The sTeX3 Package Collection *

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

2022-04-26

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 2022-04-26)

i

Contents

Part I
Manual

<div style="border: 2px solid red; border-radius: 8px; padding: 8px;">

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.

</div>

<div style="border: 2px solid green; border-radius: 8px; padding: 8px;">

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

</div>

Chapter 1

What is sTeX?

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

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

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

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

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

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

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

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

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   sTEX Archives (Manual Setup)

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

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

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

Note that sTEX archives often depend on other archives, thus you should be prepared to clone these as well – e.g. if `pdflatex` reports missing files. To make sure that sTEX too knows where to find its archives, we need to set a global system variable `MATHHUB`, that points to your local `MathHub`-directory (see section 3.2).

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

2.1.4   The sTEX IDE

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

ENP:1

2.1.5   Manual Setup for Active Documents and Knowledge Management Services

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

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 sTEX/Mmt content archives.

- **sTEX Archives** If we only care about LATEX and generating `pdf`s, we do not technically need Mmt at all; however, we still need the `MATHHUB` system variable to be set. Furthermore, Mmt can make downloading content archives we might want to use significantly easier, since it makes sure that all dependencies of (often highly interrelated) sTEX archives are cloned as well.

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

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

---

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

4

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 SƬEX-*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). SƬEX 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

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 SτεX yields pretty colors and tooltips[1]. But SτεX becomes a lot more powerful if we additionally convert our document to `xhtml` while preserving all the SτεX markup in the result.

TODO VSCode Plugin

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

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

---
[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→
⤳T↝
- sTeX archives are simultaneously MMT archives, and the same directory structure is consequently used.
- sTeX modules correspond to OMDoc/MMT *theories*. `\importmodule`s (and

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

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

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

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?

12

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

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

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

3.2.4 Using Files in sTeX Archives Directly

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

`\mhinput`

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

`\inputref`

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

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

`\ifinput`

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

`\addmhbibresource`

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

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

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

`\libinput`

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

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

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

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

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

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

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

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

3.3 Module, Symbol and Notation Declarations

3.3.1 The `smodule`-Environment

smodule A new module is declared using the basic syntax

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Example 1

Input:

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

Output:

```
Hello World
```

.

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

For example:

Example 2

Input:

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

Output:

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

.

3.3.2 Declaring New Symbols and Notations

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

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

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

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

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

Example 3

Input:

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

Output:

> Given a foo, we can...

.

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

Example 4

Input:

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

Output:

> this is a symbol taking two arguments.

.

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

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

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

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

.

> ↩M→ Applications of semantic macros, such as `\binarysymbol{a}{b}` are translated to
> —M→ MMT/OMDOC as `OMA`-terms with head `<OMS name="...?binarysymbol"/>`.
> ⤳T↝ Semantic macros with no arguments correspond to `OMS` directly.

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

Output:

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

.

> ↩M→
> —M→ `\symbolname!` is translated to OMDoc/Mmt as `<OMS name="...?symbolname"/>`
> ↝T↝ directly.

3.3.3 Argument Modes

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

Mode-b Arguments

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

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

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

Example 9

Input:

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

Output:

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

.

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

Mode-a Arguments

Mode-`a` arguments represent a *flexary argument sequence*, i.e. a sequence of arguments of arbitrary length. Formally, operators that take arbitrarily many arguments don't "exist", but in informal mathematics, they are ubiquitous. Mode-`a` arguments allow us to write e.g. `\addition{a,b,c,d,e}` rather than having to write something like `\addition{a}{\addition{b}{\addition{c}{\addition{d}{e}}}}`!

`\notation` (and consequently `\symdef`, too) take one additional argument for each mode-`a` argument that indicates how to "accumulate" a comma-separated sequence of arguments. This is best demonstrated on an example.

Let's say we want an operator representing quantification over an ascending chain of elements in some set, i.e. `\ascendingchain{S}{a,b,c,d,e}{t}` should yield $\forall a <_S b <_S c <_S d <_S e. t$. The "base"-notation for this operator is simply
`{\comp{\forall} #2\comp{.\,}#3}`, where `#2` represents the full notation fragment *accumulated* from `{a,b,c,d,e}`.

The *additional* argument to `\notation` (or `\symdef`) takes the same arguments as the base notation and two *additional* arguments `##1` and `##2` representing successive pairs in the mode-`a` argument, and accumulates them into `#2`, i.e. to produce $a <_S b <_S c <_S d <_S e$, we do `{##1 \comp{<}_{#1} ##2}`:

Example 10

Input:

```
1 \symdef{ascendingchain}[args=iai]
2   {\comp{\forall} #2\comp{.\,}#3}
3   {##1 \comp{<}_{#1} ##2}
4
5 Tadaa: $\ascendingchain{S}{a,b,c,d,e}{t}$
```

Output:

Tadaa: $\forall a<_S b<_S c<_S d<_S e.\, t$

.

If this seems overkill, keep in mind that you will rarely need the single-hash arguments `#1`,`#2` etc. in the `a`-notation-argument. For a much more representative and simpler example, we can introduce flexary addition via:

Example 11

Input:

```
1   \symdef{addition}[args=a]{#1}{##1 \comp{+} ##2}
2
3 Tadaa: $\addition{a,b,c,d,e}$
```

Output:

Tadaa: $a+b+c+d+e$

.

The `assoc`-key   We mentioned earlier that "formally", flexary arguments don't really "exist". Indeed, formally, addition is usually defined as a binary operation, quantifiers bind a single variable etc.

Consequently, we can tell sTEX (or, rather, MMT/OMDOC) how to "resolve" flexary arguments by providing `\symdecl` or `\symdef` with an optional `assoc`-argument, as in `\symdecl{addition}[args=a,assoc=bin]`. The possible values for the `assoc`-key are:

`bin`: A binary, associative argument, e.g. as in `\addition`

`binl`: A binary, left-associative argument, e.g. $a^{b^{c^d}}$, which stands for $((a^b)^c)^d$

`binr`: A binary, right-associative argument, e.g. as in $A \to B \to C \to D$, which stands for $A \to (B \to (C \to D))$

`pre`: Successively prefixed, e.g. as in $\forall x, y, z.\, P$, which stands for $\forall x.\, \forall y.\, \forall z.\, P$

`conj`: Conjunctive, e.g. as in $a = b = c = d$ or $a, b, c, d \in A$, which stand for $a = d \wedge b = d \wedge c = d$ and $a \in A \wedge b \in A \wedge c \in A \wedge d \in A$, respectively

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

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

Mode-B Arguments

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

Example 12

Input:

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

Output:

$$\forall x,y,z.P$$

.

3.3.4 Type and Definiens Components

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

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

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

Example 13

Input:

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

Output:

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

.

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

Example 14

Input:

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

Output:

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

.

3.3.5 Precedences and Automated Bracketing

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

Example 15

Input:

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

Output:

> multiplication is an operation $\mathbb{N}{\times}\mathbb{N}{\rightarrow}\mathbb{N}$

.

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

Example 16

Input:

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

Output:

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

.

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

Example 17

Input:

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

Output:

```
a+b·(c+d·e)
```

˙but we can also do better by supplying *precedences* and have ꟙTEX 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·(c+d·e)
```

.

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

`\infprec`
`\neginfprec`

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

More precisely, each notation takes

1. One *operator precedence* and

2. one *argument precedence* for each argument.

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

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

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

In the example above:

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

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

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

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

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

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

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

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

3.3.6   Variables

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

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

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

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

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

> **Example 19**
> Input:
>
> ```
> 1 \vardef{varf}[
> 2     name=f,
> 3     type=\funtype{\Nat}{\Nat},
> 4     op=f,
> 5     args=1,
> 6     prec=0;\neginfprec
> 7 ]{\comp{f}#1}
> 8 \vardef{varn}[name=n,type=\Nat]{\comp{n}}
> 9 \vardef{varx}[name=x,type=\Nat]{\comp{x}}
> 10
> 11 Given a function $\varf!:\funtype{\Nat}{\Nat}$,
> 12 by $\addition{\varf!,\varn}$ we mean the function
> 13 $\fun{\varx}{\varf{\addition{\varx,\varn}}}$
> ```
>
> Output:
>
> > Given a function $f : \mathbb{N} \to \mathbb{N}$, by $f + n$ we mean the function $x \mapsto f(x + n)$

.

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

TODO: bind=forall/exists

3.3.7 Variable Sequences

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

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

This is best shown by example:

> **Example 20**
> Input:

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

Output:

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

.

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

TODO: more notations for invoking sequences.

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

Example 21

Input:

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

Output:

$a_1 + \ldots + a_n$

.

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

Example 22

Input:

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

Output:

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

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

Example 23

Input:

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

Output:

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

.

3.4 Module Inheritance and Structures

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

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

3.4.1 Multilinguality and Translations

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

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

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

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

If a module Foo exists in e.g. english in a file `Foo.en.tex`, we can provide a file `Foo.de.tex` right next to it, and write **\begin{smodule}[sig=en]{Foo}**. The `sig`-key

then signifies, that the "signature" of the module is contained in the *english* version of the module, which is immediately imported from there, just like `\importmodule` would.

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

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

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

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

3.4.2 Simple Inheritance and Namespaces

`\importmodule`
`\usemodule`

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

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

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

Ideally, SₜₑX would use arbitrary URIs for modules, with no forced relationships between the *logical* namespace of a module and the *physical* location of the file declaring the module – like Mᴍᴛ does things.

Unfortunately, TₑX only provides very restricted access to the file system, so we are forced to generate namespaces systematically in such a way that they reflect the physical location of the associated files, so that SₜₑX can resolve them accordingly. Largely, users need not concern themselves with namespaces at all, but for completenesses sake, we describe how they are constructed:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.4.3   The `mathstructure` Environment

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

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

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

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

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

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

How this works is again best demonstrated by example:

Example 24

Input:

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

Output:

> A [monoid](#) is...

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

Example 25

Input:

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

Output:

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

.

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

Example 26

Input:

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

Output:

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

.

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

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

> `\instantiate` and `mathstructure` make use of the *Theories-as-Types* paradigm (see [MRK18]):
> `mathstructure{<name>}` simply creates a nested theory with name `<name>-structure`. The *constant* `<name>` is defined as `Mod(<name>-structure)` – a *dependent record type with manifest fields*, the fields of which are generated

from (and correspond to) the constants in `<name>-structure`.
`\instantiate` generates a constant whose definiens is a record term of type `Mod(<name>-structure)`, with the fields assigned based on the respective key-value-list.

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

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

`\varinstantiate`

The answer is that we can also instantiate a `mathstructure` with a *variable*. The syntax of `\varinstantiate` is equivalent to that of `\instantiate`, but all of the key-value-pairs are optional, and if not explicitly assigned (to a symbol *or* a variable declared with `\vardef`) inherit their notation from the one in the `mathstructure` environment.

This allows us to do things like:

Example 27

Input:

```
1 \varinstantiate{varM}{monoid}{M}
2
3 A \symname{monoid} is a structure
4 $\varM!:=\mathstruct{\varM{universe},\varM{op}!,\varM{unit}}$
5 such that
6 $\varM{op}!:\funtype{\varM{universe},\varM{universe}}{\varM{universe}}$ ...
```

Output:

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

.

and

Example 28

Input:

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

Output:

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

.

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

3.4.4 The `copymodule` Environment

Given modules:

Example 29

Input:

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

Output:

.

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

Example 30

Input:

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

Output:

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

34

.

TODO: explain donotclone

3.4.5 The `interpretmodule` Environment

TODO: explain

> **Example 31**
>
> Input:
>
> ```
>  1 \begin{smodule}{int}
>  2     \symdef{Integers}{\comp{\mathbb Z}}
>  3     \symdef{plus}[args=2,op=+]{#1 \comp+ #2}
>  4     \symdef{zero}{\comp0}
>  5     \symdef{uminus}[args=1,op=-]{\comp-#1}
>  6
>  7     \begin{interpretmodule}{group}{intisgroup}
>  8         \assign{universe}{\Integers}
>  9         \assign{operation}{\plus!}
> 10         \assign{unit}{\zero}
> 11         \assign{inverse}{\uminus!}
> 12     \end{interpretmodule}
> 13 \end{smodule}
> ```
>
> Output:
>

.

3.5 Primitive Symbols (The sTeX Metatheory)

The stex-metatheory package contains sTeX symbols so ubiquitous, that it is virtually impossible to describe any flexiformal content without them, or that are required to annotate even the most primitive symbols with meaningful (foundation-independent) "type"-annotations, or required for basic structuring principles (theorems, definitions). As such, it serves as the default meta theory for any sTeX module.

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

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

We make this theory part of the sTeX collection rather than encoding it in sTeX itself[4]

EdN:4

---

[4]EDNOTE: MK: why? continue

35

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

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

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

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

> **Example 32**
> Input:
>
> ```
> 1 \symdef{Nat}[
> 2     name=natural-number,
> 3     type=\set
> 4 ]{\comp{\mathbb{N}}}
> 5
> 6 A \symname{Nat} is...
> ```
>
> Output:
>
> > A natural number is...

.

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

`\Symname`

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

> **Example 33**
> Input:
>
> ```
> 1 \Symname[post=s]{Nat} are...
> ```
>
> Output:
>
> > Natural numbers are...

.

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

4.2   Marking Up Text and On-the-Fly Notations

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

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

> **Example 34**
> Input:
>
> ```
> 1 \addition{\comp{The sum of} \arg{$\svar{n}$} \comp{ and }\arg{$\svar{m}$}}
> 2 is...
> ```
>
> Output:
>
> > The sum of n and m is...

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

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

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

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

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

Example 35
Input:

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

Output:

```
  Addition is...
```

.

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

`\arg` also allows us to switch the order of arguments around and "hide" arguments: For example, `\arg[3]{<code>}` signifies that `<code>` represents the *third* argument to the current operator, and `\arg*[i]{<code>}` signifies that `<code>` represents the ith argument, but it should not produce any output (it is exported in the `xhtml` however, so that Mmt and other systems can pick up on it).[5]

EdN:5

Example 36
Input:

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

Output:

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

> adding k yields...

˙Note that since the second `\arg` has no explicit argument number, it automatically represents the first not-yet-given argument – i.e. in this case the first one.[6]

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

Example 37

Input:

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

Output:

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

.

4.3   Referencing Symbols and Statements

TODO: references documentation

---

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

Chapter 5

sTEX Statements

5.1 Definitions, Theorems, Examples, Paragraphs

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

The corresponding environments for that are:

- `sdefinition` for definitions,

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

- `sexample` for examples and counterexamples, and

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

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

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

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

Example 38

Input:

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

Output:

> **Example 5.1.1** (An Example)**.** $2+3$ is 5, $2\cdot3$ is 6.

.

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

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

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

All four statement environments – i.e. sdefinition, sassertion, sexample, and sparagraph – also take an optional parameter name= – if this one is given a value, the environment will generate a *symbol* by that name (but with no semantic macro). Not only does this allow for \symref et al, it allows us to resume our earlier example for monoids much more nicely:[7]

> **Example 39**
> Input:

---

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

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

Output:

A **monoid** is a structure $\langle U, \circ, e \rangle$ where $\circ : U \to U$ and $e \in U$ such that

Axiom 5.1.2 (Associativity). $\circ$ *is associative*

Axiom 5.1.3 (Unit). $x \circ e = x$ *for all* $x \in U$

An example for a monoid is...

.

The main difference to before[8] is that the two `sassertion`s now have `name=` attributes. Thus the `mathstructure` monoid now contains two additional symbols, namely the axioms for associativity and that e is a unit. Note that both symbols do not represent the mere *propositions* that e.g. $\circ$ is associative, but *the assertion that it is actually true* that $\circ$ is associative.

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

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

---

[8]EDNOTE: MK: reference

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

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

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

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

This yields the following result:

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

---

---

[2]Of course, SLEX can not check that the assertions are the "correct" ones – but if the assertions (both in `monoid` as well as those for addition and zero) are properly marked up, MMT can. TODO: should

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

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

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 $1 + 3 = 2^2 = 4$ $\square$

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. $\square$

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

                                                    $\square$

sproof   The `sproof` environment is the main container for proofs. It takes an optional `KeyVal` argument that allows to specify the `id` (identifier) and `for` (for which assertion is this a proof) keys. The regular argument of the `proof` environment contains an introductory comment, that may be used to announce the proof style. The `proof` environment contains a sequence of `spfstep`, `spfcomment`, and `spfcases` environments that are used to markup the proof steps.

\spfidea   The `\spfidea` macro allows to give a one-paragraph description of the proof idea.

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

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

\spfjust	This evidence is marked up with the `\spfjust` macro in the stex-proofs package. This environment totally invisible to the formatted result; it wraps the text in the proof step that corresponds to the evidence. The environment takes an optional `KeyVal` argument, which can have the `method` key, whose value is the name of a proof method (this will only need to mean something to the application that consumes the semantic annotations). Furthermore, the justification can contain "premises" (specifications to assertions that were used justify the step) and "arguments" (other information taken into account by the proof method).
\premise	The `\premise` macro allows to mark up part of the text as reference to an assertion that is used in the argumentation. In the running example we have used the `\premise` macro to identify the inductive hypothesis.
\justarg	The `\justarg` macro is very similar to `\premise` with the difference that it is used to mark up arguments to the proof method. Therefore the content of the first argument is interpreted as a mathematical object rather than as an identifier as in the case of `\premise`. In our example, we specified that the simplification should take place on the right hand side of the equation. Other examples include proof methods that instantiate. Here we would indicate the substituted object in a `\justarg` macro. Note that both `\premise` and `\justarg` can be used with an empty second argument to mark up premises and arguments that are not explicitly mentioned in the text.
subproof	The `spfcases` environment is used to mark up a subproof. This environment takes an optional `KeyVal` argument for semantic annotations and a second argument that allows to specify an introductory comment (just like in the `proof` environment). The `method` key can be used to give the name of the proof method executed to make this subproof.
spfcases	The `spfcases` environment is used to mark up a proof by cases. Technically it is a variant of the `subproof` where the `method` is `by-cases`. Its contents are `spfcase` environments that mark up the cases one by one.
spfcase	The content of a `spfcases` environment are a sequence of case proofs marked up in the `spfcase` environment, which takes an optional `KeyVal` argument for semantic annotations. The second argument is used to specify the the description of the case under consideration. The content of a `spfcase` environment is the same as that of a `sproof`, i.e. `spfsteps`, `spfcomments`, and `spfcases` environments.
\spfcasesketch	`\spfcasesketch` is a variant of the `spfcase` environment that takes the same arguments, but instead of the `spfsteps` in the body uses a third argument for a proof sketch.
spfcomment	The `spfcomment` environment is much like a `step`, only that it does not have an object-level assertion of its own. Rather than asserting some fact that is relevant for the proof, it is used to explain where the proof is going, what we are attempting to to, or what we have achieved so far. As such, it cannot be the target of a `\premise`.

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

The stex-proofs package provides the \sproofend macro for this.

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

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

Chapter 6

Highlighting and Presentation Customizations

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

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

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

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

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

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

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

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

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

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

47

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

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

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

\compemph
\varemph
\symrefemph
\defemph

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

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

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

For each of the four macros, there exists an additional macro that takes the full URI of the relevant symbol currently being highlighted as a second argument. That allows us to e.g. use pdf tooltips and links. For example, this document uses[9]

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

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

Chapter 7

Additional Packages

7.1   Tikzinput: Treating TIKZ code as images

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

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

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

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

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

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

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

\mhtikzinput
\cmhtikzinput

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

\libusetikzlibrary

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

7.2 Modular Document Structuring

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

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

DAG models of documents allow to replace the "Copy and Paste" in the source document with a label-and-reference model where document are shared in the document source and the formatter does the copying during document formatting/presentation.

The document-structure package accepts the following options:

class=⟨*name*⟩	load ⟨*name*⟩.cls instead of article.cls
topsect=⟨*sect*⟩	The top-level sectioning level; the default for ⟨*sect*⟩ is section

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

EdN:10 section. If the title contains semantic macros, they need to be protected by \protect[10], and we need to give the loadmodules key it needs no value. For instance we would have

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

---

[10]EDNOTE: MK: still?

$\text{S}\!\!\int\!\!\text{T}_{\!E}\!X$ automatically computes the sectioning level, from the nesting of `sfragment` environments.

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

blindfragment | Therefore the `document-structure` package provides a variant `blindfragment` that does not produce markup, but increments the sectioning level and logically groups document parts that belong together, but where traditional document markup relies on convention rather than explicit markup. The `blindfragment` environment is useful e.g. for creating frontmatter at the correct level. The example below shows a typical setup for the outer document structure of a book with parts and chapters.

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

Here we use two levels of `blindfragment`:

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

- The inner one groups the frontmatter[3] and makes the preface of the book a section-level construct.[11]

EdN:11

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

---

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

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

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

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

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

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

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

`\ifSGvar`	With `\ifSGvar` we can test for the contents of a global variable: the macro call `\ifSGvar{⟨vname⟩}{⟨val⟩}{⟨ctext⟩}` tests the content of the global variable ⟨vname⟩, only if (after expansion) it is equal to ⟨val⟩, the conditional text ⟨ctext⟩ is formatted.

7.3 Slides and Course Notes

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

The notesslides class takes the notion of a slide frame from Till Tantau's excellent beamer class and adapts its notion of frames for use in the sTeX and OMDoc. To support semantic course notes, it extends the notion of mixing frames and explanatory text, but rather than treating the frames as images (or integrating their contents into the flowing text), the notesslides package displays the slides as such in the course notes to give students a visual anchor into the slide presentation in the course (and to distinguish the different writing styles in slides and course notes).

In practice we want to generate two documents from the same source: the slides for presentation in the lecture and the course notes as a narrative document for home study. To achieve this, the notesslides class has two modes: *slides mode* and *notes mode* which are determined by the package option.

The notesslides class takes a variety of class options:

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

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

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

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

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

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

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

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

---

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

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

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

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

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

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

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

`\setlicensing` For licensing, we use the Creative Commons Attribuition-ShareAlike license by default to strengthen the public domain. If package `hyperref` is loaded, then we can attach a hyperlink to the license logo. `\setlicensing[⟨url⟩]{⟨logo name⟩}` is used for customization, where ⟨url⟩ is optional.

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

\frameimage
\mhframeimage

In this case we can use `\frameimage[⟨opt⟩]{⟨path⟩}`, where ⟨opt⟩ are the options of `\includegraphics` from the graphicx package [CR99] and ⟨path⟩ is the file path (extension can be left off like in `\includegraphics`). We have added the `label` key that allows to give a frame label that can be referenced like a regular `beamer` frame.

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

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

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

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

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

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

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

\textwarning

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

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

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

\excursion

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

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

\activateexcursion
\printexcursion
\excursionref

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

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

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

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

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

7.4 Representing Problems and Solutions

The `problem` package supplies an infrastructure that allows specify problem. Problems are text fragments that come with auxiliary functions: hints, notes, and solutions[5]. Furthermore, we can specify how long the solution to a given problem is estimated to take and how many points will be awarded for a perfect solution.

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

solutions
notes
hints
gnotes
pts
min
boxed
test

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

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

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 environment takes an optional KeyVal argument with the keys `id` as an identifier that can be reference later, `pts` for the points to be gained from this exercise in homework or quiz situations, `min` for the estimated minutes needed to solve the problem, and finally `title` for an informative title of the problem.

---

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

Example 40

Input:

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

Output:

Problem 7.4.1 (Fitting Elefants)

How many Elefants can you fit into a Volkswagen beetle?

Hint: Think positively, this is simple!

Note: Justify your answer

Solution: Four, two in the front seats, and two in the back.

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

.

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

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

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

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

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

\mcc \mcc[⟨*keyvals*⟩]{⟨*text*⟩} takes an optional key/value argument ⟨*keyvals*⟩ for choice metadata and a required argument ⟨*text*⟩ for the proposed answer text. The following keys are supported

- T for true answers, F for false ones,

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

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

If we start the solutions, then we get

Example 41

Input:

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

Output:

Problem 7.4.2 (Functions)
What is the keyword to introduce a function definition in python?

☐ def
   (true)

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

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

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

EdN:12 ˙without solutions (that is what the students see during the exam/quiz)[12]

---

[12]EDNOTE: MK: that did not work!

58

Example 42

Input:

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

Output:

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

.

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

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

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

7.5   Homeworks, Quizzes and Exams

The hwexam package and class supplies an infrastructure that allows to format nice-looking assignment sheets by simply including problems from problem files marked up

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

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

solutions
notes
hints
gnotes
pts
min

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

assignment
number

title
type
given
due
multiple

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

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

test

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

\testspace
\testnewpage
\testemptypage

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

testheading
duration
min
reqpts

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

</div>

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

Will result in

Name: Matriculation Number:

320101 General Computer Science (Fall 2010)

2022-04-26

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

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

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

prob.	To be used for grading, do not write here										Sum	grade
	7.4.1	7.4.2	7.4.3	1.1	2.1	2.2	2.3	3.1	3.2	3.3	Sum	grade
total	10			4	4	6	6	4	4	2	40	
reached												

good luck

EdN:13   [13]

\inputassignment   The \inputassignment macro can be used to input an assignment from another file. It
takes an optional KeyVal argument and a second argument which is a path to the file con-
taining 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.

---

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

Part II

Documentation

Chapter 8

sTeX-Basics

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

8.1 Macros and Environments

`\sTeX`
`\stex`
Both print this sTEX logo.

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

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

8.1.1 HTML Annotations

`\if@latexml`
LaTeX2e conditional for LaTeXML

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

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

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

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

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

Annotates the HTML generated by ⟨*content*⟩ with

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

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

8.1.2 Babel Languages

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

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

8.1.3 Auxiliary Methods

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

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

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

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

`\ignorespacesandpars`

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

Chapter 9

sTEX-MathHub

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

9.1 Macros and Environments

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

9.1.1 Files, Paths, URIs

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

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

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

\stex_path_canonicalize:N Canonicalizes the path provided; in particular, resolves . and .. path segments.

\stex_path_if_absolute_p:N ⋆
\stex_path_if_absolute:N*TF* ⋆

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

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

65

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

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

9.1.2　MathHub Archives

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

1. The `mathhub` package option, or

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

3. the `MATHHUB` system variable, or

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

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

`\l_stex_current_repository_prop`

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

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

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

narr: the narration namespace (for document references),

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

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

`\stex_set_current_repository:n`

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

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

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

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

9.1.3 Using Content in Archives

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

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

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

    Both also set `\ifinputref` to true.

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

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

\libinput   `\libinput{`⟨*filename*⟩`}`

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

\libusepackage   `\libusepackage[`⟨*args*⟩`]{`⟨*filename*⟩`}`

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

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

\mhgraphics   *If* the graphicx package is loaded, these macros are defined at `\begin{document}`.
\cmhgraphics
    `\mhgraphics` takes the same arguments as `\includegraphics`, with the additional optional key `mhrepos`. It then resolves the file path in `\mhgraphics[mhrepos=Foo/Bar]{foo/bar.png}` relative to the `source`-folder of the `Foo/Bar`-archive.

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

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

Chapter 10

sTEX-References

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

10.1 Macros and Environments

$\STEXreftitle$ \STEXreftitle{⟨*some title*⟩}

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

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

\l_stex_current_docns_str Stores its result in \l_stex_current_docns_str

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

\l_stex_current_docurl_str Stores its result in \l_stex_current_docurl_str

10.1.1 Setting Reference Targets

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

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

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

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

10.1.2 Using References

\sref   \sref[⟨*opt-args*⟩]{⟨*id*⟩}

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

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

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

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

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

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

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

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

Chapter 11

sTEX-Modules

This sub package contains code related to Modules

11.1   Macros and Environments

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

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

name The *name* of the module,

ns the *namespace* in field `ns`,

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

lang the module's *language*,

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

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

meta the metatheory of the module.

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

`\c_stex_module_<URI>_constants`

The names of all constants declared in the module

`\c_stex_module_<URI>_constants`

The full URIs of all modules imported in this module

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

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

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

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

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.

71

Computes the current namespace as follows:

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

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

11.1.1 The `smodule` environment

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

`\STEXModule`   `\STEXModule {⟨fragment⟩}`

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

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

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

Chapter 12

sTeX-Module Inheritance

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

12.1 Macros and Environments

12.1.1 SMS Mode

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

`\g_stex_smsmode_allowedmacros_tl`

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

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

`\g_stex_smsmode_allowedmacros_escape_tl`

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

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

`\g_stex_smsmode_allowedenvs_seq`

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

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

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

Tests whether SMS mode is currently active.

`\stex_file_in_smsmode:nn`	`\stex_in_smsmode:nn {⟨filename⟩} {⟨code⟩}`

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

`\stex_smsmode_do:`	Starts gobbling tokens until one is encountered that is allowed in SMS mode.

12.1.2 Imports and Inheritance

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

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

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

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

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

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

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

1. If ⟨*archive-ID*⟩ is empty:

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

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

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

2. Otherwise:

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

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

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

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

`\l_stex_import_name_str`
`\l_stex_import_archive_str`   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 SₜₑX brackets (by default (and)), which can be changed temporarily using `\withbrackets`.

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

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

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

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

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

`\comp` `\compemph` `\compemph@uri` `\defemph` `\defemph@uri` `\symrefemph` `\symrefemph@uri` `\varemph` `\varemph@uri`	`\comp`{⟨*args*⟩}

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

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

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

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

`\ellipses`	TODO

Chapter 15

sTeX-Structural Features

Code related to structural features

15.1 Macros and Environments

15.1.1 Structures

mathstructure TODO

Chapter 16

sTₑX-Statements

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

16.1 Macros and Environments

symboldoc   `\begin{`⟨*symboldoc*⟩`}{`⟨*symbols*⟩`}` ⟨*text*⟩ `\end{`⟨*symboldoc*⟩`}`

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

Chapter 17

sTeX-Proofs: Structural Markup for Proofs

Chapter 18

sTeX-Metatheory

18.1 Symbols

Part III

Extensions

Chapter 19

Tikzinput: Treating TIKZ code as images

19.1 Macros and Environments

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

Chapter 20

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

Chapter 21

NotesSlides – Slides and Course Notes

Chapter 22

`problem.sty`: An Infrastructure for formatting Problems

Chapter 23

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

Part IV
Implementation

Chapter 24

sTeX -Basics Implementation

24.1 The sTeXDocument Class

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

```
1 ⟨∗cls⟩
2
3 %%%%%%%%%%%%%   basics.dtx   %%%%%%%%%%%%%
4
5 \RequirePackage{expl3,l3keys2e}
6 \ProvidesExplClass{stex}{2022/03/03}{3.1.0}{sTeX document class}
7
8 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
9 \ProcessOptions
10
11 \bool_set_true:N \c_stex_document_class_bool
12
13 \RequirePackage{stex}
14
15 \stex_html_backend:TF {
16   \LoadClass{article}
17 }{
18   \LoadClass[border=1px,varwidth,crop=false]{standalone}
19   \setlength\textwidth{15cm}
20 }
21 \RequirePackage{standalone}
22
23
24 \clist_if_empty:NT \c_stex_languages_clist {
25   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
26   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
27   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
28   \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
29     \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
30       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
```

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

24.2   Preliminaries

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

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

`\stex`   The STEXlogo:
`\sTeX`
```
75  \RequirePackage{xspace}
76  \protected\def\stex{
77    \@ifundefined{texorpdfstring}{\let\texorpdfstring\@firstoftwo}{}
```

```
78    \texorpdfstring{\raisebox{-.5ex}S\kern-.5ex\TeX}{sTeX}\xspace
79  }
80  \let\sTeX\stex
```

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

24.3 Messages and logging

```
81  ⟨@@=stex_log⟩
```

Warnings and error messages

```
82  \msg_new:nnn{stex}{error/unknownlanguage}{
83    Unknown~language:~#1
84  }
85  \msg_new:nnn{stex}{warning/nomathhub}{
86    MATHHUB~system~variable~not~found~and~no~
87    \detokenize{\mathhub}-value~set!
88  }
89  \msg_new:nnn{stex}{error/deactivated-macro}{
90    The~\detokenize{#1}~command~is~only~allowed~in~#2!
91  }
```

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

```
92  \cs_new_protected:Nn \stex_debug:nn {
93    \clist_if_in:NnTF \c_stex_debug_clist { all } {
94      \msg_set:nnn{stex}{debug / #1}{
95        \\Debug~#1:~#2\\
96      }
97      \msg_none:nn{stex}{debug / #1}
98    }{
99      \clist_if_in:NnT \c_stex_debug_clist { #1 } {
100       \msg_set:nnn{stex}{debug / #1}{
101         \\Debug~#1:~#2\\
102       }
103       \msg_none:nn{stex}{debug / #1}
104     }
105   }
106 }
```

(*End definition for* \stex_debug:nn*. This function is documented on page* *63.*)

Redirecting messages:

```
107 \clist_if_in:NnTF \c_stex_debug_clist {all} {
108     \msg_redirect_module:nnn{ stex }{ none }{ term }
109 }{
110   \clist_map_inline:Nn \c_stex_debug_clist {
111     \msg_redirect_name:nnn{ stex }{ debug / ##1 }{ term }
112   }
113 }
114
115 \stex_debug:nn{log}{debug~mode~on}
```

24.4   HTML Annotations

`\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.

