[2]{
6461     \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
6462         \@nameuse{#1}{#2}
6463     }{
6464         \cs_if_exist:NTF\rdfmeta@sectioning{
6465             \@nameuse{rdfmeta@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
6466         }{
6467             \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
6468         }
6469     }
6470 %\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\omgroup@id
6471 }
```

(*End definition for* `\omgroup@num`*. This function is documented on page* **??***.*)

sfragment

```
6472 \keys_define:nn { document-structure / omgroup }{
6473     id              .str_set_x:N = \l__document_structure_omgroup_id_str,
6474     date            .str_set_x:N = \l__document_structure_omgroup_date_str,
6475     creators        .clist_set:N = \l__document_structure_omgroup_creators_clist,
```

```
6476    contributors  .clist_set:N = \l__document_structure_omgroup_contributors_clist,
6477    srccite       .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
6478    type          .tl_set:N    = \l__document_structure_omgroup_type_tl,
6479    short         .tl_set:N    = \l__document_structure_omgroup_short_tl,
6480    display       .tl_set:N    = \l__document_structure_omgroup_display_tl,
6481    intro         .tl_set:N    = \l__document_structure_omgroup_intro_tl,
6482    imports       .tl_set:N    = \l__document_structure_omgroup_imports_tl,
6483    loadmodules   .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
6484 }
6485 \cs_new_protected:Nn \__document_structure_omgroup_args:n {
6486    \str_clear:N \l__document_structure_omgroup_id_str
6487    \str_clear:N \l__document_structure_omgroup_date_str
6488    \clist_clear:N \l__document_structure_omgroup_creators_clist
6489    \clist_clear:N \l__document_structure_omgroup_contributors_clist
6490    \tl_clear:N \l__document_structure_omgroup_srccite_tl
6491    \tl_clear:N \l__document_structure_omgroup_type_tl
6492    \tl_clear:N \l__document_structure_omgroup_short_tl
6493    \tl_clear:N \l__document_structure_omgroup_display_tl
6494    \tl_clear:N \l__document_structure_omgroup_imports_tl
6495    \tl_clear:N \l__document_structure_omgroup_intro_tl
6496    \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
6497    \keys_set:nn { document-structure / omgroup } { #1 }
6498 }
```

we define a switch for numbering lines and a hook for the beginning of groups: The
\at@begin@omgroup macro allows customization. It is run at the beginning of the
omgroup, i.e. after the section heading.

```
6499 \newif\if@mainmatter\@mainmattertrue
6500 \newcommand\at@begin@omgroup[3][]{}
```

Then we define a helper macro that takes care of the sectioning magic. It comes
with its own key/value interface for customization.

```
6501 \keys_define:nn { document-structure / sectioning }{
6502    name    .str_set_x:N  = \l__document_structure_sect_name_str   ,
6503    ref     .str_set_x:N  = \l__document_structure_sect_ref_str    ,
6504    clear   .bool_set:N   = \l__document_structure_sect_clear_bool ,
6505    clear   .default:n    = {true}                                 ,
6506    num     .bool_set:N   = \l__document_structure_sect_num_bool   ,
6507    num     .default:n    = {true}
6508 }
6509 \cs_new_protected:Nn \__document_structure_sect_args:n {
6510    \str_clear:N \l__document_structure_sect_name_str
6511    \str_clear:N \l__document_structure_sect_ref_str
6512    \bool_set_false:N \l__document_structure_sect_clear_bool
6513    \bool_set_false:N \l__document_structure_sect_num_bool
6514    \keys_set:nn { document-structure / sectioning } { #1 }
6515 }
6516 \newcommand\omdoc@sectioning[3][]{
6517    \__document_structure_sect_args:n {#1 }
6518    \let\omdoc@sect@name\l__document_structure_sect_name_str
6519    \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
6520    \if@mainmatter% numbering not overridden by frontmatter, etc.
6521      \bool_if:NTF \l__document_structure_sect_num_bool {
6522        \omgroup@num{#2}{#3}
```

```
6523        }{
6524          \omgroup@nonum{#2}{#3}
6525        }
6526        \def\current@section@level{\omdoc@sect@name}
6527      \else
6528        \omgroup@nonum{#2}{#3}
6529      \fi
6530    }% 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.

```
6531    \newcommand\omgroup@redefine@addtocontents[1]{%
6532    %\edef\__document_structureimport{#1}%
6533    %\@for\@I:=\__document_structureimport\do{%
6534    %\edef\@path{\csname module@\@I  @path\endcsname}%
6535    %\@ifundefined{tf@toc}\relax%
6536    %     {\protected@write\tf@toc{}{\string\@requiremodules{\@path}}}}}
6537    %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
6538    %\def\addcontentsline##1##2##3{%
6539    %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}
6540    %\else% hyperref.sty not loaded
6541    %\def\addcontentsline##1##2##3{%
6542    %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}{
6543    %\fi
6544    }% hypreref.sty loaded?
```

now the `omgroup` environment itself. This takes care of the table of contents via the helper
macro above and then selects the appropriate sectioning command from `article.cls`.
It also registeres the current level of omgroups in the `\omgroup@level` counter.

```
6545    \newenvironment{sfragment}[2][]% keys, title
6546    {
6547      \__document_structure_omgroup_args:n { #1 }%\sref@target%
```

If the `loadmodules` key is set on `\begin{sfragment}`, we redefine the `\addcontetsline`
macro that determines how the sectioning commands below construct the entries for the
table of contents.

```
6548      \stex_csl_to_imports:No \usemodule \l__document_structure_omgroup_imports_tl
6549
6550      \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
6551        \omgroup@redefine@addtocontents{
6552          %\@ifundefined{module@id}\used@modules%
6553          %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}
6554        }
6555      }
```

now we only need to construct the right sectioning depending on the value of `\section@level`.

```
6556      \int_incr:N\l_document_structure_section_level_int
6557      \ifcase\l_document_structure_section_level_int
6558        \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
6559        \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
6560        \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
6561        \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
6562        \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
6563        \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#
6564        \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{paragr
```

```
6565    \fi
6566    \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
6567    \str_if_empty:NF \l__document_structure_omgroup_id_str {
6568      \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
6569    }
6570  }% for customization
6571  {}
```

and finally, we localize the sections

```
6572  \newcommand\omdoc@part@kw{Part}
6573  \newcommand\omdoc@chapter@kw{Chapter}
6574  \newcommand\omdoc@section@kw{Section}
6575  \newcommand\omdoc@subsection@kw{Subsection}
6576  \newcommand\omdoc@subsubsection@kw{Subsubsection}
6577  \newcommand\omdoc@paragraph@kw{paragraph}
6578  \newcommand\omdoc@subparagraph@kw{subparagraph}
```

37.3   Front and Backmatter

Index markup is provided by the `omtext` package [**Kohlhase:smmtf:git**], so in the
`document-structure` package we only need to supply the corresponding `\printindex`
command, if it is not already defined

\printindex

```
6579  \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).

```
6580  \cs_if_exist:NTF\frontmatter{
6581    \let\__document_structure_orig_frontmatter\frontmatter
6582    \let\frontmatter\relax
6583  }{
6584    \tl_set:Nn\__document_structure_orig_frontmatter{
6585      \clearpage
6586      \@mainmatterfalse
6587      \pagenumbering{roman}
6588    }
6589  }
6590  \cs_if_exist:NTF\backmatter{
6591    \let\__document_structure_orig_backmatter\backmatter
6592    \let\backmatter\relax
6593  }{
6594    \tl_set:Nn\__document_structure_orig_backmatter{
6595      \clearpage
6596      \@mainmatterfalse
6597      \pagenumbering{roman}
6598    }
6599  }
```

    Using these, we can now define the `frontmatter` and `backmatter` environments

we use the \orig@frontmatter macro defined above and \mainmatter if it exists, oth-
erwise we define it.

```
6600 \newenvironment{frontmatter}{
6601   \__document_structure_orig_frontmatter
6602 }{
6603   \cs_if_exist:NTF\mainmatter{
6604     \mainmatter
6605   }{
6606     \clearpage
6607     \@mainmattertrue
6608     \pagenumbering{arabic}
6609   }
6610 }
```

As backmatter is at the end of the document, we do nothing for \endbackmatter.

```
6611 \newenvironment{backmatter}{
6612   \__document_structure_orig_backmatter
6613 }{
6614   \cs_if_exist:NTF\mainmatter{
6615     \mainmatter
6616   }{
6617     \clearpage
6618     \@mainmattertrue
6619     \pagenumbering{arabic}
6620   }
6621 }
```

finally, we make sure that page numbering is arabic and we have main matter as the
default

```
6622 \@mainmattertrue\pagenumbering{arabic}
```

\prematurestop We initialize \afterprematurestop, and provide \prematurestop@endomgroup which
looks up \omgroup@level and recursively ends enough {sfragment}s.

```
6623 \def \c__document_structure_document_str{document}
6624 \newcommand\afterprematurestop{}
6625 \def\prematurestop@endomgroup{
6626   \unless\ifx\@currenvir\c__document_structure_document_str
6627     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafte
6628     \expandafter\prematurestop@endomgroup
6629   \fi
6630 }
6631 \providecommand\prematurestop{
6632   \message{Stopping~sTeX~processing~prematurely}
6633   \prematurestop@endomgroup
6634   \afterprematurestop
6635   \end{document}
6636 }
```

(*End definition for* \prematurestop. *This function is documented on page* **??**.)

37.4 Global Variables

\setSGvar set a global variable

```
6637 \RequirePackage{etoolbox}
6638 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}
```

(*End definition for* \setSGvar*. This function is documented on page* **??**.)

\useSGvar use a global variable

```
6639 \newrobustcmd\useSGvar[1]{%
6640   \@ifundefined{sTeX@Gvar@#1}
6641   {\PackageError{document-structure}
6642     {The sTeX Global variable #1 is undefined}
6643     {set it with \protect\setSGvar}}
6644 \@nameuse{sTeX@Gvar@#1}}
```

(*End definition for* \useSGvar*. This function is documented on page* **??**.)

\ifSGvar execute something conditionally based on the state of the global variable.