```
117 \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`

```
118 \cs_new_protected:Nn \_stex_html_checkempty:n {
119   \tl_set:Nn \l_stex_html_arg_tl { #1 }
120   \tl_if_empty:NT \l_stex_html_arg_tl {
121     \tl_set_eq:NN \l_stex_html_arg_tl \c_stex_html_emptyarg_tl
122   }
123 }
```

(*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

```
124 \bool_new:N \_stex_html_do_output_bool
125 \bool_set_true:N \_stex_html_do_output_bool
126
127 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
128   \bool_if:nTF \_stex_html_do_output_bool
129     \prg_return_true: \prg_return_false:
130 }
```

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

`\stex_suppress_html:n`

Whether to (locally) produce HTML output

```
131 \cs_new_protected:Nn \stex_suppress_html:n {
132   \exp_args:Nne \use:nn {
133     \bool_set_false:N \_stex_html_do_output_bool
134     #1
135   }{
136     \stex_if_do_html:T {
137       \bool_set_true:N \_stex_html_do_output_bool
138     }
139   }
140 }
```

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

`\stex_annotate:nnn`
`\stex_annotate_invisible:n`
`\stex_annotate_invisible:nnn`

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

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

```
141 \tl_if_exist:NF\stex@backend{
142   \ifcsname if@rustex\endcsname
143     \def\stex@backend{rustex}
144   \else
145     \ifcsname if@latexml\endcsname
```

```
146        \def\stex@backend{latexml}
147      \else
148        \def\stex@backend{pdflatex}
149      \fi
150    \fi
151 }
152 \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 64.*)

24.5   Babel Languages

```
153 ⟨@@=stex_language⟩
```

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

\c_stex_languages_prop
\c_stex_language_abbrevs_prop

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

(*End definition for* \c_stex_languages_prop *and* \c_stex_language_abbrevs_prop. *These variables are documented on page 64.*)

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

```
179 \cs_new_protected:Nn \stex_set_language:Nn {
180    \str_set:Nx \l_tmpa_str {#2}
181    \prop_get:NoNT \c_stex_languages_prop \l_tmpa_str #1 {
182      \ifx\@onlypreamble\@notprerr
183        \ltx@ifpackageloaded{babel}{
184          \exp_args:No \selectlanguage #1
185        }{}
186      \else
187        \exp_args:No \str_if_eq:nnTF #1 {turkish} {
```

```
188        \RequirePackage[#1,shorthands=:!]{babel}
189      }{
190        \RequirePackage[#1]{babel}
191      }
192    \fi
193  }
194 }

195
196 \clist_if_empty:NF \c_stex_languages_clist {
197   \bool_set_false:N \l_tmpa_bool
198   \clist_clear:N \l_tmpa_clist
199   \clist_map_inline:Nn \c_stex_languages_clist {
200     \str_set:Nx \l_tmpa_str {#1}
201     \str_if_eq:nnT {#1}{tr}{
202       \bool_set_true:N \l_tmpa_bool
203     }
204     \prop_get:NoNTF \c_stex_languages_prop \l_tmpa_str \l_tmpa_str {
205       \clist_put_right:No \l_tmpa_clist \l_tmpa_str
206     } {
207       \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
208     }
209   }
210   \stex_debug:nn{lang} {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
211   \bool_if:NTF \l_tmpa_bool {
212     \RequirePackage[\clist_use:Nn \l_tmpa_clist,,shorthands=:!]{babel}
213   }{
214     \RequirePackage[\clist_use:Nn \l_tmpa_clist,]{babel}
215   }
216 }

217
218 \AtBeginDocument{
219   \stex_html_backend:T {
220     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
221     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
222     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
223     \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
224     \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
225       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
226       \stex_debug:nn{basics} {Language~\l_tmpa_str~
227         inferred~from~file~name}
228       \stex_annotate_invisible:nnn{language}{ \l_tmpa_str }{}
229     }
230   }
231 }
```

24.6   Persistence

```
232 ⟨@@=stex_persist⟩
233 \bool_if:NTF \c_stex_persist_mode_bool {
234   \def \stex_persist:n #1 {}
235   \def \stex_persist:x #1 {}
236 }{
237   \bool_if:NTF \c_stex_persist_write_mode_bool {
```

```
238   \iow_new:N \c__stex_persist_iow
239   \iow_open:Nn \c__stex_persist_iow{\jobname.sms}
240   \AtEndDocument{
241     \iow_close:N \c__stex_persist_iow
242   }
243   \cs_new_protected:Nn \stex_persist:n {
244     \tl_set:Nn \l_tmpa_tl { #1 }
245     \regex_replace_all:nnN { \cP\# } { \cO\# } \l_tmpa_tl
246     \exp_args:NNo \iow_now:Nn \c__stex_persist_iow \l_tmpa_tl
247   }
248   \cs_generate_variant:Nn \stex_persist:n {x}
249   }{
250     \def \stex_persist:n #1 {}
251     \def \stex_persist:x #1 {}
252   }
253 }
```

24.7   Auxiliary Methods

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

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

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

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

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

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

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

\MMTrule

```
330  \NewDocumentCommand \MMTrule {m m}{
331    \seq_set_split:Nnn \l_tmpa_seq , {#2}
332    \int_zero:N \l_tmpa_int
333    \stex_annotate_invisible:nnn{mmtrule}{scala://#1}{
334      \seq_if_empty:NF \l_tmpa_seq {
335        $\seq_map_inline:Nn \l_tmpa_seq {
336          \int_incr:N \l_tmpa_int
337          \stex_annotate:nnn{arg}{i\int_use:N \l_tmpa_int}{##1}
338        }$
339      }
340    }
341  }
342
343  \NewDocumentCommand \MMTinclude {m}{
344    \stex_annotate_invisible:nnn{import}{#1}{}
345  }
346
347  \tl_new:N \g_stex_document_title
348  \cs_new_protected:Npn \STEXtitle #1 {
349    \tl_if_empty:NT \g_stex_document_title {
350      \tl_gset:Nn \g_stex_document_title { #1 }
351    }
352  }
353  \cs_new_protected:Nn \stex_document_title:n {
354    \tl_if_empty:NT \g_stex_document_title {
355      \tl_gset:Nn \g_stex_document_title { #1 }
356      \stex_annotate_invisible:n{\noindent
357        \stex_annotate:nnn{doctitle}{}{ #1 }
358      \par}
359    }
360  }
361  \AtBeginDocument {
362    \let \STEXtitle \stex_document_title:n
363    \tl_if_empty:NF \g_stex_document_title {
364      \stex_annotate_invisible:n{\noindent
365        \stex_annotate:nnn{doctitle}{}{ \g_stex_document_title }
366      \par}
367    }
368    \let\_stex_maketitle:\maketitle
369    \def\maketitle{
370      \tl_if_empty:NF \@title {
371        \exp_args:No \stex_document_title:n \@title
372      }
373      \_stex_maketitle:
374    }
375  }
376
377  ⟨/package⟩
```

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

Chapter 25

STEX -MathHub Implementation

```
378 ⟨∗package⟩
379
380 %%%%%%%%%%%%    mathhub.dtx    %%%%%%%%%%%%
381
382 ⟨@@=stex_path⟩
```

Warnings and error messages

```
383 \msg_new:nnn{stex}{error/norepository}{
384   No~archive~#1~found~in~#2
385 }
386 \msg_new:nnn{stex}{error/notinarchive}{
387   Not~currently~in~an~archive,~but~\detokenize{#1}~
388   needs~one!
389 }
390 \msg_new:nnn{stex}{error/nofile}{
391   \detokenize{#1}~could~not~find~file~#2
392 }
393 \msg_new:nnn{stex}{error/twofiles}{
394   \detokenize{#1}~found~two~candidates~for~#2
395 }
```

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

```
396 \cs_new_protected:Nn \stex_path_from_string:Nn {
397   \str_set:Nx \l_tmpa_str { #2 }
398   \str_if_empty:NTF \l_tmpa_str {
399     \seq_clear:N #1
400   }{
401     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
402     \sys_if_platform_windows:T{
403       \seq_clear:N \l_tmpa_tl
```

```
404       \seq_map_inline:Nn #1 {
405         \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
406         \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
407       }
408       \seq_set_eq:NN #1 \l_tmpa_tl
409     }
410     \stex_path_canonicalize:N #1
411   }
412 }
413
```

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

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

```
414 \cs_new_protected:Nn \stex_path_to_string:NN {
415   \exp_args:NNe \str_set:Nn #2 { \seq_use:Nn #1 / }
416 }
417
418 \cs_new:Nn \stex_path_to_string:N {
419   \seq_use:Nn #1 /
420 }
```

(*End definition for* `\stex_path_to_string:NN` *and* `\stex_path_to_string:N`. *These functions are documented on page* *65*.)

`\c__stex_path_dot_str`
`\c__stex_path_up_str`

`.` and `..`, respectively.

```
421 \str_const:Nn \c__stex_path_dot_str {.}
422 \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.

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

```
445              \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
446            }
447          }
448        }{
449          \str_if_empty:NF \l_tmpa_tl {
450            \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
451          }
452        }
453      }
454    }
455    \seq_gset_eq:NN #1 \l_tmpa_seq
456  }
457 }
```

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

```
458 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
459   \seq_if_empty:NTF #1 {
460     \prg_return_false:
461   }{
462     \seq_get_left:NN #1 \l_tmpa_tl
463     \sys_if_platform_windows:TF{
464       \str_if_in:NnTF \l_tmpa_tl {:}{
465         \prg_return_true:
466       }{
467         \prg_return_false:
468       }
469     }{
470       \str_if_empty:NTF \l_tmpa_tl {
471         \prg_return_true:
472       }{
473         \prg_return_false:
474       }
475     }
476   }
477 }
```

(*End definition for* `\stex_path_if_absolute:NTF`. *This function is documented on page 65.*)

25.2   PWD and kpsewhich

```
478 \str_new:N\l_stex_kpsewhich_return_str
479 \cs_new_protected:Nn \stex_kpsewhich:n {
480   \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
481   \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
482   \tl_trim_spaces:N \l_stex_kpsewhich_return_str
483 }
```

(*End definition for* `\stex_kpsewhich:n`. *This function is documented on page 65.*)

We determine the PWD

```
484 \sys_if_platform_windows:TF{
485   \begingroup\escapechar=-1\catcode`\\=12
486   \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
487   \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
488   \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_
489 }{
490   \stex_kpsewhich:n{-var-value~PWD}
491 }
492
493 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
494 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
495 \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 65*.)

25.3   File Hooks and Tracking

```
496 ⟨@@=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

```
497 \seq_gclear_new:N\g__stex_files_stack
```

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

```
498 \str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
499 \stex_path_from_string:Nn \c_stex_mainfile_seq
500   \c_stex_mainfile_str
```

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

```
501 \seq_gclear_new:N\g_stex_currentfile_seq
```

(*End definition for* `\g_stex_currentfile_seq`*. This variable is documented on page 66*.)

```
502 \cs_new_protected:Nn \stex_filestack_push:n {
503   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
504   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
505     \stex_path_from_string:Nn\g_stex_currentfile_seq{
506       \c_stex_pwd_str/#1
507     }
508   }
509   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
510   \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
511 }
```

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

`\stex_filestack_pop:`

```
512 \cs_new_protected:Nn \stex_filestack_pop: {
513   \seq_if_empty:NF\g__stex_files_stack{
514     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
515   }
516   \seq_if_empty:NTF\g__stex_files_stack{
517     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
518   }{
519     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
520     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
521   }
522 }
```

Hooks for the current file:

```
523 \AddToHook{file/before}{
524   \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
525 }
526 \AddToHook{file/after}{
527   \stex_filestack_pop:
528 }
```

25.4 MathHub Repositories

```
529 ⟨@@=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.

```
530 \str_if_empty:NTF\mathhub{
531   \sys_if_platform_windows:TF{
532     \begingroup\escapechar=-1\catcode`\\=12
533     \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str MATHHUB\c_percent_str}
534     \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
535     \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_ste
536   }{
537     \stex_kpsewhich:n{-var-value~MATHHUB}
538   }
539   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
540
541   \str_if_empty:NT \c_stex_mathhub_str {
542     \sys_if_platform_windows:TF{
543       \begingroup\escapechar=-1\catcode`\\=12
544       \exp_args:Nx\stex_kpsewhich:n{-var-value~HOME}
545       \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
546       \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_s
547     }{
548       \stex_kpsewhich:n{-var-value~HOME}
549     }
550     \ior_open:NnT \l_tmpa_ior{\l_stex_kpsewhich_return_str / .stex / mathhub.path}{
551       \begingroup\escapechar=-1\catcode`\\=12
552       \ior_str_get:NN \l_tmpa_ior \l_tmpa_str
```

```
553        \sys_if_platform_windows:T{
554            \exp_args:NNx\str_replace_all:Nnn\l_tmpa_str{\c_backslash_str}/
555        }
556        \str_gset_eq:NN \c_stex_mathhub_str\l_tmpa_str
557        \endgroup
558        \ior_close:N \l_tmpa_ior
559      }
560    }
561    \str_if_empty:NTF\c_stex_mathhub_str{
562      \msg_warning:nn{stex}{warning/nomathhub}
563    }{
564      \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
565      \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
566    }
567  }{
568    \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
569    \stex_path_if_absolute:NF \c_stex_mathhub_seq {
570      \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
571        \c_stex_pwd_str/\mathhub
572      }
573    }
574    \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
575    \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
576  }
```

(*End definition for* \mathhub, \c_stex_mathhub_seq, *and* \c_stex_mathhub_str. *These variables are documented on page 66.*)

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

```
577  \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
578    \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
579      \str_set:Nx \l_tmpa_str { #1 }
580      \prop_new:c { c_stex_mathhub_#1_manifest_prop }
581      \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
582      \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
583      \__stex_mathhub_find_manifest:N \l_tmpa_seq
584      \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
585        \msg_error:nnxx{stex}{error/norepository}{#1}{
586          \stex_path_to_string:N \c_stex_mathhub_str
587        }
588      } {
589        \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
590      }
591    }
592  }
```

(*End definition for* __stex_mathhub_do_manifest:n.)

\l__stex_mathhub_manifest_file_seq

```
593  \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`:

```
594 \cs_new_protected:Nn \__stex_mathhub_find_manifest:N {
595   \seq_set_eq:NN\l_tmpa_seq #1
596   \bool_set_true:N\l_tmpa_bool
597   \bool_while_do:Nn \l_tmpa_bool {
598     \seq_if_empty:NTF \l_tmpa_seq {
599       \bool_set_false:N\l_tmpa_bool
600     }{
601       \file_if_exist:nTF{
602         \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
603       }{
604         \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
605         \bool_set_false:N\l_tmpa_bool
606       }{
607         \file_if_exist:nTF{
608           \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
609         }{
610           \seq_put_right:Nn\l_tmpa_seq{META-INF}
611           \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
612           \bool_set_false:N\l_tmpa_bool
613         }{
614           \file_if_exist:nTF{
615             \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
616           }{
617             \seq_put_right:Nn\l_tmpa_seq{meta-inf}
618             \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
619             \bool_set_false:N\l_tmpa_bool
620           }{
621             \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
622           }
623         }
624       }
625     }
626   }
627   \seq_set_eq:NN\l__stex_mathhub_manifest_file_seq\l_tmpa_seq
628 }
```

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

`\c__stex_mathhub_manifest_ior` File variable used for `MANIFEST`-files

```
629 \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:

```
630 \cs_new_protected:Nn \__stex_mathhub_parse_manifest:n {
631   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
632   \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
633   \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
634     \str_set:Nn \l_tmpa_str {##1}
635     \exp_args:NNoo \seq_set_split:Nnn
636         \l_tmpb_seq \c_colon_str \l_tmpa_str
637     \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
```

```
638        \exp_args:NNe \str_set:Nn \l_tmpb_tl {
639          \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
640        }
641        \exp_args:No \str_case:nnTF \l_tmpa_tl {
642          {id} {
643            \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
644              { id } \l_tmpb_tl
645          }
646          {narration-base} {
647            \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
648              { narr } \l_tmpb_tl
649          }
650          {url-base} {
651            \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
652              { docurl } \l_tmpb_tl
653          }
654          {source-base} {
655            \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
656              { ns } \l_tmpb_tl
657          }
658          {ns} {
659            \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
660              { ns } \l_tmpb_tl
661          }
662          {dependencies} {
663            \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
664              { deps } \l_tmpb_tl
665          }
666        }{}{}
667      }{}
668    }
669    \ior_close:N \c__stex_mathhub_manifest_ior
670    \stex_persist:x {
671      \prop_set_from_keyval:cn{ c_stex_mathhub_#1_manifest_prop }{
672        \exp_after:wN \prop_to_keyval:N \csname c_stex_mathhub_#1_manifest_prop\endcsname
673      }
674    }
675 }
```

(End definition for __stex_mathhub_parse_manifest:n.*)*

```
676 \cs_new_protected:Nn \stex_set_current_repository:n {
677    \stex_require_repository:n { #1 }
678    \prop_set_eq:Nc \l_stex_current_repository_prop {
679      c_stex_mathhub_#1_manifest_prop
680    }
681 }
```

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

```
682 \cs_new_protected:Nn \stex_require_repository:n {
683    \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
684      \stex_debug:nn{mathhub}{Opening~archive:~#1}
```

108

```
685    \__stex_mathhub_do_manifest:n { #1 }
686    }
687 }
```

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

`\l_stex_current_repository_prop` Current MathHub repository

```
688 %\prop_new:N \l_stex_current_repository_prop
689 \bool_if:NF \c_stex_persist_mode_bool {
690    \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
691    \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
692       \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
693    } {
694       \__stex_mathhub_parse_manifest:n { main }
695       \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
696          \l_tmpa_str
697       \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str _manifest_prop }
698          \c_stex_mathhub_main_manifest_prop
699       \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
700       \stex_debug:nn{mathhub}{Current~repository:~
701          \prop_item:Nn \l_stex_current_repository_prop {id}
702    }
703    }
704 }
```

(End definition for `\l_stex_current_repository_prop`*. This variable is documented on page 66.)*

`\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.

```
705 \cs_new_protected:Nn \stex_in_repository:nn {
706    \str_set:Nx \l_tmpa_str { #1 }
707    \cs_set:Npn \l_tmpa_cs ##1 { #2 }
708    \str_if_empty:NTF \l_tmpa_str {
709       \prop_if_exist:NTF \l_stex_current_repository_prop {
710          \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
711          \exp_args:Ne \l_tmpa_cs{
712             \prop_item:Nn \l_stex_current_repository_prop { id }
713          }
714       }{
715          \l_tmpa_cs{}
716       }
717    }{
718       \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}
719       \stex_require_repository:n \l_tmpa_str
720       \str_set:Nx \l_tmpa_str { #1 }
721       \exp_args:Nne \use:nn {
722          \stex_set_current_repository:n \l_tmpa_str
723          \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
724       }{
725          \stex_debug:nn{mathhub}{switching~back~to:~
726             \prop_if_exist:NTF \l_stex_current_repository_prop {
727                \prop_item:Nn \l_stex_current_repository_prop { id }:~
728                \meaning\l_stex_current_repository_prop
729             }{
```

109

```
730              no~repository
731           }
732        }
733        \prop_if_exist:NTF \l_stex_current_repository_prop {
734         \stex_set_current_repository:n {
735          \prop_item:Nn \l_stex_current_repository_prop { id }
736         }
737        }{
738          \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
739         }
740      }
741    }
742 }
```

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

25.5   Using Content in Archives

```
743 \def \mhpath #1 #2 {
744    \exp_args:Ne \tl_if_empty:nTF{#1}{
745      \c_stex_mathhub_str /
746        \prop_item:Nn \l_stex_current_repository_prop { id }
747        / source / #2
748    }{
749      \c_stex_mathhub_str / #1 / source / #2
750    }
751 }
```

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

```
752 \newif \ifinputref \inputreffalse
753
754 \cs_new_protected:Nn \__stex_mathhub_mhinput:nn {
755    \stex_in_repository:nn {#1} {
756      \ifinputref
757        \input{ \c_stex_mathhub_str / ##1 / source / #2 }
758      \else
759        \inputreftrue
760        \input{ \c_stex_mathhub_str / ##1 / source / #2 }
761        \inputreffalse
762      \fi
763    }
764 }
765 \NewDocumentCommand \mhinput { O{} m}{
766    \__stex_mathhub_mhinput:nn{ #1 }{ #2 }
767 }
768
769 \cs_new_protected:Nn \__stex_mathhub_inputref:nn {
770    \stex_in_repository:nn {#1} {
771      \stex_html_backend:TF {
772        \str_clear:N \l_tmpa_str
```

```
773       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
774         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
775       }
776       \stex_annotate_invisible:nnn{inputref}{
777         \l_tmpa_str / #2
778       }{}
779     }{
780       \begingroup
781         \inputreftrue
782         \tl_if_empty:nTF{ ##1 }{
783           \input{#2}
784         }{
785           \input{ \c_stex_mathhub_str / ##1 / source / #2 }
786         }
787       \endgroup
788     }
789   }
790 }
791 \NewDocumentCommand \inputref { O{} m }{
792   \__stex_mathhub_inputref:nn{ #1 }{ #2 }
793 }
```

(*End definition for* `\inputref` *and* `\mhinput`*. These functions are documented on page 67.*)

\addmhbibresource

```
794 \cs_new_protected:Nn \__stex_mathhub_mhbibresource:nn {
795   \stex_in_repository:nn {#1} {
796     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
797   }
798 }
799 \newcommand\addmhbibresource[2][]{
800   \__stex_mathhub_mhbibresource:nn{ #1 }{ #2 }
801 }
```

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

\libinput

```
802 \cs_new_protected:Npn \libinput #1 {
803   \prop_if_exist:NF \l_stex_current_repository_prop {
804     \msg_error:nnn{stex}{error/notinarchive}\libinput
805   }
806   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
807     \msg_error:nnn{stex}{error/notinarchive}\libinput
808   }
809   \seq_clear:N \l__stex_mathhub_libinput_files_seq
810   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
811   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
812
813   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
814     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #1.tex}
815     \IfFileExists{ \l_tmpa_str }{
816       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
817     }{}
818     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
819     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
```

111

```
820      }
821
822      \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}
823      \IfFileExists{ \l_tmpa_str }{
824        \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
825      }{}
826
827      \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
828        \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
829      }{
830        \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
831          \input{ ##1 }
832        }
833      }
834    }
```

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

\libusepackage

```
835    \NewDocumentCommand \libusepackage {O{} m} {
836      \prop_if_exist:NF \l_stex_current_repository_prop {
837        \msg_error:nnn{stex}{error/notinarchive}\libusepackage
838      }
839      \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
840        \msg_error:nnn{stex}{error/notinarchive}\libusepackage
841      }
842      \seq_clear:N \l__stex_mathhub_libinput_files_seq
843      \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
844      \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
845
846      \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
847        \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #2}
848        \IfFileExists{ \l_tmpa_str.sty }{
849          \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
850        }{}
851        \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
852        \seq_put_right:No \l_tmpa_seq \l_tmpa_str
853      }
854
855      \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2}
856      \IfFileExists{ \l_tmpa_str.sty }{
857        \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
858      }{}
859
860      \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
861        \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
862      }{
863        \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
864          \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
865            \usepackage[#1]{ ##1 }
866          }
867        }{
868          \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
869        }
```

```
870      }
871    }
```

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

`\mhgraphics`
`\cmhgraphics`

```
872
873  \AddToHook{begindocument}{
874  \ltx@ifpackageloaded{graphicx}{
875      \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
876      \newcommand\mhgraphics[2][]{%
877        \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
878        \includegraphics[#1]{\mhpath\Gin@mhrepos{#2}}}
879      \newcommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
880    }{}
```

(*End definition for* `\mhgraphics` *and* `\cmhgraphics`. *These functions are documented on page* *67.*)

`\lstinputmhlisting`
`\clstinputmhlisting`

```
881  \ltx@ifpackageloaded{listings}{
882      \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
883      \newcommand\lstinputmhlisting[2][]{%
884        \def\lst@mhrepos{}\setkeys{lst}{#1}%
885        \lstinputlisting[#1]{\mhpath\lst@mhrepos{#2}}}
886      \newcommand\clstinputmhlisting[2][]{\begin{center}\lstinputmhlisting[#1]{#2}\end{center}
887    }{}
888  }
889
890  ⟨/package⟩
```

(*End definition for* `\lstinputmhlisting` *and* `\clstinputmhlisting`. *These functions are documented on*
page *67.*)

Chapter 26

sTEX
-References Implementation

891 ⟨∗package⟩

892

893 %%%%%%%%%%%   references.dtx   %%%%%%%%%%%%

894

895 ⟨@@=stex_refs⟩

Warnings and error messages

896

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

897 %\iow_new:N \c__stex_refs_refs_iow

898 \AtBeginDocument{

899 % \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}

900 }

901 \AtEndDocument{

902 % \iow_close:N \c__stex_refs_refs_iow

903 }

\STEXreftitle

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

905

906 \NewDocumentCommand \STEXreftitle { m } {

907   \tl_gset:Nx \g__stex_refs_title_tl { #1 }

908 }

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

26.1 Document URIs and URLs

\l_stex_current_docns_str

909 \str_new:N \l_stex_current_docns_str

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

```
910 \cs_new_protected:Nn \stex_get_document_uri: {
911   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
912   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
913   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
914   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
915   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
916
917   \str_clear:N \l_tmpa_str
918   \prop_if_exist:NT \l_stex_current_repository_prop {
919     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
920       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
921     }
922   }
923
924   \str_if_empty:NTF \l_tmpa_str {
925     \str_set:Nx \l_stex_current_docns_str {
926       file:/\stex_path_to_string:N \l_tmpa_seq
927     }
928   }{
929     \bool_set_true:N \l_tmpa_bool
930     \bool_while_do:Nn \l_tmpa_bool {
931       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
932       \exp_args:No \str_case:nnTF { \l_tmpb_str } {
933         {source} { \bool_set_false:N \l_tmpa_bool }
934       }{}{
935         \seq_if_empty:NT \l_tmpa_seq {
936           \bool_set_false:N \l_tmpa_bool
937         }
938       }
939     }
940
941     \seq_if_empty:NTF \l_tmpa_seq {
942       \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
943     }{
944       \str_set:Nx \l_stex_current_docns_str {
945         \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
946       }
947     }
948   }
949 }
```

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

```
950 \str_new:N \l_stex_current_docurl_str
```

(End definition for \l_stex_current_docurl_str. *This variable is documented on page 68.)*

```
951 \cs_new_protected:Nn \stex_get_document_url: {
952   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
953   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
954   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
```

```
955    \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
956    \seq_put_right:No \l_tmpa_seq \l_tmpb_str
957
958    \str_clear:N \l_tmpa_str
959    \prop_if_exist:NT \l_stex_current_repository_prop {
960      \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
961        \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
962          \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
963        }
964      }
965    }
966
967    \str_if_empty:NTF \l_tmpa_str {
968      \str_set:Nx \l_stex_current_docurl_str {
969        file:/\stex_path_to_string:N \l_tmpa_seq
970      }
971    }{
972      \bool_set_true:N \l_tmpa_bool
973      \bool_while_do:Nn \l_tmpa_bool {
974        \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
975        \exp_args:No \str_case:nnTF { \l_tmpb_str } {
976          {source} { \bool_set_false:N \l_tmpa_bool }
977        }{}{
978          \seq_if_empty:NT \l_tmpa_seq {
979            \bool_set_false:N \l_tmpa_bool
980          }
981        }
982      }
983
984      \seq_if_empty:NTF \l_tmpa_seq {
985        \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
986      }{
987        \str_set:Nx \l_stex_current_docurl_str {
988          \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
989        }
990      }
991    }
992 }
```

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

26.2   Setting Reference Targets

```
993 \str_const:Nn \c__stex_refs_url_str{URL}
994 \str_const:Nn \c__stex_refs_ref_str{REF}
995 \str_new:N \l__stex_refs_curr_label_str
996 % @currentlabel -> number
997 % @currentlabelname -> title
998 % @currentHref -> name.number <- id of some kind
999 % \theH# -> \arabic{section}
1000 % \the#  -> number
1001 % \hyper@makecurrent{#}
1002 \int_new:N \l__stex_refs_unnamed_counter_int
```

```
1003 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
1004   \stex_get_document_uri:
1005   \str_clear:N \l__stex_refs_curr_label_str
1006   \str_set:Nx \l_tmpa_str { #1 }
1007   \str_if_empty:NT \l_tmpa_str {
1008     \int_incr:N \l__stex_refs_unnamed_counter_int
1009     \str_set:Nx \l_tmpa_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
1010   }
1011   \str_set:Nx \l__stex_refs_curr_label_str {
1012     \l_stex_current_docns_str?\l_tmpa_str
1013   }
1014   \seq_if_exist:cF{g__stex_refs_labels_\l_tmpa_str _seq}{
1015     \seq_new:c {g__stex_refs_labels_\l_tmpa_str _seq}
1016   }
1017   \seq_if_in:coF{g__stex_refs_labels_\l_tmpa_str _seq}\l__stex_refs_curr_label_str {
1018     \seq_gput_right:co{g__stex_refs_labels_\l_tmpa_str _seq}\l__stex_refs_curr_label_str
1019   }
1020   \stex_if_smsmode:TF {
1021     \stex_get_document_url:
1022     \str_gset_eq:cN {sref_url_\l__stex_refs_curr_label_str _str}\l_stex_current_docurl_str
1023     \str_gset_eq:cN {sref_\l__stex_refs_curr_label_str _type}\c__stex_refs_url_str
1024   }{
1025     %\iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=~\expandafter\unexpanded\expandafter{
1026     \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
1027     \immediate\write\@auxout{\stexauxadddocref{\l_stex_current_docns_str}{\l_tmpa_str}}
1028     \str_gset:cx {sref_\l__stex_refs_curr_label_str _type}\c__stex_refs_ref_str
1029   }
1030 }
```

(*End definition for* \stex_ref_new_doc_target:n*. This function is documented on page* *68.*)

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

```
1031 \cs_new_protected:Npn \stexauxadddocref #1 #2 {
1032   \str_set:Nn \l_tmpa_str {#1?#2}
1033   \str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
1034   \seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
1035     \seq_new:c {g__stex_refs_labels_#2_seq}
1036   }
1037   \seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
1038     \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
1039   }
1040 }
```

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

```
1041 \AtEndDocument{
1042   \def\stexauxadddocref#1 #2 {}{}
1043 }
```

```
1044 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
1045   \stex_if_smsmode:TF {
1046     \str_if_exist:cF{sref_sym_#1_type}{
1047       \stex_get_document_url:
1048       \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str
```

```
1049        \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
1050      }
1051    }{
1052      \str_if_empty:NF \l__stex_refs_curr_label_str {
1053        \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
1054        \immediate\write\@auxout{
1055          \exp_not:N\expandafter\def\exp_not:N\csname \exp_not:N\detokenize{sref_sym_#1_label_
1056            \l__stex_refs_curr_label_str
1057          }
1058        }
1059      }
1060    }
1061 }
```

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

26.3   Using References

```
1062 \str_new:N \l__stex_refs_indocument_str
```

\sref Optional arguments:

```
1063
1064 \keys_define:nn { stex / sref } {
1065    linktext      .tl_set:N  = \l__stex_refs_linktext_tl ,
1066    fallback      .tl_set:N  = \l__stex_refs_fallback_tl ,
1067    pre           .tl_set:N  = \l__stex_refs_pre_tl ,
1068    post          .tl_set:N  = \l__stex_refs_post_tl ,
1069 }
1070 \cs_new_protected:Nn \__stex_refs_args:n {
1071    \tl_clear:N \l__stex_refs_linktext_tl
1072    \tl_clear:N \l__stex_refs_fallback_tl
1073    \tl_clear:N \l__stex_refs_pre_tl
1074    \tl_clear:N \l__stex_refs_post_tl
1075    \str_clear:N \l__stex_refs_repo_str
1076    \keys_set:nn { stex / sref } { #1 }
1077 }
```

The actual macro:

```
1078 \NewDocumentCommand \sref { O{} m}{
1079    \__stex_refs_args:n { #1 }
1080    \str_if_empty:NTF \l__stex_refs_indocument_str {
1081      \str_set:Nx \l_tmpa_str { #2 }
1082      \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
1083      \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
1084        \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
1085          \seq_get_left:cNF {g__stex_refs_labels_\l_tmpa_str _seq} \l_tmpa_str {
1086            \str_clear:N \l_tmpa_str
1087          }
1088        }{
1089          \str_clear:N \l_tmpa_str
1090        }
1091      }{
1092        \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1093        \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
```

```
1094          \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
1095          \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
1096            \str_set_eq:NN \l_tmpc_str \l_tmpa_str
1097            \str_clear:N \l_tmpa_str
1098            \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str _seq} {
1099              \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
1100                \str_range:nnn { ##1 }{ -\l_tmpa_int}{ -1 }
1101              }{
1102                \seq_map_break:n {
1103                  \str_set:Nn \l_tmpa_str { ##1 }
1104                }
1105              }
1106            }
1107          }{
1108            \str_clear:N \l_tmpa_str
1109          }
1110        }
1111        \str_if_empty:NTF \l_tmpa_str {
1112          \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_lin
1113        }{
1114          \str_if_eq:cNTF {sref_\l_tmpa_str _type} \c__stex_refs_ref_str {
1115            \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1116              \cs_if_exist:cTF{autoref}{
1117                \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1118              }{
1119                \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1120              }
1121            }{
1122              \ltx@ifpackageloaded{hyperref}{
1123                \hyperref[sref_\l_tmpa_str]\l__stex_refs_linktext_tl
1124              }{
1125                \l__stex_refs_linktext_tl
1126              }
1127            }
1128          }{
1129            \ltx@ifpackageloaded{hyperref}{
1130              \href{\use:c{sref_url_\l_tmpa_str _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_t
1131            }{
1132              \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs
1133            }
1134          }
1135        }
1136      }{
1137        % TODO
1138      }
1139 }
```

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

\srefsym

```
1140 \NewDocumentCommand \srefsym { O{} m}{
1141   \stex_get_symbol:n { #2 }
1142   \__stex_refs_sym_aux:nn{#1}{\l_stex_get_symbol_uri_str}
1143 }
```

```
1144
1145  \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1146    \str_if_exist:cTF {sref_sym_#2 _label_str }{
1147      \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1148    }{
1149      \__stex_refs_args:n { #1 }
1150      \str_if_empty:NTF \l__stex_refs_indocument_str {
1151        \tl_if_exist:cTF{sref_sym_#2 _type}{
1152          % doc uri in \l_tmpb_str
1153          \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1154          \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1155            % reference
1156            \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1157              \cs_if_exist:cTF{autoref}{
1158                \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1159              }{
1160                \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1161              }
1162            }{
1163              \ltx@ifpackageloaded{hyperref}{
1164                \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1165              }{
1166                \l__stex_refs_linktext_tl
1167              }
1168            }
1169          }{
1170            % URL
1171            \ltx@ifpackageloaded{hyperref}{
1172              \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl \
1173            }{
1174              \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_re
1175            }
1176          }
1177        }{
1178          \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_l
1179        }
1180      }{
1181        % TODO
1182      }
1183    }
1184  }
```

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

```
1185  \cs_new_protected:Npn \srefsymuri #1 #2 {
1186    \__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1187  }
```

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

```
1188  ⟨/package⟩
```

Chapter 27

sTeX -Modules Implementation

```
1189 ⟨∗package⟩
1190
1191 %%%%%%%%%%%%%   modules.dtx   %%%%%%%%%%%%%
1192
1193 ⟨@@=stex_modules⟩
```

Warnings and error messages
```
1194 \msg_new:nnn{stex}{error/unknownmodule}{
1195   No~module~#1~found
1196 }
1197 \msg_new:nnn{stex}{error/syntax}{
1198   Syntax~error:~#1
1199 }
1200 \msg_new:nnn{stex}{error/siglanguage}{
1201   Module~#1~declares~signature~#2,~but~does~not~
1202   declare~its~language
1203 }
1204 \msg_new:nnn{stex}{warning/deprecated}{
1205   #1~is~deprecated;~please~use~#2~instead!
1206 }
1207
1208 \msg_new:nnn{stex}{error/conflictingmodules}{
1209   Conflicting~imports~for~module~#1
1210 }
```

`\l_stex_current_module_str` The current module:
```
1211 \str_new:N \l_stex_current_module_str
```

(*End definition for* `\l_stex_current_module_str`. *This variable is documented on page 71.*)

`\l_stex_all_modules_seq` Stores all available modules
```
1212 \seq_new:N \l_stex_all_modules_seq
```

(*End definition for* `\l_stex_all_modules_seq`. *This variable is documented on page 71.*)

```
1213 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1214     \str_if_empty:NTF \l_stex_current_module_str
1215         \prg_return_false: \prg_return_true:
1216 }
```

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

```
1217 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1218     \prop_if_exist:cTF { c_stex_module_#1_prop }
1219         \prg_return_true: \prg_return_false:
1220 }
```

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

Only allowed within modules:

```
1221 \cs_new_protected:Nn \stex_execute_in_module:n { \stex_if_in_module:T {
1222     \stex_add_to_current_module:n { #1 }
1223     \stex_do_up_to_module:n { #1 }
1224 }}
1225 \cs_generate_variant:Nn \stex_execute_in_module:n {x}
1226
1227 \cs_new_protected:Nn \stex_add_to_current_module:n {
1228     \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1229 }
1230 \cs_generate_variant:Nn \stex_add_to_current_module:n {x}
1231 \cs_new_protected:Npn \STEXexport {
1232     \begingroup
1233     \newlinechar=-1\relax
1234     \endlinechar=-1\relax
1235     %\catcode'\ = 9\relax
1236     \expandafter\endgroup\__stex_modules_export:n
1237 }
1238 \cs_new_protected:Nn \__stex_modules_export:n {
1239     \ignorespaces #1
1240     \stex_add_to_current_module:n { \ignorespaces #1 }
1241     \stex_smsmode_do:
1242 }
1243 \stex_deactivate_macro:Nn \STEXexport {module~environments}
```

(*End definition for* \stex_add_to_current_module:n *and* \STEXexport*. These functions are documented on page* *71.*)

```
1244 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1245     \str_set:Nx \l_tmpa_str { #1 }
1246     \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
1247 }
```

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

```
1248 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1249   \str_set:Nx \l_tmpa_str { #1 }
1250   \exp_args:Nno
1251   \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1252     \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1253   }
1254 }
```

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

```
1255 \cs_new_protected:Nn \stex_collect_imports:n {
1256   \seq_clear:N \l_stex_collect_imports_seq
1257   \__stex_modules_collect_imports:n {#1}
1258 }
1259 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1260   \seq_map_inline:cn {c_stex_module_#1_imports} {
1261     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1262       \__stex_modules_collect_imports:n { ##1 }
1263     }
1264   }
1265   \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1266     \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1267   }
1268 }
```

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

```
1269 \int_new:N \l__stex_modules_group_depth_int
1270 \cs_new_protected:Nn \stex_do_up_to_module:n {
1271   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1272     #1
1273   }{
1274     #1
1275     \expandafter \tl_gset:Nn
1276     \csname l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1277     \expandafter\expandafter\expandafter\endcsname
1278     \expandafter\expandafter\expandafter { \csname
1279       l__stex_modules_aftergroup_\l_stex_current_module_str _tl\endcsname #1 }
1280     \aftergroup\__stex_modules_aftergroup_do:
1281   }
1282 }
1283 \cs_generate_variant:Nn \stex_do_up_to_module:n {x}
1284 \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1285   \stex_debug:nn{aftergroup}{\cs_meaning:c{
1286     l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1287   }}
1288   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1289     \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1290     \tl_gclear:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1291   }{
1292     \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
```

```
1293        \aftergroup\__stex_modules_aftergroup_do:
1294    }
1295 }
1296 \cs_new_protected:Nn \_stex_reset_up_to_module:n {
1297    \expandafter\let\csname l__stex_modules_aftergroup_#1_tl\endcsname\undefined
1298 }
```

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

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

```
1299
```

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

Computes the current namespace based on the current MathHub repository (if existent) and the current file.

```
1300 \str_new:N \l_stex_module_ns_str
1301 \str_new:N \l_stex_module_subpath_str
1302 \cs_new_protected:Nn \__stex_modules_compute_namespace:nN {
1303    \seq_set_eq:NN \l_tmpa_seq #2
1304    % split off file extension
1305    \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str % <- filename
1306    \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1307    \seq_get_left:NN \l_tmpb_seq \l_tmpb_str % <- filename without suffixes
1308    \seq_put_right:No \l_tmpa_seq \l_tmpb_str % <- file path including name without suffixes
1309
1310    \bool_set_true:N \l_tmpa_bool
1311    \bool_while_do:Nn \l_tmpa_bool {
1312       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1313       \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1314          {source} { \bool_set_false:N \l_tmpa_bool }
1315       }{}{
1316          \seq_if_empty:NT \l_tmpa_seq {
1317             \bool_set_false:N \l_tmpa_bool
1318          }
1319       }
1320    }
1321
1322    \stex_path_to_string:NN \l_tmpa_seq \l_stex_module_subpath_str
1323    % \l_tmpa_seq <- sub-path relative to archive
1324    \str_if_empty:NTF \l_stex_module_subpath_str {
1325       \str_set:Nx \l_stex_module_ns_str {#1}
1326    }{
1327       \str_set:Nx \l_stex_module_ns_str {
1328          #1/\l_stex_module_subpath_str
1329       }
1330    }
1331 }
1332
1333 \cs_new_protected:Nn \stex_modules_current_namespace: {
1334    \str_clear:N \l_stex_module_subpath_str
1335    \prop_if_exist:NTF \l_stex_current_repository_prop {
1336       \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
```

```
1337        \__stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1338    }{
1339      % split off file extension
1340      \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1341      \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1342      \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1343      \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1344      \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1345      \str_set:Nx \l_stex_module_ns_str {
1346        file:/\stex_path_to_string:N \l_tmpa_seq
1347      }
1348    }
1349 }
```

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

27.1   The `smodule` environment

`smodule` arguments:

```
1350 \keys_define:nn { stex / module } {
1351    title          .tl_set:N     = \smoduletitle ,
1352    type           .str_set_x:N  = \smoduletype ,
1353    id             .str_set_x:N  = \smoduleid ,
1354    deprecate      .str_set_x:N  = \l_stex_module_deprecate_str ,
1355    ns             .str_set_x:N  = \l_stex_module_ns_str ,
1356    lang           .str_set_x:N  = \l_stex_module_lang_str ,
1357    sig            .str_set_x:N  = \l_stex_module_sig_str ,
1358    creators       .str_set_x:N  = \l_stex_module_creators_str ,
1359    contributors   .str_set_x:N  = \l_stex_module_contributors_str ,
1360    meta           .str_set_x:N  = \l_stex_module_meta_str ,
1361    srccite        .str_set_x:N  = \l_stex_module_srccite_str
1362 }
1363
1364 \cs_new_protected:Nn \__stex_modules_args:n {
1365    \str_clear:N \smoduletitle
1366    \str_clear:N \smoduletype
1367    \str_clear:N \smoduleid
1368    \str_clear:N \l_stex_module_ns_str
1369    \str_clear:N \l_stex_module_deprecate_str
1370    \str_clear:N \l_stex_module_lang_str
1371    \str_clear:N \l_stex_module_sig_str
1372    \str_clear:N \l_stex_module_creators_str
1373    \str_clear:N \l_stex_module_contributors_str
1374    \str_clear:N \l_stex_module_meta_str
1375    \str_clear:N \l_stex_module_srccite_str
1376    \keys_set:nn { stex / module } { #1 }
1377 }
1378
1379 % module parameters here? In the body?
1380
```

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

```
1381 \cs_new_protected:Nn \stex_module_setup:nn {
```

125

```
1382    \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1383    \str_set:Nx \l_stex_module_name_str { #2 }
1384    \__stex_modules_args:n { #1 }
```

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

```
1385    \stex_if_in_module:TF {
1386      % Nested module
1387      \prop_get:cnN {c_stex_module_\l_stex_current_module_str _prop}
1388        { ns } \l_stex_module_ns_str
1389      \str_set:Nx \l_stex_module_name_str {
1390        \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
1391          { name } / \l_stex_module_name_str
1392      }
1393      \str_if_empty:NT \l_stex_module_lang_str {
1394        \str_set:Nx \l_stex_module_lang_str {
1395          \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
1396            { lang }
1397        }
1398      }
1399    }{
1400      % not nested:
1401      \str_if_empty:NT \l_stex_module_ns_str {
1402        \stex_modules_current_namespace:
1403        \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1404          / {\l_stex_module_ns_str}
1405        \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1406        \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1407          \str_set:Nx \l_stex_module_ns_str {
1408            \stex_path_to_string:N \l_tmpa_seq
1409          }
1410        }
1411      }
1412    }
```

Next, we determine the language of the module:

```
1413    \str_if_empty:NT \l_stex_module_lang_str {
1414      \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1415      \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1416      \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1417      \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
1418        \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
1419          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
1420        }
1421      }
1422      \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1423      \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
1424        \seq_pop_right:NN \l_tmpa_seq \l_stex_module_lang_str
1425        \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1426          inferred~from~file~name}
1427      }
1428    }
1429
1430    \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
```

126

```
1431        \exp_args:NNo \stex_set_language:Nn \l_tmpa_str \l_stex_module_lang_str
1432      }}
```

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

```
1433      \str_if_empty:NTF \l_stex_module_sig_str {
1434        \exp_args:Nnx \prop_gset_from_keyval:cn {
1435          c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1436        } {
1437          name      = \l_stex_module_name_str ,
1438          ns        = \l_stex_module_ns_str ,
1439          file      = \exp_not:o { \g_stex_currentfile_seq } ,
1440          lang      = \l_stex_module_lang_str ,
1441          sig       = \l_stex_module_sig_str ,
1442          deprecate = \l_stex_module_deprecate_str ,
1443          meta      = \l_stex_module_meta_str
1444        }
1445        \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1446        \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1447        \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _copymodules}
1448        \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1449        \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
```

We load the metatheory:

```
1450        \str_if_empty:NT \l_stex_module_meta_str {
1451          \str_set:Nx \l_stex_module_meta_str {
1452            \c_stex_metatheory_ns_str ? Metatheory
1453          }
1454        }
1455        \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1456          \bool_set_true:N \l_stex_in_meta_bool
1457          \exp_args:Nx \stex_add_to_current_module:n {
1458            \bool_set_true:N \l_stex_in_meta_bool
1459            \stex_activate_module:n {\l_stex_module_meta_str}
1460            \bool_set_false:N \l_stex_in_meta_bool
1461          }
1462          \stex_activate_module:n {\l_stex_module_meta_str}
1463          \bool_set_false:N \l_stex_in_meta_bool
1464        }
1465      }{
1466        \str_if_empty:NT \l_stex_module_lang_str {
1467          \msg_error:nnxx{stex}{error/siglanguage}{
1468            \l_stex_module_ns_str?\l_stex_module_name_str
1469          }{\l_stex_module_sig_str}
1470        }
1471        \stex_debug:nn{modules}{Signature~\l_stex_module_sig_str~for~\l_stex_module_ns_str?\l_st
1472        \stex_if_module_exists:nTF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1473          \stex_debug:nn{modules}{(already exists)}
1474        }{
1475          \stex_debug:nn{modules}{(needs loading)}
1476          \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1477          \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1478          \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1479          \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
```

```
1480        \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1481        \str_set:Nx \l_tmpa_str {
1482          \stex_path_to_string:N \l_tmpa_seq /
1483          \l_tmpa_str . \l_stex_module_sig_str .tex
1484        }
1485        \IfFileExists \l_tmpa_str {
1486          \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1487            \str_clear:N \l_stex_current_module_str
1488            \seq_clear:N \l_stex_all_modules_seq
1489            \stex_debug:nn{modules}{Loading~signature}
1490          }
1491        }{
1492          \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1493        }
1494      }
1495      \stex_if_smsmode:F {
1496        \stex_activate_module:n {
1497          \l_stex_module_ns_str ? \l_stex_module_name_str
1498        }
1499      }
1500      \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1501    }
1502    \str_if_empty:NF \l_stex_module_deprecate_str {
1503      \msg_warning:nnxx{stex}{warning/deprecated}{
1504        Module~\l_stex_current_module_str
1505      }{
1506        \l_stex_module_deprecate_str
1507      }
1508    }
1509    \seq_put_right:Nx \l_stex_all_modules_seq {
1510      \l_stex_module_ns_str ? \l_stex_module_name_str
1511    }
1512    \tl_clear:c{l__stex_modules_aftergroup_\l_stex_module_ns_str ? \l_stex_module_name_str _t
1513 }
```

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

smodule   The module environment.

__stex_modules_begin_module: implements \begin{smodule}

```
1514 \cs_new_protected:Nn \__stex_modules_begin_module: {
1515    \stex_reactivate_macro:N \STEXexport
1516    \stex_reactivate_macro:N \importmodule
1517    \stex_reactivate_macro:N \symdecl
1518    \stex_reactivate_macro:N \notation
1519    \stex_reactivate_macro:N \symdef
1520
1521    \stex_debug:nn{modules}{
1522      New~module:\\
1523      Namespace:~\l_stex_module_ns_str\\
1524      Name:~\l_stex_module_name_str\\
1525      Language:~\l_stex_module_lang_str\\
1526      Signature:~\l_stex_module_sig_str\\
1527      Metatheory:~\l_stex_module_meta_str\\
```

```
1528        File:~\stex_path_to_string:N \g_stex_currentfile_seq
1529    }
1530
1531    \stex_if_do_html:T{
1532      \begin{stex_annotate_env} {theory} {
1533        \l_stex_module_ns_str ? \l_stex_module_name_str
1534      }
1535
1536      \stex_annotate_invisible:nnn{header}{} {
1537        \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1538        \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1539        \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1540          \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1541        }
1542        \str_if_empty:NF \smoduletype {
1543          \stex_annotate:nnn{type}{\smoduletype}{}
1544        }
1545      }
1546    }
1547    % TODO: Inherit metatheory for nested modules?
1548 }
1549 \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}

```
1550 \cs_new_protected:Nn \__stex_modules_end_module: {
1551    \stex_debug:nn{modules}{Closing~module~\prop_item:cn {c_stex_module_\l_stex_current_module
1552    \_stex_reset_up_to_module:n \l_stex_current_module_str
1553    \stex_if_smsmode:T {
1554      \stex_persist:x {
1555        \prop_set_from_keyval:cn{c_stex_module_\l_stex_current_module_str _prop}{
1556          \exp_after:wN \prop_to_keyval:N \csname c_stex_module_\l_stex_current_module_str _pr
1557        }
1558        \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _constants}{
1559          \seq_use:cn{c_stex_module_\l_stex_current_module_str _constants},
1560        }
1561        \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _imports}{
1562          \seq_use:cn{c_stex_module_\l_stex_current_module_str _imports},
1563        }
1564        \tl_set:cn {c_stex_module_\l_stex_current_module_str _code}
1565      }
1566      \exp_after:wN \let \exp_after:wN \l_tmpa_tl \csname c_stex_module_\l_stex_current_module
1567      \exp_after:wN \stex_persist:n \exp_after:wN { \exp_after:wN { \l_tmpa_tl } }
1568    }
1569 }
```

(*End definition for* __stex_modules_end_module:.)

     The core environment

```
1570 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1571 \NewDocumentEnvironment { smodule } { O{} m } {
1572    \stex_module_setup:nn{#1}{#2}
1573    \par
1574    \stex_if_smsmode:F{
```

```
1575        \tl_if_empty:NF \smoduletitle {
1576          \exp_args:No \stex_document_title:n \smoduletitle
1577        }
1578        \tl_clear:N \l_tmpa_tl
1579        \clist_map_inline:Nn \smoduletype {
1580          \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1581            \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1582          }
1583        }
1584        \tl_if_empty:NTF \l_tmpa_tl {
1585          \__stex_modules_smodule_start:
1586        }{
1587          \l_tmpa_tl
1588        }
1589      }
1590      \__stex_modules_begin_module:
1591      \str_if_empty:NF \smoduleid {
1592        \stex_ref_new_doc_target:n \smoduleid
1593      }
1594      \stex_smsmode_do:
1595    } {
1596      \__stex_modules_end_module:
1597      \stex_if_smsmode:F {
1598        \end{stex_annotate_env}
1599        \clist_set:No \l_tmpa_clist \smoduletype
1600        \tl_clear:N \l_tmpa_tl
1601        \clist_map_inline:Nn \l_tmpa_clist {
1602          \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1603            \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1604          }
1605        }
1606        \tl_if_empty:NTF \l_tmpa_tl {
1607          \__stex_modules_smodule_end:
1608        }{
1609          \l_tmpa_tl
1610        }
1611      }
1612    }
```

\stexpatchmodule

```
1613  \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1614  \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1615
1616  \newcommand\stexpatchmodule[3][] {
1617      \str_set:Nx \l_tmpa_str{ #1 }
1618      \str_if_empty:NTF \l_tmpa_str {
1619          \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1620          \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1621      }{
1622          \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1623          \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1624      }
1625  }
```

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

130

27.2   Invoking modules

```
1626 \NewDocumentCommand \STEXModule { m } {
1627   \exp_args:NNx \str_set:Nn \l_tmpa_str { # 1 }
1628   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1629   \tl_set:Nn \l_tmpa_tl {
1630     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1631   }
1632   \seq_map_inline:Nn \l_stex_all_modules_seq {
1633     \str_set:Nn \l_tmpb_str { ##1 }
1634     \str_if_eq:eeT { \l_tmpa_str } {
1635       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1636     } {
1637       \seq_map_break:n {
1638         \tl_set:Nn \l_tmpa_tl {
1639           \stex_invoke_module:n { ##1 }
1640         }
1641       }
1642     }
1643   }
1644   \l_tmpa_tl
1645 }
1646
1647 \cs_new_protected:Nn \stex_invoke_module:n {
1648   \stex_debug:nn{modules}{Invoking~module~#1}
1649   \peek_charcode_remove:NTF ! {
1650     \__stex_modules_invoke_uri:nN { #1 }
1651   } {
1652     \peek_charcode_remove:NTF ? {
1653       \__stex_modules_invoke_symbol:nn { #1 }
1654     } {
1655       \msg_error:nnx{stex}{error/syntax}{
1656         ?~or~!~expected~after~
1657         \c_backslash_str STEXModule{#1}
1658       }
1659     }
1660   }
1661 }
1662
1663 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1664   \str_set:Nn #2 { #1 }
1665 }
1666
1667 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1668   \stex_invoke_symbol:n{#1?#2}
1669 }
```

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

```
1670 \bool_new:N \l_stex_in_meta_bool
1671 \bool_set_false:N \l_stex_in_meta_bool
```

131

```
1672 \cs_new_protected:Nn \stex_activate_module:n {
1673   \stex_debug:nn{modules}{Activating~module~#1}
1674   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1675     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1676     \use:c{ c_stex_module_#1_code }
1677   }
1678 }
```

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

```
1679 ⟨/package⟩
```

Chapter 28

$\mathrm{S}\!T\!E\!X$ -Module Inheritance Implementation

```
1680 ⟨*package⟩
1681
1682 %%%%%%%%%%%%   inheritance.dtx   %%%%%%%%%%%%
1683
```

28.1   SMS Mode

```
1684 ⟨@@=stex_smsmode⟩
```

```
1685 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1686 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1687 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1688
1689 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1690    \makeatletter
1691    \makeatother
1692    \ExplSyntaxOn
1693    \ExplSyntaxOff
1694    \rustexBREAK
1695 }
1696
1697 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1698    \symdef
1699    \importmodule
1700    \notation
1701    \symdecl
1702    \STEXexport
1703    \inlineass
1704    \inlinedef
1705    \inlineex
1706    \endinput
1707    \setnotation
```

```
1708     \copynotation
1709     \assign
1710     \renamedecl
1711     \donotcopy
1712     \instantiate
1713 }
1714
1715 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1716     \tl_to_str:n {
1717         smodule,
1718         copymodule,
1719         interpretmodule,
1720         sdefinition,
1721         sexample,
1722         sassertion,
1723         sparagraph,
1724         mathstructure
1725     }
1726 }
```

(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 74.)*

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

```
1727 \bool_new:N \g__stex_smsmode_bool
1728 \bool_set_false:N \g__stex_smsmode_bool
1729 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1730     \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1731 }
```

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

`\__stex_smsmode_in_smsmode:nn`

```
1732 \cs_new_protected:Nn \__stex_smsmode_in_smsmode:nn { \stex_suppress_html:n {
1733     \vbox_set:Nn \l_tmpa_box {
1734         \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1735         \bool_gset_true:N \g__stex_smsmode_bool
1736         #2
1737         \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1738     }
1739     \box_clear:N \l_tmpa_box
1740 } }
```

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

`\stex_file_in_smsmode:nn`

```
1741 \quark_new:N \q__stex_smsmode_break
1742
1743 \NewDocumentCommand \__stex_smsmode_importmodule: { O{} m } {
1744     \seq_gput_right:Nn \l__stex_smsmode_importmodules_seq {{#1}{#2}}
1745     \stex_smsmode_do:
1746 }
1747
1748 \cs_new_protected:Nn \__stex_smsmode_module:nn {
1749     \__stex_modules_args:n{#1}
```

```
1750    \stex_if_in_module:F {
1751      \str_if_empty:NF \l_stex_module_sig_str {
1752        \stex_modules_current_namespace:
1753        \str_set:Nx \l_stex_module_name_str { #2 }
1754        \stex_if_module_exists:nF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1755          \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1756          \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1757          \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1758          \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1759          \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1760          \str_set:Nx \l_tmpa_str {
1761            \stex_path_to_string:N \l_tmpa_seq /
1762            \l_tmpa_str . \l_stex_module_sig_str .tex
1763          }
1764          \IfFileExists \l_tmpa_str {
1765            \exp_args:NNx \seq_gput_right:Nn \l__stex_smsmode_sigmodules_seq \l_tmpa_str
1766          }{
1767            \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1768          }
1769        }
1770      }
1771    }
1772  }
1773
1774  \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1775    \stex_filestack_push:n{#1}
1776    \seq_gclear:N \l__stex_smsmode_importmodules_seq
1777    \seq_gclear:N \l__stex_smsmode_sigmodules_seq
1778    % ----- new ----------------------------
1779    \__stex_smsmode_in_smsmode:nn{#1}{
1780      \let\importmodule\__stex_smsmode_importmodule:
1781      \let\stex_module_setup:nn\__stex_smsmode_module:nn
1782      \let\__stex_modules_begin_module:\relax
1783      \let\__stex_modules_end_module:\relax
1784      \seq_clear:N \g_stex_smsmode_allowedenvs_seq
1785      \exp_args:NNx \seq_put_right:Nn \g_stex_smsmode_allowedenvs_seq {\tl_to_str:n{smodule}}
1786      \tl_clear:N \g_stex_smsmode_allowedmacros_tl
1787      \tl_clear:N \g_stex_smsmode_allowedmacros_escape_tl
1788      \tl_put_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\importmodule}
1789      \everyeof{\q__stex_smsmode_break\noexpand}
1790      \expandafter\expandafter\expandafter
1791      \stex_smsmode_do:
1792      \csname @ @ input\endcsname "#1"\relax
1793
1794      \seq_map_inline:Nn \l__stex_smsmode_sigmodules_seq {
1795        \stex_filestack_push:n{##1}
1796        \expandafter\expandafter\expandafter
1797        \stex_smsmode_do:
1798        \csname @ @ input\endcsname "##1"\relax
1799        \stex_filestack_pop:
1800      }
1801    }
1802    % ----- new ----------------------------
1803    \__stex_smsmode_in_smsmode:nn{#1} {
```

```
1804        #2
1805        % ----- new ---------------------------
1806        \begingroup
1807        %\stex_debug:nn{smsmode}{Here:~\seq_use:Nn\l__stex_smsmode_importmodules_seq, }
1808        \seq_map_inline:Nn \l__stex_smsmode_importmodules_seq {
1809          \stex_import_module_uri:nn ##1
1810          \stex_import_require_module:nnnn
1811            \l_stex_import_ns_str
1812            \l_stex_import_archive_str
1813            \l_stex_import_path_str
1814            \l_stex_import_name_str
1815        }
1816        \endgroup
1817        \stex_debug:nn{smsmode}{Actually~loading~file~#1}
1818        % ----- new ---------------------------
1819        \everyeof{\q__stex_smsmode_break\noexpand}
1820        \expandafter\expandafter\expandafter
1821        \stex_smsmode_do:
1822        \csname @ @ input\endcsname "#1"\relax
1823      }
1824      \stex_filestack_pop:
1825 }
```

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

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

```
1826 \cs_new_protected:Npn \stex_smsmode_do: {
1827   \stex_if_smsmode:T {
1828     \__stex_smsmode_do:w
1829   }
1830 }
1831 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1832   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 }}{
1833     \expandafter\if\expandafter\relax\noexpand#1
1834       \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1835     \else\expandafter\__stex_smsmode_do:w\fi
1836   }{
1837     \__stex_smsmode_do:w %#1
1838   }
1839 }
1840 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1841   \cs_if_eq:NNF #1 \q__stex_smsmode_break {
1842     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1843       #1\__stex_smsmode_do:w
1844     }{
1845       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1846         #1
1847       }{
1848         \cs_if_eq:NNTF \begin #1 {
1849           \__stex_smsmode_check_begin:n
1850         }{
1851           \cs_if_eq:NNTF \end #1 {
1852             \__stex_smsmode_check_end:n
```

```
1853              }{
1854                \__stex_smsmode_do:w
1855              }
1856            }
1857          }
1858        }
1859      }
1860    }
1861
1862    \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1863      \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1864        \begin{#1}
1865      }{
1866        \__stex_smsmode_do:w
1867      }
1868    }
1869    \cs_new_protected:Nn \__stex_smsmode_check_end:n {
1870      \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1871        \end{#1}\__stex_smsmode_do:w
1872      }{
1873        \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1874      }
1875    }
```

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

28.2 Inheritance

```
1876  ⟨@@=stex_importmodule⟩
```

```
1877  \cs_new_protected:Nn \stex_import_module_uri:nn {
1878    \str_set:Nx \l_stex_import_archive_str { #1 }
1879    \str_set:Nn \l_stex_import_path_str { #2 }
1880
1881    \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1882    \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1883    \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1884
1885    \stex_modules_current_namespace:
1886    \bool_lazy_all:nTF {
1887      {\str_if_empty_p:N \l_stex_import_archive_str}
1888      {\str_if_empty_p:N \l_stex_import_path_str}
1889      {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1890    }{
1891      \str_set_eq:NN \l_stex_import_path_str \l_stex_module_subpath_str
1892      \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1893    }{
1894      \str_if_empty:NT \l_stex_import_archive_str {
1895        \prop_if_exist:NT \l_stex_current_repository_prop {
1896          \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1897        }
1898      }
1899      \str_if_empty:NTF \l_stex_import_archive_str {
```

137

```
1900        \str_if_empty:NF \l_stex_import_path_str {
1901          \str_set:Nx \l_stex_import_ns_str {
1902            \l_stex_module_ns_str / \l_stex_import_path_str
1903          }
1904        }
1905      }{
1906        \stex_require_repository:n \l_stex_import_archive_str
1907        \prop_get:cnN { c_stex_mathhub_\l_stex_import_archive_str _manifest_prop } { ns }
1908          \l_stex_import_ns_str
1909        \str_if_empty:NF \l_stex_import_path_str {
1910          \str_set:Nx \l_stex_import_ns_str {
1911            \l_stex_import_ns_str / \l_stex_import_path_str
1912          }
1913        }
1914      }
1915    }
1916  }
```

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

`\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`.

```
1917 \str_new:N \l_stex_import_name_str
1918 \str_new:N \l_stex_import_archive_str
1919 \str_new:N \l_stex_import_path_str
1920 \str_new:N \l_stex_import_ns_str
```

(End definition for `\l_stex_import_name_str` *and others. These variables are documented on page 76.)*

`\stex_import_require_module:nnnn`

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

```
1921 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1922   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1923
1924     %\stex_debug:nn{requiremodule}{Here:\\~~1:~#1\\~~2:~#2\\~~3:~#3\\~~4:~#4}
1925
1926     \exp_args:NNxx \seq_set_split:Nnn \l_tmpa_seq {\tl_to_str:n{/}} {#4}
1927     \seq_get_left:NN \l_tmpa_seq \l_tmpc_str
1928
1929     %\stex_debug:nn{requiremodule}{Top~module:\l_tmpc_str}
1930
1931     % archive
1932     \str_set:Nx \l_tmpa_str { #2 }
1933     \str_if_empty:NTF \l_tmpa_str {
1934       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1935     } {
1936       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1937       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1938       \seq_put_right:Nn \l_tmpa_seq { source }
1939     }
1940
1941     % path
1942     \str_set:Nx \l_tmpb_str { #3 }
1943     \str_if_empty:NTF \l_tmpb_str {
1944       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / \l_tmpc_str }
1945
```

```
1946          \ltx@ifpackageloaded{babel} {
1947            \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1948                { \languagename } \l_tmpb_str {
1949                  \msg_error:nnx{stex}{error/unknownlanguage}{\languagename}
1950                }
1951          } {
1952            \str_clear:N \l_tmpb_str
1953          }
1954
1955          %\stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1956          \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1957            \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1958          }{
1959            %\stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1960            \IfFileExists{ \l_tmpa_str.tex }{
1961              \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1962            }{
1963              % try english as default
1964              %\stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1965              \IfFileExists{ \l_tmpa_str.en.tex }{
1966                \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1967              }{
1968                \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1969              }
1970            }
1971          }
1972
1973        } {
1974          \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1975          \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1976
1977          \ltx@ifpackageloaded{babel} {
1978            \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1979                { \languagename } \l_tmpb_str {
1980                  \msg_error:nnx{stex}{error/unknownlanguage}{\languagename}
1981                }
1982          } {
1983            \str_clear:N \l_tmpb_str
1984          }
1985
1986          \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1987
1988          %\stex_debug:nn{modules}{Checking~\l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex}
1989          \IfFileExists{ \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }{
1990            \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.te
1991          }{
1992            %\stex_debug:nn{modules}{Checking~\l_tmpa_str/\l_tmpc_str.tex}
1993            \IfFileExists{ \l_tmpa_str/\l_tmpc_str.tex }{
1994              \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.tex }
1995            }{
1996              % try english as default
1997              %\stex_debug:nn{modules}{Checking~\l_tmpa_str/\l_tmpc_str.en.tex}
1998              \IfFileExists{ \l_tmpa_str/\l_tmpc_str.en.tex }{
1999                \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.en.tex }
```

```
2000              }{
2001                %\stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
2002                \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2003                  \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2004                }{
2005                  %\stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
2006                  \IfFileExists{ \l_tmpa_str.tex }{
2007                    \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2008                  }{
2009                    % try english as default
2010                    %\stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
2011                    \IfFileExists{ \l_tmpa_str.en.tex }{
2012                      \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2013                    }{
2014                      \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2015                    }
2016                  }
2017                }
2018              }
2019            }
2020          }
2021        }
2022
2023        \str_if_eq:eeF{\g__stex_importmodule_file_str}{\seq_use:Nn \g_stex_currentfile_seq /}{
2024          \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
2025            \seq_clear:N \l_stex_all_modules_seq
2026            \str_clear:N \l_stex_current_module_str
2027            \str_set:Nx \l_tmpb_str { #2 }
2028            \str_if_empty:NF \l_tmpb_str {
2029              \stex_set_current_repository:n { #2 }
2030            }
2031            \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
2032          }
2033
2034          \stex_if_module_exists:nF { #1 ? #4 } {
2035            \msg_error:nnx{stex}{error/unknownmodule}{
2036              #1?#4~(in~file~\g__stex_importmodule_file_str)
2037            }
2038          }
2039        }
2040
2041      }
2042      \stex_activate_module:n { #1 ? #4 }
2043    }
```

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

\importmodule

```
2044  \NewDocumentCommand \importmodule { O{} m } {
2045    \stex_import_module_uri:nn { #1 } { #2 }
2046    \stex_debug:nn{modules}{Importing~module:~
2047      \l_stex_import_ns_str ? \l_stex_import_name_str
2048    }
2049    \stex_import_require_module:nnnn
```

```
2050     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2051     { \l_stex_import_path_str } { \l_stex_import_name_str }
2052     \stex_if_smsmode:F {
2053       \stex_annotate_invisible:nnn
2054         {import} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
2055     }
2056     \exp_args:Nx \stex_add_to_current_module:n {
2057       \stex_import_require_module:nnnn
2058       { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2059       { \l_stex_import_path_str } { \l_stex_import_name_str }
2060     }
2061     \exp_args:Nx \stex_add_import_to_current_module:n {
2062       \l_stex_import_ns_str ? \l_stex_import_name_str
2063     }
2064     \stex_smsmode_do:
2065     \ignorespacesandpars
2066 }
2067 \stex_deactivate_macro:Nn \importmodule {module~environments}
```

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

\usemodule

```
2068 \NewDocumentCommand \usemodule { O{} m } {
2069   \stex_if_smsmode:F {
2070     \stex_import_module_uri:nn { #1 } { #2 }
2071     \stex_import_require_module:nnnn
2072     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2073     { \l_stex_import_path_str } { \l_stex_import_name_str }
2074     \stex_annotate_invisible:nnn
2075       {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
2076   }
2077   \stex_smsmode_do:
2078   \ignorespacesandpars
2079 }
```

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

```
2080 \cs_new_protected:Nn \stex_csl_to_imports:Nn {
2081   \tl_if_empty:nF{#2}{
2082     \clist_set:Nn \l_tmpa_clist {#2}
2083     \clist_map_inline:Nn \l_tmpa_clist {
2084       \tl_if_head_eq_charcode:nNTF {##1}[{
2085         #1 ##1
2086       }{
2087         #1{##1}
2088       }
2089     }
2090   }
2091 }
2092 \cs_generate_variant:Nn \stex_csl_to_imports:Nn {No}
2093
2094
2095 ⟨/package⟩
```

141

Chapter 29

sTEX
-Symbols Implementation

```
2096 ⟨∗package⟩
2097
2098 %%%%%%%%%%%%    symbols.dtx    %%%%%%%%%%%%
2099
```

Warnings and error messages
```
2100 \msg_new:nnn{stex}{error/wrongargs}{
2101   args~value~in~symbol~declaration~for~#1~
2102   needs~to~be~i,~a,~b~or~B,~but~#2~given
2103 }
2104 \msg_new:nnn{stex}{error/unknownsymbol}{
2105   No~symbol~#1~found!
2106 }
2107 \msg_new:nnn{stex}{error/seqlength}{
2108   Expected~#1~arguments;~got~#2!
2109 }
2110 \msg_new:nnn{stex}{error/unknownnotation}{
2111   Unknown~notation~#1~for~#2!
2112 }
```

29.1   Symbol Declarations

```
2113 ⟨@@=stex_symdecl⟩
```

\stex_all_symbols:n Map over all available symbols
```
2114 \cs_new_protected:Nn \stex_all_symbols:n {
2115   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
2116   \seq_map_inline:Nn \l_stex_all_modules_seq {
2117     \seq_map_inline:cn{c_stex_module_##1_constants}{
2118       \__stex_symdecl_all_symbols_cs{##1?####1}
2119     }
2120   }
2121 }
```

(*End definition for* \stex_all_symbols:n. *This function is documented on page 78.*)

```
2122 \NewDocumentCommand \STEXsymbol { m } {
2123   \stex_get_symbol:n { #1 }
2124   \exp_args:No
2125   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
2126 }
```