```
6645 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
6646   \@ifundefined{sTeX@Gvar@#1}
6647   {\PackageError{document-structure}
6648     {The sTeX Global variable #1 is undefined}
6649     {set it with \protect\setSGvar}}
6650   {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}
```

(*End definition for* \ifSGvar*. This function is documented on page* **??**.)

Chapter 38

NotesSlides – Implementation

38.1   Class and Package Options

We define some Package Options and switches for the `notesslides` class and activate
them by passing them on to `beamer.cls` and `omdoc.cls` and the `notesslides` package.
We pass the `nontheorem` option to the `statements` package when we are not in notes
mode, since the `beamer` package has its own (overlay-aware) theorem environments.

```
6651 ⟨*cls⟩
6652 ⟨@@=notesslides⟩
6653 \ProvidesExplClass{notesslides}{2022/02/28}{3.1.0}{notesslides Class}
6654 \RequirePackage{l3keys2e}
6655
6656 \keys_define:nn{notesslides / cls}{
6657   class    .str_set_x:N = \c__notesslides_class_str,
6658   notes    .bool_set:N  = \c__notesslides_notes_bool ,
6659   slides   .code:n       = { \bool_set_false:N \c__notesslides_notes_bool },
6660   docopt   .str_set_x:N = \c__notesslides_docopt_str,
6661   unknown  .code:n       = {
6662     \PassOptionsToPackage{\CurrentOption}{document-structure}
6663     \PassOptionsToClass{\CurrentOption}{beamer}
6664     \PassOptionsToPackage{\CurrentOption}{notesslides}
6665     \PassOptionsToPackage{\CurrentOption}{stex}
6666   }
6667 }
6668 \ProcessKeysOptions{ notesslides / cls }
6669
6670 \str_if_empty:NF \c__notesslides_class_str {
6671   \PassOptionsToPackage{class=\c__notesslides_class_str}{document-structure}
6672 }
6673
6674 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{book}{
6675   \PassOptionsToPackage{defaulttopsect=part}{notesslides}
6676 }
6677 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{report}{
6678   \PassOptionsToPackage{defaulttopsect=part}{notesslides}
6679 }
6680
6681 \RequirePackage{stex}
```

246

```
6682  \stex_html_backend:T {
6683    \bool_set_true:N\c__notesslides_notes_bool
6684  }
6685
6686  \bool_if:NTF \c__notesslides_notes_bool {
6687    \PassOptionsToPackage{notes=true}{notesslides}
6688  }{
6689    \PassOptionsToPackage{notes=false}{notesslides}
6690  }
6691  ⟨/cls⟩
```

now we do the same for the `notesslides` package.

```
6692  ⟨*package⟩
6693  \ProvidesExplPackage{notesslides}{2022/02/28}{3.1.0}{notesslides Package}
6694  \RequirePackage{l3keys2e}
6695
6696  \keys_define:nn{notesslides / pkg}{
6697    topsect         .str_set_x:N  = \c__notesslides_topsect_str,
6698    defaulttopsect  .str_set_x:N  = \c__notesslides_defaulttopsec_str,
6699    notes           .bool_set:N   = \c__notesslides_notes_bool ,
6700    slides          .code:n       = { \bool_set_false:N \c__notesslides_notes_bool },
6701    sectocframes    .bool_set:N   = \c__notesslides_sectocframes_bool ,
6702    frameimages     .bool_set:N   = \c__notesslides_frameimages_bool ,
6703    fiboxed         .bool_set:N   = \c__notesslides_fiboxed_bool ,
6704    noproblems      .bool_set:N   = \c__notesslides_noproblems_bool,
6705    unknown         .code:n       = {
6706      \PassOptionsToClass{\CurrentOption}{stex}
6707      \PassOptionsToClass{\CurrentOption}{tikzinput}
6708    }
6709  }
6710  \ProcessKeysOptions{ notesslides / pkg }
6711
6712  \RequirePackage{stex}
6713  \stex_html_backend:T {
6714    \bool_set_true:N\c__notesslides_notes_bool
6715  }
6716
6717  \newif\ifnotes
6718  \bool_if:NTF \c__notesslides_notes_bool {
6719    \notestrue
6720  }{
6721    \notesfalse
6722  }
6723
```

we give ourselves a macro `\@@topsect` that needs only be evaluated once, so that the `\ifdefstring` conditionals work below.

```
6724  \str_if_empty:NTF \c__notesslides_topsect_str {
6725    \str_set_eq:NN \__notesslidestopsect \c__notesslides_defaulttopsec_str
6726  }{
6727    \str_set_eq:NN \__notesslidestopsect \c__notesslides_topsect_str
6728  }
6729  \PassOptionsToPackage{topsect=\__notesslidestopsect}{document-structure}
6730  ⟨/package⟩
```

Depending on the options, we either load the `article`-based `document-structure` or the `beamer` class (and set some counters).

```
6731 ⟨*cls⟩
6732 \bool_if:NTF \c__notesslides_notes_bool {
6733   \str_if_empty:NT \c__notesslides_class_str {
6734     \str_set:Nn \c__notesslides_class_str {article}
6735   }
6736   \exp_after:wN\LoadClass\exp_after:wN[\c__notesslides_docopt_str]
6737     {\c__notesslides_class_str}
6738 }{
6739   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
6740   \newcounter{Item}
6741   \newcounter{paragraph}
6742   \newcounter{subparagraph}
6743   \newcounter{Hfootnote}
6744 }
6745 \RequirePackage{document-structure}
```

now it only remains to load the `notesslides` package that does all the rest.

```
6746 \RequirePackage{notesslides}
6747 ⟨/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.

```
6748 ⟨*package⟩
6749 \bool_if:NT \c__notesslides_notes_bool {
6750   \RequirePackage{a4wide}
6751   \RequirePackage{marginnote}
6752   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
6753   \RequirePackage{mdframed}
6754   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
6755   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
6756 }
6757 \RequirePackage{stex-tikzinput}
6758 \RequirePackage{etoolbox}
6759 \RequirePackage{amssymb}
6760 \RequirePackage{amsmath}
6761 \RequirePackage{comment}
6762 \RequirePackage{textcomp}
6763 \RequirePackage{url}
6764 \RequirePackage{graphicx}
6765 \RequirePackage{pgf}
```

38.2 Notes and Slides

For the lecture notes cases, we also provide the `\usetheme` macro that would otherwise come from the the `beamer` class. While the latter loads beamertheme⟨*theme*⟩.sty, the

notes version loads `beamernotestheme`⟨*theme*⟩`.sty`.[15]

```
6766  \bool_if:NT \c__notesslides_notes_bool {
6767    \renewcommand\usetheme[2][]{\usepackage[#1]{beamernotestheme#2}}
6768  }
6769
6770
6771  \NewDocumentCommand \libusetheme {O{} m} {
6772    \bool_if:NTF \c__notesslides_notes_bool {
6773      \libusepackage[#1]{beamernotestheme#2}
6774    }{
6775      \libusepackage[#1]{beamertheme#2}
6776    }
6777  }
```

We define the sizes of slides in the notes. Somehow, we cannot get by with the same here.

```
6778  \newcounter{slide}
6779  \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
6780  \newlength{\slideheight}\setlength{\slideheight}{9cm}
```

note The `note` environment is used to leave out text in the `slides` mode. It does not have a counterpart in OMDoc. So for course notes, we define the `note` environment to be a no-operation otherwise we declare the `note` environment as a comment via the `comment` package.

```
6781  \bool_if:NTF \c__notesslides_notes_bool {
6782    \renewenvironment{note}{\ignorespaces}{}
6783  }{
6784    \excludecomment{note}
6785  }
```

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.

```
6786  \bool_if:NT \c__notesslides_notes_bool {
6787    \newlength{\slideframewidth}
6788    \setlength{\slideframewidth}{1.5pt}
```

frame We first define the keys.

```
6789    \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
6790      \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
6791        \bool_set_true:N #1
6792      }{
6793        \bool_set_false:N #1
6794      }
6795    }
6796    \keys_define:nn{notesslides / frame}{
6797      label                .str_set_x:N  = \l__notesslides_frame_label_str,
6798      allowframebreaks     .code:n       = {
6799        \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
6800      },
6801      allowdisplaybreaks   .code:n       = {
```

---

[15]EdNote: MK: This is not ideal, but I am not sure that I want to be able to provide the full theme functionality there.

```
6802        \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
6803      },
6804      fragile                .code:n       = {
6805        \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
6806      },
6807      shrink                 .code:n       = {
6808        \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
6809      },
6810      squeeze                .code:n       = {
6811        \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
6812      },
6813      t                      .code:n       = {
6814        \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
6815      },
6816    }
6817    \cs_new_protected:Nn \__notesslides_frame_args:n {
6818      \str_clear:N \l__notesslides_frame_label_str
6819      \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
6820      \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
6821      \bool_set_true:N \l__notesslides_frame_fragile_bool
6822      \bool_set_true:N \l__notesslides_frame_shrink_bool
6823      \bool_set_true:N \l__notesslides_frame_squeeze_bool
6824      \bool_set_true:N \l__notesslides_frame_t_bool
6825      \keys_set:nn { notesslides / frame }{ #1 }
6826    }
```

We define the environment, read them, and construct the slide number and label.

```
6827    \renewenvironment{frame}[1][]{
6828      \__notesslides_frame_args:n{#1}
6829      \sffamily
6830      \stepcounter{slide}
6831      \def\@currentlabel{\theslide}
6832      \str_if_empty:NF \l__notesslides_frame_label_str {
6833        \label{\l__notesslides_frame_label_str}
6834      }
```

We redefine the `itemize` environment so that it looks more like the one in `beamer`.

```
6835      \def\itemize@level{outer}
6836      \def\itemize@outer{outer}
6837      \def\itemize@inner{inner}
6838      \renewcommand\newpage{\addtocounter{framenumber}{1}}
6839      \newcommand\metakeys@show@keys[2]{\marginnote{{\scriptsize ##2}}}
6840      \renewenvironment{itemize}{
6841        \ifx\itemize@level\itemize@outer
6842          \def\itemize@label{$\rhd$}
6843        \fi
6844        \ifx\itemize@level\itemize@inner
6845          \def\itemize@label{$\scriptstyle\rhd$}
6846        \fi
6847        \begin{list}
6848        {\itemize@label}
6849        {\setlength{\labelsep}{.3em}
6850         \setlength{\labelwidth}{.5em}
6851         \setlength{\leftmargin}{1.5em}
6852        }
```

250

```
6853        \edef\itemize@level{\itemize@inner}
6854      }{
6855        \end{list}
6856      }
```

We create the box with the `mdframed` environment from the equinymous package.

```
6857      \stex_html_backend:TF {
6858        \begin{stex_annotate_env}{frame}{}\vbox\bgroup
6859      }{
6860        \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwid
6861      }
6862    }{
6863      \stex_html_backend:TF {
6864        \miko@slidelabel\egroup\end{stex_annotate_env}
6865      }{\medskip\miko@slidelabel\end{mdframed}}
6866    }
```

Now, we need to redefine the frametitle (we are still in course notes mode).