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

symdecl arguments:

```
2127 \keys_define:nn { stex / symdecl } {
2128   name        .str_set_x:N  = \l_stex_symdecl_name_str ,
2129   local       .bool_set:N   = \l_stex_symdecl_local_bool ,
2130   args        .str_set_x:N  = \l_stex_symdecl_args_str ,
2131   type        .tl_set:N     = \l_stex_symdecl_type_tl ,
2132   deprecate   .str_set_x:N  = \l_stex_symdecl_deprecate_str ,
2133   align       .str_set:N    = \l_stex_symdecl_align_str , % TODO(?)
2134   gfc         .str_set:N    = \l_stex_symdecl_gfc_str , % TODO(?)
2135   specializes .str_set:N    = \l_stex_symdecl_specializes_str , % TODO(?)
2136   def         .tl_set:N     = \l_stex_symdecl_definiens_tl ,
2137   reorder     .str_set_x:N  = \l_stex_symdecl_reorder_str ,
2138   assoc       .choices:nn   =
2139     {bin,binl,binr,pre,conj,pwconj}
2140     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}
2141 }
2142
2143 \bool_new:N \l_stex_symdecl_make_macro_bool
2144
2145 \cs_new_protected:Nn \__stex_symdecl_args:n {
2146   \str_clear:N \l_stex_symdecl_name_str
2147   \str_clear:N \l_stex_symdecl_args_str
2148   \str_clear:N \l_stex_symdecl_deprecate_str
2149   \str_clear:N \l_stex_symdecl_reorder_str
2150   \str_clear:N \l_stex_symdecl_assoctype_str
2151   \bool_set_false:N \l_stex_symdecl_local_bool
2152   \tl_clear:N \l_stex_symdecl_type_tl
2153   \tl_clear:N \l_stex_symdecl_definiens_tl
2154
2155   \keys_set:nn { stex / symdecl } { #1 }
2156 }
```

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

```
2157
2158 \NewDocumentCommand \symdecl { s m O{} } {
2159   \__stex_symdecl_args:n { #3 }
2160   \IfBooleanTF #1 {
2161     \bool_set_false:N \l_stex_symdecl_make_macro_bool
2162   } {
2163     \bool_set_true:N \l_stex_symdecl_make_macro_bool
2164   }
2165   \stex_symdecl_do:n { #2 }
2166   \stex_smsmode_do:
2167 }
```

143

```
2168
2169  \cs_new_protected:Nn \stex_symdecl_do:nn {
2170    \__stex_symdecl_args:n{#1}
2171    \bool_set_false:N \l_stex_symdecl_make_macro_bool
2172    \stex_symdecl_do:n{#2}
2173  }
2174
2175  \stex_deactivate_macro:Nn \symdecl {module~environments}
```

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

\stex_symdecl_do:n

```
2176  \cs_new_protected:Nn \stex_symdecl_do:n {
2177    \stex_if_in_module:F {
2178      % TODO throw error? some default namespace?
2179    }
2180
2181    \str_if_empty:NT \l_stex_symdecl_name_str {
2182      \str_set:Nx \l_stex_symdecl_name_str { #1 }
2183    }
2184
2185    \prop_if_exist:cT { l_stex_symdecl_
2186        \l_stex_current_module_str ?
2187        \l_stex_symdecl_name_str
2188      _prop
2189    }{
2190      % TODO throw error (beware of circular dependencies)
2191    }
2192
2193    \prop_clear:N \l_tmpa_prop
2194    \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
2195    \seq_clear:N \l_tmpa_seq
2196    \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
2197    \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
2198
2199    \str_if_empty:NT \l_stex_symdecl_deprecate_str {
2200      \str_if_empty:NF \l_stex_module_deprecate_str {
2201        \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
2202      }
2203    }
2204    \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
2205
2206    \exp_args:No \stex_add_constant_to_current_module:n {
2207      \l_stex_symdecl_name_str
2208    }
2209
2210    % arity/args
2211    \int_zero:N \l_tmpb_int
2212
2213    \bool_set_true:N \l_tmpa_bool
2214    \str_map_inline:Nn \l_stex_symdecl_args_str {
2215      \token_case_meaning:NnF ##1 {
2216        0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2217        {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
```

144

```
2218        {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2219        {\tl_to_str:n a} {
2220          \bool_set_false:N \l_tmpa_bool
2221          \int_incr:N \l_tmpb_int
2222        }
2223        {\tl_to_str:n B} {
2224          \bool_set_false:N \l_tmpa_bool
2225          \int_incr:N \l_tmpb_int
2226        }
2227      }{
2228        \msg_error:nnxx{stex}{error/wrongargs}{
2229          \l_stex_current_module_str ?
2230          \l_stex_symdecl_name_str
2231        }{##1}
2232      }
2233    }
2234    \bool_if:NTF \l_tmpa_bool {
2235      % possibly numeric
2236      \str_if_empty:NTF \l_stex_symdecl_args_str {
2237        \prop_put:Nnn \l_tmpa_prop { args } {}
2238        \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2239      }{
2240        \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
2241        \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2242        \str_clear:N \l_tmpa_str
2243        \int_step_inline:nn \l_tmpa_int {
2244          \str_put_right:Nn \l_tmpa_str i
2245        }
2246        \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
2247      }
2248    } {
2249      \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
2250      \prop_put:Nnx \l_tmpa_prop { arity }
2251        { \str_count:N \l_stex_symdecl_args_str }
2252    }
2253    \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2254
2255    \tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
2256      \prop_put:Nnx \l_tmpa_prop { defined }{ false }
2257    }{
2258      \prop_put:Nnx \l_tmpa_prop { defined }{ true }
2259    }
2260
2261    % semantic macro
2262
2263    \bool_if:NT \l_stex_symdecl_make_macro_bool {
2264      \exp_args:Nx \stex_do_up_to_module:n {
2265        \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2266          \l_stex_current_module_str ? \l_stex_symdecl_name_str
2267        }}
2268      }
2269    }
2270
2271    \stex_debug:nn{symbols}{New~symbol:~
```

```
2272    \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2273    Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2274    Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
2275    Definiens:~\exp_not:o {\l_stex_symdecl_definiens_tl}
2276  }
2277
2278  % circular dependencies require this:
2279  \stex_if_do_html:T {
2280    \stex_annotate_invisible:nnn {symdecl} {
2281      \l_stex_current_module_str ? \l_stex_symdecl_name_str
2282    } {
2283      \tl_if_empty:NF \l_stex_symdecl_type_tl {
2284        \stex_annotate_invisible:nnn{type}{}{$\l_stex_symdecl_type_tl$}
2285      }
2286      \stex_annotate_invisible:nnn{args}{}{
2287        \prop_item:Nn \l_tmpa_prop { args }
2288      }
2289      \stex_annotate_invisible:nnn{macroname}{#1}{}
2290      \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2291        \stex_annotate_invisible:nnn{definiens}{}
2292          {$\l_stex_symdecl_definiens_tl$}
2293      }
2294      \str_if_empty:NF \l_stex_symdecl_assoctype_str {
2295        \stex_annotate_invisible:nnn{assoctype}{\l_stex_symdecl_assoctype_str}{}
2296      }
2297      \str_if_empty:NF \l_stex_symdecl_reorder_str {
2298        \stex_annotate_invisible:nnn{reorderargs}{\l_stex_symdecl_reorder_str}{}
2299      }
2300    }
2301  }
2302  \prop_if_exist:cF {
2303    l_stex_symdecl_
2304    \l_stex_current_module_str ? \l_stex_symdecl_name_str
2305    _prop
2306  } {
2307    \bool_if:NTF \l_stex_symdecl_local_bool \stex_do_up_to_module:x \stex_execute_in_module:
2308      \__stex_symdecl_restore_symbol:nnnnnnn
2309        {\l_stex_symdecl_name_str}
2310        { \prop_item:Nn \l_tmpa_prop {args} }
2311        { \prop_item:Nn \l_tmpa_prop {arity} }
2312        { \prop_item:Nn \l_tmpa_prop {assocs} }
2313        { \prop_item:Nn \l_tmpa_prop {defined} }
2314        {\bool_if:NT \l_stex_symdecl_make_macro_bool {#1} }
2315        {\l_stex_current_module_str}
2316    }
2317  }
2318 }
2319 \cs_new_protected:Nn \__stex_symdecl_restore_symbol:nnnnnnn {
2320   \prop_clear:N \l_tmpa_prop
2321   \prop_put:Nnn \l_tmpa_prop { module } { #7 }
2322   \prop_put:Nnn \l_tmpa_prop { name } { #1}
2323   \prop_put:Nnn \l_tmpa_prop { args } {#2}
2324   \prop_put:Nnn \l_tmpa_prop { arity } { #3 }
2325   \prop_put:Nnn \l_tmpa_prop { assocs } { #4 }
```

146

```
2326    \prop_put:Nnn \l_tmpa_prop { defined } { #5 }
2327    \tl_if_empty:nF{#6}{
2328      \tl_set:cx{#6}{\stex_invoke_symbol:n{\detokenize{#7 ? #1}}}
2329    }
2330    \prop_set_eq:cN{l_stex_symdecl_ \detokenize{#7 ? #1} _prop}\l_tmpa_prop
2331    \seq_clear:c{l_stex_symdecl_ \detokenize{#7 ? #1} _notations}
2332  }
```

(*End definition for* \stex_symdecl_do:n. *This function is documented on page* *78.*)

\stex_get_symbol:n

```
2333  \str_new:N \l_stex_get_symbol_uri_str
2334
2335  \cs_new_protected:Nn \stex_get_symbol:n {
2336    \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2337      \tl_set:Nn \l_tmpa_tl { #1 }
2338      \__stex_symdecl_get_symbol_from_cs:
2339    }{
2340      % argument is a string
2341      % is it a command name?
2342      \cs_if_exist:cTF { #1 }{
2343        \cs_set_eq:Nc \l_tmpa_tl { #1 }
2344        \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2345        \str_if_empty:NTF \l_tmpa_str {
2346          \exp_args:Nx \cs_if_eq:NNTF {
2347            \tl_head:N \l_tmpa_tl
2348          } \stex_invoke_symbol:n {
2349            \__stex_symdecl_get_symbol_from_cs:
2350          }{
2351            \__stex_symdecl_get_symbol_from_string:n { #1 }
2352          }
2353        } {
2354          \__stex_symdecl_get_symbol_from_string:n { #1 }
2355        }
2356      }{
2357        % argument is not a command name
2358        \__stex_symdecl_get_symbol_from_string:n { #1 }
2359        % \l_stex_all_symbols_seq
2360      }
2361    }
2362    \str_if_eq:eeF {
2363      \prop_item:cn {
2364        l_stex_symdecl_\l_stex_get_symbol_uri_str _prop
2365      }{ deprecate }
2366    }{}{
2367      \msg_warning:nnxx{stex}{warning/deprecated}{
2368        Symbol~\l_stex_get_symbol_uri_str
2369      }{
2370        \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{ deprecate }
2371      }
2372    }
2373  }
2374
2375  \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
```

```
2376    \tl_set:Nn \l_tmpa_tl {
2377      \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2378    }
2379    \str_set:Nn \l_tmpa_str { #1 }
2380
2381    %\int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2382
2383    \str_if_in:NnTF \l_tmpa_str ? {
2384      \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
2385      \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
2386      \str_set:Nx \l_tmpb_str {\seq_use:Nn \l_tmpa_seq ?}
2387    }{
2388      \str_clear:N \l_tmpb_str
2389    }
2390    \str_if_empty:NTF \l_tmpb_str {
2391      \seq_map_inline:Nn \l_stex_all_modules_seq {
2392        \seq_map_inline:cn{c_stex_module_##1_constants}{
2393          \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2394            \seq_map_break:n{\seq_map_break:n{
2395              \tl_set:Nn \l_tmpa_tl {
2396                \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2397              }
2398            }}
2399          }
2400        }
2401      }
2402    }{
2403      \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpb_str }
2404      \seq_map_inline:Nn \l_stex_all_modules_seq {
2405        \str_if_eq:eeT{ \l_tmpb_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2406          \seq_map_inline:cn{c_stex_module_##1_constants}{
2407            \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2408              \seq_map_break:n{\seq_map_break:n{
2409                \tl_set:Nn \l_tmpa_tl {
2410                  \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2411                }
2412              }}
2413            }
2414          }
2415        }
2416      }
2417    }
2418
2419    \l_tmpa_tl
2420  }
2421
2422  \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
2423    \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2424      { \tl_tail:N \l_tmpa_tl }
2425    \tl_if_single:NTF \l_tmpa_tl {
2426      \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2427        \exp_after:wN \str_set:Nn \exp_after:wN
2428          \l_stex_get_symbol_uri_str \l_tmpa_tl
2429      }{
```

```
2430        % TODO
2431        % tail is not a single group
2432      }
2433   }{
2434      % TODO
2435      % tail is not a single group
2436   }
2437 }
```

(*End definition for* `\stex_get_symbol:n`. *This function is documented on page* *78.*)

29.2   Notations

```
2438 ⟨@@=stex_notation⟩
```

notation arguments:
```
2439 \keys_define:nn { stex / notation } {
2440 %  lang     .tl_set_x:N  = \l__stex_notation_lang_str ,
2441   variant .tl_set_x:N  = \l__stex_notation_variant_str ,
2442   prec     .str_set_x:N = \l__stex_notation_prec_str ,
2443   op       .tl_set:N    = \l__stex_notation_op_tl ,
2444   primary .bool_set:N  = \l__stex_notation_primary_bool ,
2445   primary .default:n   = {true} ,
2446   unknown .code:n       = \str_set:Nx
2447       \l__stex_notation_variant_str \l_keys_key_str
2448 }
2449
2450 \cs_new_protected:Nn \_stex_notation_args:n {
2451 %  \str_clear:N \l__stex_notation_lang_str
2452   \str_clear:N \l__stex_notation_variant_str
2453   \str_clear:N \l__stex_notation_prec_str
2454   \tl_clear:N \l__stex_notation_op_tl
2455   \bool_set_false:N \l__stex_notation_primary_bool
2456
2457   \keys_set:nn { stex / notation } { #1 }
2458 }
```

`\notation`

```
2459 \NewDocumentCommand \notation { s m O{}} {
2460   \_stex_notation_args:n { #3 }
2461   \tl_clear:N \l_stex_symdecl_definiens_tl
2462   \stex_get_symbol:n { #2 }
2463   \tl_set:Nn \l_stex_notation_after_do_tl {
2464     \__stex_notation_final:
2465     \IfBooleanTF#1{
2466       \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2467     }{}
2468     \stex_smsmode_do:\ignorespacesandpars
2469   }
2470   \stex_notation_do:nnnnn
2471     { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } }
2472     { \prop_item:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } }
2473     { \l__stex_notation_variant_str }
2474     { \l__stex_notation_prec_str}
```

149

```
2475    }
2476    \stex_deactivate_macro:Nn \notation {module~environments}
```

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

\stex_notation_do:nnnnn

```
2477    \seq_new:N \l__stex_notation_precedences_seq
2478    \tl_new:N \l__stex_notation_opprec_tl
2479    \int_new:N \l__stex_notation_currarg_int
2480    \tl_new:N \stex_symbol_after_invokation_tl
2481
2482    \cs_new_protected:Nn \stex_notation_do:nnnnn {
2483      \let\l_stex_current_symbol_str\relax
2484      \seq_clear:N \l__stex_notation_precedences_seq
2485      \tl_clear:N \l__stex_notation_opprec_tl
2486      \str_set:Nx \l__stex_notation_args_str { #1 }
2487      \str_set:Nx \l__stex_notation_arity_str { #2 }
2488      \str_set:Nx \l__stex_notation_suffix_str { #3 }
2489      \str_set:Nx \l__stex_notation_prec_str { #4 }
2490
2491      % precedences
2492      \str_if_empty:NTF \l__stex_notation_prec_str {
2493        \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2494          \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2495        }{
2496          \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2497        }
2498      } {
2499        \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2500          \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2501          \int_step_inline:nn { \l__stex_notation_arity_str } {
2502            \exp_args:NNo
2503            \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2504          }
2505        }{
2506          \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2507          \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2508            \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2509            \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2510              \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2511                \l_tmpa_seq {\tl_to_str:n{x} } { \l_tmpa_str }
2512              \seq_map_inline:Nn \l_tmpa_seq {
2513                \seq_put_right:Nn \l_tmpb_seq { ##1 }
2514              }
2515            }
2516          }{
2517            \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2518              \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2519            }{
2520              \tl_set:No \l__stex_notation_opprec_tl { 0 }
2521            }
2522          }
2523        }
2524      }
```

```
2525
2526    \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2527    \int_step_inline:nn { \l__stex_notation_arity_str } {
2528       \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
2529          \exp_args:NNo
2530          \seq_put_right:No \l__stex_notation_precedences_seq {
2531             \l__stex_notation_opprec_tl
2532          }
2533       }
2534    }
2535    \tl_clear:N \l_stex_notation_dummyargs_tl
2536
2537    \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2538       \exp_args:NNe
2539       \cs_set:Npn \l_stex_notation_macrocode_cs {
2540          \_stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2541             { \l__stex_notation_suffix_str }
2542             { \l__stex_notation_opprec_tl }
2543             { \exp_not:n { #5 } }
2544       }
2545       \l_stex_notation_after_do_tl
2546    }{
2547       \str_if_in:NnTF \l__stex_notation_args_str b {
2548          \exp_args:Nne \use:nn
2549          {
2550          \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2551          \cs_set:Npn \l__stex_notation_arity_str } { {
2552             \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2553                { \l__stex_notation_suffix_str }
2554                { \l__stex_notation_opprec_tl }
2555                { \exp_not:n { #5 } }
2556       }}
2557    }{
2558       \str_if_in:NnTF \l__stex_notation_args_str B {
2559          \exp_args:Nne \use:nn
2560          {
2561          \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2562          \cs_set:Npn \l__stex_notation_arity_str } { {
2563             \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2564                { \l__stex_notation_suffix_str }
2565                { \l__stex_notation_opprec_tl }
2566                { \exp_not:n { #5 } }
2567          } }
2568       }{
2569          \exp_args:Nne \use:nn
2570          {
2571          \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2572          \cs_set:Npn \l__stex_notation_arity_str } { {
2573             \_stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2574                { \l__stex_notation_suffix_str }
2575                { \l__stex_notation_opprec_tl }
2576                { \exp_not:n { #5 } }
2577          } }
2578       }
```

```
2579        }
2580
2581        \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2582        \int_zero:N \l__stex_notation_currarg_int
2583        \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2584        \__stex_notation_arguments:
2585    }
2586 }
```

(*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

```
2587 \cs_new_protected:Nn \__stex_notation_arguments: {
2588    \int_incr:N \l__stex_notation_currarg_int
2589    \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2590      \l_stex_notation_after_do_tl
2591    }{
2592      \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2593      \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2594      \str_if_eq:VnTF \l_tmpa_str a {
2595        \__stex_notation_argument_assoc:nn{a}
2596      }{
2597        \str_if_eq:VnTF \l_tmpa_str B {
2598          \__stex_notation_argument_assoc:nn{B}
2599        }{
2600          \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpb_str
2601          \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2602            { \_stex_term_math_arg:nnn
2603              { \l_tmpa_str\int_use:N \l__stex_notation_currarg_int }
2604              { \l_tmpb_str }
2605              { ####\int_use:N \l__stex_notation_currarg_int }
2606          }
2607        }
2608        \__stex_notation_arguments:
2609      }
2610    }
2611  }
2612 }
```

(*End definition for* __stex_notation_arguments:.)

__stex_notation_argument_assoc:nn

```
2613 \cs_new_protected:Nn \__stex_notation_argument_assoc:nn {
2614
2615    \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2616      {\l__stex_notation_arity_str}{
2617      #2
2618    }
2619    \int_zero:N \l_tmpa_int
2620    \tl_clear:N \l_tmpa_tl
2621    \str_map_inline:Nn \l__stex_notation_args_str {
2622      \int_incr:N \l_tmpa_int
2623      \tl_put_right:Nx \l_tmpa_tl {
2624        \str_if_eq:nnTF {##1}{a}{ {} }{
```

152

```
2625        \str_if_eq:nnTF {##1}{B}{ {} }{
2626          {\_stex_term_arg:nn{##1\int_use:N \l_tmpa_int}{############### \int_use:N \l_tmpa
2627        }
2628      }
2629    }
2630  }
2631  \exp_after:wN\exp_after:wN\exp_after:wN \def
2632  \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2633  \exp_after:wN\exp_after:wN\exp_after:wN ##
2634  \exp_after:wN\exp_after:wN\exp_after:wN 1
2635  \exp_after:wN\exp_after:wN\exp_after:wN ##
2636  \exp_after:wN\exp_after:wN\exp_after:wN 2
2637  \exp_after:wN\exp_after:wN\exp_after:wN {
2638    \exp_after:wN \exp_after:wN \exp_after:wN
2639    \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2640      \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2641    }
2642  }
2643
2644  \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2645  \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
2646    \_stex_term_math_assoc_arg:nnnn
2647      { #1\int_use:N \l__stex_notation_currarg_int }
2648      { \l_tmpa_str }
2649      { ####\int_use:N \l__stex_notation_currarg_int }
2650      { \l_tmpa_cs {####1} {####2} }
2651  } }
2652  \__stex_notation_arguments:
2653 }
```

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

`\__stex_notation_final:` Called after processing all notation arguments

```
2654 \cs_new_protected:Nn \__stex_notation_restore_notation:nnnnn {
2655   \cs_generate_from_arg_count:cNnn{stex_notation_\detokenize{#1} \c_hash_str \detokenize{#2}
2656   \cs_set_nopar:Npn {#3}{#4}
2657   \tl_if_empty:nF {#5}{
2658     \tl_set:cn{stex_op_notation_\detokenize{#1} \c_hash_str \detokenize{#2}_cs}{ \comp{ #5 }
2659   }
2660   \seq_if_exist:cT { l_stex_symdecl_\detokenize{#1} _notations }{
2661     \seq_put_right:cx { l_stex_symdecl_\detokenize{#1} _notations } { \detokenize{#2} }
2662   }
2663 }
2664
2665 \cs_new_protected:Nn \__stex_notation_final: {
2666
2667   \stex_execute_in_module:x {
2668     \__stex_notation_restore_notation:nnnnn
2669       {\l_stex_get_symbol_uri_str}
2670       {\l__stex_notation_suffix_str}
2671       {\l__stex_notation_arity_str}
2672       {
2673         \exp_after:wN \exp_after:wN \exp_after:wN
2674         \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
```

153

```
2675          { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2676        }
2677        {\exp_args:No \exp_not:n \l__stex_notation_op_tl }
2678    }
2679
2680    \stex_debug:nn{symbols}{
2681      Notation~\l__stex_notation_suffix_str
2682      ~for~\l_stex_get_symbol_uri_str^^J
2683      Operator~precedence:~\l__stex_notation_opprec_tl^^J
2684      Argument~precedences:~
2685        \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2686      Notation: \cs_meaning:c {
2687        stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2688        \l__stex_notation_suffix_str
2689        _cs
2690      }
2691    }
2692      % HTML annotations
2693    \stex_if_do_html:T {
2694      \stex_annotate_invisible:nnn { notation }
2695      { \l_stex_get_symbol_uri_str } {
2696        \stex_annotate_invisible:nnn { notationfragment }
2697          { \l__stex_notation_suffix_str }{}
2698        \stex_annotate_invisible:nnn { precedence }
2699          { \l__stex_notation_prec_str }{}
2700
2701        \int_zero:N \l_tmpa_int
2702        \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2703        \tl_clear:N \l_tmpa_tl
2704        \int_step_inline:nn { \l__stex_notation_arity_str }{
2705          \int_incr:N \l_tmpa_int
2706          \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2707          \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_rem
2708          \str_if_eq:VnTF \l_tmpb_str a {
2709            \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2710              \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2711              \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2712            } }
2713          }{
2714            \str_if_eq:VnTF \l_tmpb_str B {
2715              \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2716                \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2717                \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2718              } }
2719            }{
2720              \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2721                \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
2722              } }
2723            }
2724          }
2725        }
2726        \stex_annotate_invisible:nnn { notationcomp }{}{
2727          \str_set:Nx \l_stex_current_symbol_str {\l_stex_get_symbol_uri_str }
2728          $ \exp_args:Nno \use:nn { \use:c {
```

```
2729            stex_notation_ \l_stex_current_symbol_str
2730            \c_hash_str \l__stex_notation_suffix_str _cs
2731          } } { \l_tmpa_tl } $
2732        }
2733      }
2734    }
2735 }
```

(*End definition for* __stex_notation_final:.)

\setnotation

```
2736 \keys_define:nn { stex / setnotation } {
2737 %  lang    .tl_set_x:N  = \l__stex_notation_lang_str ,
2738    variant .tl_set_x:N  = \l__stex_notation_variant_str ,
2739    unknown .code:n       = \str_set:Nx
2740        \l__stex_notation_variant_str \l_keys_key_str
2741 }
2742
2743 \cs_new_protected:Nn \_stex_setnotation_args:n {
2744  % \str_clear:N \l__stex_notation_lang_str
2745    \str_clear:N \l__stex_notation_variant_str
2746    \keys_set:nn { stex / setnotation } { #1 }
2747 }
2748
2749 \cs_new_protected:Nn \__stex_notation_setnotation:nn {
2750    \seq_if_exist:cT{l_stex_symdecl_#1_notations}{
2751      \seq_remove_all:cn { l_stex_symdecl_#1 _notations }{ #2 }
2752      \seq_put_left:cn { l_stex_symdecl_#1 _notations }{ #2 }
2753    }
2754 }
2755
2756 \cs_new_protected:Nn \stex_setnotation:n {
2757    \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1 _notations }
2758      { \l__stex_notation_variant_str }{
2759        \stex_execute_in_module:x{ \__stex_notation_setnotation:nn {#1}{\l__stex_notation_vari
2760        \stex_debug:nn {notations}{
2761          Setting~default~notation~
2762          {\l__stex_notation_variant_str }~for~
2763          #1 \\
2764          \expandafter\meaning\csname
2765          l_stex_symdecl_#1 _notations\endcsname
2766        }
2767      }{
2768        \msg_error:nnxx{stex}{unknownnotation}{\l__stex_notation_variant_str}{#1}
2769      }
2770 }
2771
2772 \NewDocumentCommand \setnotation {m m} {
2773    \stex_get_symbol:n { #1 }
2774    \_stex_setnotation_args:n { #2 }
2775    \stex_setnotation:n{\l_stex_get_symbol_uri_str}
2776    \stex_smsmode_do:\ignorespacesandpars
2777 }
2778
```

```
2779  \cs_new_protected:Nn \stex_copy_notations:nn {
2780    \stex_debug:nn {notations}{
2781      Copying~notations~from~#2~to~#1\\
2782      \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
2783    }
2784    \tl_clear:N \l_tmpa_tl
2785    \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } } {
2786      \tl_put_right:Nn \l_tmpa_tl { {######## ##1} }
2787    }
2788    \seq_map_inline:cn {l_stex_symdecl_#2_notations}{
2789      \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2790      \edef \l_tmpa_tl {
2791        \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2792        \exp_after:wN\exp_after:wN\exp_after:wN {
2793          \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2794        }
2795      }
2796
2797      \exp_after:wN \def \exp_after:wN \l_tmpa_tl
2798      \exp_after:wN ####\exp_after:wN 1 \exp_after:wN ####\exp_after:wN 2
2799      \exp_after:wN  { \l_tmpa_tl }
2800
2801      \edef \l_tmpa_tl {
2802        \exp_after:wN \exp_not:n \exp_after:wN {
2803          \l_tmpa_tl {######## 1}{######## 2}
2804        }
2805      }
2806
2807      \stex_execute_in_module:x {
2808        \__stex_notation_restore_notation:nnnnn
2809          {#1}{##1}
2810          { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }
2811          { \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl} }
2812          {
2813            \cs_if_exist:cT{stex_op_notation_ #2\c_hash_str ##1 _cs}{
2814              \exp_args:NNo\exp_args:No\exp_not:n{\csname stex_op_notation_ #2\c_hash_str ##1
2815            }
2816          }
2817      }
2818    }
2819  }
2820
2821  \NewDocumentCommand \copynotation {m m} {
2822    \stex_get_symbol:n { #1 }
2823    \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2824    \stex_get_symbol:n { #2 }
2825    \exp_args:Noo
2826    \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2827    \stex_smsmode_do:\ignorespacesandpars
2828  }
2829
```

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

156

```
2830 \keys_define:nn { stex / symdef } {
2831   name    .str_set_x:N = \l_stex_symdecl_name_str ,
2832   local   .bool_set:N  = \l_stex_symdecl_local_bool ,
2833   args    .str_set_x:N = \l_stex_symdecl_args_str ,
2834   type    .tl_set:N    = \l_stex_symdecl_type_tl ,
2835   def     .tl_set:N    = \l_stex_symdecl_definiens_tl ,
2836   reorder .str_set_x:N = \l_stex_symdecl_reorder_str ,
2837   op      .tl_set:N    = \l__stex_notation_op_tl ,
2838 % lang    .str_set_x:N = \l__stex_notation_lang_str ,
2839   variant .str_set_x:N = \l__stex_notation_variant_str ,
2840   prec    .str_set_x:N = \l__stex_notation_prec_str ,
2841   assoc   .choices:nn  =
2842       {bin,binl,binr,pre,conj,pwconj}
2843       {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
2844   unknown .code:n      = \str_set:Nx
2845       \l__stex_notation_variant_str \l_keys_key_str
2846 }
2847
2848 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2849   \str_clear:N \l_stex_symdecl_name_str
2850   \str_clear:N \l_stex_symdecl_args_str
2851   \str_clear:N \l_stex_symdecl_assoctype_str
2852   \str_clear:N \l_stex_symdecl_reorder_str
2853   \bool_set_false:N \l_stex_symdecl_local_bool
2854   \tl_clear:N \l_stex_symdecl_type_tl
2855   \tl_clear:N \l_stex_symdecl_definiens_tl
2856 % \str_clear:N \l__stex_notation_lang_str
2857   \str_clear:N \l__stex_notation_variant_str
2858   \str_clear:N \l__stex_notation_prec_str
2859   \tl_clear:N \l__stex_notation_op_tl
2860
2861   \keys_set:nn { stex / symdef } { #1 }
2862 }
2863
2864 \NewDocumentCommand \symdef { m O{} } {
2865   \__stex_notation_symdef_args:n { #2 }
2866   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2867   \stex_symdecl_do:n { #1 }
2868   \tl_set:Nn \l_stex_notation_after_do_tl {
2869     \__stex_notation_final:
2870     \stex_smsmode_do:\ignorespacesandpars
2871   }
2872   \str_set:Nx \l_stex_get_symbol_uri_str {
2873     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2874   }
2875   \exp_args:Nx \stex_notation_do:nnnnn
2876     { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } }
2877     { \prop_item:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } }
2878     { \l__stex_notation_variant_str }
2879     { \l__stex_notation_prec_str}
2880 }
2881 \stex_deactivate_macro:Nn \symdef {module~environments}
```

(End definition for `\symdef`*. This function is documented on page* [78](#)*.)*

29.3 Variables

```
2882  ⟨@@=stex_variables⟩
2883
2884  \keys_define:nn { stex / vardef } {
2885    name     .str_set_x:N  = \l__stex_variables_name_str ,
2886    args     .str_set_x:N  = \l__stex_variables_args_str ,
2887    type     .tl_set:N     = \l__stex_variables_type_tl ,
2888    def      .tl_set:N     = \l__stex_variables_def_tl ,
2889    op       .tl_set:N     = \l__stex_variables_op_tl ,
2890    prec     .str_set_x:N  = \l__stex_variables_prec_str ,
2891    assoc    .choices:nn    =
2892        {bin,binl,binr,pre,conj,pwconj}
2893        {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
2894    bind     .choices:nn    =
2895        {forall,exists}
2896        {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2897  }
2898
2899  \cs_new_protected:Nn \__stex_variables_args:n {
2900    \str_clear:N \l__stex_variables_name_str
2901    \str_clear:N \l__stex_variables_args_str
2902    \str_clear:N \l__stex_variables_prec_str
2903    \str_clear:N \l__stex_variables_assoctype_str
2904    \str_clear:N \l__stex_variables_bind_str
2905    \tl_clear:N \l__stex_variables_type_tl
2906    \tl_clear:N \l__stex_variables_def_tl
2907    \tl_clear:N \l__stex_variables_op_tl
2908
2909    \keys_set:nn { stex / vardef } { #1 }
2910  }
2911
2912  \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{}} {
2913    \__stex_variables_args:n {#2}
2914    \str_if_empty:NT \l__stex_variables_name_str {
2915      \str_set:Nx \l__stex_variables_name_str { #1 }
2916    }
2917    \prop_clear:N \l_tmpa_prop
2918    \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
2919
2920    \int_zero:N \l_tmpb_int
2921    \bool_set_true:N \l_tmpa_bool
2922    \str_map_inline:Nn \l__stex_variables_args_str {
2923      \token_case_meaning:NnF ##1 {
2924        0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2925        {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2926        {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2927        {\tl_to_str:n a} {
2928          \bool_set_false:N \l_tmpa_bool
2929          \int_incr:N \l_tmpb_int
2930        }
```

```
2931        {\tl_to_str:n B} {
2932          \bool_set_false:N \l_tmpa_bool
2933          \int_incr:N \l_tmpb_int
2934        }
2935      }{
2936        \msg_error:nnxx{stex}{error/wrongargs}{
2937          variable~\l__stex_variables_name_str
2938        }{##1}
2939      }
2940    }
2941    \bool_if:NTF \l_tmpa_bool {
2942      % possibly numeric
2943      \str_if_empty:NTF \l__stex_variables_args_str {
2944        \prop_put:Nnn \l_tmpa_prop { args } {}
2945        \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2946      }{
2947        \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
2948        \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2949        \str_clear:N \l_tmpa_str
2950        \int_step_inline:nn \l_tmpa_int {
2951          \str_put_right:Nn \l_tmpa_str i
2952        }
2953        \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
2954        \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2955      }
2956    } {
2957      \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2958      \prop_put:Nnx \l_tmpa_prop { arity }
2959        { \str_count:N \l__stex_variables_args_str }
2960    }
2961    \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2962    \tl_set:cx { #1 }{ \stex_invoke_variable:n { \l__stex_variables_name_str } }
2963
2964    \prop_set_eq:cN { l_stex_variable_\l__stex_variables_name_str _prop} \l_tmpa_prop
2965
2966    \tl_if_empty:NF \l__stex_variables_op_tl {
2967      \cs_set:cpx {
2968        stex_var_op_notation_ \l__stex_variables_name_str _cs
2969      } { \exp_not:N\comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } } }
2970    }
2971
2972    \tl_set:Nn \l_stex_notation_after_do_tl {
2973      \exp_args:Nne \use:nn {
2974        \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str _cs }
2975          \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
2976      } {{
2977        \exp_after:wN \exp_after:wN \exp_after:wN
2978        \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2979        { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_symb
2980      }}
2981      \stex_if_do_html:T {
2982        \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
2983          \stex_annotate_invisible:nnn { precedence }
2984            { \l__stex_variables_prec_str }{}
```

159

```
2985          \tl_if_empty:NF \l__stex_variables_type_tl {\stex_annotate_invisible:nnn{type}{}{$\l
2986          \stex_annotate_invisible:nnn{args}{}{ \l__stex_variables_args_str }
2987          \stex_annotate_invisible:nnn{macroname}{#1}{}
2988          \tl_if_empty:NF \l__stex_variables_def_tl {
2989            \stex_annotate_invisible:nnn{definiens}{}
2990              {$\l__stex_variables_def_tl$}
2991          }
2992          \str_if_empty:NF \l__stex_variables_assoctype_str {
2993            \stex_annotate_invisible:nnn{assoctype}{\l__stex_variables_assoctype_str}{}
2994          }
2995          \str_if_empty:NF \l__stex_variables_bind_str {
2996            \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{}
2997          }
2998          \int_zero:N \l_tmpa_int
2999          \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
3000          \tl_clear:N \l_tmpa_tl
3001          \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
3002            \int_incr:N \l_tmpa_int
3003            \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
3004            \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables
3005            \str_if_eq:VnTF \l_tmpb_str a {
3006              \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3007                \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3008                \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3009              } }
3010            }{
3011              \str_if_eq:VnTF \l_tmpb_str B {
3012                \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3013                  \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3014                  \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3015                } }
3016              }{
3017                \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3018                  \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
3019                } }
3020              }
3021            }
3022          }
3023          \stex_annotate_invisible:nnn { notationcomp }{}{
3024            \str_set:Nx \l_stex_current_symbol_str {var://\l__stex_variables_name_str }
3025            $ \exp_args:Nno \use:nn { \use:c {
3026              stex_var_notation_\l__stex_variables_name_str _cs
3027            } } { \l_tmpa_tl } $
3028          }
3029        }
3030      }\ignorespacesandpars
3031    }
3032
3033    \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
3034 }
3035
3036 \cs_new:Nn \_stex_reset:N {
3037    \tl_if_exist:NTF #1 {
3038      \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
```

160

```
3039    }{
3040      \let \exp_not:N #1 \exp_not:N \undefined
3041    }
3042  }
3043
3044  \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
3045    \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
3046    \exp_args:Nnx \use:nn {
3047      % TODO
3048      \stex_annotate_invisible:nnn {vardecl}{\clist_use:Nn\l__stex_variables_names,}{
3049        #2
3050      }
3051    }{
3052      \_stex_reset:N \varnot
3053      \_stex_reset:N \vartype
3054      \_stex_reset:N \vardefi
3055    }
3056  }
3057
3058  \NewDocumentCommand \vardef { s } {
3059    \IfBooleanTF#1 {
3060      \__stex_variables_do_complex:nn
3061    }{
3062      \__stex_variables_do_simple:nnn
3063    }
3064  }
3065
3066  \NewDocumentCommand \svar { O{} m }{
3067    \tl_if_empty:nTF {#1}{
3068      \str_set:Nn \l_tmpa_str { #2 }
3069    }{
3070      \str_set:Nn \l_tmpa_str { #1 }
3071    }
3072    \_stex_term_omv:nn {
3073      var://\l_tmpa_str
3074    }{
3075      \exp_args:Nnx \use:nn {
3076        \def\comp{\_varcomp}
3077        \str_set:Nx \l_stex_current_symbol_str { var://\l_tmpa_str }
3078        \comp{ #2 }
3079      }{
3080        \_stex_reset:N \comp
3081        \_stex_reset:N \l_stex_current_symbol_str
3082      }
3083    }
3084  }
3085
3086
3087
3088  \keys_define:nn { stex / varseq } {
3089    name    .str_set_x:N  = \l__stex_variables_name_str ,
3090    args    .int_set:N    = \l__stex_variables_args_int ,
3091    type    .tl_set:N     = \l__stex_variables_type_tl  ,
3092    mid     .tl_set:N     = \l__stex_variables_mid_tl    ,
```

```
3093   bind     .choices:nn    =
3094        {forall,exists}
3095        {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
3096  }
3097
3098  \cs_new_protected:Nn \__stex_variables_seq_args:n {
3099    \str_clear:N \l__stex_variables_name_str
3100    \int_set:Nn \l__stex_variables_args_int 1
3101    \tl_clear:N \l__stex_variables_type_tl
3102    \str_clear:N \l__stex_variables_bind_str
3103
3104    \keys_set:nn { stex / varseq } { #1 }
3105  }
3106
3107  \NewDocumentCommand \varseq {m O{} m m m}{
3108    \__stex_variables_seq_args:n { #2 }
3109    \str_if_empty:NT \l__stex_variables_name_str {
3110      \str_set:Nx \l__stex_variables_name_str { #1 }
3111    }
3112    \prop_clear:N \l_tmpa_prop
3113    \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
3114
3115    \seq_set_from_clist:Nn \l_tmpa_seq {#3}
3116    \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
3117      \msg_error:nnxx{stex}{error/seqlength}
3118        {\int_use:N \l__stex_variables_args_int}
3119        {\seq_count:N \l_tmpa_seq}
3120    }
3121    \seq_set_from_clist:Nn \l_tmpb_seq {#4}
3122    \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
3123      \msg_error:nnxx{stex}{error/seqlength}
3124        {\int_use:N \l__stex_variables_args_int}
3125        {\seq_count:N \l_tmpb_seq}
3126    }
3127    \prop_put:Nnn \l_tmpa_prop {starts} {#3}
3128    \prop_put:Nnn \l_tmpa_prop {ends} {#4}
3129
3130    \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
3131      \cs_set:Npn  {\int_use:N \l__stex_variables_args_int} { #5 }
3132
3133    \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3134    \int_step_inline:nn \l__stex_variables_args_int {
3135      \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
3136    }
3137    \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
3138    \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
3139    \tl_if_empty:NF \l__stex_variables_mid_tl {
3140      \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
3141      \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
3142    }
3143    \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3144    \int_step_inline:nn \l__stex_variables_args_int {
3145      \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
3146    }
```

```
3147    \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
3148    \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
3149

3150
3151    \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl
3152
3153    \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}
3154
3155    \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3156
3157    \int_step_inline:nn \l__stex_variables_args_int {
3158      \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
3159        \_stex_term_math_arg:nnn{i##1}{0}{\exp_not:n{####}##1}
3160      }}
3161    }
3162
3163    \tl_set:Nx \l_tmpa_tl {
3164      \_stex_term_math_oma:nnnn { varseq://\l__stex_variables_name_str}{}{0}{
3165        \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
3166      }
3167    }
3168
3169    \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \stex_symbol_after_invokation_tl} }
3170
3171    \exp_args:Nno \use:nn {
3172    \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
3173      \cs_set:Npn {\int_use:N \l__stex_variables_args_int}}{\l_tmpa_tl}
3174
3175    \stex_debug:nn{sequences}{New~Sequence:~
3176      \expandafter\meaning\csname stex_varseq_\l__stex_variables_name_str _cs\endcsname\\~\\
3177      \prop_to_keyval:N \l_tmpa_prop
3178    }
3179    \stex_if_do_html:T{\stex_annotate_invisible:nnn{varseq}{\l__stex_variables_name_str}{
3180      \tl_if_empty:NF \l__stex_variables_type_tl {
3181        \stex_annotate:nnn {type}{}{$\seqtype\l__stex_variables_type_tl$}
3182      }
3183      \stex_annotate:nnn {args}{\int_use:N \l__stex_variables_args_int}{}
3184      \str_if_empty:NF \l__stex_variables_bind_str {
3185        \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{}
3186      }
3187    }}
3188
3189    \prop_set_eq:cN {stex_varseq_\l__stex_variables_name_str _prop}\l_tmpa_prop
3190    \ignorespacesandpars
3191 }
3192
3193 ⟨/package⟩
```

163

Chapter 30

STEX
-Terms Implementation

3194 ⟨∗package⟩

3195

3196 %%%%%%%%%%%%   terms.dtx   %%%%%%%%%%%%

3197

3198 ⟨@@=stex_terms⟩

Warnings and error messages

```
3199  \msg_new:nnn{stex}{error/nonotation}{
3200    Symbol~#1~invoked,~but~has~no~notation#2!
3201  }
3202  \msg_new:nnn{stex}{error/notationarg}{
3203    Error~in~parsing~notation~#1
3204  }
3205  \msg_new:nnn{stex}{error/noop}{
3206    Symbol~#1~has~no~operator~notation~for~notation~#2
3207  }
3208  \msg_new:nnn{stex}{error/notallowed}{
3209    Symbol~invokation~#1~not~allowed~in~notation~component~of~#2
3210  }
3211  \msg_new:nnn{stex}{error/doubleargument}{
3212    Argument~#1~of~symbol~#2~already~assigned
3213  }
3214  \msg_new:nnn{stex}{error/overarity}{
3215    Argument~#1~invalid~for~symbol~#2~with~arity~#3
3216  }
3217
```

30.1   Symbol Invocations

\stex_invoke_symbol:n Invokes a semantic macro

```
3218
3219
3220  \bool_new:N \l_stex_allow_semantic_bool
3221  \bool_set_true:N \l_stex_allow_semantic_bool
3222
```

```
3223 \cs_new_protected:Nn \stex_invoke_symbol:n {
3224   \bool_if:NTF \l_stex_allow_semantic_bool {
3225     \str_if_eq:eeF {
3226       \prop_item:cn {
3227         l_stex_symdecl_#1_prop
3228       }{ deprecate }
3229     }{}{
3230       \msg_warning:nnxx{stex}{warning/deprecated}{
3231         Symbol~#1
3232       }{
3233         \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
3234       }
3235     }
3236     \if_mode_math:
3237       \exp_after:wN \__stex_terms_invoke_math:n
3238     \else:
3239       \exp_after:wN \__stex_terms_invoke_text:n
3240     \fi: { #1 }
3241   }{
3242     \msg_error:nnxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
3243   }
3244 }
3245
3246 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
3247   \peek_charcode_remove:NTF ! {
3248     \__stex_terms_invoke_op_custom:nn {#1}
3249   }{
3250     \__stex_terms_invoke_custom:nn {#1}
3251   }
3252 }
3253
3254 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
3255   \peek_charcode_remove:NTF ! {
3256     % operator
3257     \peek_charcode_remove:NTF * {
3258       % custom op
3259       \__stex_terms_invoke_op_custom:nn {#1}
3260     }{
3261       % op notation
3262       \peek_charcode:NTF [ {
3263         \__stex_terms_invoke_op_notation:nw {#1}
3264       }{
3265         \__stex_terms_invoke_op_notation:nw {#1}[]
3266       }
3267     }
3268   }{
3269     \peek_charcode_remove:NTF * {
3270       \__stex_terms_invoke_custom:nn {#1}
3271       % custom
3272     }{
3273       % normal
3274       \peek_charcode:NTF [ {
3275         \__stex_terms_invoke_notation:nw {#1}
3276       }{
```

```
3277          \__stex_terms_invoke_notation:nw {#1}[]
3278        }
3279      }
3280    }
3281 }
3282
3283
3284 \cs_new_protected:Nn \__stex_terms_invoke_op_custom:nn {
3285    \exp_args:Nnx \use:nn {
3286      \def\comp{\_comp}
3287      \str_set:Nn \l_stex_current_symbol_str { #1 }
3288      \bool_set_false:N \l_stex_allow_semantic_bool
3289      \_stex_term_oms:nnn {#1}{#1 \c_hash_str CUSTOM-}{
3290        \comp{ #2 }
3291      }
3292    }{
3293      \_stex_reset:N \comp
3294      \_stex_reset:N \l_stex_current_symbol_str
3295      \bool_set_true:N \l_stex_allow_semantic_bool
3296    }
3297 }
3298
3299 \keys_define:nn { stex / terms } {
3300 %  lang     .tl_set_x:N = \l_stex_notation_lang_str ,
3301    variant .tl_set_x:N = \l_stex_notation_variant_str ,
3302    unknown .code:n     = \str_set:Nx
3303        \l_stex_notation_variant_str \l_keys_key_str
3304 }
3305
3306 \cs_new_protected:Nn \__stex_terms_args:n {
3307  % \str_clear:N \l_stex_notation_lang_str
3308    \str_clear:N \l_stex_notation_variant_str
3309
3310    \keys_set:nn { stex / terms } { #1 }
3311 }
3312
3313 \cs_new_protected:Nn \stex_find_notation:nn {
3314    \__stex_terms_args:n { #2 }
3315    \seq_if_empty:cTF {
3316      l_stex_symdecl_ #1 _notations
3317    } {
3318      \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
3319    } {
3320      \str_if_empty:NTF \l_stex_notation_variant_str {
3321        \seq_get_left:cN {l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
3322      }{
3323        \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
3324          \l_stex_notation_variant_str
3325        }{
3326        % \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3327        }{
3328          \msg_error:nnxx{stex}{error/nonotation}{#1}{
3329            ~\l_stex_notation_variant_str
3330          }
```

```
3331            }
3332          }
3333        }
3334      }
3335
3336  \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3337    \exp_args:Nnx \use:nn {
3338      \def\comp{\_comp}
3339      \str_set:Nn \l_stex_current_symbol_str { #1 }
3340      \stex_find_notation:nn { #1 }{ #2 }
3341      \bool_set_false:N \l_stex_allow_semantic_bool
3342      \cs_if_exist:cTF {
3343        stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3344      }{
3345        \_stex_term_oms:nnn { #1 }{
3346          #1 \c_hash_str \l_stex_notation_variant_str
3347        }{
3348          \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3349        }
3350      }{
3351        \int_compare:nNnTF {\prop_item:cn {l_stex_symdecl_#1_prop}{arity}} = 0{
3352          \cs_if_exist:cTF {
3353            stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3354          }{
3355            \tl_set:Nx \stex_symbol_after_invokation_tl {
3356              \_stex_reset:N \comp
3357              \_stex_reset:N \stex_symbol_after_invokation_tl
3358              \_stex_reset:N \l_stex_current_symbol_str
3359              \bool_set_true:N \l_stex_allow_semantic_bool
3360            }
3361            \def\comp{\_comp}
3362            \str_set:Nn \l_stex_current_symbol_str { #1 }
3363            \bool_set_false:N \l_stex_allow_semantic_bool
3364            \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3365          }{
3366            \msg_error:nnxx{stex}{error/nonotation}{#1}{
3367              ~\l_stex_notation_variant_str
3368            }
3369          }
3370        }{
3371          \msg_error:nnxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3372        }
3373      }
3374    }{
3375      \_stex_reset:N \comp
3376      \_stex_reset:N \l_stex_current_symbol_str
3377      \bool_set_true:N \l_stex_allow_semantic_bool
3378    }
3379  }
3380
3381  \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {
3382    \stex_find_notation:nn { #1 }{ #2 }
3383    \cs_if_exist:cTF {
3384      stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
```

```
3385    }{
3386      \tl_set:Nx \stex_symbol_after_invokation_tl {
3387        \_stex_reset:N \comp
3388        \_stex_reset:N \stex_symbol_after_invokation_tl
3389        \_stex_reset:N \l_stex_current_symbol_str
3390        \bool_set_true:N \l_stex_allow_semantic_bool
3391      }
3392      \def\comp{\_comp}
3393      \str_set:Nn \l_stex_current_symbol_str { #1 }
3394      \bool_set_false:N \l_stex_allow_semantic_bool
3395      \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3396    }{
3397      \msg_error:nnxx{stex}{error/nonotation}{#1}{
3398        ~\l_stex_notation_variant_str
3399      }
3400    }
3401 }
3402
3403 \prop_new:N \l__stex_terms_custom_args_prop
3404
3405 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3406    \exp_args:Nnx \use:nn {
3407      \bool_set_false:N \l_stex_allow_semantic_bool
3408      \def\comp{\_comp}
3409      \str_set:Nn \l_stex_current_symbol_str { #1 }
3410      \prop_clear:N \l__stex_terms_custom_args_prop
3411      \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3412      \prop_get:cnN {
3413        l_stex_symdecl_#1 _prop
3414      }{ args } \l_tmpa_str
3415      \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3416      \tl_set:Nn \arg { \__stex_terms_arg: }
3417      \str_if_empty:NTF \l_tmpa_str {
3418        \_stex_term_oms:nnn {#1}{#1\c_hash_str CUSTOM-}{#2}
3419      }{
3420        \str_if_in:NnTF \l_tmpa_str b {
3421          \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3422        }{
3423          \str_if_in:NnTF \l_tmpa_str B {
3424            \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3425          }{
3426            \_stex_term_oma:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3427          }
3428        }
3429      }
3430      % TODO check that all arguments exist
3431    }{
3432      \_stex_reset:N \l_stex_current_symbol_str
3433      \_stex_reset:N \arg
3434      \_stex_reset:N \comp
3435      \_stex_reset:N \l__stex_terms_custom_args_prop
3436      \bool_set_true:N \l_stex_allow_semantic_bool
3437    }
3438 }
```

```
3439
3440 \NewDocumentCommand \__stex_terms_arg: { s O{} m}{
3441   \tl_if_empty:nTF {#2}{
3442     \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
3443     \bool_set_true:N \l_tmpa_bool
3444     \bool_do_while:Nn \l_tmpa_bool {
3445       \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int
3446         \int_incr:N \l_tmpa_int
3447       }{
3448         \bool_set_false:N \l_tmpa_bool
3449       }
3450     }
3451   }{
3452     \int_set:Nn \l_tmpa_int { #2 }
3453   }
3454   \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
3455   \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
3456     \msg_error:nnxxx{stex}{error/overarity}
3457       {\int_use:N \l_tmpa_int}
3458       {\l_stex_current_symbol_str}
3459       {\str_count:N \l_tmpa_str}
3460   }
3461   \str_set:Nx \l_tmpa_str {\str_item:Nn \l_tmpa_str \l_tmpa_int}
3462   \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
3463     \bool_lazy_any:nF {
3464       {\str_if_eq_p:Vn \l_tmpa_str {a}}
3465       {\str_if_eq_p:Vn \l_tmpa_str {B}}
3466     }{
3467       \msg_error:nnxx{stex}{error/doubleargument}
3468         {\int_use:N \l_tmpa_int}
3469         {\l_stex_current_symbol_str}
3470     }
3471   }
3472   \exp_args:NNx \prop_put:Nnn \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {#3}
3473   \bool_set_true:N \l_stex_allow_semantic_bool
3474   \IfBooleanTF#1{
3475     \stex_annotate_invisible:n { %TODO
3476       \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{#3}
3477     }
3478   }{ %TODO
3479     \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{#3}
3480   }
3481   \bool_set_false:N \l_stex_allow_semantic_bool
3482 }
3483
3484
3485 \cs_new_protected:Nn \_stex_term_arg:nn {
3486   \bool_set_true:N \l_stex_allow_semantic_bool
3487   \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3488   \bool_set_false:N \l_stex_allow_semantic_bool
3489 }
3490
3491 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
3492   \exp_args:Nnx \use:nn
```

169

```
3493        { \int_set:Nn \l__stex_terms_downprec { #2 }
3494            \_stex_term_arg:nn { #1 }{ #3 }
3495        }
3496        { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3497 }
```

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

```
3498 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
3499   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3500   \tl_set:Nn \l_tmpb_tl {\_stex_term_math_arg:nnn{#1}{#2}}
3501   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 }}{
3502      \expandafter\if\expandafter\relax\noexpand#3
3503        \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_maybe_sequence:Nn#3{#1}}
3504      \else
3505        \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_simple:nn{#1}{#3}}
3506      \fi
3507      \l_tmpa_tl
3508   }{
3509      \__stex_terms_math_assoc_arg_simple:nn{#1}{#3}
3510   }
3511 }
3512
3513 \cs_new_protected:Nn \__stex_terms_math_assoc_arg_maybe_sequence:Nn {
3514   \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
3515   \str_if_empty:NTF \l_tmpa_str {
3516      \exp_args:Nx \cs_if_eq:NNTF {
3517        \tl_head:N #1
3518      } \stex_invoke_sequence:n {
3519        \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
3520        \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
3521        \tl_set:Nx \l_tmpa_tl {\prop_item:cn {stex_varseq_\l_tmpa_str _prop}{notation}}
3522        \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
3523        \tl_set:Nx \l_tmpa_tl {{\exp_not:N \exp_not:n{
3524          \exp_not:n{\exp_args:Nnx \use:nn} {
3525            \exp_not:n {
3526              \def\comp{\_varcomp}
3527              \str_set:Nn \l_stex_current_symbol_str
3528            } {varseq://\l_tmpa_str}
3529            \exp_not:n{ ##1 }
3530          }{
3531            \exp_not:n {
3532              \_stex_reset:N \comp
3533              \_stex_reset:N \l_stex_current_symbol_str
3534            }
3535          }
3536        }}}
3537        \exp_args:Nno \use:nn {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
3538        \seq_reverse:N \l_tmpa_seq
3539        \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
3540        \seq_map_inline:Nn \l_tmpa_seq {
3541          \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3542            \exp_args:Nno
```

170

```
3543          \l_tmpa_cs { ##1 } \l_tmpa_tl
3544        }
3545      }
3546      \tl_set:Nx \l_tmpa_tl {
3547        \_stex_term_omv:nn {varseq://\l_tmpa_str}{
3548          \exp_args:No \exp_not:n \l_tmpa_tl
3549        }
3550      }
3551      \exp_args:No\l_tmpb_tl\l_tmpa_tl
3552    }{
3553      \__stex_terms_math_assoc_arg_simple:nn{#2} { #1 }
3554    }
3555  } {
3556    \__stex_terms_math_assoc_arg_simple:nn{#2} { #1 }
3557  }
3558
3559 }
3560
3561 \cs_new_protected:Nn \__stex_terms_math_assoc_arg_simple:nn {
3562   \clist_set:Nn \l_tmpa_clist{ #2 }
3563   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3564     \tl_set:Nn \l_tmpa_tl { \_stex_term_arg:nn{A#1}{ #2 } }
3565   }{
3566     \clist_reverse:N \l_tmpa_clist
3567     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3568     \tl_set:Nx \l_tmpa_tl { \_stex_term_arg:nn{A#1}{
3569       \exp_args:No \exp_not:n \l_tmpa_tl
3570     }}
3571     \clist_map_inline:Nn \l_tmpa_clist {
3572       \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3573         \exp_args:Nno
3574         \l_tmpa_cs { \_stex_term_arg:nn{A#1}{##1} } \l_tmpa_tl
3575       }
3576     }
3577   }
3578   \exp_args:No\l_tmpb_tl\l_tmpa_tl
3579 }
```

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

30.2 Terms

Precedences:

```
3580 \tl_const:Nx \infprec {\int_use:N \c_max_int}
3581 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
3582 \int_new:N \l__stex_terms_downprec
3583 \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* *80.*)

Bracketing:

171

```
3584 \tl_set:Nn \l__stex_terms_left_bracket_str (
3585 \tl_set:Nn \l__stex_terms_right_bracket_str )
```

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

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

```
3586 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
3587   \bool_if:NTF \l__stex_terms_brackets_done_bool {
3588     \bool_set_false:N \l__stex_terms_brackets_done_bool
3589     #2
3590   } {
3591     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
3592       \bool_if:NTF \l_stex_inparray_bool { #2 }{
3593         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
3594         \dobrackets { #2 }
3595       }
3596     }{ #2 }
3597   }
3598 }
```

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

`\dobrackets`

```
3599 \bool_new:N \l__stex_terms_brackets_done_bool
3600 %\RequirePackage{scalerel}
3601 \cs_new_protected:Npn \dobrackets #1 {
3602   %\ThisStyle{\if D\m@switch
3603   %   \exp_args:Nnx \use:nn
3604   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
3605   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
3606   %  \else
3607      \exp_args:Nnx \use:nn
3608      {
3609        \bool_set_true:N \l__stex_terms_brackets_done_bool
3610        \int_set:Nn \l__stex_terms_downprec \infprec
3611        \l__stex_terms_left_bracket_str
3612        #1
3613      }
3614      {
3615        \bool_set_false:N \l__stex_terms_brackets_done_bool
3616        \l__stex_terms_right_bracket_str
3617        \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3618      }
3619   %\fi}
3620 }
```

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

`\withbrackets`

```
3621 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
3622   \exp_args:Nnx \use:nn
3623   {
3624     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
```

```
3625     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
3626       #3
3627   }
3628   {
3629     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
3630       {\l__stex_terms_left_bracket_str}
3631     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
3632       {\l__stex_terms_right_bracket_str}
3633   }
3634 }
```

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

\STEXinvisible

```
3635 \cs_new_protected:Npn \STEXinvisible #1 {
3636   \stex_annotate_invisible:n { #1 }
3637 }
```

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

OMDoc terms:

_stex_term_math_oms:nnnn

```
3638 \cs_new_protected:Nn \_stex_term_oms:nnn {
3639   \stex_annotate:nnn{ OMID }{ #2 }{
3640     #3
3641   }
3642 }
3643
3644 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
3645   \__stex_terms_maybe_brackets:nn { #3 }{
3646     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3647   }
3648 }
```

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

_stex_term_math_omv:nn

```
3649 \cs_new_protected:Nn \_stex_term_omv:nn {
3650   \stex_annotate:nnn{ OMV }{ #1 }{
3651     #2
3652   }
3653 }
```

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

_stex_term_math_oma:nnnn

```
3654 \cs_new_protected:Nn \_stex_term_oma:nnn {
3655   \stex_annotate:nnn{ OMA }{ #2 }{
3656     #3
3657   }
3658 }
3659
3660 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
3661   \__stex_terms_maybe_brackets:nn { #3 }{
3662     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
```

173

```
3663        }
3664    }
```

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

```
3665    \cs_new_protected:Nn \_stex_term_ombind:nnn {
3666      \stex_annotate:nnn{ OMBIND }{ #2 }{
3667        #3
3668      }
3669    }
3670
3671    \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
3672      \__stex_terms_maybe_brackets:nn { #3 }{
3673        \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3674      }
3675    }
```

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

```
3676    \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3677
3678    \keys_define:nn { stex / symname } {
3679      pre      .tl_set_x:N    = \l__stex_terms_pre_tl ,
3680      post     .tl_set_x:N    = \l__stex_terms_post_tl ,
3681      root     .tl_set_x:N    = \l__stex_terms_root_tl
3682    }
3683
3684    \cs_new_protected:Nn \stex_symname_args:n {
3685      \tl_clear:N \l__stex_terms_post_tl
3686      \tl_clear:N \l__stex_terms_pre_tl
3687      \tl_clear:N \l__stex_terms_root_str
3688      \keys_set:nn { stex / symname } { #1 }
3689    }
3690
3691    \NewDocumentCommand \symref { m m }{
3692      \let\compemph_uri_prev:\compemph@uri
3693      \let\compemph@uri\symrefemph@uri
3694      \STEXsymbol{#1}!{ #2 }
3695      \let\compemph@uri\compemph_uri_prev:
3696    }
3697
3698    \NewDocumentCommand \synonym { O{} m m}{
3699      \stex_symname_args:n { #1 }
3700      \let\compemph_uri_prev:\compemph@uri
3701      \let\compemph@uri\symrefemph@uri
3702      % TODO
3703      \STEXsymbol{#2}!{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
3704      \let\compemph@uri\compemph_uri_prev:
3705    }
3706
3707    \NewDocumentCommand \symname { O{} m }{
3708      \stex_symname_args:n { #1 }
3709      \stex_get_symbol:n { #2 }
```

```
3710    \str_set:Nx \l_tmpa_str {
3711      \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3712    }
3713    \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3714
3715    \let\compemph_uri_prev:\compemph@uri
3716    \let\compemph@uri\symrefemph@uri
3717    \exp_args:NNx \use:nn
3718    \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmmode*\fi{
3719      \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
3720    } }
3721    \let\compemph@uri\compemph_uri_prev:
3722  }
3723
3724  \NewDocumentCommand \Symname { O{} m }{
3725    \stex_symname_args:n { #1 }
3726    \stex_get_symbol:n { #2 }
3727    \str_set:Nx \l_tmpa_str {
3728      \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3729    }
3730    \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3731    \let\compemph_uri_prev:\compemph@uri
3732    \let\compemph@uri\symrefemph@uri
3733    \exp_args:NNx \use:nn
3734    \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmmode*\fi{
3735      \exp_after:wN \stex_capitalize:n \l_tmpa_str
3736        \l__stex_terms_post_tl
3737    } }
3738    \let\compemph@uri\compemph_uri_prev:
3739  }
```

(*End definition for* `\symref` *and* `\symname`*. These functions are documented on page* *79.*)

30.3   Notation Components

```
3740  ⟨@@=stex_notationcomps⟩
```

```
3741  \cs_new_protected:Npn \_comp #1 {
3742    \str_if_empty:NF \l_stex_current_symbol_str {
3743      \stex_html_backend:TF {
3744        \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
3745      }{
3746        \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3747      }
3748    }
3749  }
3750
3751  \cs_new_protected:Npn \_varcomp #1 {
3752    \str_if_empty:NF \l_stex_current_symbol_str {
3753      \stex_html_backend:TF {
3754        \stex_annotate:nnn { varcomp }{ \l_stex_current_symbol_str }{ #1 }
3755      }{
3756        \exp_args:Nnx \varemph@uri { #1 } { \l_stex_current_symbol_str }
```

```
3757              }
3758          }
3759      }
3760
3761  \def\comp{\_comp}
3762
3763  \cs_new_protected:Npn \compemph@uri #1 #2 {
3764      \compemph{ #1 }
3765  }
3766
3767
3768  \cs_new_protected:Npn \compemph #1 {
3769      #1
3770  }
3771
3772  \cs_new_protected:Npn \defemph@uri #1 #2 {
3773      \defemph{#1}
3774  }
3775
3776  \cs_new_protected:Npn \defemph #1 {
3777      \textbf{#1}
3778  }
3779
3780  \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3781      \symrefemph{#1}
3782  }
3783
3784  \cs_new_protected:Npn \symrefemph #1 {
3785      \emph{#1}
3786  }
3787
3788  \cs_new_protected:Npn \varemph@uri #1 #2 {
3789      \varemph{#1}
3790  }
3791
3792  \cs_new_protected:Npn \varemph #1 {
3793      #1
3794  }
```

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

\ellipses

```
3795  \NewDocumentCommand \ellipses {} { \ldots }
```

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

\parray
\prmatrix
\parrayline
\parraylineh
\parraycell

```
3796  \bool_new:N \l_stex_inparray_bool
3797  \bool_set_false:N \l_stex_inparray_bool
3798  \NewDocumentCommand \parray { m m } {
3799      \begingroup
3800      \bool_set_true:N \l_stex_inparray_bool
3801      \begin{array}{#1}
3802          #2
3803      \end{array}
```

```
3804        \endgroup
3805    }
3806
3807    \NewDocumentCommand \prmatrix { m } {
3808        \begingroup
3809        \bool_set_true:N \l_stex_inparray_bool
3810        \begin{matrix}
3811            #1
3812        \end{matrix}
3813        \endgroup
3814    }
3815
3816    \def \maybephline {
3817        \bool_if:NT \l_stex_inparray_bool {\hline}
3818    }
3819
3820    \def \parrayline #1 #2 {
3821        #1 #2 \bool_if:NT \l_stex_inparray_bool {\\}
3822    }
3823
3824    \def \pmrow #1 { \parrayline{}{ #1 } }
3825
3826    \def \parraylineh #1 #2 {
3827        #1 #2 \bool_if:NT \l_stex_inparray_bool {\\\hline}
3828    }
3829
3830    \def \parraycell #1 {
3831        #1 \bool_if:NT \l_stex_inparray_bool {&}
3832    }
```

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

30.4   Variables

```
3833    ⟨@@=stex_variables⟩
```

\stex_invoke_variable:n   Invokes a variable

```
3834    \cs_new_protected:Nn \stex_invoke_variable:n {
3835        \if_mode_math:
3836            \exp_after:wN \__stex_variables_invoke_math:n
3837        \else:
3838            \exp_after:wN \__stex_variables_invoke_text:n
3839        \fi: {#1}
3840    }
3841
3842    \cs_new_protected:Nn \__stex_variables_invoke_text:n {
3843        %TODO
3844    }
3845
3846
3847    \cs_new_protected:Nn \__stex_variables_invoke_math:n {
3848        \peek_charcode_remove:NTF ! {
3849            \peek_charcode_remove:NTF ! {
3850                \peek_charcode:NTF [ {
```

```
3851            \__stex_variables_invoke_op_custom:nw
3852        }{
3853          % TODO throw error
3854        }
3855      }{
3856        \__stex_variables_invoke_op:n { #1 }
3857      }
3858    }{
3859      \peek_charcode_remove:NTF * {
3860        \__stex_variables_invoke_text:n { #1 }
3861      }{
3862        \__stex_variables_invoke_math_ii:n { #1 }
3863      }
3864    }
3865 }
3866
3867 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
3868   \cs_if_exist:cTF {
3869     stex_var_op_notation_ #1 _cs
3870   }{
3871     \exp_args:Nnx \use:nn {
3872       \def\comp{\_varcomp}
3873       \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3874       \_stex_term_omv:nn { var://#1 }{
3875         \use:c{stex_var_op_notation_ #1 _cs }
3876       }
3877     }{
3878       \_stex_reset:N \comp
3879       \_stex_reset:N \l_stex_current_symbol_str
3880     }
3881   }{
3882     \int_compare:nNnTF {\prop_item:cn {l_stex_variable_#1_prop}{arity}} = 0{
3883       \__stex_variables_invoke_math_ii:n {#1}
3884     }{
3885       \msg_error:nnxx{stex}{error/noop}{variable~#1}{}
3886     }
3887   }
3888 }
3889
3890 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n  #1 {
3891   \cs_if_exist:cTF {
3892     stex_var_notation_#1_cs
3893   }{
3894     \tl_set:Nx \stex_symbol_after_invokation_tl {
3895       \_stex_reset:N \comp
3896       \_stex_reset:N \stex_symbol_after_invokation_tl
3897       \_stex_reset:N \l_stex_current_symbol_str
3898       \bool_set_true:N \l_stex_allow_semantic_bool
3899     }
3900     \def\comp{\_varcomp}
3901     \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3902     \bool_set_false:N \l_stex_allow_semantic_bool
3903     \use:c{stex_var_notation_#1_cs}
3904   }{
```

```
3905        \msg_error:nnxx{stex}{error/nonotation}{variable~#1}{s}
3906    }
3907 }
```

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

30.5 Sequences

```
3908 ⟨@@=stex_sequences⟩
3909
3910 \cs_new_protected:Nn \stex_invoke_sequence:n {
3911    \peek_charcode_remove:NTF ! {
3912      \_stex_term_omv:nn {varseq://#1}{
3913        \exp_args:Nnx \use:nn {
3914          \def\comp{\_varcomp}
3915          \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3916          \prop_item:cn{stex_varseq_#1_prop}{notation}
3917        }{
3918          \_stex_reset:N \comp
3919          \_stex_reset:N \l_stex_current_symbol_str
3920        }
3921      }
3922    }{
3923      \bool_set_false:N \l_stex_allow_semantic_bool
3924      \def\comp{\_varcomp}
3925      \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3926      \tl_set:Nx \stex_symbol_after_invokation_tl {
3927        \_stex_reset:N \comp
3928        \_stex_reset:N \stex_symbol_after_invokation_tl
3929        \_stex_reset:N \l_stex_current_symbol_str
3930        \bool_set_true:N \l_stex_allow_semantic_bool
3931      }
3932      \use:c { stex_varseq_#1_cs }
3933    }
3934 }
3935 ⟨/package⟩
```

Chapter 31

sTeX -Structural Features Implementation