\frametitle

```
6867      \renewcommand{\frametitle}[1]{{\Large\bf\sf\color{blue}{#1}}\medskip}
6868  }
```

(*End definition for* \frametitle. *This function is documented on page* **??**.)

EdN:16   \pause [16]

```
6869  \bool_if:NT \c__notesslides_notes_bool {
6870    \newcommand\pause{}
6871  }
```

(*End definition for* \pause. *This function is documented on page* **??**.)

nparagraph

```
6872  \bool_if:NTF \c__notesslides_notes_bool {
6873    \newenvironment{nparagraph}[1][]{\begin{sparagraph}[#1]}{\end{sparagraph}}
6874  }{
6875    \excludecomment{nparagraph}
6876  }
```

nfragment

```
6877  \bool_if:NTF \c__notesslides_notes_bool {
6878    \newenvironment{nfragment}[2][]{\begin{sfragment}[#1]{#2}}{\end{sfragment}}
6879  }{
6880    \excludecomment{nfragment}
6881  }
```

ndefinition

```
6882  \bool_if:NTF \c__notesslides_notes_bool {
6883    \newenvironment{ndefinition}[1][]{\begin{sdefinition}[#1]}{\end{sdefinition}}
6884  }{
6885    \excludecomment{ndefinition}
6886  }
```

---

[16]EDNOTE: MK: fake it in notes mode for now

nassertion

```
6887  \bool_if:NTF \c__notesslides_notes_bool {
6888    \newenvironment{nassertion}[1][]{\begin{sassertion}[#1]}{\end{sassertion}}
6889  }{
6890    \excludecomment{nassertion}
6891  }
```

nsproof

```
6892  \bool_if:NTF \c__notesslides_notes_bool {
6893    \newenvironment{nproof}[2][]{\begin{sproof}[#1]{#2}}{\end{sproof}}
6894  }{
6895    \excludecomment{nproof}
6896  }
```

nexample

```
6897  \bool_if:NTF \c__notesslides_notes_bool {
6898    \newenvironment{nexample}[1][]{\begin{sexample}[#1]}{\end{sexample}}
6899  }{
6900    \excludecomment{nexample}
6901  }
```

\inputref@*skip   We customize the hooks for in \inputref.

```
6902  \def\inputref@preskip{\smallskip}
6903  \def\inputref@postskip{\medskip}
```

(*End definition for \inputref@*skip. This function is documented on page* **??**.)

\inputref*

```
6904  \let\orig@inputref\inputref
6905  \def\inputref{\@ifstar\ninputref\orig@inputref}
6906  \newcommand\ninputref[2][]{
6907    \bool_if:NT \c__notesslides_notes_bool {
6908      \orig@inputref[#1]{#2}
6909    }
6910  }
```

(*End definition for \inputref*. This function is documented on page* **??**.)

38.3   Header and Footer Lines

Now, we set up the infrastructure for the footer line of the slides, we use boxes for the logos, so that they are only loaded once, that considerably speeds up processing.

\setslidelogo   The default logo is the sTEX logo. Customization can be done by \setslidelogo{⟨*logo name*⟩}.

```
6911  \newlength{\slidelogoheight}
6912
6913  \bool_if:NTF \c__notesslides_notes_bool {
6914    \setlength{\slidelogoheight}{.4cm}
6915  }{
6916    \setlength{\slidelogoheight}{1cm}
6917  }
6918  \newsavebox{\slidelogo}
```

252

```
6919  \sbox{\slidelogo}{\sTeX}
6920  \newrobustcmd{\setslidelogo}[1]{
6921    \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
6922  }
```

(*End definition for* `\setslidelogo`*. This function is documented on page* **??***.*)

`\setsource` `\source` stores the writer's name. By default it is *Michael Kohlhase* since he is the main user and designer of this package. `\setsource{`⟨*name*⟩`}` can change the writer's name.

```
6923  \def\source{Michael Kohlhase}% customize locally
6924  \newrobustcmd{\setsource}[1]{\def\source{#1}}
```

(*End definition for* `\setsource`*. This function is documented on page* **??***.*)

`\setlicensing` Now, we set up the copyright and licensing. By default we use the Creative Commons Attribuition-ShareAlike license to strengthen the public domain. If package `hyperref` is loaded, then we can attach a hyperlink to the license logo. `\setlicensing[`⟨*url*⟩`]{`⟨*logo name*⟩`}` is used for customization, where ⟨*url*⟩ is optional.

```
6925  \def\copyrightnotice{\footnotesize\copyright :\hspace{.3ex}{\source}}
6926  \newsavebox{\cclogo}
6927  \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{stex-cc_somerights}}
6928  \newif\ifcchref\cchreffalse
6929  \AtBeginDocument{
6930    \@ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
6931  }
6932  \def\licensing{
6933    \ifcchref
6934      \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
6935    \else
6936      {\usebox{\cclogo}}
6937    \fi
6938  }
6939  \newrobustcmd{\setlicensing}[2][]{
6940    \def\@url{#1}
6941    \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
6942    \ifx\@url\@empty
6943      \def\licensing{{\usebox{\cclogo}}}
6944    \else
6945      \def\licensing{
6946        \ifcchref
6947        \href{#1}{\usebox{\cclogo}}
6948        \else
6949        {\usebox{\cclogo}}
6950        \fi
6951      }
6952    \fi
6953  }
```

(*End definition for* `\setlicensing`*. This function is documented on page* **??***.*)

`\slidelabel` Now, we set up the slide label for the `article` mode.[17]

```
6954  \newrobustcmd\miko@slidelabel{
6955    \vbox to \slidelogoheight{
```

---

[17]EDNOTE: see that we can use the themes for the slides some day. This is all fake.

```
6956        \vss\hbox to \slidewidth
6957        {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
6958   }
6959 }
```

(*End definition for* \slidelabel. *This function is documented on page* **??**.)

38.4   Frame Images

\frameimage   We have to make sure that the width is overwritten, for that we check the \Gin@ewidth macro from the graphicx package. We also add the label key.

```
6960 \def\Gin@mhrepos{}
6961 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
6962 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
6963 \newrobustcmd\frameimage[2][]{
6964    \stepcounter{slide}
6965    \bool_if:NT \c__notesslides_frameimages_bool {
6966       \def\Gin@ewidth{}\setkeys{Gin}{#1}
6967       \bool_if:NF \c__notesslides_notes_bool { \vfill }
6968       \begin{center}
6969          \bool_if:NTF \c__notesslides_fiboxed_bool {
6970             \fbox{
6971                \ifx\Gin@ewidth\@empty
6972                   \ifx\Gin@mhrepos\@empty
6973                      \mhgraphics[width=\slidewidth,#1]{#2}
6974                   \else
6975                      \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
6976                   \fi
6977                \else% Gin@ewidth empty
6978                   \ifx\Gin@mhrepos\@empty
6979                      \mhgraphics[#1]{#2}
6980                   \else
6981                      \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
6982                   \fi
6983                \fi% Gin@ewidth empty
6984             }
6985          }{
6986             \ifx\Gin@ewidth\@empty
6987                \ifx\Gin@mhrepos\@empty
6988                   \mhgraphics[width=\slidewidth,#1]{#2}
6989                \else
6990                   \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
6991                \fi
6992                \ifx\Gin@mhrepos\@empty
6993                   \mhgraphics[#1]{#2}
6994                \else
6995                   \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
6996                \fi
6997             \fi% Gin@ewidth empty
6998          }
6999       \end{center}
7000       \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
7001       \bool_if:NF \c__notesslides_notes_bool { \vfill }
```

```
7002      }
7003  } % ifmks@sty@frameimages
```

(*End definition for* `\frameimage`. *This function is documented on page* **??**.)

38.5   Colors and Highlighting

We first specify sans serif fonts as the default.

```
7004  \sffamily
```

Now, we set up an infrastructure for highlighting phrases in slides. Note that we use content-oriented macros for highlighting rather than directly using color markup. The first thing to to is to adapt the green so that it is dark enough for most beamers

```
7005  \AddToHook{begindocument}{
7006      \definecolor{green}{rgb}{0,.5,0}
7007      \definecolor{purple}{cmyk}{.3,1,0,.17}
7008  }
```

We customize the `\defemph`, `\symrefemph`, `\compemph`, and `\titleemph` macros with colors. Furthermore we customize the `\__omtextlec` macro for the appearance of line end comments in `\lec`.

```
7009  % \def\STpresent#1{\textcolor{blue}{#1}}
7010  \def\defemph#1{{\textcolor{magenta}{#1}}}
7011  \def\symrefemph#1{{\textcolor{cyan}{#1}}}
7012  \def\compemph#1{{\textcolor{blue}{#1}}}
7013  \def\titleemph#1{{\textcolor{blue}{#1}}}
7014  \def\__omtext_lec#1{(\textcolor{green}{#1})}
```

I like to use the dangerous bend symbol for warnings, so we provide it here.