```
3936  ⟨∗package⟩
3937
3938  %%%%%%%%%%%%    features.dtx    %%%%%%%%%%%%
3939
```

Warnings and error messages
```
3940  \msg_new:nnn{stex}{error/copymodule/notallowed}{
3941    Symbol~#1~can~not~be~assigned~in~copymodule~#2
3942  }
3943  \msg_new:nnn{stex}{error/interpretmodule/nodefiniens}{
3944    Symbol~#1~not~assigned~in~interpretmodule~#2
3945  }
3946
3947  \msg_new:nnn{stex}{error/unknownstructure}{
3948    No~structure~#1~found!
3949  }
3950
3951  \msg_new:nnn{stex}{error/unknownfield}{
3952    No~field~#1~in~instance~#2~found!\\#3
3953  }
3954
3955  \msg_new:nnn{stex}{error/keyval}{
3956    Invalid~key=value~pair:#1
3957  }
3958  \msg_new:nnn{stex}{error/instantiate/missing}{
3959    Assignments~missing~in~instantiate:~#1
3960  }
3961  \msg_new:nnn{stex}{error/incompatible}{
3962    Incompatible~signature:~#1~(#2)~and~#3~(#4)
3963  }
3964
```

31.1 Imports with modification

```
3965 ⟨@@=stex_copymodule⟩
3966 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
3967   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3968     \tl_set:Nn \l_tmpa_tl { #1 }
3969     \__stex_copymodule_get_symbol_from_cs:
3970   }{
3971     % argument is a string
3972     % is it a command name?
3973     \cs_if_exist:cTF { #1 }{
3974       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3975       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3976       \str_if_empty:NTF \l_tmpa_str {
3977         \exp_args:Nx \cs_if_eq:NNTF {
3978           \tl_head:N \l_tmpa_tl
3979         } \stex_invoke_symbol:n {
3980           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
3981         }{
3982           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3983         }
3984       } {
3985         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3986       }
3987     }{
3988       % argument is not a command name
3989       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3990       % \l_stex_all_symbols_seq
3991     }
3992   }
3993 }
3994
3995 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
3996   \str_set:Nn \l_tmpa_str { #1 }
3997   \bool_set_false:N \l_tmpa_bool
3998   \bool_if:NF \l_tmpa_bool {
3999     \tl_set:Nn \l_tmpa_tl {
4000       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
4001     }
4002   \str_set:Nn \l_tmpa_str { #1 }
4003   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4004   \seq_map_inline:Nn #2 {
4005     \str_set:Nn \l_tmpb_str { ##1 }
4006     \str_if_eq:eeT { \l_tmpa_str } {
4007       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4008     } {
4009       \seq_map_break:n {
4010         \tl_set:Nn \l_tmpa_tl {
4011           \str_set:Nn \l_stex_get_symbol_uri_str {
4012             ##1
4013           }
4014         }
4015       }
4016     }
```

```
4017         }
4018       \l_tmpa_tl
4019     }
4020 }
4021
4022 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
4023   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4024     { \tl_tail:N \l_tmpa_tl }
4025   \tl_if_single:NTF \l_tmpa_tl {
4026     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
4027       \exp_after:wN \str_set:Nn \exp_after:wN
4028         \l_stex_get_symbol_uri_str \l_tmpa_tl
4029       \__stex_copymodule_get_symbol_check:n { #1 }
4030     }{
4031       % TODO
4032       % tail is not a single group
4033     }
4034   }{
4035     % TODO
4036     % tail is not a single group
4037   }
4038 }
4039
4040 \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
4041   \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
4042     \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
4043       :~\seq_use:Nn #1 {,~}
4044     }
4045   }
4046 }
4047
4048 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
4049   % import module
4050   \stex_import_module_uri:nn { #1 } { #2 }
4051   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
4052   \stex_import_require_module:nnnn
4053     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4054     { \l_stex_import_path_str } { \l_stex_import_name_str }
4055
4056   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
4057   \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
4058
4059   % fields
4060   \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
4061   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4062     \seq_map_inline:cn {c_stex_module_##1_constants}{
4063       \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
4064         ##1 ? ####1
4065       }
4066     }
4067   }
4068
4069   % setup prop
4070   \seq_clear:N \l_tmpa_seq
```

```
4071    \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
4072      name     = \l_stex_current_copymodule_name_str ,
4073      module   = \l_stex_current_module_str ,
4074      from     = \l_stex_import_ns_str ?\l_stex_import_name_str ,
4075      includes = \l_tmpa_seq %,
4076    %  fields   = \l_tmpa_seq
4077    }
4078    \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
4079      as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
4080    \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_se
4081    \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,

4083    \stex_if_do_html:T {
4084      \begin{stex_annotate_env} {#4} {
4085        \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4086      }
4087      \stex_annotate_invisible:nnn{domain}{\l_stex_import_ns_str ?\l_stex_import_name_str}{}
4088    }
4089 }

4091 \cs_new_protected:Nn \stex_copymodule_end:n {
4092    % apply to every field
4093    \def \l_tmpa_cs ##1 ##2 {#1}

4095    \tl_clear:N \l__stex_copymodule_module_tl
4096    \tl_clear:N \l__stex_copymodule_exec_tl

4098    %\prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
4099    \seq_clear:N \l__stex_copymodule_fields_seq

4101    \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4102      \seq_map_inline:cn {c_stex_module_##1_constants}{

4104        \tl_clear:N \l__stex_copymodule_curr_symbol_tl % <- wrap in current symbol html
4105        \l_tmpa_cs{##1}{####1}

4107        \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
4108          \str_set_eq:Nc \l__stex_copymodule_curr_name_str {l__stex_copymodule_copymodule_##1?#
4109          \stex_if_do_html:T {
4110            \tl_put_right:Nx \l__stex_copymodule_curr_symbol_tl {
4111              \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?###
4112            }
4113          }
4114        }{
4115          \str_set:Nx \l__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str /
4116        }

4118        \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
4119        \prop_put:Nnx \l_tmpa_prop { name } \l__stex_copymodule_curr_name_str
4120        \prop_put:Nnx \l_tmpa_prop { module } \l_stex_current_module_str

4122        \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
4123          \stex_if_do_html:T {
4124            \tl_put_right:Nx \l__stex_copymodule_curr_symbol_tl {
```

```
4125            $\stex_annotate_invisible:nnn{definiens}{}{\exp_after:wN \exp_not:N\csname l__st
4126          }
4127        }
4128        \prop_put:Nnn \l_tmpa_prop { defined } { true }
4129      }
4130
4131      \stex_add_constant_to_current_module:n \__stex_copymodule_curr_name_str
4132      \tl_put_right:Nx \__stex_copymodule_module_tl {
4133        \seq_clear:c {l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_n
4134        \prop_set_from_keyval:cn {
4135          l_stex_symdecl_\l_stex_current_module_str ? \__stex_copymodule_curr_name_str _prop
4136        }{
4137          \prop_to_keyval:N \l_tmpa_prop
4138        }
4139      }
4140
4141      \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
4142        \stex_if_do_html:T {
4143          \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4144            \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1
4145          }
4146        }
4147        \tl_put_right:Nx \__stex_copymodule_module_tl {
4148          \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
4149            \stex_invoke_symbol:n {
4150              \l_stex_current_module_str ? \__stex_copymodule_curr_name_str
4151            }
4152          }
4153        }
4154      }
4155
4156      \seq_put_right:Nx \__stex_copymodule_fields_seq {\l_stex_current_module_str ? \__stex_
4157
4158      \tl_put_right:Nx \__stex_copymodule_exec_tl {
4159        \stex_copy_notations:nn {\l_stex_current_module_str ? \__stex_copymodule_curr_name_s
4160      }
4161
4162      \tl_put_right:Nx \__stex_copymodule_exec_tl {
4163        \stex_if_do_html:TF{
4164          \stex_annotate_invisible:nnn{assignment} {##1?####1} { \exp_after:wN \exp_not:n \e
4165        }{
4166          \exp_after:wN \exp_not:n \exp_after:wN {\__stex_copymodule_curr_symbol_tl}
4167        }
4168      }
4169    }
4170  }
4171
4172
4173  \prop_put:Nno \l_stex_current_copymodule_prop {fields} \__stex_copymodule_fields_seq
4174  \tl_put_left:Nx \__stex_copymodule_module_tl {
4175    \prop_set_from_keyval:cn {
4176      l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _pro
4177    }{
4178      \prop_to_keyval:N \l_stex_current_copymodule_prop
```

```
4179        }
4180      }
4181
4182      \seq_gput_right:cx{c_stex_module_\l_stex_current_module_str _copymodules}{
4183        \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4184      }
4185
4186      \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4187      \stex_debug:nn{copymodule}{result:\meaning \__stex_copymodule_module_tl}
4188      \stex_debug:nn{copymodule}{output:\meaning \__stex_copymodule_exec_tl}
4189
4190      \__stex_copymodule_exec_tl
4191      \stex_if_do_html:T {
4192        \end{stex_annotate_env}
4193      }
4194 }
4195
4196 \NewDocumentEnvironment {copymodule} { O{} m m}{
4197      \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ copymodule }
4198      \stex_deactivate_macro:Nn \symdecl {module~environments}
4199      \stex_deactivate_macro:Nn \symdef {module~environments}
4200      \stex_deactivate_macro:Nn \notation {module~environments}
4201      \stex_reactivate_macro:N \assign
4202      \stex_reactivate_macro:N \renamedecl
4203      \stex_reactivate_macro:N \donotcopy
4204      \stex_smsmode_do:
4205 }{
4206      \stex_copymodule_end:n {}
4207 }
4208
4209 \NewDocumentEnvironment {interpretmodule} { O{} m m}{
4210      \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ interpretmodule }
4211      \stex_deactivate_macro:Nn \symdecl {module~environments}
4212      \stex_deactivate_macro:Nn \symdef {module~environments}
4213      \stex_deactivate_macro:Nn \notation {module~environments}
4214      \stex_reactivate_macro:N \assign
4215      \stex_reactivate_macro:N \renamedecl
4216      \stex_reactivate_macro:N \donotcopy
4217      \stex_smsmode_do:
4218 }{
4219      \stex_copymodule_end:n {
4220        \tl_if_exist:cF {
4221          l__stex_copymodule_copymodule_##1?##2_def_tl
4222        }{
4223          \str_if_eq:eeF {
4224            \prop_item:cn{
4225              l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
4226          }{ true }{
4227            \msg_error:nnxx{stex}{error/interpretmodule/nodefiniens}{
4228              ##1?##2
4229            }{\l_stex_current_copymodule_name_str}
4230          }
4231        }
4232      }
```

185

```
4233 }
4234
4235 \iffalse \begin{stex_annotate_env} \fi
4236 \NewDocumentEnvironment {realization} { O{} m}{
4237   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #2 }{ realize }
4238   \stex_deactivate_macro:Nn \symdecl {module~environments}
4239   \stex_deactivate_macro:Nn \symdef {module~environments}
4240   \stex_deactivate_macro:Nn \notation {module~environments}
4241   \stex_reactivate_macro:N \donotcopy
4242   \stex_reactivate_macro:N \assign
4243   \stex_smsmode_do:
4244 }{
4245   \stex_import_module_uri:nn { #1 } { #2 }
4246   \tl_clear:N \__stex_copymodule_exec_tl
4247   \tl_set:Nx \__stex_copymodule_module_tl {
4248     \stex_import_require_module:nnnn
4249       { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4250       { \l_stex_import_path_str } { \l_stex_import_name_str }
4251   }
4252
4253   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4254     \seq_map_inline:cn {c_stex_module_##1_constants}{
4255       \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str / #
4256       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
4257         \stex_if_do_html:T {
4258           \tl_put_right:Nx \__stex_copymodule_exec_tl {
4259             \stex_annotate_invisible:nnn{assignment} {##1?####1} {
4260               $\stex_annotate_invisible:nnn{definiens}{}{\exp_after:wN \exp_not:N\csname l__
4261             }
4262           }
4263         }
4264         \tl_put_right:Nx \__stex_copymodule_module_tl {
4265           \prop_put:cnn {l_stex_symdecl_##1?####1_prop}{ defined }{ true }
4266         }
4267       }
4268   }}
4269
4270   \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4271
4272   \__stex_copymodule_exec_tl
4273   \stex_if_do_html:T {\end{stex_annotate_env}}
4274 }
4275
4276 \NewDocumentCommand \donotcopy { m }{
4277   \str_clear:N \l_stex_import_name_str
4278   \str_set:Nn \l_tmpa_str { #1 }
4279   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4280   \seq_map_inline:Nn \l_stex_all_modules_seq {
4281     \str_set:Nn \l_tmpb_str { ##1 }
4282     \str_if_eq:eeT { \l_tmpa_str } {
4283       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4284     } {
4285       \seq_map_break:n {
4286         \stex_if_do_html:T {
```

186

```
4287        \stex_if_smsmode:F {
4288          \stex_annotate_invisible:nnn{donotcopy}{##1}{
4289            \stex_annotate:nnn{domain}{##1}{}
4290          }
4291        }
4292      }
4293      \str_set_eq:NN \l_stex_import_name_str \l_tmpb_str
4294    }
4295  }
4296  \seq_map_inline:cn {c_stex_module_##1_copymodules}{
4297    \str_set:Nn \l_tmpb_str { ####1 }
4298    \str_if_eq:eeT { \l_tmpa_str } {
4299      \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4300    } {
4301      \seq_map_break:n {\seq_map_break:n {
4302        \stex_if_do_html:T {
4303          \stex_if_smsmode:F {
4304            \stex_annotate_invisible:nnn{donotcopy}{####1}{
4305              \stex_annotate:nnn{domain}{
4306                \prop_item:cn {l_stex_copymodule_ ####1 _prop}{module}
4307              }{}
4308            }
4309          }
4310        }
4311        \str_set:Nx \l_stex_import_name_str {
4312          \prop_item:cn {l_stex_copymodule_ ####1 _prop}{module}
4313        }
4314      }}
4315    }
4316  }
4317  }
4318  \str_if_empty:NTF \l_stex_import_name_str {
4319    % TODO throw error
4320  }{
4321    \stex_collect_imports:n {\l_stex_import_name_str }
4322    \seq_map_inline:Nn \l_stex_collect_imports_seq {
4323      \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
4324      \seq_map_inline:cn {c_stex_module_##1_constants}{
4325        \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? ####1 }
4326        \bool_lazy_any:nT {
4327          { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_name_str}}
4328          { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_macroname_str}}
4329          { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_def_tl}}
4330        }{
4331          % TODO throw error
4332        }
4333      }
4334    }
4335    \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
4336    \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_name_str }
4337    \prop_put:Nno \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
4338  }
4339  \stex_smsmode_do:
4340 }
```

```
4341
4342  \NewDocumentCommand \assign { m m }{
4343    \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
4344    \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
4345    \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _def_tl}{#2}
4346    \stex_smsmode_do:
4347  }
4348
4349  \keys_define:nn { stex / renamedecl } {
4350    name          .str_set_x:N  = \l_stex_renamedecl_name_str
4351  }
4352  \cs_new_protected:Nn \__stex_copymodule_renamedecl_args:n {
4353    \str_clear:N \l_stex_renamedecl_name_str
4354    \keys_set:nn { stex / renamedecl } { #1 }
4355  }
4356
4357  \NewDocumentCommand \renamedecl { O{} m m}{
4358    \__stex_copymodule_renamedecl_args:n { #1 }
4359    \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
4360    \stex_debug:nn{renamedecl}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
4361    \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
4362    \str_if_empty:NTF \l_stex_renamedecl_name_str {
4363      \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4364        \l_stex_get_symbol_uri_str
4365      } }
4366    } {
4367      \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex
4368      \stex_debug:nn{renamedecl}{@~\l_stex_current_module_str ? \l_stex_renamedecl_name_str}
4369      \prop_set_eq:cc {l_stex_symdecl_
4370        \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4371        _prop
4372      }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
4373      \seq_set_eq:cc {l_stex_symdecl_
4374        \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4375        _notations
4376      }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
4377      \prop_put:cnx {l_stex_symdecl_
4378        \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4379        _prop
4380      }{ name }{ \l_stex_renamedecl_name_str }
4381      \prop_put:cnx {l_stex_symdecl_
4382        \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4383        _prop
4384      }{ module }{ \l_stex_current_module_str }
4385      \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
4386        \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4387      }
4388      \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4389        \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4390      } }
4391    }
4392    \stex_smsmode_do:
4393  }
4394
```

188

```
4395  \stex_deactivate_macro:Nn \assign {copymodules}
4396  \stex_deactivate_macro:Nn \renamedecl {copymodules}
4397  \stex_deactivate_macro:Nn \donotcopy {copymodules}

4398
4399
```

31.2   The feature environment

```
4400  ⟨@@=stex_features⟩
4401
4402  \NewDocumentEnvironment{structural_feature_module}{ m m m }{
4403    \stex_if_in_module:F {
4404      \msg_set:nnn{stex}{error/nomodule}{
4405        Structural~Feature~has~to~occur~in~a~module:\\
4406        Feature~#2~of~type~#1\\
4407        In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
4408      }
4409      \msg_error:nn{stex}{error/nomodule}
4410    }
4411
4412    \str_set_eq:NN \l_stex_feature_parent_str \l_stex_current_module_str
4413
4414    \stex_module_setup:nn{meta=NONE}{#2 - #1}
4415
4416    \stex_if_do_html:T {
4417      \begin{stex_annotate_env}{ feature:#1 }{\l_stex_feature_parent_str ? #2 - #1}
4418        \stex_annotate_invisible:nnn{header}{}{ #3 }
4419    }
4420  }{
4421    \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
4422    \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
4423    \stex_debug:nn{features}{
4424      Feature: \l_stex_last_feature_str
4425    }
4426    \stex_if_do_html:T {
4427      \end{stex_annotate_env}
4428    }
4429  }
```

31.3   Structure

```
4430  ⟨@@=stex_structures⟩
4431  \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
4432    \prop_if_exist:cF {c_stex_module_\l_stex_current_module_str _structures}{
4433      \prop_new:c {c_stex_module_\l_stex_current_module_str _structures}
4434    }
4435    \prop_gput:cxx{c_stex_module_\l_stex_current_module_str _structures}
4436      {#1}{#2}
4437  }
4438
```

```
4439 \keys_define:nn { stex / features / structure } {
4440   name           .str_set_x:N  = \l__stex_structures_name_str ,
4441 }
4442
4443 \cs_new_protected:Nn \__stex_structures_structure_args:n {
4444   \str_clear:N \l__stex_structures_name_str
4445   \keys_set:nn { stex / features / structure } { #1 }
4446 }
4447
4448 \NewDocumentEnvironment{mathstructure}{m O{}}{
4449   \__stex_structures_structure_args:n { #2 }
4450   \str_if_empty:NT \l__stex_structures_name_str {
4451     \str_set:Nx \l__stex_structures_name_str { #1 }
4452   }
4453   \stex_suppress_html:n {
4454     \exp_args:Nx \stex_symdecl_do:nn {
4455       name = \l__stex_structures_name_str ,
4456       def  = {\STEXsymbol{module-type}{
4457         \_stex_term_math_oms:nnnn {
4458           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4459             { ns } ?
4460             \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4461               { name } / \l__stex_structures_name_str - structure
4462         }{}{0}{}
4463       }}
4464     }{ #1 }
4465   }
4466   \exp_args:Nnnx
4467   \begin{structural_feature_module}{ structure }
4468     { \l__stex_structures_name_str }{}
4469   \stex_smsmode_do:
4470 }{
4471   \end{structural_feature_module}
4472   \_stex_reset_up_to_module:n \l_stex_last_feature_str
4473   \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
4474   \seq_clear:N \l_tmpa_seq
4475   \seq_map_inline:Nn \l_stex_collect_imports_seq {
4476     \seq_map_inline:cn{c_stex_module_##1_constants}{
4477       \seq_put_right:Nn \l_tmpa_seq { ##1 ? ####1 }
4478     }
4479   }
4480   \exp_args:Nnno
4481   \prop_gput:cnn {c_stex_module_ \l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
4482   \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
4483   \stex_add_structure_to_current_module:nn
4484     \l__stex_structures_name_str
4485     \l_stex_last_feature_str
4486
4487   \stex_execute_in_module:x {
4488     \tl_set:cn { #1 }{
4489       \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structure
4490     }
4491   }
4492 }
```

190

```
4493
4494  \cs_new:Nn \stex_invoke_structure:nn {
4495    \stex_invoke_symbol:n { #1?#2 }
4496  }
4497
4498  \cs_new_protected:Nn \stex_get_structure:n {
4499    \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4500      \tl_set:Nn \l_tmpa_tl { #1 }
4501      \__stex_structures_get_from_cs:
4502    }{
4503      \cs_if_exist:cTF { #1 }{
4504        \cs_set_eq:Nc \l_tmpa_cs { #1 }
4505        \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
4506        \str_if_empty:NTF \l_tmpa_str {
4507          \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs} \stex_invoke_structure:nn {
4508            \__stex_structures_get_from_cs:
4509          }{
4510            \__stex_structures_get_from_string:n { #1 }
4511          }
4512        }{
4513          \__stex_structures_get_from_string:n { #1 }
4514        }
4515      }{
4516        \__stex_structures_get_from_string:n { #1 }
4517      }
4518    }
4519  }
4520
4521  \cs_new_protected:Nn \__stex_structures_get_from_cs: {
4522    \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4523      { \tl_tail:N \l_tmpa_tl }
4524    \str_set:Nx \l_tmpa_str {
4525      \exp_after:wN \use_i:nn \l_tmpa_tl
4526    }
4527    \str_set:Nx \l_tmpb_str {
4528      \exp_after:wN \use_ii:nn \l_tmpa_tl
4529    }
4530    \str_set:Nx \l_stex_get_structure_str {
4531      \l_tmpa_str ? \l_tmpb_str
4532    }
4533    \str_set:Nx \l_stex_get_structure_module_str {
4534      \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
4535    }
4536  }
4537
4538  \cs_new_protected:Nn \__stex_structures_get_from_string:n {
4539    \tl_set:Nn \l_tmpa_tl {
4540      \msg_error:nnn{stex}{error/unknownstructure}{#1}
4541    }
4542    \str_set:Nn \l_tmpa_str { #1 }
4543    \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4544
4545    \seq_map_inline:Nn \l_stex_all_modules_seq {
4546      \prop_if_exist:cT {c_stex_module_##1_structures} {
```

```
4547        \prop_map_inline:cn {c_stex_module_##1_structures} {
4548          \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
4549            \prop_map_break:n{\seq_map_break:n{
4550              \tl_set:Nn \l_tmpa_tl {
4551                \str_set:Nn \l_stex_get_structure_str {##1?####1}
4552                \str_set:Nn \l_stex_get_structure_module_str {####2}
4553              }
4554            }}
4555          }
4556        }
4557      }
4558    }
4559    \l_tmpa_tl
4560 }
```

```
4561
4562 \keys_define:nn { stex / instantiate } {
4563    name          .str_set_x:N  = \l__stex_structures_name_str
4564 }
4565 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
4566    \str_clear:N \l__stex_structures_name_str
4567    \keys_set:nn { stex / instantiate } { #1 }
4568 }
4569
4570 \NewDocumentCommand \instantiate {m O{} m m O{}}{
4571    \begingroup
4572      \stex_get_structure:n {#3}
4573      \__stex_structures_instantiate_args:n { #2 }
4574      \str_if_empty:NT \l__stex_structures_name_str {
4575        \str_set:Nn \l__stex_structures_name_str { #1 }
4576      }
4577      \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4578      \seq_clear:N \l__stex_structures_fields_seq
4579      \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4580      \seq_map_inline:Nn \l_stex_collect_imports_seq {
4581        \seq_map_inline:cn {c_stex_module_##1_constants}{
4582          \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4583        }
4584      }
4585
4586      \tl_if_empty:nF{#5}{
4587        \seq_set_split:Nnn \l_tmpa_seq , {#5}
4588        \prop_clear:N \l_tmpa_prop
4589        \seq_map_inline:Nn \l_tmpa_seq {
4590          \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4591          \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4592            \msg_error:nnn{stex}{error/keyval}{##1}
4593          }
4594          \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
4595          \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4596          \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
4597          \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}
4598          \exp_args:Nxx \str_if_eq:nnF
```

```
4599            {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4600            {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
4601            \msg_error:nnxxxx{stex}{error/incompatible}
4602              {\l__stex_structures_dom_str}
4603              {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4604              {\l_stex_get_symbol_uri_str}
4605              {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
4606          }
4607          \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str
4608        }
4609      }
4610
4611      \seq_map_inline:Nn \l__stex_structures_fields_seq {
4612        \str_set:Nx \l_tmpa_str {field:\l__stex_structures_name_str . \prop_item:cn {l_stex_sy
4613        \stex_debug:nn{instantiate}{Field~\l_tmpa_str :~##1}
4614
4615        \stex_add_constant_to_current_module:n {\l_tmpa_str}
4616        \stex_execute_in_module:x {
4617          \prop_set_from_keyval:cn { l_stex_symdecl_ \l_stex_current_module_str?\l_tmpa_str _p
4618            name  = \l_tmpa_str ,
4619            args  = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4620            arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4621            assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4622          }
4623          \seq_clear:c {l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _notations}
4624        }
4625
4626        \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4627          \stex_find_notation:nn{##1}{}
4628          \stex_execute_in_module:x {
4629            \seq_put_right:cn {l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _notation
4630          }
4631
4632          \stex_copy_control_sequence_ii:ccN
4633            {stex_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_notation_
4634            {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4635            \l_tmpa_tl
4636          \exp_args:No \stex_execute_in_module:n \l_tmpa_tl
4637
4638
4639          \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
4640            \tl_set_eq:Nc \l_tmpa_cs {stex_op_notation_##1\c_hash_str \l_stex_notation_variant
4641            \stex_execute_in_module:x {
4642              \tl_set:cn
4643              {stex_op_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_nota
4644              { \exp_args:No \exp_not:n \l_tmpa_cs}
4645            }
4646          }
4647
4648        }
4649
4650        \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\l_stex_cur
4651      }
4652
```

193

```
4653      \stex_execute_in_module:x {
4654        \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
4655          domain = \l_stex_get_structure_module_str ,
4656          \prop_to_keyval:N \l_tmpa_prop
4657        }
4658        \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
4659      }
4660      \stex_debug:nn{instantiate}{
4661        Instance~\l_stex_current_module_str?\l__stex_structures_name_str \\
4662        \prop_to_keyval:N \l_tmpa_prop
4663      }
4664      \exp_args:Nxx \stex_symdecl_do:nn {
4665        type={\STEXsymbol{module-type}{
4666          \_stex_term_math_oms:nnnn {
4667            \l_stex_get_structure_module_str
4668          }{}{0}{}
4669        }}
4670      }{\l__stex_structures_name_str}
4671 %    {
4672        \str_set:Nx \l_stex_get_symbol_uri_str {\l_stex_current_module_str?\l__stex_structures
4673        \tl_set:Nn \l_stex_notation_after_do_tl {\__stex_notation_final:}
4674        \stex_notation_do:nnnnn{}{0}{}{}{\comp{#4}}
4675 %    }
4676      %\exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}
4677    \endgroup
4678    \stex_smsmode_do:\ignorespacesandpars
4679 }
4680
4681 \cs_new_protected:Nn \stex_symbol_or_var:n {
4682    \cs_if_exist:cTF{#1}{
4683      \cs_set_eq:Nc \l_tmpa_tl { #1 }
4684      \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4685      \str_if_empty:NTF \l_tmpa_str {
4686        \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
4687          \stex_invoke_variable:n {
4688            \bool_set_true:N \l_stex_symbol_or_var_bool
4689            \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
4690            \str_set:Nx \l_stex_get_symbol_uri_str {
4691              \exp_after:wN \use:n \l_tmpa_tl
4692            }
4693          }{
4694            \bool_set_false:N \l_stex_symbol_or_var_bool
4695            \stex_get_symbol:n{#1}
4696          }
4697      }{
4698        \__stex_structures_symbolorvar_from_string:n{ #1 }
4699      }
4700    }{
4701      \__stex_structures_symbolorvar_from_string:n{ #1 }
4702    }
4703 }
4704
4705 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
4706    \prop_if_exist:cTF {l_stex_variable_#1 _prop}{
```

194

```
4707      \bool_set_true:N \l_stex_symbol_or_var_bool
4708      \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
4709   }{
4710      \bool_set_false:N \l_stex_symbol_or_var_bool
4711      \stex_get_symbol:n{#1}
4712   }
4713 }
4714
4715 \keys_define:nn { stex / varinstantiate } {
4716   name         .str_set_x:N  = \l__stex_structures_name_str,
4717   bind         .choices:nn   =
4718       {forall,exists}
4719       {\str_set:Nx \l__stex_structures_bind_str {\l_keys_choice_tl}}
4720
4721 }
4722 \cs_new_protected:Nn \__stex_structures_varinstantiate_args:n {
4723   \str_clear:N \l__stex_structures_name_str
4724   \str_clear:N \l__stex_structures_bind_str
4725   \keys_set:nn { stex / varinstantiate } { #1 }
4726 }
4727
4728 \NewDocumentCommand \varinstantiate {m O{} m m O{}}{
4729   \begingroup
4730      \stex_get_structure:n {#3}
4731      \__stex_structures_varinstantiate_args:n { #2 }
4732      \str_if_empty:NT \l__stex_structures_name_str {
4733        \str_set:Nn \l__stex_structures_name_str { #1 }
4734      }
4735      \stex_if_do_html:TF{
4736        \stex_annotate:nnn{varinstance}{\l__stex_structures_name_str}
4737      }{\use:n}
4738      {
4739        \stex_if_do_html:T{
4740          \stex_annotate_invisible:nnn{domain}{\l_stex_get_structure_module_str}{}
4741        }
4742        \seq_clear:N \l__stex_structures_fields_seq
4743        \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4744        \seq_map_inline:Nn \l_stex_collect_imports_seq {
4745          \seq_map_inline:cn {c_stex_module_##1_constants}{
4746            \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4747          }
4748        }
4749        \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4750        \prop_clear:N \l_tmpa_prop
4751        \tl_if_empty:nF {#5} {
4752          \seq_set_split:Nnn \l_tmpa_seq , {#5}
4753          \seq_map_inline:Nn \l_tmpa_seq {
4754            \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4755            \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4756              \msg_error:nnn{stex}{error/keyval}{##1}
4757            }
4758            \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_stru
4759            \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4760            \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol
```

```
4761                \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
4762                \stex_if_do_html:T{
4763                  \stex_annotate:nnn{assign}{\l__stex_structures_dom_str,\l_stex_get_symbol_uri_st
4764                }
4765                \bool_if:NTF \l_stex_symbol_or_var_bool {
4766                  \exp_args:Nxx \str_if_eq:nnF
4767                    {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4768                    {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}{
4769                  \msg_error:nnxxxx{stex}{error/incompatible}
4770                    {\l__stex_structures_dom_str}
4771                    {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4772                    {\l_stex_get_symbol_uri_str}
4773                    {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}
4774                }
4775                \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n
4776            }{
4777                \exp_args:Nxx \str_if_eq:nnF
4778                    {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4779                    {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
4780                  \msg_error:nnxxxx{stex}{error/incompatible}
4781                    {\l__stex_structures_dom_str}
4782                    {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4783                    {\l_stex_get_symbol_uri_str}
4784                    {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
4785                }
4786                \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n {
4787              }
4788            }
4789          }
4790          \tl_gclear:N \g__stex_structures_aftergroup_tl
4791          \seq_map_inline:Nn \l__stex_structures_fields_seq {
4792            \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdec
4793            \stex_debug:nn{varinstantiate}{Field~\l_tmpa_str :~##1}
4794            \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4795              \stex_find_notation:nn{##1}{}
4796              \cs_gset_eq:cc{g__stex_structures_tmpa_\l_tmpa_str _cs}
4797                {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4798              \stex_debug:nn{varinstantiate}{Notation:~\cs_meaning:c{g__stex_structures_tmpa_\l_
4799              \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
4800                \cs_gset_eq:cc {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
4801                  {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4802                  \stex_debug:nn{varinstantiate}{Operator~Notation:~\cs_meaning:c{g__stex_struct
4803            }
4804          }
4805
4806          \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4807            \prop_set_from_keyval:cn { l_stex_variable_ \l_tmpa_str _prop}{
4808              name  = \l_tmpa_str ,
4809              args  = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4810              arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4811              assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4812            }
4813            \cs_set_eq:cc {stex_var_notation_\l_tmpa_str _cs}
4814              {g__stex_structures_tmpa_\l_tmpa_str _cs}
```

```
4815            \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str _cs}
4816              {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
4817          }
4818          \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\stex_inv
4819        }
4820        \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4821          \prop_set_from_keyval:cn {l_stex_varinstance_\l__stex_structures_name_str _prop }{
4822            domain = \l_stex_get_structure_module_str ,
4823            \prop_to_keyval:N \l_tmpa_prop
4824          }
4825          \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
4826          \tl_set:cn {l_stex_varinstance_\l__stex_structures_name_str _op_tl}{
4827            \exp_args:Nnx \exp_not:N \use:nn {
4828              \str_set:Nn \exp_not:N \l_stex_current_symbol_str {var://\l__stex_structures_nam
4829              \_stex_term_omv:nn {var://\l__stex_structures_name_str}{
4830                \exp_not:n{
4831                  \_varcomp{#4}
4832                }
4833              }
4834            }{
4835              \exp_not:n{\_stex_reset:N \l_stex_current_symbol_str}
4836            }
4837          }
4838        }
4839      }
4840      \stex_debug:nn{varinstantiate}{\expandafter\detokenize\expandafter{\g__stex_structures_a
4841      \aftergroup\g__stex_structures_aftergroup_tl
4842    \endgroup
4843    \stex_smsmode_do:\ignorespacesandpars
4844 }
4845
4846 \cs_new_protected:Nn \stex_invoke_instance:n {
4847    \peek_charcode_remove:NTF ! {
4848      \stex_invoke_symbol:n{#1}
4849    }{
4850      \_stex_invoke_instance:nn {#1}
4851    }
4852 }
4853
4854
4855 \cs_new_protected:Nn \stex_invoke_varinstance:n {
4856    \peek_charcode_remove:NTF ! {
4857      \exp_args:Nnx \use:nn {
4858        \def\comp{\_varcomp}
4859        \use:c{l_stex_varinstance_#1_op_tl}
4860      }{
4861        \_stex_reset:N \comp
4862      }
4863    }{
4864      \_stex_invoke_varinstance:nn {#1}
4865    }
4866 }
4867
4868 \cs_new_protected:Nn \_stex_invoke_instance:nn {
```

197