`\textwarning` as the macro can be used quite often we put it into a box register, so that it is only loaded once.

```
7015  \pgfdeclareimage[width=.8em]{miko@small@dbend}{stex-dangerous-bend}
7016  \def\smalltextwarning{
7017      \pgfuseimage{miko@small@dbend}
7018      \xspace
7019  }
7020  \pgfdeclareimage[width=1.2em]{miko@dbend}{stex-dangerous-bend}
7021  \newrobustcmd\textwarning{
7022      \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
7023      \xspace
7024  }
7025  \pgfdeclareimage[width=2.5em]{miko@big@dbend}{stex-dangerous-bend}
7026  \newrobustcmd\bigtextwarning{
7027      \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
7028      \xspace
7029  }
```

(*End definition for* `\textwarning`. *This function is documented on page* **??**.)

```
7030  \newrobustcmd\putgraphicsat[3]{
7031      \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
7032  }
7033  \newrobustcmd\putat[2]{
7034      \begin{picture}(0,0)\put(#1){#2}\end{picture}
7035  }
```

38.6 Sectioning

If the `sectocframes` option is set, then we make section frames. We first define counters for `part` and `chapter`, which `beamer.cls` does not have and we make the `section` counter which it does dependent on `chapter`.

```
7036 \bool_if:NT \c__notesslides_sectocframes_bool {
7037   \str_if_eq:VnTF \__notesslidestopsect{part}{
7038     \newcounter{chapter}\counterwithin*{section}{chapter}
7039   }{
7040     \str_if_eq:VnT\__notesslidestopsect{chapter}{
7041       \newcounter{chapter}\counterwithin*{section}{chapter}
7042     }
7043   }
7044 }
```

\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

```
7045 \def\part@prefix{}
7046 \@ifpackageloaded{document-structure}{}{
7047   \str_case:VnF \__notesslidestopsect {
7048     {part}{
7049       \int_set:Nn \l_document_structure_section_level_int {0}
7050       \def\thesection{\arabic{chapter}.\arabic{section}}
7051       \def\part@prefix{\arabic{chapter}.}
7052     }
7053     {chapter}{
7054       \int_set:Nn \l_document_structure_section_level_int {1}
7055       \def\thesection{\arabic{chapter}.\arabic{section}}
7056       \def\part@prefix{\arabic{chapter}.}
7057     }
7058   }{
7059     \int_set:Nn \l_document_structure_section_level_int {2}
7060     \def\part@prefix{}
7061   }
7062 }
7063
7064 \bool_if:NF \c__notesslides_notes_bool { % only in slides
```

(*End definition for* `\section@level`. *This function is documented on page* **??**.)

The new counters are used in the `omgroup` environment that choses the LATEX sectioning macros according to `\section@level`.

sfragment

```
7065   \renewenvironment{sfragment}[2][]{
7066     \__document_structure_omgroup_args:n { #1 }
7067     \int_incr:N \l_document_structure_section_level_int
7068     \bool_if:NT \c__notesslides_sectocframes_bool {
7069       \stepcounter{slide}
7070       \begin{frame}[noframenumbering]
7071       \vfill\Large\centering
7072       \red{
7073         \ifcase\l_document_structure_section_level_int\or
```

```
7074        \stepcounter{part}
7075        \def\__notesslideslabel{{\omdoc@part@kw}~\Roman{part}}
7076        \def\currentsectionlevel{\omdoc@part@kw}
7077      \or
7078        \stepcounter{chapter}
7079        \def\__notesslideslabel{{\omdoc@chapter@kw}~\arabic{chapter}}
7080        \def\currentsectionlevel{\omdoc@chapter@kw}
7081      \or
7082        \stepcounter{section}
7083        \def\__notesslideslabel{\part@prefix\arabic{section}}
7084        \def\currentsectionlevel{\omdoc@section@kw}
7085      \or
7086        \stepcounter{subsection}
7087        \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
7088        \def\currentsectionlevel{\omdoc@subsection@kw}
7089      \or
7090        \stepcounter{subsubsection}
7091        \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
7092        \def\currentsectionlevel{\omdoc@subsubsection@kw}
7093      \or
7094        \stepcounter{paragraph}
7095        \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
7096        \def\currentsectionlevel{\omdoc@paragraph@kw}
7097      \else
7098        \def\__notesslideslabel{}
7099        \def\currentsectionlevel{\omdoc@paragraph@kw}
7100      \fi% end ifcase
7101      \__notesslideslabel%\sref@label@id\__notesslideslabel
7102      \quad #2%
7103    }%
7104    \vfill%
7105    \end{frame}%
7106  }
7107  \str_if_empty:NF \l__document_structure_omgroup_id_str {
7108    \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
7109  }
7110 }{}
7111 }
```

We set up a `beamer` template for theorems like ams style, but without a block environment.

```
7112 \def\inserttheorembodyfont{\normalfont}
7113 %\bool_if:NF \c__notesslides_notes_bool {
7114 % \defbeamertemplate{theorem begin}{miko}
7115 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
7116 %   \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
7117 %   \inserttheorempunctuation\inserttheorembodyfont\xspace}
7118 % \defbeamertemplate{theorem end}{miko}{}
```

and we set it as the default one.

```
7119 % \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.

```
7120 % \expandafter\def\csname Parent2\endcsname{}
```

```
7121 %}
7122
7123 \AddToHook{begindocument}{ % this does not work for some reasone
7124   \setbeamertemplate{theorems}[ams style]
7125 }
7126 \bool_if:NT \c__notesslides_notes_bool {
7127   \renewenvironment{columns}[1][]{%
7128     \par\noindent%
7129     \begin{minipage}%
7130     \slidewidth\centering\leavevmode%
7131   }{%
7132     \end{minipage}\par\noindent%
7133   }%
7134   \newsavebox\columnbox%
7135   \renewenvironment<>{column}[2][]{%
7136     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
7137   }{%
7138     \end{minipage}\end{lrbox}\usebox\columnbox%
7139   }%
7140 }
7141 \bool_if:NTF \c__notesslides_noproblems_bool {
7142   \newenvironment{problems}{}{}
7143 }{
7144   \excludecomment{problems}
7145 }
```

38.7   Excursions

\excursion The excursion macros are very simple, we define a new internal macro `\excursionref` and use it in `\excursion`, which is just an `\inputref` that checks if the new macro is defined before formatting the file in the argument.

```
7146 \gdef\printexcursions{}
7147 \newcommand\excursionref[2]{% label, text
7148   \bool_if:NT \c__notesslides_notes_bool {
7149     \begin{sparagraph}[title=Excursion]
7150       #2 \sref[fallback=the appendix]{#1}.
7151     \end{sparagraph}
7152   }
7153 }
7154 \newcommand\activate@excursion[2][]{
7155   \gappto\printexcursions{\inputref[#1]{#2}}
7156 }
7157 \newcommand\excursion[4][]{% repos, label, path, text
7158   \bool_if:NT \c__notesslides_notes_bool {
7159     \activate@excursion[#1]{#3}\excursionref{#2}{#4}
7160   }
7161 }
```

(*End definition for* \excursion. *This function is documented on page* **??**.)

\excursiongroup

```
7162 \keys_define:nn{notesslides / excursiongroup }{
```

258

```
7163    id        .str_set_x:N  = \l__notesslides_excursion_id_str,
7164    intro     .tl_set:N     = \l__notesslides_excursion_intro_tl,
7165    mhrepos   .str_set_x:N  = \l__notesslides_excursion_mhrepos_str
7166  }
7167  \cs_new_protected:Nn \__notesslides_excursion_args:n {
7168    \tl_clear:N \l__notesslides_excursion_intro_tl
7169    \str_clear:N \l__notesslides_excursion_id_str
7170    \str_clear:N \l__notesslides_excursion_mhrepos_str
7171    \keys_set:nn {notesslides / excursiongroup }{ #1 }
7172  }
7173  \newcommand\excursiongroup[1][]{
7174    \__notesslides_excursion_args:n{ #1 }
7175    \ifdefempty\printexcursions{}% only if there are excursions
7176    {\begin{note}
7177      \begin{sfragment}[#1]{Excursions}%
7178        \ifdefempty\l__notesslides_excursion_intro_tl{}{
7179          \inputref[\l__notesslides_excursion_mhrepos_str]{
7180            \l__notesslides_excursion_intro_tl
7181          }
7182        }
7183        \printexcursions%
7184      \end{sfragment}
7185    \end{note}}
7186  }
7187  \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
7188  ⟨/package⟩
```

(*End definition for* \excursiongroup. *This function is documented on page* **??**.)

Chapter 39

The Implementation

39.1 Package Options

The first step is to declare (a few) package options that handle whether certain information is printed or not. They all come with their own conditionals that are set by the options.

```
7189  ⟨*package⟩
7190  ⟨@@=problems⟩
7191  \ProvidesExplPackage{problem}{2022/02/26}{3.0.1}{Semantic Markup for Problems}
7192  \RequirePackage{l3keys2e,stex}
7193
7194  \keys_define:nn { problem / pkg }{
7195    notes     .default:n   = { true },
7196    notes     .bool_set:N  = \c__problems_notes_bool,
7197    gnotes    .default:n   = { true },
7198    gnotes    .bool_set:N  = \c__problems_gnotes_bool,
7199    hints     .default:n   = { true },
7200    hints     .bool_set:N  = \c__problems_hints_bool,
7201    solutions .default:n   = { true },
7202    solutions .bool_set:N  = \c__problems_solutions_bool,
7203    pts       .default:n   = { true },
7204    pts       .bool_set:N  = \c__problems_pts_bool,
7205    min       .default:n   = { true },
7206    min       .bool_set:N  = \c__problems_min_bool,
7207    boxed     .default:n   = { true },
7208    boxed     .bool_set:N  = \c__problems_boxed_bool,
7209    unknown   .code:n      = {}
7210  }
7211  \newif\ifsolutions
7212
7213  \ProcessKeysOptions{ problem / pkg }
7214  \bool_if:NTF \c__problems_solutions_bool {
7215    \solutionstrue
7216  }{
7217    \solutionsfalse
7218  }
```

Then we make sure that the necessary packages are loaded (in the right versions).

`\RequirePackage{comment}`

The next package relies on the LATEX3 kernel, which LATEXMLonly partially supports. As it is purely presentational, we only load it when the `boxed` option is given and we run LATEXML.

7220 `\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.

7221 `\def\prob@problem@kw{Problem}`
7222 `\def\prob@solution@kw{Solution}`
7223 `\def\prob@hint@kw{Hint}`
7224 `\def\prob@note@kw{Note}`
7225 `\def\prob@gnote@kw{Grading}`
7226 `\def\prob@pt@kw{pt}`
7227 `\def\prob@min@kw{min}`

(*End definition for* `\prob@*@kw`*. This function is documented on page* **??**.)

For the other languages, we set up triggers

7228 `\AddToHook{begindocument}{`
7229 `  \ltx@ifpackageloaded{babel}{`
7230 `      \makeatletter`
7231 `      \clist_set:Nx \l_tmpa_clist {\bbl@loaded}`
7232 `      \clist_if_in:NnT \l_tmpa_clist {ngerman}{`
7233 `        \input{problem-ngerman.ldf}`
7234 `      }`
7235 `      \clist_if_in:NnT \l_tmpa_clist {finnish}{`
7236 `        \input{problem-finnish.ldf}`
7237 `      }`
7238 `      \clist_if_in:NnT \l_tmpa_clist {french}{`
7239 `        \input{problem-french.ldf}`
7240 `      }`
7241 `      \clist_if_in:NnT \l_tmpa_clist {russian}{`
7242 `        \input{problem-russian.ldf}`
7243 `      }`
7244 `      \makeatother`
7245 `  }{}`
7246 `}`

39.2 Problems and Solutions

We now prepare the KeyVal support for problems. The key macros just set appropriate internal macros.

7247 `\keys_define:nn{ problem / problem }{`
7248 `  id      .str_set_x:N  = \l__problems_prob_id_str,`
7249 `  pts     .tl_set:N      = \l__problems_prob_pts_tl,`
7250 `  min     .tl_set:N      = \l__problems_prob_min_tl,`
7251 `  title   .tl_set:N      = \l__problems_prob_title_tl,`
7252 `  type    .tl_set:N      = \l__problems_prob_type_tl,`
7253 `  imports .tl_set:N      = \l__problems_prob_imports_tl,`
7254 `  name    .str_set_x:N  = \l__problems_prob_name_str,`
7255 `  refnum  .int_set:N    = \l__problems_prob_refnum_int`

```
7256 }
7257 \cs_new_protected:Nn \__problems_prob_args:n {
7258   \str_clear:N \l__problems_prob_id_str
7259   \str_clear:N \l__problems_prob_name_str
7260   \tl_clear:N \l__problems_prob_pts_tl
7261   \tl_clear:N \l__problems_prob_min_tl
7262   \tl_clear:N \l__problems_prob_title_tl
7263   \tl_clear:N \l__problems_prob_type_tl
7264   \tl_clear:N \l__problems_prob_imports_tl
7265   \int_zero_new:N \l__problems_prob_refnum_int
7266   \keys_set:nn { problem / problem }{ #1 }
7267   \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
7268     \let\l__problems_prob_refnum_int\undefined
7269   }
7270 }
```

Then we set up a counter for problems.

\numberproblemsin

```
7271 \newcounter{problem}[section]
7272 \newcommand\numberproblemsin[1]{\@addtoreset{problem}{#1}}
```

(*End definition for* \numberproblemsin. *This function is documented on page* **??**.)

\prob@label   We provide the macro \prob@label to redefine later to get context involved.

```
7273 \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

```
7274 \newcommand\prob@number{
7275   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
7276     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
7277   }{
7278     \int_if_exist:NTF \l__problems_prob_refnum_int {
7279       \prob@label{\int_use:N \l__problems_prob_refnum_int }
7280     }{
7281       \prob@label\theproblem
7282     }
7283   }
7284 }
```

(*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.

```
7285 \newcommand\prob@title[3]{%
7286   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
7287     #2 \l__problems_inclprob_title_tl #3
7288   }{
7289     \tl_if_exist:NTF \l__problems_prob_title_tl {
7290       #2 \l__problems_prob_title_tl #3
7291     }{
7292       #1
```

262

```
7293        }
7294      }
7295    }
```

(*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.

```
7296  \def\prob@heading{
7297    {\prob@problem@kw}\ \prob@number\prob@title{~}{~(}{)}\strut}
7298    %\sref@label@id{\prob@problem@kw~\prob@number}{}
7299  }
```

(*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
```
7300  \newenvironment{sproblem}[1][]{
7301    \__problems_prob_args:n{#1}%\sref@target%
7302    \@in@omtexttrue% we are in a statement (for inline definitions)
7303    \stepcounter{problem}\record@problem
7304    \def\current@section@level{\prob@problem@kw}
7305
7306    \str_if_empty:NT \l__problems_prob_name_str {
7307      \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
7308      \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
7309      \seq_get_left:NN \l_tmpa_seq \l__problems_prob_name_str
7310    }
7311    \exp_args:Nno\stex_module_setup:nn{type=problem}\l__problems_prob_name_str
7312
7313    \stex_reactivate_macro:N \STEXexport
7314    \stex_reactivate_macro:N \importmodule
7315    \stex_reactivate_macro:N \symdecl
7316    \stex_reactivate_macro:N \notation
7317    \stex_reactivate_macro:N \symdef
7318
7319    \stex_if_do_html:T{
7320      \begin{stex_annotate_env} {problem} {
7321        \l_stex_module_ns_str ? \l_stex_module_name_str
7322      }
7323
7324      \stex_annotate_invisible:nnn{header}{} {
7325        \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
7326        \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
7327        \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
7328          \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
7329        }
7330      }
7331    }
7332
```

```
7333    \stex_csl_to_imports:No \importmodule \l__problems_prob_imports_tl
7334

7335

7336    \tl_if_exist:NTF \l__problems_inclprob_type_tl {
7337      \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
7338    }{
7339      \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
7340    }
7341    \str_if_exist:NTF \l__problems_inclprob_id_str {
7342      \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
7343    }{
7344      \str_set_eq:NN \sproblemid \l__problems_prob_id_str
7345    }

7346

7347

7348    \stex_if_smsmode:F {
7349      \clist_set:No \l_tmpa_clist \sproblemtype
7350      \tl_clear:N \l_tmpa_tl
7351      \clist_map_inline:Nn \l_tmpa_clist {
7352        \tl_if_exist:cT {__problems_sproblem_##1_start:}{
7353          \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_start:}}
7354        }
7355      }
7356      \tl_if_empty:NTF \l_tmpa_tl {
7357        \__problems_sproblem_start:
7358      }{
7359        \l_tmpa_tl
7360      }
7361    }
7362    \stex_ref_new_doc_target:n \sproblemid
7363    \stex_smsmode_do:
7364  }{
7365    \__stex_modules_end_module:
7366    \stex_if_smsmode:F{
7367      \clist_set:No \l_tmpa_clist \sproblemtype
7368      \tl_clear:N \l_tmpa_tl
7369      \clist_map_inline:Nn \l_tmpa_clist {
7370        \tl_if_exist:cT {__problems_sproblem_##1_end:}{
7371          \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_end:}}
7372        }
7373      }
7374      \tl_if_empty:NTF \l_tmpa_tl {
7375        \__problems_sproblem_end:
7376      }{
7377        \l_tmpa_tl
7378      }
7379    }
7380    \stex_if_do_html:T{
7381      \end{stex_annotate_env}
7382    }

7383

7384    \smallskip
7385  }

7386
```

```
7387  \seq_put_right:Nx\g_stex_smsmode_allowedenvs_seq{\tl_to_str:n{sproblem}}
7388
7389
7390
7391  \cs_new_protected:Nn \__problems_sproblem_start: {
7392    \par\noindent\textbf\prob@heading\show@pts\show@min\\\ignorespacesandpars
7393  }
7394  \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
7395
7396  \newcommand\stexpatchproblem[3][] {
7397    \str_set:Nx \l_tmpa_str{ #1 }
7398    \str_if_empty:NTF \l_tmpa_str {
7399      \tl_set:Nn \__problems_sproblem_start: { #2 }
7400      \tl_set:Nn \__problems_sproblem_end: { #3 }
7401    }{
7402      \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
7403      \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
7404    }
7405  }
7406
7407
7408  \bool_if:NT \c__problems_boxed_bool {
7409    \surroundwithmdframed{problem}
7410  }
```

\record@problem This macro records information about the problems in the `*.aux` file.

```
7411  \def\record@problem{
7412    \protected@write\@auxout{}
7413    {
7414      \string\@problem{\prob@number}
7415      {
7416        \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7417          \l__problems_inclprob_pts_tl
7418        }{
7419          \l__problems_prob_pts_tl
7420        }
7421      }%
7422      {
7423        \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7424          \l__problems_inclprob_min_tl
7425        }{
7426          \l__problems_prob_min_tl
7427        }
7428      }
7429    }
7430  }
```

(*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).

```
7431  \def\@problem#1#2#3{}
```

solution   The `solution` environment is similar to the `problem` environment, only that it is independent of the boxed mode. It also has it's own keys that we need to define first.

```
7432 \keys_define:nn { problem / solution }{
7433   id             .str_set_x:N  = \l__problems_solution_id_str ,
7434   for            .tl_set:N     = \l__problems_solution_for_tl ,
7435   height         .dim_set:N    = \l__problems_solution_height_dim ,
7436   creators       .clist_set:N  = \l__problems_solution_creators_clist ,
7437   contributors   .clist_set:N  = \l__problems_solution_contributors_clist ,
7438   srccite        .tl_set:N     = \l__problems_solution_srccite_tl
7439 }
7440 \cs_new_protected:Nn \__problems_solution_args:n {
7441   \str_clear:N \l__problems_solution_id_str
7442   \tl_clear:N \l__problems_solution_for_tl
7443   \tl_clear:N \l__problems_solution_srccite_tl
7444   \clist_clear:N \l__problems_solution_creators_clist
7445   \clist_clear:N \l__problems_solution_contributors_clist
7446   \dim_zero:N \l__problems_solution_height_dim
7447   \keys_set:nn { problem / solution }{ #1 }
7448 }
```

the next step is to define a helper macro that does what is needed to start a solution.

```
7449 \newcommand\@startsolution[1][]{
7450   \__problems_solution_args:n { #1 }
7451   \@in@omtexttrue% we are in a statement.
7452   \bool_if:NF \c__problems_boxed_bool { \hrule }
7453   \smallskip\noindent
7454   {\textbf\prob@solution@kw :\enspace}
7455   \begin{small}
7456   \def\current@section@level{\prob@solution@kw}
7457   \ignorespacesandpars
7458 }
```