```
4869    \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
4870      \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
4871    }{
4872      \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
4873      \msg_error:nnxxx{stex}{error/unknownfield}{#2}{#1}{
4874        \prop_to_keyval:N \l_tmpa_prop
4875      }
4876    }
4877  }
4878
4879  \cs_new_protected:Nn \_stex_invoke_varinstance:nn {
4880    \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
4881      \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
4882      \l_tmpa_tl
4883    }{
4884      \msg_error:nnnnn{stex}{error/unknownfield}{#2}{#1}{}
4885    }
4886  }
```

(End definition for `\instantiate`*. This function is documented on page 32.)*

`\stex_invoke_structure:nnn`

```
4887  % #1: URI of the instance
4888  % #2: URI of the instantiated module
4889  \cs_new_protected:Nn \stex_invoke_structure:nnn {
4890    \tl_if_empty:nTF{ #3 }{
4891      \prop_set_eq:Nc \l__stex_structures_structure_prop {
4892        c_stex_feature_ #2 _prop
4893      }
4894      \tl_clear:N \l_tmpa_tl
4895      \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
4896      \seq_map_inline:Nn \l_tmpa_seq {
4897        \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
4898        \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
4899        \cs_if_exist:cT {
4900          stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
4901        }{
4902          \tl_if_empty:NF \l_tmpa_tl {
4903            \tl_put_right:Nn \l_tmpa_tl {,}
4904          }
4905          \tl_put_right:Nx \l_tmpa_tl {
4906            \stex_invoke_symbol:n {#1/\l_tmpa_str}!
4907          }
4908        }
4909      }
4910      \exp_args:No \mathstruct \l_tmpa_tl
4911    }{
4912      \stex_invoke_symbol:n{#1/#3}
4913    }
4914  }
```

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

```
4915  ⟨/package⟩
```

198

Chapter 32

STEX
-Statements Implementation

```
4916 ⟨∗package⟩
4917
4918 %%%%%%%%%%%%   features.dtx   %%%%%%%%%%%%
4919
4920 ⟨@@=stex_statements⟩
```

Warnings and error messages

```
4921
```

\titleemph

```
4922 \def\titleemph#1{\textbf{#1}}
```

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

32.1   Definitions

definiendum

```
4923 \keys_define:nn {stex / definiendum }{
4924   pre     .tl_set:N     = \l__stex_statements_definiendum_pre_tl,
4925   post    .tl_set:N     = \l__stex_statements_definiendum_post_tl,
4926   root    .str_set_x:N  = \l__stex_statements_definiendum_root_str,
4927   gfa     .str_set_x:N  = \l__stex_statements_definiendum_gfa_str
4928 }
4929 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
4930   \str_clear:N \l__stex_statements_definiendum_root_str
4931   \tl_clear:N \l__stex_statements_definiendum_post_tl
4932   \str_clear:N \l__stex_statements_definiendum_gfa_str
4933   \keys_set:nn { stex / definiendum }{ #1 }
4934 }
4935 \NewDocumentCommand \definiendum { O{} m m} {
4936   \__stex_statements_definiendum_args:n { #1 }
4937   \stex_get_symbol:n { #2 }
4938   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4939   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
4940     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

199

```
4941        \tl_set:Nn \l_tmpa_tl { #3 }
4942      } {
4943        \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
4944        \tl_set:Nn \l_tmpa_tl {
4945          \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
4946        }
4947      }
4948    } {
4949      \tl_set:Nn \l_tmpa_tl { #3 }
4950    }
4951
4952    % TODO root
4953    \stex_html_backend:TF {
4954      \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
4955    } {
4956      \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
4957    }
4958 }
4959 \stex_deactivate_macro:Nn \definiendum {definition~environments}
```

(*End definition for* `definiendum`*. This function is documented on page* *41.*)

definame

```
4960
4961 \NewDocumentCommand \definame { O{} m } {
4962    \__stex_statements_definiendum_args:n { #1 }
4963    % TODO: root
4964    \stex_get_symbol:n { #2 }
4965    \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4966    \str_set:Nx \l_tmpa_str {
4967      \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4968    }
4969    \str_replace_all:Nnn \l_tmpa_str {-} {~}
4970    \stex_html_backend:TF {
4971      \stex_if_do_html:T {
4972        \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4973          \l_tmpa_str\l__stex_statements_definiendum_post_tl
4974        }
4975      }
4976    } {
4977      \exp_args:Nnx \defemph@uri {
4978        \l_tmpa_str\l__stex_statements_definiendum_post_tl
4979      } { \l_stex_get_symbol_uri_str }
4980    }
4981 }
4982 \stex_deactivate_macro:Nn \definame {definition~environments}
4983
4984 \NewDocumentCommand \Definame { O{} m } {
4985    \__stex_statements_definiendum_args:n { #1 }
4986    \stex_get_symbol:n { #2 }
4987    \str_set:Nx \l_tmpa_str {
4988      \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4989    }
4990    \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
```

200

```
4991      \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4992      \stex_html_backend:TF {
4993        \stex_if_do_html:T {
4994          \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4995            \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
4996          }
4997        }
4998      } {
4999        \exp_args:Nnx \defemph@uri {
5000          \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5001        } { \l_stex_get_symbol_uri_str }
5002      }
5003 }
5004 \stex_deactivate_macro:Nn \Definame {definition~environments}
5005
5006 \NewDocumentCommand \premise { m }{
5007    \stex_annotate:nnn{ premise }{}{ #1 }
5008 }
5009 \NewDocumentCommand \conclusion { m }{
5010    \stex_annotate:nnn{ conclusion }{}{ #1 }
5011 }
5012 \NewDocumentCommand \definiens { O{} m }{
5013    \str_clear:N \l_stex_get_symbol_uri_str
5014    \tl_if_empty:nF {#1} {
5015      \stex_get_symbol:n { #1 }
5016    }
5017    \str_if_empty:NT \l_stex_get_symbol_uri_str {
5018      \int_compare:nNnTF {\clist_count:N \l__stex_statements_sdefinition_for_clist} = 1 {
5019        \str_set:Nx \l_stex_get_symbol_uri_str {\clist_item:Nn \l__stex_statements_sdefinition
5020      }{
5021        % TODO throw error
5022      }
5023    }
5024    \str_if_eq:eeT {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{module}}
5025      {\l_stex_current_module_str}{
5026        \str_if_eq:eeF {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defin
5027        {true}{
5028          \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5029          \exp_args:Nx \stex_add_to_current_module:n {
5030            \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5031          }
5032        }
5033    }
5034    \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
5035 }
5036
5037 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
5038 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
5039 \stex_deactivate_macro:Nn \definiens {definition~environments}
5040
```

(*End definition for* `definame`*. This function is documented on page* *41.*)

sdefinition

```
5041
5042 \keys_define:nn {stex / sdefinition }{
5043   type      .str_set_x:N  = \sdefinitiontype,
5044   id        .str_set_x:N  = \sdefinitionid,
5045   name      .str_set_x:N  = \sdefinitionname,
5046   for       .clist_set:N  = \l__stex_statements_sdefinition_for_clist ,
5047   title     .tl_set:N     = \sdefinitiontitle
5048 }
5049 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
5050   \str_clear:N \sdefinitiontype
5051   \str_clear:N \sdefinitionid
5052   \str_clear:N \sdefinitionname
5053   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5054   \tl_clear:N \sdefinitiontitle
5055   \keys_set:nn { stex / sdefinition }{ #1 }
5056 }
5057
5058 \NewDocumentEnvironment{sdefinition}{O{}}{
5059   \__stex_statements_sdefinition_args:n{ #1 }
5060   \stex_reactivate_macro:N \definiendum
5061   \stex_reactivate_macro:N \definame
5062   \stex_reactivate_macro:N \Definame
5063   \stex_reactivate_macro:N \premise
5064   \stex_reactivate_macro:N \definiens
5065   \stex_if_smsmode:F{
5066     \seq_clear:N \l_tmpa_seq
5067     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5068       \tl_if_empty:nF{ ##1 }{
5069         \stex_get_symbol:n { ##1 }
5070         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5071           \l_stex_get_symbol_uri_str
5072         }
5073       }
5074     }
5075     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpa_seq
5076     \exp_args:Nnnx
5077     \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {,}}
5078     \str_if_empty:NF \sdefinitiontype {
5079       \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5080     }
5081     \str_if_empty:NF \sdefinitionname {
5082       \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5083     }
5084     \clist_set:No \l_tmpa_clist \sdefinitiontype
5085     \tl_clear:N \l_tmpa_tl
5086     \clist_map_inline:Nn \l_tmpa_clist {
5087       \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
5088         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
5089       }
5090     }
5091     \tl_if_empty:NTF \l_tmpa_tl {
5092       \__stex_statements_sdefinition_start:
5093     }{
5094       \l_tmpa_tl
```

```
5095        }
5096      }
5097      \stex_ref_new_doc_target:n \sdefinitionid
5098      \stex_smsmode_do:
5099  }{
5100      \stex_suppress_html:n {
5101        \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5102      }
5103      \stex_if_smsmode:F {
5104        \clist_set:No \l_tmpa_clist \sdefinitiontype
5105        \tl_clear:N \l_tmpa_tl
5106        \clist_map_inline:Nn \l_tmpa_clist {
5107          \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
5108            \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
5109          }
5110        }
5111        \tl_if_empty:NTF \l_tmpa_tl {
5112          \__stex_statements_sdefinition_end:
5113        }{
5114          \l_tmpa_tl
5115        }
5116        \end{stex_annotate_env}
5117      }
5118  }
```

\stexpatchdefinition

```
5119  \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
5120      \par\noindent\titleemph{Definition\tl_if_empty:NF \sdefinitiontitle {
5121        ~(\sdefinitiontitle)
5122      }~}
5123  }
5124  \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\par\medskip}
5125
5126  \newcommand\stexpatchdefinition[3][] {
5127      \str_set:Nx \l_tmpa_str{ #1 }
5128      \str_if_empty:NTF \l_tmpa_str {
5129        \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
5130        \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
5131      }{
5132        \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2
5133        \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
5134      }
5135  }
```

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

\inlinedef   inline:

```
5136  \keys_define:nn {stex / inlinedef }{
5137      type     .str_set_x:N  = \sdefinitiontype,
5138      id       .str_set_x:N  = \sdefinitionid,
5139      for      .clist_set:N  = \l__stex_statements_sdefinition_for_clist ,
5140      name     .str_set_x:N  = \sdefinitionname
5141  }
5142  \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
```

```
5143    \str_clear:N \sdefinitiontype
5144    \str_clear:N \sdefinitionid
5145    \str_clear:N \sdefinitionname
5146    \clist_clear:N \l__stex_statements_sdefinition_for_clist
5147    \keys_set:nn { stex / inlinedef }{ #1 }
5148  }
5149  \NewDocumentCommand \inlinedef { O{} m } {
5150    \begingroup
5151    \__stex_statements_inlinedef_args:n{ #1 }
5152    \stex_reactivate_macro:N \definiendum
5153    \stex_reactivate_macro:N \definame
5154    \stex_reactivate_macro:N \Definame
5155    \stex_reactivate_macro:N \premise
5156    \stex_reactivate_macro:N \definiens
5157    \stex_ref_new_doc_target:n \sdefinitionid
5158    \stex_if_smsmode:TF{\stex_suppress_html:n {
5159      \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5160    }}{
5161      \seq_clear:N \l_tmpa_seq
5162      \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5163        \tl_if_empty:nF{ ##1 }{
5164          \stex_get_symbol:n { ##1 }
5165          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5166            \l_stex_get_symbol_uri_str
5167          }
5168        }
5169      }
5170      \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpa_seq
5171      \exp_args:Nnx
5172      \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
5173        \str_if_empty:NF \sdefinitiontype {
5174          \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5175        }
5176        #2
5177        \str_if_empty:NF \sdefinitionname {
5178          \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sdefinitionname}}
5179          \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5180        }
5181      }
5182    }
5183    \endgroup
5184    \stex_smsmode_do:
5185  }
```

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

32.2 Assertions

sassertion

```
5186
5187  \keys_define:nn {stex / sassertion }{
5188    type    .str_set_x:N  = \sassertiontype,
5189    id      .str_set_x:N  = \sassertionid,
```

```
5190   title    .tl_set:N     = \sassertiontitle ,
5191   for      .clist_set:N  = \l__stex_statements_sassertion_for_clist ,
5192   name     .str_set_x:N  = \sassertionname
5193 }
5194 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
5195   \str_clear:N \sassertiontype
5196   \str_clear:N \sassertionid
5197   \str_clear:N \sassertionname
5198   \clist_clear:N \l__stex_statements_sassertion_for_clist
5199   \tl_clear:N \sassertiontitle
5200   \keys_set:nn { stex / sassertion }{ #1 }
5201 }
5202
5203 %\tl_new:N \g__stex_statements_aftergroup_tl
5204
5205 \NewDocumentEnvironment{sassertion}{O{}}{
5206   \__stex_statements_sassertion_args:n{ #1 }
5207   \stex_reactivate_macro:N \premise
5208   \stex_reactivate_macro:N \conclusion
5209   \stex_if_smsmode:F {
5210     \seq_clear:N \l_tmpa_seq
5211     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5212       \tl_if_empty:nF{ ##1 }{
5213         \stex_get_symbol:n { ##1 }
5214         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5215           \l_stex_get_symbol_uri_str
5216         }
5217       }
5218     }
5219     \exp_args:Nnnx
5220     \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}
5221     \str_if_empty:NF \sassertiontype {
5222       \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
5223     }
5224     \str_if_empty:NF \sassertionname {
5225       \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
5226     }
5227     \clist_set:No \l_tmpa_clist \sassertiontype
5228     \tl_clear:N \l_tmpa_tl
5229     \clist_map_inline:Nn \l_tmpa_clist {
5230       \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
5231         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
5232       }
5233     }
5234     \tl_if_empty:NTF \l_tmpa_tl {
5235       \__stex_statements_sassertion_start:
5236     }{
5237       \l_tmpa_tl
5238     }
5239   }
5240   \str_if_empty:NTF \sassertionid {
5241     \str_if_empty:NF \sassertionname {
5242       \stex_ref_new_doc_target:n {}
5243     }
```

```
5244        } {
5245          \stex_ref_new_doc_target:n \sassertionid
5246        }
5247      \stex_smsmode_do:
5248 }{
5249    \str_if_empty:NF \sassertionname {
5250      \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5251      \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5252    }
5253    \stex_if_smsmode:F {
5254      \clist_set:No \l_tmpa_clist \sassertiontype
5255      \tl_clear:N \l_tmpa_tl
5256      \clist_map_inline:Nn \l_tmpa_clist {
5257        \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
5258          \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
5259        }
5260      }
5261      \tl_if_empty:NTF \l_tmpa_tl {
5262        \__stex_statements_sassertion_end:
5263      }{
5264        \l_tmpa_tl
5265      }
5266      \end{stex_annotate_env}
5267    }
5268 }
```

\stexpatchassertion

```
5269
5270 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
5271    \par\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {
5272      (\sassertiontitle)
5273    }~}
5274 }
5275 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
5276
5277 \newcommand\stexpatchassertion[3][] {
5278    \str_set:Nx \l_tmpa_str{ #1 }
5279    \str_if_empty:NTF \l_tmpa_str {
5280      \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
5281      \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
5282    }{
5283      \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
5284      \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 
5285    }
5286 }
```

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

\inlineass inline:

```
5287 \keys_define:nn {stex / inlineass }{
5288    type     .str_set_x:N  = \sassertiontype,
5289    id       .str_set_x:N  = \sassertionid,
5290    for      .clist_set:N  = \l__stex_statements_sassertion_for_clist ,
5291    name     .str_set_x:N  = \sassertionname
```

```
5292  }
5293  \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
5294    \str_clear:N \sassertiontype
5295    \str_clear:N \sassertionid
5296    \str_clear:N \sassertionname
5297    \clist_clear:N \l__stex_statements_sassertion_for_clist
5298    \keys_set:nn { stex / inlineass }{ #1 }
5299  }
5300  \NewDocumentCommand \inlineass { O{} m } {
5301    \begingroup
5302    \stex_reactivate_macro:N \premise
5303    \stex_reactivate_macro:N \conclusion
5304    \__stex_statements_inlineass_args:n{ #1 }
5305    \str_if_empty:NTF \sassertionid {
5306      \str_if_empty:NF \sassertionname {
5307        \stex_ref_new_doc_target:n {}
5308      }
5309    } {
5310      \stex_ref_new_doc_target:n \sassertionid
5311    }
5312
5313    \stex_if_smsmode:TF{
5314      \str_if_empty:NF \sassertionname {
5315        \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5316        \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5317      }
5318    }{
5319      \seq_clear:N \l_tmpa_seq
5320      \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5321        \tl_if_empty:nF{ ##1 }{
5322          \stex_get_symbol:n { ##1 }
5323          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5324            \l_stex_get_symbol_uri_str
5325          }
5326        }
5327      }
5328      \exp_args:Nnx
5329      \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{
5330        \str_if_empty:NF \sassertiontype {
5331          \stex_annotate_invisible:nnn{typestrings}{\sassertiontype}{}
5332        }
5333        #2
5334        \str_if_empty:NF \sassertionname {
5335          \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5336          \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5337          \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
5338        }
5339      }
5340    }
5341    \endgroup
5342    \stex_smsmode_do:
5343  }
```

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

32.3   Examples

```
5344
5345 \keys_define:nn {stex / sexample }{
5346   type     .str_set_x:N  = \exampletype,
5347   id       .str_set_x:N  = \sexampleid,
5348   title    .tl_set:N     = \sexampletitle,
5349   name     .str_set_x:N  = \sexamplename ,
5350   for      .clist_set:N  = \l__stex_statements_sexample_for_clist,
5351 }
5352 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
5353   \str_clear:N \sexampletype
5354   \str_clear:N \sexampleid
5355   \str_clear:N \sexamplename
5356   \tl_clear:N \sexampletitle
5357   \clist_clear:N \l__stex_statements_sexample_for_clist
5358   \keys_set:nn { stex / sexample }{ #1 }
5359 }
5360
5361 \NewDocumentEnvironment{sexample}{O{}}{
5362   \__stex_statements_sexample_args:n{ #1 }
5363   \stex_reactivate_macro:N \premise
5364   \stex_reactivate_macro:N \conclusion
5365   \stex_if_smsmode:F {
5366     \seq_clear:N \l_tmpa_seq
5367     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5368       \tl_if_empty:nF{ ##1 }{
5369         \stex_get_symbol:n { ##1 }
5370         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5371           \l_stex_get_symbol_uri_str
5372         }
5373       }
5374     }
5375     \exp_args:Nnnx
5376     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
5377     \str_if_empty:NF \sexampletype {
5378       \stex_annotate_invisible:nnn{typestrings}{\sexampletype}{}
5379     }
5380     \str_if_empty:NF \sexamplename {
5381       \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}
5382     }
5383     \clist_set:No \l_tmpa_clist \sexampletype
5384     \tl_clear:N \l_tmpa_tl
5385     \clist_map_inline:Nn \l_tmpa_clist {
5386       \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
5387         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
5388       }
5389     }
5390     \tl_if_empty:NTF \l_tmpa_tl {
5391       \__stex_statements_sexample_start:
5392     }{
5393       \l_tmpa_tl
5394     }
```

```
5395        }
5396     \str_if_empty:NF \sexampleid {
5397        \stex_ref_new_doc_target:n \sexampleid
5398     }
5399     \stex_smsmode_do:
5400  }{
5401     \str_if_empty:NF \sexamplename {
5402        \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5403     }
5404     \stex_if_smsmode:F {
5405        \clist_set:No \l_tmpa_clist \sexampletype
5406        \tl_clear:N \l_tmpa_tl
5407        \clist_map_inline:Nn \l_tmpa_clist {
5408           \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
5409              \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
5410           }
5411        }
5412        \tl_if_empty:NTF \l_tmpa_tl {
5413           \__stex_statements_sexample_end:
5414        }{
5415           \l_tmpa_tl
5416        }
5417        \end{stex_annotate_env}
5418     }
5419  }
```

\stexpatchexample

```
5420
5421  \cs_new_protected:Nn \__stex_statements_sexample_start: {
5422     \par\noindent\titleemph{Example~\tl_if_empty:NF \sexampletitle {
5423        (\sexampletitle)
5424     }~}
5425  }
5426  \cs_new_protected:Nn \__stex_statements_sexample_end: {\par\medskip}
5427
5428  \newcommand\stexpatchexample[3][] {
5429     \str_set:Nx \l_tmpa_str{ #1 }
5430     \str_if_empty:NTF \l_tmpa_str {
5431        \tl_set:Nn \__stex_statements_sexample_start: { #2 }
5432        \tl_set:Nn \__stex_statements_sexample_end: { #3 }
5433     }{
5434        \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
5435        \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
5436     }
5437  }
```

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

\inlineex inline:

```
5438  \keys_define:nn {stex / inlineex }{
5439     type    .str_set_x:N  = \sexampletype,
5440     id      .str_set_x:N  = \sexampleid,
5441     for     .clist_set:N  = \l__stex_statements_sexample_for_clist ,
5442     name    .str_set_x:N  = \sexamplename
```

```
5443 }
5444 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
5445   \str_clear:N \sexampletype
5446   \str_clear:N \sexampleid
5447   \str_clear:N \sexamplename
5448   \clist_clear:N \l__stex_statements_sexample_for_clist
5449   \keys_set:nn { stex / inlineex }{ #1 }
5450 }
5451 \NewDocumentCommand \inlineex { O{} m } {
5452   \begingroup
5453   \stex_reactivate_macro:N \premise
5454   \stex_reactivate_macro:N \conclusion
5455   \__stex_statements_inlineex_args:n{ #1 }
5456   \str_if_empty:NF \sexampleid {
5457     \stex_ref_new_doc_target:n \sexampleid
5458   }
5459   \stex_if_smsmode:TF{
5460     \str_if_empty:NF \sexamplename {
5461       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\examplename}}
5462     }
5463   }{
5464     \seq_clear:N \l_tmpa_seq
5465     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5466       \tl_if_empty:nF{ ##1 }{
5467         \stex_get_symbol:n { ##1 }
5468         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5469           \l_stex_get_symbol_uri_str
5470         }
5471       }
5472     }
5473     \exp_args:Nnx
5474     \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpa_seq {,}}{
5475       \str_if_empty:NF \sexampletype {
5476         \stex_annotate_invisible:nnn{typestrings}{\sexampletype}{}
5477       }
5478       #2
5479       \str_if_empty:NF \sexamplename {
5480         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5481         \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}
5482       }
5483     }
5484   }
5485   \endgroup
5486   \stex_smsmode_do:
5487 }
```

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

32.4 Logical Paragraphs

sparagraph

```
5488 \keys_define:nn { stex / sparagraph} {
5489   id       .str_set_x:N   = \sparagraphid ,
```

```
5490    title   .tl_set:N     = \l_stex_sparagraph_title_tl ,
5491    type    .str_set_x:N  = \sparagraphtype ,
5492    for     .clist_set:N  = \l__stex_statements_sparagraph_for_clist ,
5493    from    .tl_set:N     = \sparagraphfrom ,
5494    to      .tl_set:N     = \sparagraphto ,
5495    start   .tl_set:N     = \l_stex_sparagraph_start_tl ,
5496    name    .str_set:N    = \sparagraphname ,
5497    imports .tl_set:N     = \l__stex_statements_sparagraph_imports_tl
5498 }
5499
5500 \cs_new_protected:Nn \stex_sparagraph_args:n {
5501   \tl_clear:N \l_stex_sparagraph_title_tl
5502   \tl_clear:N \sparagraphfrom
5503   \tl_clear:N \sparagraphto
5504   \tl_clear:N \l_stex_sparagraph_start_tl
5505   \tl_clear:N \l__stex_statements_sparagraph_imports_tl
5506   \str_clear:N \sparagraphid
5507   \str_clear:N \sparagraphtype
5508   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5509   \str_clear:N \sparagraphname
5510   \keys_set:nn { stex / sparagraph }{ #1 }
5511 }
5512 \newif\if@in@omtext\@in@omtextfalse
5513
5514 \NewDocumentEnvironment {sparagraph} { O{} } {
5515   \stex_sparagraph_args:n { #1 }
5516   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5517     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
5518   }{
5519     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
5520   }
5521   \@in@omtexttrue
5522   \stex_if_smsmode:F {
5523     \seq_clear:N \l_tmpa_seq
5524     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5525       \tl_if_empty:nF{ ##1 }{
5526         \stex_get_symbol:n { ##1 }
5527         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5528           \l_stex_get_symbol_uri_str
5529         }
5530       }
5531     }
5532     \exp_args:Nnnx
5533     \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
5534     \str_if_empty:NF \sparagraphtype {
5535       \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
5536     }
5537     \str_if_empty:NF \sparagraphfrom {
5538       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5539     }
5540     \str_if_empty:NF \sparagraphto {
5541       \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5542     }
5543     \str_if_empty:NF \sparagraphname {
```

```
5544        \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
5545      }
5546    \clist_set:No \l_tmpa_clist \sparagraphtype
5547    \tl_clear:N \l_tmpa_tl
5548    \clist_map_inline:Nn \sparagraphtype {
5549      \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
5550        \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
5551      }
5552    }
5553    \stex_csl_to_imports:No \usemodule \l__stex_statements_sparagraph_imports_tl
5554    \tl_if_empty:NTF \l_tmpa_tl {
5555      \__stex_statements_sparagraph_start:
5556    }{
5557      \l_tmpa_tl
5558    }
5559  }
5560  \clist_set:No \l_tmpa_clist \sparagraphtype
5561  \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}
5562  {
5563    \stex_reactivate_macro:N \definiendum
5564    \stex_reactivate_macro:N \definame
5565    \stex_reactivate_macro:N \Definame
5566    \stex_reactivate_macro:N \premise
5567    \stex_reactivate_macro:N \definiens
5568  }
5569  \str_if_empty:NTF \sparagraphid {
5570    \str_if_empty:NTF \sparagraphname {
5571      \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5572        \stex_ref_new_doc_target:n {}
5573      }
5574    } {
5575      \stex_ref_new_doc_target:n {}
5576    }
5577  } {
5578    \stex_ref_new_doc_target:n \sparagraphid
5579  }
5580  \exp_args:NNx
5581  \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5582    \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5583      \tl_if_empty:nF{ ##1 }{
5584        \stex_get_symbol:n { ##1 }
5585        \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5586      }
5587    }
5588  }
5589  \stex_smsmode_do:
5590  \ignorespacesandpars
5591 }{
5592  \str_if_empty:NF \sparagraphname {
5593    \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5594    \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5595  }
5596  \stex_if_smsmode:F {
5597    \clist_set:No \l_tmpa_clist \sparagraphtype
```

```
5598        \tl_clear:N \l_tmpa_tl
5599        \clist_map_inline:Nn \l_tmpa_clist {
5600          \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
5601            \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}}
5602        }
5603      }
5604      \tl_if_empty:NTF \l_tmpa_tl {
5605        \__stex_statements_sparagraph_end:
5606      }{
5607        \l_tmpa_tl
5608      }
5609      \end{stex_annotate_env}
5610    }
5611 }
```

```
5612
5613 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
5614   \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5615     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
5616       \titleemph{\l_stex_sparagraph_title_tl}:~
5617     }
5618   }{
5619     \titleemph{\l_stex_sparagraph_start_tl}~
5620   }
5621 }
5622 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
5623
5624 \newcommand\stexpatchparagraph[3][] {
5625     \str_set:Nx \l_tmpa_str{ #1 }
5626     \str_if_empty:NTF \l_tmpa_str {
5627       \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
5628       \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
5629     }{
5630       \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
5631       \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
5632     }
5633 }
5634
5635 \keys_define:nn { stex / inlinepara} {
5636   id      .str_set_x:N  = \sparagraphid ,
5637   type    .str_set_x:N  = \sparagraphtype ,
5638   for     .clist_set:N  = \l__stex_statements_sparagraph_for_clist ,
5639   from    .tl_set:N     = \sparagraphfrom ,
5640   to      .tl_set:N     = \sparagraphto ,
5641   name    .str_set:N    = \sparagraphname
5642 }
5643 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
5644   \tl_clear:N \sparagraphfrom
5645   \tl_clear:N \sparagraphto
5646   \str_clear:N \sparagraphid
5647   \str_clear:N \sparagraphtype
5648   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5649   \str_clear:N \sparagraphname
```

```
5650    \keys_set:nn { stex / inlinepara }{ #1 }
5651  }
5652  \NewDocumentCommand \inlinepara { O{} m } {
5653    \begingroup
5654    \__stex_statements_inlinepara_args:n{ #1 }
5655    \clist_set:No \l_tmpa_clist \sparagraphtype
5656    \str_if_empty:NTF \sparagraphid {
5657      \str_if_empty:NTF \sparagraphname {
5658        \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5659          \stex_ref_new_doc_target:n {}
5660        }
5661      } {
5662        \stex_ref_new_doc_target:n {}
5663      }
5664    } {
5665      \stex_ref_new_doc_target:n \sparagraphid
5666    }
5667    \stex_if_smsmode:TF{
5668      \str_if_empty:NF \sparagraphname {
5669        \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5670        \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5671      }
5672    }{
5673      \seq_clear:N \l_tmpa_seq
5674      \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5675        \tl_if_empty:nF{ ##1 }{
5676          \stex_get_symbol:n { ##1 }
5677          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5678            \l_stex_get_symbol_uri_str
5679          }
5680        }
5681      }
5682      \exp_args:Nnx
5683      \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}{
5684        \str_if_empty:NF \sparagraphtype {
5685          \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
5686        }
5687        \str_if_empty:NF \sparagraphfrom {
5688          \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5689        }
5690        \str_if_empty:NF \sparagraphto {
5691          \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5692        }
5693        \str_if_empty:NF \sparagraphname {
5694          \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5695          \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
5696          \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5697        }
5698        \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5699          \clist_map_inline:Nn \l_tmpa_seq {
5700            \stex_ref_new_sym_target:n {##1}
5701          }
5702        }
5703        #2
```

```
5704        }
5705    }
5706    \endgroup
5707    \stex_smsmode_do:
5708 }
5709
```

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

```
5710 ⟨/package⟩
```

Chapter 33

The Implementation

```
5711 ⟨*package⟩
5712 ⟨@@=stex_sproof⟩
5713
5714 %%%%%%%%%%%%    sproof.dtx    %%%%%%%%%%%%
5715
```

33.1   Proofs

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

```
5716 \keys_define:nn { stex / spf } {
5717    id            .str_set_x:N  = \spfid,
5718    for           .clist_set:N  = \l__stex_sproof_spf_for_clist ,
5719    from          .tl_set:N     = \l__stex_sproof_spf_from_tl ,
5720    proofend      .tl_set:N     = \l__stex_sproof_spf_proofend_tl,
5721    type          .str_set_x:N  = \spftype,
5722    title         .tl_set:N     = \spftitle,
5723    continues     .tl_set:N     = \l__stex_sproof_spf_continues_tl,
5724    functions     .tl_set:N     = \l__stex_sproof_spf_functions_tl,
5725    method        .tl_set:N     = \l__stex_sproof_spf_method_tl
5726 }
5727 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
5728 \str_clear:N \spfid
5729 \tl_clear:N \l__stex_sproof_spf_for_tl
5730 \tl_clear:N \l__stex_sproof_spf_from_tl
5731 \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
5732 \str_clear:N \spftype
5733 \tl_clear:N \spftitle
5734 \tl_clear:N \l__stex_sproof_spf_continues_tl
5735 \tl_clear:N \l__stex_sproof_spf_functions_tl
5736 \tl_clear:N \l__stex_sproof_spf_method_tl
5737    \bool_set_false:N \l__stex_sproof_inc_counter_bool
5738 \keys_set:nn { stex / spf }{ #1 }
5739 }
```

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

```
5740 \str_set:Nn\c__stex_sproof_flow_str{inline}
```

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

```
5741 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
5742 \cs_new_protected:Npn \sproofnumber {
5743   \int_set:Nn \l_tmpa_int {1}
5744   \bool_while_do:nn {
5745     \int_compare_p:nNn {
5746       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5747     } > 0
5748   }{
5749     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
5750     \int_incr:N \l_tmpa_int
5751   }
5752 }
5753 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
5754   \int_set:Nn \l_tmpa_int {1}
5755   \bool_while_do:nn {
5756     \int_compare_p:nNn {
5757       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5758     } > 0
5759   }{
5760     \int_incr:N \l_tmpa_int
5761   }
5762   \int_compare:nNnF \l_tmpa_int = 1 {
5763     \int_decr:N \l_tmpa_int
5764   }
5765   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
5766     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1
5767   }
5768 }
5769
5770 \cs_new_protected:Npn \__stex_sproof_add_counter: {
5771   \int_set:Nn \l_tmpa_int {1}
5772   \bool_while_do:nn {
5773     \int_compare_p:nNn {
5774       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5775     } > 0
5776   }{
5777     \int_incr:N \l_tmpa_int
5778   }
5779   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 1 }
5780 }
5781
5782 \cs_new_protected:Npn \__stex_sproof_remove_counter: {
5783   \int_set:Nn \l_tmpa_int {1}
5784   \bool_while_do:nn {
```

```
5785        \int_compare_p:nNn {
5786          \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5787        } > 0
5788      }{
5789        \int_incr:N \l_tmpa_int
5790      }
5791      \int_decr:N \l_tmpa_int
5792      \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 0 }
5793    }
```

\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

```
5794    \def\sproof@box{
5795      \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
5796    }
5797    \def\sproofend{
5798      \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
5799        \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
5800      }
5801    }
```

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

spf@*@kw

```
5802    \def\spf@proofsketch@kw{Proof~Sketch}
5803    \def\spf@proof@kw{Proof}
5804    \def\spf@step@kw{Step}
```

(*End definition for* spf@*@kw. *This function is documented on page* **??**.)
      For the other languages, we set up triggers

```
5805    \AddToHook{begindocument}{
5806      \ltx@ifpackageloaded{babel}{
5807        \makeatletter
5808        \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5809        \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5810          \input{sproof-ngerman.ldf}
5811        }
5812        \clist_if_in:NnT \l_tmpa_clist {finnish}{
5813          \input{sproof-finnish.ldf}
5814        }
5815        \clist_if_in:NnT \l_tmpa_clist {french}{
5816          \input{sproof-french.ldf}
5817        }
5818        \clist_if_in:NnT \l_tmpa_clist {russian}{
5819          \input{sproof-russian.ldf}
5820        }
5821        \makeatother
5822      }{}
5823    }
```

spfsketch

```
5824    \newcommand\spfsketch[2][]{
5825      \begingroup
5826      \let \premise \stex_proof_premise:
```

```
5827        \__stex_sproof_spf_args:n{#1}
5828        \stex_if_smsmode:TF {
5829          \str_if_empty:NF \spfid {
5830            \stex_ref_new_doc_target:n \spfid
5831          }
5832        }{
5833          \seq_clear:N \l_tmpa_seq
5834          \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5835            \tl_if_empty:nF{ ##1 }{
5836              \stex_get_symbol:n { ##1 }
5837              \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5838                \l_stex_get_symbol_uri_str
5839              }
5840            }
5841          }
5842          \exp_args:Nnx
5843          \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
5844            \str_if_empty:NF \spftype {
5845              \stex_annotate_invisible:nnn{type}{\spftype}{}
5846            }
5847            \clist_set:No \l_tmpa_clist \spftype
5848            \tl_set:Nn \l_tmpa_tl {
5849              \titleemph{
5850                \tl_if_empty:NTF \spftitle {
5851                  \spf@proofsketch@kw
5852                }{
5853                  \spftitle
5854                }
5855              }:~
5856            }
5857            \clist_map_inline:Nn \l_tmpa_clist {
5858              \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5859                \tl_clear:N \l_tmpa_tl
5860              }
5861            }
5862            \str_if_empty:NF \spfid {
5863              \stex_ref_new_doc_target:n \spfid
5864            }
5865            \l_tmpa_tl #2 \sproofend
5866          }
5867        }
5868        \endgroup
5869        \stex_smsmode_do:
5870    }
5871
```