\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.

```
7459 \box_new:N \l__problems_solution_box
7460 \newenvironment{solution}{
7461   \stex_html_backend:TF{
7462     \stex_if_do_html:T{
7463       \begin{stex_annotate_env}{solution}{}
7464     }
7465   }{
7466     \setbox\l__problems_solution_box\vbox\bgroup
7467       \par\smallskip\hrule\smallskip
7468       \noindent\textbf{Solution:}~
7469   }
7470 }{
7471   \stex_html_backend:TF{
7472     \stex_if_do_html:T{
7473       \end{stex_annotate_env}
7474     }
7475   }{
```

```
7476        \smallskip\hrule
7477        \egroup
7478        \bool_if:NT \c__problems_solutions_bool {
7479          \box\l__problems_solution_box
7480        }
7481      }
7482    }
7483
7484    \newcommand\startsolutions{
7485      \bool_set_true:N \c__problems_solutions_bool
7486    % \specialcomment{solution}{\@startsolution}{
7487    %   \bool_if:NF \c__problems_boxed_bool {
7488    %     \hrule\medskip
7489    %   }
7490    %   \end{small}%
7491    % }
7492    % \bool_if:NT \c__problems_boxed_bool {
7493    %   \surroundwithmdframed{solution}
7494    % }
7495    }
```

(*End definition for* \startsolutions. *This function is documented on page* **??**.)

```
7496    \newcommand\stopsolutions{\bool_set_false:N \c__problems_solutions_bool}%\excludecomment{sol
```

(*End definition for* \stopsolutions. *This function is documented on page* **??**.)

so it only remains to start/stop solutions depending on what option was specified.

```
7497    \ifsolutions
7498      \startsolutions
7499    \else
7500      \stopsolutions
7501    \fi
```

```
7502    \bool_if:NTF \c__problems_notes_bool {
7503      \newenvironment{exnote}[1][]{
7504        \par\smallskip\hrule\smallskip
7505        \noindent\textbf{\prob@note@kw : }\small
7506      }{
7507        \smallskip\hrule
7508      }
7509    }{
7510      \excludecomment{exnote}
7511    }
```

```
7512    \bool_if:NTF \c__problems_notes_bool {
7513      \newenvironment{hint}[1][]{
7514        \par\smallskip\hrule\smallskip
7515        \noindent\textbf{\prob@hint@kw :~ }\small
7516      }{
7517        \smallskip\hrule
7518      }
```

```
7519    \newenvironment{exhint}[1][]{
7520      \par\smallskip\hrule\smallskip
7521      \noindent\textbf{\prob@hint@kw :~ }\small
7522    }{
7523      \smallskip\hrule
7524    }
7525  }{
7526    \excludecomment{hint}
7527    \excludecomment{exhint}
7528  }
```

gnote

```
7529  \bool_if:NTF \c__problems_notes_bool {
7530    \newenvironment{gnote}[1][]{
7531      \par\smallskip\hrule\smallskip
7532      \noindent\textbf{\prob@gnote@kw : }\small
7533    }{
7534      \smallskip\hrule
7535    }
7536  }{
7537    \excludecomment{gnote}
7538  }
```

39.3   Multiple Choice Blocks

mcb   [18]

```
7539  \newenvironment{mcb}{
7540    \begin{enumerate}
7541  }{
7542    \end{enumerate}
7543  }
```

we define the keys for the mcc macro

```
7544  \cs_new_protected:Nn \__problems_do_yes_param:Nn {
7545    \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
7546      \bool_set_true:N #1
7547    }{
7548      \bool_set_false:N #1
7549    }
7550  }
7551  \keys_define:nn { problem / mcc }{
7552    id        .str_set_x:N  = \l__problems_mcc_id_str ,
7553    feedback  .tl_set:N     = \l__problems_mcc_feedback_tl ,
7554    T         .default:n    = { false } ,
7555    T         .bool_set:N   = \l__problems_mcc_t_bool ,
7556    F         .default:n    = { false } ,
7557    F         .bool_set:N   = \l__problems_mcc_f_bool ,
7558    Ttext     .tl_set:N     = \l__problems_mcc_Ttext_str ,
7559    Ftext     .tl_set:N     = \l__problems_mcc_Ftext_str
7560  }
7561  \cs_new_protected:Nn \l__problems_mcc_args:n {
```

---

[18]EDNOTE: MK: maybe import something better here from a dedicated MC package

268

```
7562    \str_clear:N \l__problems_mcc_id_str
7563    \tl_clear:N \l__problems_mcc_feedback_tl
7564    \bool_set_false:N \l__problems_mcc_t_bool
7565    \bool_set_false:N \l__problems_mcc_f_bool
7566    \tl_clear:N \l__problems_mcc_Ttext_tl
7567    \tl_clear:N \l__problems_mcc_Ftext_tl
7568    \str_clear:N \l__problems_mcc_id_str
7569    \keys_set:nn { problem / mcc }{ #1 }
7570  }
```

\mcc

```
7571  \def\mccTrueText{\textbf{(true)~}}
7572  \def\mccFalseText{\textbf{(false)~}}
7573  \newcommand\mcc[2][]{
7574    \l__problems_mcc_args:n{ #1 }
7575    \item[$\Box$] #2
7576    \ifsolutions
7577      \\
7578      \bool_if:NT \l__problems_mcc_t_bool {
7579        \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccTrueText\l__problems_mcc_Ttext_tl
7580      }
7581      \bool_if:NT \l__problems_mcc_f_bool {
7582        \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccFalseText\l__problems_mcc_Ftext_tl
7583      }
7584      \tl_if_empty:NF \l__problems_mcc_feedback_tl {
7585        \emph{(\l__problems_mcc_feedback_tl)}
7586      }
7587    \fi
7588  } %solutions
```

(*End definition for* \mcc. *This function is documented on page* **??**.)

39.4   Including Problems

\includeproblem   The \includeproblem command is essentially a glorified \input statement, it sets some internal macros first that overwrite the local points. Importantly, it resets the inclprob keys after the input.

```
7589
7590  \keys_define:nn{ problem / inclproblem }{
7591    id      .str_set_x:N = \l__problems_inclprob_id_str,
7592    pts     .tl_set:N    = \l__problems_inclprob_pts_tl,
7593    min     .tl_set:N    = \l__problems_inclprob_min_tl,
7594    title   .tl_set:N    = \l__problems_inclprob_title_tl,
7595    refnum  .int_set:N   = \l__problems_inclprob_refnum_int,
7596    type    .tl_set:N    = \l__problems_inclprob_type_tl,
7597    mhrepos .str_set_x:N = \l__problems_inclprob_mhrepos_str
7598  }
7599  \cs_new_protected:Nn \__problems_inclprob_args:n {
7600    \str_clear:N \l__problems_prob_id_str
7601    \tl_clear:N \l__problems_inclprob_pts_tl
7602    \tl_clear:N \l__problems_inclprob_min_tl
7603    \tl_clear:N \l__problems_inclprob_title_tl
7604    \tl_clear:N \l__problems_inclprob_type_tl
```

```
7605    \int_zero_new:N \l__problems_inclprob_refnum_int
7606    \str_clear:N \l__problems_inclprob_mhrepos_str
7607    \keys_set:nn { problem / inclproblem }{ #1 }
7608    \tl_if_empty:NT \l__problems_inclprob_pts_tl {
7609      \let\l__problems_inclprob_pts_tl\undefined
7610    }
7611    \tl_if_empty:NT \l__problems_inclprob_min_tl {
7612      \let\l__problems_inclprob_min_tl\undefined
7613    }
7614    \tl_if_empty:NT \l__problems_inclprob_title_tl {
7615      \let\l__problems_inclprob_title_tl\undefined
7616    }
7617    \tl_if_empty:NT \l__problems_inclprob_type_tl {
7618      \let\l__problems_inclprob_type_tl\undefined
7619    }
7620    \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
7621      \let\l__problems_inclprob_refnum_int\undefined
7622    }
7623 }
7624
7625 \cs_new_protected:Nn \__problems_inclprob_clear: {
7626    \let\l__problems_inclprob_id_str\undefined
7627    \let\l__problems_inclprob_pts_tl\undefined
7628    \let\l__problems_inclprob_min_tl\undefined
7629    \let\l__problems_inclprob_title_tl\undefined
7630    \let\l__problems_inclprob_type_tl\undefined
7631    \let\l__problems_inclprob_refnum_int\undefined
7632    \let\l__problems_inclprob_mhrepos_str\undefined
7633 }
7634 \__problems_inclprob_clear:
7635
7636 \newcommand\includeproblem[2][]{
7637    \__problems_inclprob_args:n{ #1 }
7638    \exp_args:No \stex_in_repository:nn\l__problems_inclprob_mhrepos_str{
7639      \stex_html_backend:TF {
7640        \str_clear:N \l_tmpa_str
7641        \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
7642          \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
7643        }
7644        \stex_annotate_invisible:nnn{includeproblem}{
7645          \l_tmpa_str / #2
7646        }{}
7647      }{
7648        \begingroup
7649          \inputreftrue
7650          \tl_if_empty:nTF{ ##1 }{
7651            \input{#2}
7652          }{
7653            \input{ \c_stex_mathhub_str / ##1 / source / #2 }
7654          }
7655        \endgroup
7656      }
7657    }
7658    \__problems_inclprob_clear:
```

```
7659  }
```

(*End definition for* `\includeproblem`. *This function is documented on page* **??**.)

39.5   Reporting Metadata

For messages it is OK to have them in English as the whole documentation is, and we can therefore assume authors can deal with it.

```
7660  \AddToHook{enddocument}{
7661    \bool_if:NT \c__problems_pts_bool {
7662      \message{Total:~\arabic{pts}~points}
7663    }
7664    \bool_if:NT \c__problems_min_bool {
7665      \message{Total:~\arabic{min}~minutes}
7666    }
7667  }
```

The margin pars are reader-visible, so we need to translate

```
7668  \def\pts#1{
7669    \bool_if:NT \c__problems_pts_bool {
7670      \marginpar{#1~\prob@pt@kw}
7671    }
7672  }
7673  \def\min#1{
7674    \bool_if:NT \c__problems_min_bool {
7675      \marginpar{#1~\prob@min@kw}
7676    }
7677  }
```

\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.

```
7678  \newcounter{pts}
7679  \def\show@pts{
7680    \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7681      \bool_if:NT \c__problems_pts_bool {
7682        \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
7683        \addtocounter{pts}{\l__problems_inclprob_pts_tl}
7684      }
7685    }{
7686      \tl_if_exist:NT \l__problems_prob_pts_tl {
7687        \bool_if:NT \c__problems_pts_bool {
7688          \tl_if_empty:NT\l__problems_prob_pts_tl{
7689            \tl_set:Nn \l__problems_prob_pts_tl {0}
7690          }
7691          \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
7692          \addtocounter{pts}{\l__problems_prob_pts_tl}
7693        }
7694      }
7695    }
7696  }
```

(*End definition for* \show@pts. *This function is documented on page* **??**.)

and now the same for the minutes

\show@min

```
7697  \newcounter{min}
7698  \def\show@min{
7699    \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7700      \bool_if:NT \c__problems_min_bool {
7701        \marginpar{\l__problems_inclprob_pts_tl\ min}
7702        \addtocounter{min}{\l__problems_inclprob_min_tl}
7703      }
7704    }{
7705      \tl_if_exist:NT \l__problems_prob_min_tl {
7706        \bool_if:NT \c__problems_min_bool {
7707          \tl_if_empty:NT\l__problems_prob_min_tl{
7708            \tl_set:Nn \l__problems_prob_min_tl {0}
7709          }
7710          \marginpar{\l__problems_prob_min_tl\ min}
7711          \addtocounter{min}{\l__problems_prob_min_tl}
7712        }
7713      }
7714    }
7715  }
7716  ⟨/package⟩
```

(*End definition for* \show@min. *This function is documented on page* **??**.)

Chapter 40

Implementation: The hwexam Package

40.1 Package Options

The first step is to declare (a few) package options that handle whether certain information is printed or not. Some come with their own conditionals that are set by the options, the rest is just passed on to the `problems` package.

```
7717 ⟨∗package⟩
7718 \ProvidesExplPackage{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
7719 \RequirePackage{l3keys2e}
7720
7721 \newif\iftest\testfalse
7722 \DeclareOption{test}{\testtrue}
7723 \newif\ifmultiple\multiplefalse
7724 \DeclareOption{multiple}{\multipletrue}
7725 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
7726 \ProcessOptions
```

Then we make sure that the necessary packages are loaded (in the right versions).

```
7727 \RequirePackage{keyval}[1997/11/10]
7728 \RequirePackage{problem}
```

\hwexam@*@kw For multilinguality, we define internal macros for keywords that can be specialized in `*.ldf` files.

```
7729 \newcommand\hwexam@assignment@kw{Assignment}
7730 \newcommand\hwexam@given@kw{Given}
7731 \newcommand\hwexam@due@kw{Due}
7732 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
7733 blank~for~extra~space}
7734 \def\hwexam@minutes@kw{minutes}
7735 \newcommand\correction@probs@kw{prob.}
7736 \newcommand\correction@pts@kw{total}
7737 \newcommand\correction@reached@kw{reached}
7738 \newcommand\correction@sum@kw{Sum}
7739 \newcommand\correction@grade@kw{grade}
7740 \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

```
7741 \AddToHook{begindocument}{
7742 \ltx@ifpackageloaded{babel}{
7743 \makeatletter
7744 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7745 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7746   \input{hwexam-ngerman.ldf}
7747 }
7748 \clist_if_in:NnT \l_tmpa_clist {finnish}{
7749   \input{hwexam-finnish.ldf}
7750 }
7751 \clist_if_in:NnT \l_tmpa_clist {french}{
7752   \input{hwexam-french.ldf}
7753 }
7754 \clist_if_in:NnT \l_tmpa_clist {russian}{
7755   \input{hwexam-russian.ldf}
7756 }
7757 \makeatother
7758 }{}
7759 }
7760
```

40.2 Assignments

Then we set up a counter for problems and make the problem counter inherited from `problem.sty` depend on it. Furthermore, we specialize the `\prob@label` macro to take the assignment counter into account.

```
7761 \newcounter{assignment}
7762 %\numberproblemsin{assignment}
```

We will prepare the keyval support for the `assignment` environment.

```
7763 \keys_define:nn { hwexam / assignment } {
7764 id   .str_set_x:N = \l_@@_assign_id_str,
7765 number  .int_set:N  = \l_@@_assign_number_int,
7766 title  .tl_set:N  = \l_@@_assign_title_tl,
7767 type  .tl_set:N  = \l_@@_assign_type_tl,
7768 given .tl_set:N  = \l_@@_assign_given_tl,
7769 due .tl_set:N  = \l_@@_assign_due_tl,
7770 loadmodules .code:n  = {
7771 \bool_set_true:N \l_@@_assign_loadmodules_bool
7772 }
7773 }
7774 \cs_new_protected:Nn \_@@_assignment_args:n {
7775 \str_clear:N \l_@@_assign_id_str
7776 \int_set:Nn \l_@@_assign_number_int {-1}
7777 \tl_clear:N \l_@@_assign_title_tl
7778 \tl_clear:N \l_@@_assign_type_tl
7779 \tl_clear:N \l_@@_assign_given_tl
7780 \tl_clear:N \l_@@_assign_due_tl
7781 \bool_set_false:N \l_@@_assign_loadmodules_bool
7782 \keys_set:nn { hwexam / assignment }{ #1 }
7783 }
```

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.

```
7784  \newcommand\given@due[2]{
7785  \bool_lazy_all:nF {
7786  {\tl_if_empty_p:V \l_@@_inclassign_given_tl}
7787  {\tl_if_empty_p:V \l_@@_assign_given_tl}
7788  {\tl_if_empty_p:V \l_@@_inclassign_due_tl}
7789  {\tl_if_empty_p:V \l_@@_assign_due_tl}
7790  }{ #1 }
7791
7792  \tl_if_empty:NTF \l_@@_inclassign_given_tl {
7793  \tl_if_empty:NF \l_@@_assign_given_tl {
7794  \hwexam@given@kw\xspace\l_@@_assign_given_tl
7795  }
7796  }{
7797  \hwexam@given@kw\xspace\l_@@_inclassign_given_tl
7798  }
7799
7800  \bool_lazy_or:nnF {
7801  \bool_lazy_and_p:nn {
7802  \tl_if_empty_p:V \l_@@_inclassign_due_tl
7803  }{
7804  \tl_if_empty_p:V \l_@@_assign_due_tl
7805  }
7806  }{
7807  \bool_lazy_and_p:nn {
7808  \tl_if_empty_p:V \l_@@_inclassign_due_tl
7809  }{
7810  \tl_if_empty_p:V \l_@@_assign_due_tl
7811  }
7812  }{ ,~ }
7813
7814  \tl_if_empty:NTF \l_@@_inclassign_due_tl {
7815  \tl_if_empty:NF \l_@@_assign_due_tl {
7816  \hwexam@due@kw\xspace \l_@@_assign_due_tl
7817  }
7818  }{
7819  \hwexam@due@kw\xspace \l_@@_inclassign_due_tl
7820  }
7821
7822  \bool_lazy_all:nF {
7823  { \tl_if_empty_p:V \l_@@_inclassign_given_tl }
7824  { \tl_if_empty_p:V \l_@@_assign_given_tl }
7825  { \tl_if_empty_p:V \l_@@_inclassign_due_tl }
7826  { \tl_if_empty_p:V \l_@@_assign_due_tl }
7827  }{ #2 }
7828  }
```

`\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.

```
7829 \newcommand\assignment@title[3]{
7830 \tl_if_empty:NTF \l_@@_inclassign_title_tl {
7831 \tl_if_empty:NTF \l_@@_assign_title_tl {
7832 #1
7833 }{
7834 #2\l_@@_assign_title_tl#3
7835 }
7836 }{
7837 #2\l_@@_inclassign_title_tl#3
7838 }
7839 }
```

(*End definition for* \assignment@title. *This function is documented on page* **??**.)

\assignment@number Like \assignment@title only for the number, and no around part.

```
7840 \newcommand\assignment@number{
7841 \int_compare:nNnTF \l_@@_inclassign_number_int = {-1} {
7842 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
7843 \arabic{assignment}
7844 } {
7845 \int_use:N \l_@@_assign_number_int
7846 }
7847 }{
7848 \int_use:N \l_@@_inclassign_number_int
7849 }
7850 }
```

(*End definition for* \assignment@number. *This function is documented on page* **??**.)

With them, we can define the central assignment environment. This has two forms (separated by \ifmultiple) in one we make a title block for an assignment sheet, and in the other we make a section heading and add it to the table of contents. We first define an assignment counter

assignment For the assignment environment we delegate the work to the @assignment environment that depends on whether multiple option is given.

```
7851 \newenvironment{assignment}[1][]{
7852 \_@@_assignment_args:n { #1 }
7853 %\sref@target
7854 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
7855 \global\stepcounter{assignment}
7856 }{
7857 \global\setcounter{assignment}{\int_use:N\l_@@_assign_number_int}
7858 }
7859 \setcounter{problem}{0}
7860 \renewcommand\prob@label[1]{\assignment@number.##1}
7861 \def\current@section@level{\document@hwexamtype}
7862 %\sref@label@id{\document@hwexamtype \thesection}
7863 \begin{@assignment}
7864 }{
7865 \end{@assignment}
7866 }
```

276

In the multi-assignment case we just use the omdoc environment for suitable sectioning.

```
7867 \def\ass@title{
7868 {\protect\document@hwexamtype}~\arabic{assignment}
7869 \assignment@title{}{\;(}{)\;} -- \given@due{}{}
7870 }
7871 \ifmultiple
7872 \newenvironment{@assignment}{
7873 \bool_if:NTF \l_@@_assign_loadmodules_bool {
7874 \begin{sfragment}[loadmodules]{\ass@title}
7875 }{
7876 \begin{sfragment}{\ass@title}
7877 }
7878 }{
7879 \end{sfragment}
7880 }
```

for the single-page case we make a title block from the same components.

```
7881 \else
7882 \newenvironment{@assignment}{
7883 \begin{center}\bf
7884 \Large\@title\strut\\
7885 \document@hwexamtype~\arabic{assignment}\assignment@title{\;}{:\;}{\\}
7886 \large\given@due{--\;}{\;--}
7887 \end{center}
7888 }{}
7889 \fi% multiple
```

40.3   Including Assignments

\in*assignment   This macro is essentially a glorified \include statement, it just sets some internal macros first that overwrite the local points Importantly, it resets the inclassig keys after the input.

```
7890 \keys_define:nn { hwexam / inclassignment } {
7891 %id   .str_set_x:N = \l_@@_assign_id_str,
7892 number  .int_set:N  = \l_@@_inclassign_number_int,
7893 title  .tl_set:N  = \l_@@_inclassign_title_tl,
7894 type  .tl_set:N  = \l_@@_inclassign_type_tl,
7895 given .tl_set:N  = \l_@@_inclassign_given_tl,
7896 due .tl_set:N  = \l_@@_inclassign_due_tl,
7897 mhrepos  .str_set_x:N = \l_@@_inclassign_mhrepos_str
7898 }
7899 \cs_new_protected:Nn \_@@_inclassignment_args:n {
7900 \int_set:Nn \l_@@_inclassign_number_int {-1}
7901 \tl_clear:N \l_@@_inclassign_title_tl
7902 \tl_clear:N \l_@@_inclassign_type_tl
7903 \tl_clear:N \l_@@_inclassign_given_tl
7904 \tl_clear:N \l_@@_inclassign_due_tl
7905 \str_clear:N \l_@@_inclassign_mhrepos_str
7906 \keys_set:nn { hwexam / inclassignment }{ #1 }
7907 }
7908 \_@@_inclassignment_args:n {}
7909
7910 \newcommand\inputassignment[2][]{
```

277

```
7911 \_@@_inclassignment_args:n { #1 }
7912 \str_if_empty:NTF \l_@@_inclassign_mhrepos_str {
7913 \input{#2}
7914 }{
7915 \stex_in_repository:nn{\l_@@_inclassign_mhrepos_str}{
7916 \input{\mhpath{\l_@@_inclassign_mhrepos_str}{#2}}
7917 }
7918 }
7919 \_@@_inclassignment_args:n {}
7920 }
7921 \newcommand\includeassignment[2][]{
7922 \newpage
7923 \inputassignment[#1]{#2}
7924 }
```

(*End definition for* \in*assignment. *This function is documented on page* **??**.)

40.4 Typesetting Exams

\quizheading

```
7925 \ExplSyntaxOff
7926 \newcommand\quizheading[1]{%
7927 \def\@tas{#1}%
7928 \large\noindent NAME: \hspace{8cm}   MAILBOX:\\[2ex]%
7929 \ifx\@tas\@empty\else%
7930 \noindent TA:~\@for\@I:=\@tas\do{{\Large$\Box$}\@I\hspace*{1em}}\\[2ex]%
7931 \fi%
7932 }
7933 \ExplSyntaxOn
```

(*End definition for* \quizheading. *This function is documented on page* **??**.)

\testheading

```
7934
7935 \def\hwexamheader{\input{hwexam-default.header}}
7936
7937 \def\hwexamminutes{
7938 \tl_if_empty:NTF \testheading@duration {
7939 {\testheading@min}~\hwexam@minutes@kw
7940 }{
7941 \testheading@duration
7942 }
7943 }
7944
7945 \keys_define:nn { hwexam / testheading } {
7946 min   .tl_set:N  = \testheading@min,
7947 duration .tl_set:N  = \testheading@duration,
7948 reqpts .tl_set:N  = \testheading@reqpts,
7949 tools .tl_set:N  = \testheading@tools
7950 }
7951 \cs_new_protected:Nn \_@@_testheading_args:n {
7952 \tl_clear:N \testheading@min
7953 \tl_clear:N \testheading@duration
```

```
7954  \tl_clear:N \testheading@reqpts
7955  \tl_clear:N \testheading@tools
7956  \keys_set:nn { hwexam / testheading }{ #1 }
7957  }
7958  \newenvironment{testheading}[1][]{
7959  \_@@_testheading_args:n{ #1 }
7960  \newcount\check@time\check@time=\testheading@min
7961  \advance\check@time by -\theassignment@totalmin
7962  \newif\if@bonuspoints
7963  \tl_if_empty:NTF \testheading@reqpts {
7964  \@bonuspointsfalse
7965  }{
7966  \newcount\bonus@pts
7967  \bonus@pts=\theassignment@totalpts
7968  \advance\bonus@pts by -\testheading@reqpts
7969  \edef\bonus@pts{\the\bonus@pts}
7970  \@bonuspointstrue
7971  }
7972  \edef\check@time{\the\check@time}
7973
7974  \makeatletter\hwexamheader\makeatother
7975  }{
7976  \newpage
7977  }
```

(*End definition for* \testheading. *This function is documented on page* **??**.)

\testspace

```
7978  \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}
```

(*End definition for* \testspace. *This function is documented on page* **??**.)

\testnewpage

```
7979  \newcommand\testnewpage{\iftest\newpage\fi}
```

(*End definition for* \testnewpage. *This function is documented on page* **??**.)

\testemptypage

```
7980  \newcommand\testemptypage[1][]{\iftest\begin{center}\hwexam@testemptypage@kw\end{center}\vfi
```

(*End definition for* \testemptypage. *This function is documented on page* **??**.)

\@problem This macro acts on a problem's record in the *.aux file. Here we redefine it (it was
defined to do nothing in problem.sty) to generate the correction table.

```
7981  ⟨@@=problems⟩
7982  \renewcommand\@problem[3]{
7983  \stepcounter{assignment@probs}
7984  \def\__problemspts{#2}
7985  \ifx\__problemspts\@empty\else
7986  \addtocounter{assignment@totalpts}{#2}
7987  \fi
7988  \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\f
7989  \xdef\correction@probs{\correction@probs & #1}%
7990  \xdef\correction@pts{\correction@pts & #2}
7991  \xdef\correction@reached{\correction@reached &}
```

279

```
7992  }
7993  ⟨@@=hwexam⟩
```

(*End definition for* `\@problem`*. This function is documented on page* **??***.*)

`\correction@table` This macro generates the correction table

```
7994  \newcounter{assignment@probs}
7995  \newcounter{assignment@totalpts}
7996  \newcounter{assignment@totalmin}
7997  \def\correction@probs{\correction@probs@kw}
7998  \def\correction@pts{\correction@pts@kw}
7999  \def\correction@reached{\correction@reached@kw}
8000  \stepcounter{assignment@probs}
8001  \newcommand\correction@table{
8002  \resizebox{\textwidth}{!}{%
8003  \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
8004  &\multicolumn{\theassignment@probs}{c||}%|
8005  {\footnotesize\correction@forgrading@kw} &\\\hline
8006  \correction@probs & \correction@sum@kw & \correction@grade@kw\\\hline
8007  \correction@pts &\theassignment@totalpts & \\\hline
8008  \correction@reached & & \\[.7cm]\hline
8009  \end{tabular}}}
8010  ⟨/package⟩
```

(*End definition for* `\correction@table`*. This function is documented on page* **??***.*)

40.5   Leftovers

at some point, we may want to reactivate the logos font, then we use

```
here we define the logos that characterize the assignment
\font\bierfont=../assignments/bierglas
\font\denkerfont=../assignments/denker
\font\uhrfont=../assignments/uhr
\font\warnschildfont=../assignments/achtung

\newcommand\bierglas{{\bierfont\char65}}
\newcommand\denker{{\denkerfont\char65}}
\newcommand\uhr{{\uhrfont\char65}}
\newcommand\warnschild{{\warnschildfont\char 65}}
\newcommand\hardA{\warnschild}
\newcommand\longA{\uhr}
\newcommand\thinkA{\denker}
\newcommand\discussA{\bierglas}
```

Chapter 41

Bibliography

[]   *sLaTeX/RusTeX*. URL: https://github.com/sLaTeX/RusTeX (visited on 04/22/2022).

[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).

[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.

[SIa]   *sLaTeX/sTeX-IDE*. URL: https://github.com/slatex/sTeX-IDE (visited on 04/22/2022).

[SIb]   *sLaTeX/stexls-vscode-plugin*. URL: https://github.com/slatex/stexls-vscode-plugin (visited on 04/22/2022).

[SLS]   *sLaTeX/stexls*. URL: https://github.com/slatex/stexls (visited on 04/22/2022).

[ST]   *sTeX – An Infrastructure for Semantic Preloading of LaTeX Documents*. URL: https://ctan.org/pkg/stex (visited on 04/22/2022).

[sTeX]   *sTeX: A semantic Extension of TeX/LaTeX*. URL: https://github.com/sLaTeX/sTeX (visited on 05/11/2020).

[Tana]   Till Tantau. *beamer – A LaTeX class for producing presentations and slides.* URL: http://ctan.org/pkg/beamer (visited on 01/07/2014).

[Tanb]   Till Tantau. *User Guide to the Beamer Class.* URL: http://ctan.org/macros/latex/contrib/beamer/doc/beameruserguide.pdf.

[TL]   *TeX Live.* URL: http://www.tug.org/texlive/ (visited on 12/11/2012).