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

spfeq   This is very similar to \spfsketch, but uses a computation array[14][15]

```
5872    \newenvironment{spfeq}[2][]{
5873        \__stex_sproof_spf_args:n{#1}
```

---

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

[15]EDNOTE: document above

```
5874    \let \premise \stex_proof_premise:
5875    \stex_if_smsmode:TF {
5876      \str_if_empty:NF \spfid {
5877        \stex_ref_new_doc_target:n \spfid
5878      }
5879    }{
5880      \seq_clear:N \l_tmpa_seq
5881      \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5882        \tl_if_empty:nF{ ##1 }{
5883          \stex_get_symbol:n { ##1 }
5884          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5885            \l_stex_get_symbol_uri_str
5886          }
5887        }
5888      }
5889      \exp_args:Nnnx
5890      \begin{stex_annotate_env}{spfeq}{\seq_use:Nn \l_tmpa_seq {,}}
5891      \str_if_empty:NF \spftype {
5892        \stex_annotate_invisible:nnn{type}{\spftype}{}
5893      }
5894
5895      \clist_set:No \l_tmpa_clist \spftype
5896      \tl_clear:N \l_tmpa_tl
5897      \clist_map_inline:Nn \l_tmpa_clist {
5898        \tl_if_exist:cT {__stex_sproof_spfeq_##1_start:}{
5899          \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_start:}}
5900        }
5901        \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5902          \tl_set:Nn \l_tmpa_tl {\use:n{}}
5903        }
5904      }
5905      \tl_if_empty:NTF \l_tmpa_tl {
5906        \__stex_sproof_spfeq_start:
5907      }{
5908        \l_tmpa_tl
5909      }{~#2}
5910      \str_if_empty:NF \spfid {
5911        \stex_ref_new_doc_target:n \spfid
5912      }
5913      \begin{displaymath}\begin{array}{rcll}
5914    }
5915    \stex_smsmode_do:
5916  }{
5917    \stex_if_smsmode:F {
5918      \end{array}\end{displaymath}
5919      \clist_set:No \l_tmpa_clist \spftype
5920      \tl_clear:N \l_tmpa_tl
5921      \clist_map_inline:Nn \l_tmpa_clist {
5922        \tl_if_exist:cT {__stex_sproof_spfeq_##1_end:}{
5923          \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_end:}}
5924        }
5925      }
5926      \tl_if_empty:NTF \l_tmpa_tl {
5927        \__stex_sproof_spfeq_end:
```

```
5928      }{
5929        \l_tmpa_tl
5930      }
5931      \end{stex_annotate_env}
5932    }
5933  }
5934
5935  \cs_new_protected:Nn \__stex_sproof_spfeq_start: {
5936    \titleemph{
5937      \tl_if_empty:NTF \spftitle {
5938        \spf@proof@kw
5939      }{
5940        \spftitle
5941      }
5942    }:
5943  }
5944  \cs_new_protected:Nn \__stex_sproof_spfeq_end: {\sproofend}
5945
5946  \newcommand\stexpatchspfeq[3][] {
5947      \str_set:Nx \l_tmpa_str{ #1 }
5948      \str_if_empty:NTF \l_tmpa_str {
5949        \tl_set:Nn \__stex_sproof_spfeq_start: { #2 }
5950        \tl_set:Nn \__stex_sproof_spfeq_end: { #3 }
5951      }{
5952        \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_start:\endcsname{ #2 }
5953        \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_end:\endcsname{ #3 }
5954      }
5955  }
5956
```

(*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.

```
5957  \newenvironment{sproof}[2][]{
5958    \let \premise \stex_proof_premise:
5959    \intarray_gzero:N \l__stex_sproof_counter_intarray
5960    \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
5961    \__stex_sproof_spf_args:n{#1}
5962    \stex_if_smsmode:TF {
5963      \str_if_empty:NF \spfid {
5964        \stex_ref_new_doc_target:n \spfid
5965      }
5966    }{
5967      \seq_clear:N \l_tmpa_seq
5968      \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5969        \tl_if_empty:nF{ ##1 }{
5970          \stex_get_symbol:n { ##1 }
5971          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5972            \l_stex_get_symbol_uri_str
5973          }
5974        }
5975      }
```

221

```
5976        \exp_args:Nnnx
5977        \begin{stex_annotate_env}{sproof}{\seq_use:Nn \l_tmpa_seq {,}}
5978        \str_if_empty:NF \spftype {
5979          \stex_annotate_invisible:nnn{type}{\spftype}{}
5980        }
5981
5982        \clist_set:No \l_tmpa_clist \spftype
5983        \tl_clear:N \l_tmpa_tl
5984        \clist_map_inline:Nn \l_tmpa_clist {
5985          \tl_if_exist:cT {__stex_sproof_sproof_##1_start:}{
5986            \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_start:}}
5987          }
5988          \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5989            \tl_set:Nn \l_tmpa_tl {\use:n{}}
5990          }
5991        }
5992        \tl_if_empty:NTF \l_tmpa_tl {
5993          \__stex_sproof_sproof_start:
5994        }{
5995          \l_tmpa_tl
5996        }{~#2}
5997        \str_if_empty:NF \spfid {
5998          \stex_ref_new_doc_target:n \spfid
5999        }
6000        \begin{description}
6001      }
6002      \stex_smsmode_do:
6003    }{
6004      \stex_if_smsmode:F{
6005        \end{description}
6006        \clist_set:No \l_tmpa_clist \spftype
6007        \tl_clear:N \l_tmpa_tl
6008        \clist_map_inline:Nn \l_tmpa_clist {
6009          \tl_if_exist:cT {__stex_sproof_sproof_##1_end:}{
6010            \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_end:}}
6011          }
6012        }
6013        \tl_if_empty:NTF \l_tmpa_tl {
6014          \__stex_sproof_sproof_end:
6015        }{
6016          \l_tmpa_tl
6017        }
6018        \end{stex_annotate_env}
6019      }
6020  }
6021
6022  \cs_new_protected:Nn \__stex_sproof_sproof_start: {
6023    \par\noindent\titleemph{
6024      \tl_if_empty:NTF \spftype {
6025        \spf@proof@kw
6026      }{
6027        \spftype
6028      }
6029    }:
```

222

```
6030  }
6031  \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
6032
6033  \newcommand\stexpatchproof[3][] {
6034    \str_set:Nx \l_tmpa_str{ #1 }
6035    \str_if_empty:NTF \l_tmpa_str {
6036      \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
6037      \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
6038    }{
6039      \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
6040      \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
6041    }
6042  }
```

```
6043  \newcommand\spfidea[2][]{
6044    \__stex_sproof_spf_args:n{#1}
6045    \titleemph{
6046      \tl_if_empty:NTF \spftype {Proof~Idea}{
6047        \spftype
6048      }:
6049    }~#2
6050    \sproofend
6051  }
```

(*End definition for* \spfidea. *This function is documented on page* *44*.)

The next two environments (proof steps) and comments, are mostly semantical, they take KeyVal arguments that specify their semantic role. In draft mode, they read these values and show them. If the surrounding proof had display=flow, then no new \item is generated, otherwise it is. In any case, the proof step number (at the current level) is incremented.

spfstep

```
6052  \newenvironment{spfstep}[1][]{
6053    \__stex_sproof_spf_args:n{#1}
6054    \stex_if_smsmode:TF {
6055      \str_if_empty:NF \spfid {
6056        \stex_ref_new_doc_target:n \spfid
6057      }
6058    }{
6059      \@in@omtexttrue
6060      \seq_clear:N \l_tmpa_seq
6061      \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6062        \tl_if_empty:nF{ ##1 }{
6063          \stex_get_symbol:n { ##1 }
6064          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6065            \l_stex_get_symbol_uri_str
6066          }
6067        }
6068      }
6069      \exp_args:Nnnx
6070      \begin{stex_annotate_env}{spfstep}{\seq_use:Nn \l_tmpa_seq {,}}
6071      \str_if_empty:NF \spftype {
6072        \stex_annotate_invisible:nnn{type}{\spftype}{}
```

223

```
6073      }
6074    \clist_set:No \l_tmpa_clist \spftype
6075    \tl_set:Nn \l_tmpa_tl {
6076      \item[\sproofnumber]
6077      \bool_set_true:N \l__stex_sproof_inc_counter_bool
6078    }
6079    \clist_map_inline:Nn \l_tmpa_clist {
6080      \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6081        \tl_clear:N \l_tmpa_tl
6082      }
6083    }
6084    \l_tmpa_tl
6085    \tl_if_empty:NF \spftitle {
6086      {(\titleemph{\spftitle})\enspace}
6087    }
6088    \str_if_empty:NF \spfid {
6089      \stex_ref_new_doc_target:n \spfid
6090    }
6091  }
6092  \stex_smsmode_do:
6093  \ignorespacesandpars
6094 }{
6095  \bool_if:NT \l__stex_sproof_inc_counter_bool {
6096    \__stex_sproof_inc_counter:
6097  }
6098  \stex_if_smsmode:F {
6099    \end{stex_annotate_env}
6100  }
6101 }
```

```
6102 \newenvironment{spfcomment}[1][]{
6103   \__stex_sproof_spf_args:n{#1}
6104   \clist_set:No \l_tmpa_clist \spftype
6105   \tl_set:Nn \l_tmpa_tl {
6106     \item[\sproofnumber]
6107     \bool_set_true:N \l__stex_sproof_inc_counter_bool
6108   }
6109   \clist_map_inline:Nn \l_tmpa_clist {
6110     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6111       \tl_clear:N \l_tmpa_tl
6112     }
6113   }
6114   \l_tmpa_tl
6115 }{
6116   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6117     \__stex_sproof_inc_counter:
6118   }
6119 }
```

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.

224

```
6120  \newenvironment{subproof}[2][]{
6121    \__stex_sproof_spf_args:n{#1}
6122    \stex_if_smsmode:TF{
6123      \str_if_empty:NF \spfid {
6124        \stex_ref_new_doc_target:n \spfid
6125      }
6126    }{
6127      \seq_clear:N \l_tmpa_seq
6128      \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6129        \tl_if_empty:nF{ ##1 }{
6130          \stex_get_symbol:n { ##1 }
6131          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6132            \l_stex_get_symbol_uri_str
6133          }
6134        }
6135      }
6136      \exp_args:Nnnx
6137      \begin{stex_annotate_env}{subproof}{\seq_use:Nn \l_tmpa_seq {,}}
6138      \str_if_empty:NF \spftype {
6139        \stex_annotate_invisible:nnn{type}{\spftype}{}
6140      }
6141
6142      \clist_set:No \l_tmpa_clist \spftype
6143      \tl_set:Nn \l_tmpa_tl {
6144        \item[\sproofnumber]
6145        \bool_set_true:N \l__stex_sproof_inc_counter_bool
6146      }
6147      \clist_map_inline:Nn \l_tmpa_clist {
6148        \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6149          \tl_clear:N \l_tmpa_tl
6150        }
6151      }
6152      \l_tmpa_tl
6153      \tl_if_empty:NF \spftitle {
6154        {(\titleemph{\spftitle})\enspace}
6155      }
6156      {~#2}
6157      \str_if_empty:NF \spfid {
6158        \stex_ref_new_doc_target:n \spfid
6159      }
6160    }
6161    \__stex_sproof_add_counter:
6162    \stex_smsmode_do:
6163  }{
6164    \__stex_sproof_remove_counter:
6165    \bool_if:NT \l__stex_sproof_inc_counter_bool {
6166      \__stex_sproof_inc_counter:
6167    }
6168    \stex_if_smsmode:F{
6169      \end{stex_annotate_env}
6170    }
6171  }
```

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

```
6172  \newenvironment{spfcases}[2][]{
6173    \tl_if_empty:nTF{#1}{
6174      \begin{subproof}[method=by-cases]{#2}
6175    }{
6176      \begin{subproof}[#1,method=by-cases]{#2}
6177    }
6178  }{
6179    \end{subproof}
6180  }
```

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

```
6181  \newenvironment{spfcase}[2][]{
6182    \__stex_sproof_spf_args:n{#1}
6183    \stex_if_smsmode:TF {
6184      \str_if_empty:NF \spfid {
6185        \stex_ref_new_doc_target:n \spfid
6186      }
6187    }{
6188      \seq_clear:N \l_tmpa_seq
6189      \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6190        \tl_if_empty:nF{ ##1 }{
6191          \stex_get_symbol:n { ##1 }
6192          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6193            \l_stex_get_symbol_uri_str
6194          }
6195        }
6196      }
6197      \exp_args:Nnnx
6198      \begin{stex_annotate_env}{spfcase}{\seq_use:Nn \l_tmpa_seq {,}}
6199      \str_if_empty:NF \spftype {
6200        \stex_annotate_invisible:nnn{type}{\spftype}{}
6201      }
6202      \clist_set:No \l_tmpa_clist \spftype
6203      \tl_set:Nn \l_tmpa_tl {
6204        \item[\sproofnumber]
6205        \bool_set_true:N \l__stex_sproof_inc_counter_bool
6206      }
6207      \clist_map_inline:Nn \l_tmpa_clist {
6208        \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6209          \tl_clear:N \l_tmpa_tl
6210        }
6211      }
6212      \l_tmpa_tl
6213      \tl_if_empty:nF{#2}{
6214        \titleemph{#2}:~
6215      }
6216    }
6217    \__stex_sproof_add_counter:
6218    \stex_smsmode_do:
6219  }{
6220    \__stex_sproof_remove_counter:
6221    \bool_if:NT \l__stex_sproof_inc_counter_bool {
6222      \__stex_sproof_inc_counter:
```

```
6223    }
6224    \stex_if_smsmode:F{
6225      \clist_set:No \l_tmpa_clist \spftype
6226      \tl_set:Nn \l_tmpa_tl{\sproofend}
6227      \clist_map_inline:Nn \l_tmpa_clist {
6228        \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6229          \tl_clear:N \l_tmpa_tl
6230        }
6231      }
6232      \l_tmpa_tl
6233      \end{stex_annotate_env}
6234    }
6235  }
```

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

```
6236  \newcommand\spfcasesketch[3][]{
6237    \begin{spfcase}[#1]{#2}#3\end{spfcase}
6238  }
```

33.2   Justifications

We define the actions that are undertaken, when the keys for justifications are encountered. Here this is very simple, we just define an internal macro with the value, so that we can use it later.

```
6239  \keys_define:nn { stex / just }{
6240    id        .str_set_x:N  = \l__stex_sproof_just_id_str,
6241    method    .tl_set:N     = \l__stex_sproof_just_method_tl,
6242    premises  .tl_set:N     = \l__stex_sproof_just_premises_tl,
6243    args      .tl_set:N     = \l__stex_sproof_just_args_tl
6244  }
```

EdN:16

The next three environments and macros are purely semantic, so we ignore the keyval arguments for now and only display the content.[16]

\spfjust

```
6245  \newcommand\spfjust[1][]{}
```

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

\premise

```
6246  \newcommand\stex_proof_premise:[2][]{#2}
```

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

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

```
6247  \newcommand\justarg[2][]{#2}
6248  ⟨/package⟩
```

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

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

---

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

Chapter 34

sTEX
-Others Implementation

```
6249 ⟨∗package⟩
6250
6251 %%%%%%%%%%%   others.dtx   %%%%%%%%%%%%
6252
6253 ⟨@@=stex_others⟩
```

Warnings and error messages

```
6254   % None
```

Math subject classifier

```
6255 \NewDocumentCommand \MSC {m} {
6256   % TODO
6257 }
```

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

Patching tikzinput, if loaded

```
6258 \@ifpackageloaded{tikzinput}{
6259   \RequirePackage{stex-tikzinput}
6260 }{}
6261
6262 \bool_if:NT \c_stex_persist_mode_bool {
6263   \input{\jobname.sms}
6264   \prop_if_exist:NT\c_stex_mathhub_main_manifest_prop{
6265     \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
6266       \l_tmpa_str
6267     \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str _manifest_prop }
6268       \c_stex_mathhub_main_manifest_prop
6269     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
6270   }
6271 }
6272 ⟨/package⟩
```

Chapter 35

sTeX -Metatheory Implementation

```
6273  ⟨∗package⟩
6274  ⟨@@=stex_modules⟩
6275
6276  %%%%%%%%%%%%   metatheory.dtx   %%%%%%%%%%%%
6277
6278  \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX/meta}
6279  \begingroup
6280  \stex_module_setup:nn{
6281    ns=\c_stex_metatheory_ns_str,
6282    meta=NONE
6283  }{Metatheory}
6284  \stex_reactivate_macro:N \symdecl
6285  \stex_reactivate_macro:N \notation
6286  \stex_reactivate_macro:N \symdef
6287  \ExplSyntaxOff
6288  \csname stex_suppress_html:n\endcsname{
6289    % is-a (a:A, a \in A, a is an A, etc.)
6290    \symdecl{isa}[args=ai]
6291    \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
6292    \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
6293    \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
6294
6295    % bind (\forall, \Pi, \lambda etc.)
6296    \symdecl{bind}[args=Bi]
6297    \notation{bind}[forall]{\comp\forall #1.\;#2}{##1 \comp, ##2}
6298    \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6299    \notation{bind}[depfun]{\comp( #1 \comp()\;\to\;} #2}{##1 \comp, ##2}
6300
6301    % implicit bind
6302    \symdef{implicitbind}[args=Bi]{\comp\prod_{#1}#2}{##1\comp,##2}
6303
6304    % dummy variable
6305    \symdecl{dummyvar}
6306    \notation{dummyvar}[underscore]{\comp\_}
6307    \notation{dummyvar}[dot]{\comp\cdot}
```

229

```
6308    \notation{dummyvar}[dash]{\comp{{\rm --}}}}

6309

6310    %fromto (function space, Hom-set, implication etc.)
6311    \symdecl{fromto}[args=ai]
6312    \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
6313    \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}

6314

6315    % mapto (lambda etc.)
6316    %\symdecl{mapto}[args=Bi]
6317    %\notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
6318    %\notation{mapto}[lambda]{\comp\lambda #1 \comp.\; #2}{#1 \comp, #2}
6319    %\notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.\; #2}{#1 \comp, #2}

6320

6321    % function/operator application
6322    \symdecl{apply}[args=ia]
6323    \notation{apply}[prec=0;0x\infprec,parens]{#1 \comp( #2 \comp)}{##1 \comp, ##2}
6324    \notation{apply}[prec=0;0x\infprec,lambda]{#1 \; #2 }{##1 \; ##2}

6325

6326    % collection of propositions/booleans/truth values
6327    \symdecl{prop}[name=proposition]
6328    \notation{prop}[prop]{\comp{{\rm prop}}}
6329    \notation{prop}[BOOL]{\comp{{\rm BOOL}}}

6330

6331    \symdecl{judgmentholds}[args=1]
6332    \notation{judgmentholds}[vdash,op=\vdash]{\comp\vdash\; #1}

6333

6334    % sequences
6335    \symdecl{seqtype}[args=1]
6336    \notation{seqtype}[kleene]{#1^{\comp\ast}}

6337

6338    \symdecl{seqexpr}[args=a]
6339    \notation{seqexpr}[angle,prec=nobrackets]{\comp\langle #1\comp\rangle}{##1\comp,##2}

6340

6341    \symdef{seqmap}[args=abi,setlike]{\comp\{#3 \comp| #2\comp\in \dobrackets{#1} \comp\}}{##1
6342    \symdef{seqprepend}[args=ia]{#1 \comp{::} #2}{##1 \comp, ##2}
6343    \symdef{seqappend}[args=ai]{#1 \comp{::} #2}{##1 \comp, ##2}
6344    \symdef{seqfoldleft}[args=iabbi]{ \comp{foldl}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\com
6345    \symdef{seqfoldright}[args=iabbi,op=foldr]{ \comp{foldr}\dobrackets{#1,#2}\dobrackets{#3\c
6346    \symdef{seqhead}[args=a]{\comp{head}\dobrackets{#1}}{##1 \comp, ##2}
6347    \symdef{seqtail}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6348    \symdef{seqlast}[args=a]{\comp{last}\dobrackets{#1}}{##1 \comp, ##2}
6349    \symdef{seqinit}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}

6350

6351    \symdef{sequence-index}[args=2,li,prec=nobrackets]{{#1}_{#2}}
6352    \notation{sequence-index}[ui,prec=nobrackets]{{#1}^{#2}}

6353

6354    \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{,\ellipses}}{##1\comp,##2}
6355    \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{,\ellipses,}#2}{##1\comp,##2}
6356    \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{,\ellipses,}#2\comp{,\ellipses,}#

6357

6358    % letin (``let'', local definitions, variable substitution)
6359    \symdecl{letin}[args=bii]
6360    \notation{letin}[let]{\comp{{\rm let}}\;#1\comp{=}#2\;\comp{{\rm in}}\;#3}
6361    \notation{letin}[subst]{#3 \comp[ #1 \comp/ #2 \comp]}
```

230

```
6362    \notation{letin}[frac]{#3 \comp[ \frac{#2}{#1} \comp]}

6363
6364    % structures
6365    \symdecl*{module-type}[args=1]
6366    \notation{module-type}{\comp{\mathtt{MOD}} #1}
6367    \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
6368    \notation{mathstruct}[angle,prec=nobrackets]{\comp\langle #1 \comp\rangle}{##1 \comp, ##2

6369
6370    % objects
6371    \symdecl{object}
6372    \notation{object}{\comp{\mathtt{OBJECT}}}

6373
6374 }

6375
6376 % The following are abbreviations in the sTeX corpus that are left over from earlier
6377 % developments. They will eventually be phased out.

6378
6379    \ExplSyntaxOn
6380    \stex_add_to_current_module:n{
6381      \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
6382      \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
6383      \def\livar{\csname sequence-index\endcsname[li]}
6384      \def\uivar{\csname sequence-index\endcsname[ui]}
6385      \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
6386      \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
6387    }
6388 \__stex_modules_end_module:
6389 \endgroup

6390 ⟨/package⟩
```

Chapter 36

Tikzinput Implementation

```
6391  ⟨@@=tikzinput⟩
6392  ⟨∗package⟩
6393
6394  %%%%%%%%%%%  tikzinput.dtx   %%%%%%%%%%%
6395
6396  \ProvidesExplPackage{tikzinput}{2022/02/26}{3.0.1}{tikzinput package}
6397  \RequirePackage{l3keys2e}
6398
6399  \keys_define:nn { tikzinput } {
6400    image    .bool_set:N   = \c_tikzinput_image_bool,
6401    image    .default:n    = false ,
6402    unknown    .code:n       = {}
6403  }
6404
6405  \ProcessKeysOptions { tikzinput }
6406
6407  \bool_if:NTF \c_tikzinput_image_bool {
6408    \RequirePackage{graphicx}
6409
6410    \providecommand\usetikzlibrary[]{}
6411    \newcommand\tikzinput[2][]{\includegraphics[#1]{#2}}
6412  }{
6413    \RequirePackage{tikz}
6414    \RequirePackage{standalone}
6415
6416    \newcommand \tikzinput [2] [] {
6417      \setkeys{Gin}{#1}
6418      \ifx \Gin@ewidth \Gin@exclamation
6419        \ifx \Gin@eheight \Gin@exclamation
6420          \input { #2 }
6421        \else
6422          \resizebox{!}{ \Gin@eheight }{
6423            \input { #2 }
6424          }
6425        \fi
6426      \else
6427        \ifx \Gin@eheight \Gin@exclamation
6428          \resizebox{ \Gin@ewidth }{!}{
```

```
6429            \input { #2 }
6430          }
6431        \else
6432          \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
6433            \input { #2 }
6434          }
6435        \fi
6436      \fi
6437    }
6438  }

6439
6440  \newcommand \ctikzinput [2] [] {
6441    \begin{center}
6442      \tikzinput [#1] {#2}
6443    \end{center}
6444  }

6445
6446  \@ifpackageloaded{stex}{
6447    \RequirePackage{stex-tikzinput}
6448  }{}

6449
6450  ⟨/package⟩
6451  ⟨∗stex⟩

6452  \ProvidesExplPackage{stex-tikzinput}{2022/02/26}{3.0.1}{stex-tikzinput}
6453  \RequirePackage{stex}
6454  \RequirePackage{tikzinput}

6455
6456  \newcommand\mhtikzinput[2][]{%
6457    \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
6458    \stex_in_repository:nn\Gin@mhrepos{
6459      \tikzinput[#1]{\mhpath{##1}{#2}}
6460    }
6461  }
6462  \newcommand\cmhtikzinput[2][]{\begin{center}\mhtikzinput[#1]{#2}\end{center}}

6463
6464  \cs_new_protected:Nn \__tikzinput_usetikzlibrary:nn {
6465    \pgfkeys@spdef\pgf@temp{#1}
6466    \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
6467    \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgfut
6468    \expandafter\edef\csname tikz@library@#1atcode\endcsname{\the\catcode`\@}
6469    \expandafter\edef\csname tikz@library@#1barcode\endcsname{\the\catcode`\|}
6470    \expandafter\edef\csname tikz@library@#1dollarcode\endcsname{\the\catcode`\$}
6471    \catcode`\@=11
6472    \catcode`\|=12
6473    \catcode`\$=3
6474    \pgfutil@InputIfFileExists{#2}{}{}
6475    \catcode`\@=\csname tikz@library@#1atcode\endcsname
6476    \catcode`\|=\csname tikz@library@#1barcode\endcsname
6477    \catcode`\$=\csname tikz@library@#1dollarcode\endcsname
6478  }

6479

6480
6481  \newcommand\libusetikzlibrary[1]{
```

```
6482    \prop_if_exist:NF \l_stex_current_repository_prop {
6483      \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6484    }
6485    \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
6486      \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6487    }
6488    \seq_clear:N \l__tikzinput_libinput_files_seq
6489    \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
6490    \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
6491
6492    \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
6493      \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / tikzlibra
6494      \IfFileExists{ \l_tmpa_str }{
6495        \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6496      }{}
6497      \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
6498      \seq_put_right:No \l_tmpa_seq \l_tmpa_str
6499    }
6500
6501    \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / tikzlibrary #1 .code.t
6502    \IfFileExists{ \l_tmpa_str }{
6503      \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6504    }{}
6505
6506    \seq_if_empty:NTF \l__tikzinput_libinput_files_seq {
6507      \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .code.t
6508    }{
6509      \int_compare:nNnTF {\seq_count:N \l__tikzinput_libinput_files_seq} = 1 {
6510        \seq_map_inline:Nn \l__tikzinput_libinput_files_seq {
6511          \__tikzinput_usetikzlibrary:nn{#1}{ ##1 }
6512        }
6513      }{
6514        \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .co
6515      }
6516    }
6517 }
6518 ⟨/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

6519 ⟨∗package⟩
6520 ⟨@@=document_structure⟩
6521 \ProvidesExplPackage{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure}
6522 \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).

```
6523
6524 \keys_define:nn{ document-structure }{
6525   class       .str_set_x:N  = \c_document_structure_class_str,
6526   topsect     .str_set_x:N  = \c_document_structure_topsect_str,,
6527   unknown     .code:n       = {
6528     \PassOptionsToClass{\CurrentOption}{stex}
6529     \PassOptionsToClass{\CurrentOption}{tikzinput}
6530   }
6531 %  showignores .bool_set:N   = \c_document_structure_showignores_bool,
6532 }
6533 \ProcessKeysOptions{ document-structure }
6534 \str_if_empty:NT \c_document_structure_class_str {
6535   \str_set:Nn \c_document_structure_class_str {article}
6536 }
6537 \str_if_empty:NT \c_document_structure_topsect_str {
6538   \str_set:Nn \c_document_structure_topsect_str {section}
6539 }
```

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

```
6540 \RequirePackage{xspace}
6541 \RequirePackage{comment}
6542 \RequirePackage{stex}
6543 \AddToHook{begindocument}{
```

```
6544  \ltx@ifpackageloaded{babel}{
6545      \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6546      \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6547          \makeatletter\input{document-structure-ngerman.ldf}\makeatother
6548      }
6549  }{}
6550 }
```

\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.

```
6551 \int_new:N \l_document_structure_section_level_int
6552 \str_case:VnF \c_document_structure_topsect_str {
6553   {part}{
6554       \int_set:Nn \l_document_structure_section_level_int {0}
6555   }
6556   {chapter}{
6557       \int_set:Nn \l_document_structure_section_level_int {1}
6558   }
6559 }{
6560   \str_case:VnF \c_document_structure_class_str {
6561     {book}{
6562         \int_set:Nn \l_document_structure_section_level_int {0}
6563     }
6564     {report}{
6565         \int_set:Nn \l_document_structure_section_level_int {0}
6566     }
6567   }{
6568     \int_set:Nn \l_document_structure_section_level_int {2}
6569   }
6570 }
```

37.2   Document Structure

The structure of the document is given by the sfragment environment. The hierarchy is adjusted automatically according to the LaTeX class in effect.

\currentsectionlevel   For the \currentsectionlevel and \Currentsectionlevel macros we use an internal macro \current@section@level that only contains the keyword (no markup). We initialize it with "document" as a default. In the generated OMDoc, we only generate a text

EdN:17   element of class omdoc_currentsectionlevel, wich will be instantiated by CSS later.[17]

```
6571 \def\current@section@level{document}%
6572 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
6573 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

\skipfragment

```
6574 \cs_new_protected:Npn \skipfragment {
```

---

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

```
6575    \ifcase\l_document_structure_section_level_int
6576    \or\stepcounter{part}
6577    \or\stepcounter{chapter}
6578    \or\stepcounter{section}
6579    \or\stepcounter{subsection}
6580    \or\stepcounter{subsubsection}
6581    \or\stepcounter{paragraph}
6582    \or\stepcounter{subparagraph}
6583    \fi
6584 }
```

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

blindfragment

```
6585 \newcommand\at@begin@blindsfragment[1]{}
6586 \newenvironment{blindfragment}
6587 {
6588    \int_incr:N\l_document_structure_section_level_int
6589    \at@begin@blindsfragment\l_document_structure_section_level_int
6590 }{}
```

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

```
6591 \newcommand\sfragment@nonum[2]{
6592    \ifx\hyper@anchor\@undefined\else\phantomsection\fi
6593    \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
6594 }
```

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

\sfragment@num   convenience macro: `\sfragment@nonum{⟨level⟩}{⟨title⟩}` makes numbered sectioning
with title ⟨*title*⟩ at level ⟨*level*⟩. We have to check the `short` key was given in the
`sfragment` environment and – if it is use it. But how to do that depends on whether
the `rdfmeta` package has been loaded. In the end we call `\sref@label@id` to enable
crossreferencing.

```
6595 \newcommand\sfragment@num[2]{
6596    \tl_if_empty:NTF \l__document_structure_sfragment_short_tl {
6597        \@nameuse{#1}{#2}
6598    }{
6599        \cs_if_exist:NTF\rdfmeta@sectioning{
6600            \@nameuse{rdfmeta@#1@old}[\l__document_structure_sfragment_short_tl]{#2}
6601        }{
6602            \@nameuse{#1}[\l__document_structure_sfragment_short_tl]{#2}
6603        }
6604    }
6605 %\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\sfragment@id
6606 }
```

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

sfragment

```
6607 \keys_define:nn { document-structure / sfragment }{
6608    id              .str_set_x:N = \l__document_structure_sfragment_id_str,
6609    date            .str_set_x:N = \l__document_structure_sfragment_date_str,
```

237

```
6610    creators      .clist_set:N = \l__document_structure_sfragment_creators_clist,
6611    contributors  .clist_set:N = \l__document_structure_sfragment_contributors_clist,
6612    srccite       .tl_set:N    = \l__document_structure_sfragment_srccite_tl,
6613    type          .tl_set:N    = \l__document_structure_sfragment_type_tl,
6614    short         .tl_set:N    = \l__document_structure_sfragment_short_tl,
6615    display       .tl_set:N    = \l__document_structure_sfragment_display_tl,
6616    intro         .tl_set:N    = \l__document_structure_sfragment_intro_tl,
6617    imports       .tl_set:N    = \l__document_structure_sfragment_imports_tl,
6618    loadmodules   .bool_set:N  = \l__document_structure_sfragment_loadmodules_bool
6619 }
6620 \cs_new_protected:Nn \__document_structure_sfragment_args:n {
6621    \str_clear:N \l__document_structure_sfragment_id_str
6622    \str_clear:N \l__document_structure_sfragment_date_str
6623    \clist_clear:N \l__document_structure_sfragment_creators_clist
6624    \clist_clear:N \l__document_structure_sfragment_contributors_clist
6625    \tl_clear:N \l__document_structure_sfragment_srccite_tl
6626    \tl_clear:N \l__document_structure_sfragment_type_tl
6627    \tl_clear:N \l__document_structure_sfragment_short_tl
6628    \tl_clear:N \l__document_structure_sfragment_display_tl
6629    \tl_clear:N \l__document_structure_sfragment_imports_tl
6630    \tl_clear:N \l__document_structure_sfragment_intro_tl
6631    \bool_set_false:N \l__document_structure_sfragment_loadmodules_bool
6632    \keys_set:nn { document-structure / sfragment } { #1 }
6633 }
```

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

```
6634 \newif\if@mainmatter\@mainmattertrue
6635 \newcommand\at@begin@sfragment[3][]{}
```

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

```
6636 \keys_define:nn { document-structure / sectioning }{
6637    name     .str_set_x:N  = \l__document_structure_sect_name_str   ,
6638    ref      .str_set_x:N  = \l__document_structure_sect_ref_str    ,
6639    clear    .bool_set:N   = \l__document_structure_sect_clear_bool ,
6640    clear    .default:n    = {true}                                 ,
6641    num      .bool_set:N   = \l__document_structure_sect_num_bool   ,
6642    num      .default:n    = {true}
6643 }
6644 \cs_new_protected:Nn \__document_structure_sect_args:n {
6645    \str_clear:N \l__document_structure_sect_name_str
6646    \str_clear:N \l__document_structure_sect_ref_str
6647    \bool_set_false:N \l__document_structure_sect_clear_bool
6648    \bool_set_false:N \l__document_structure_sect_num_bool
6649    \keys_set:nn { document-structure / sectioning } { #1 }
6650 }
6651 \newcommand\omdoc@sectioning[3][]{
6652    \__document_structure_sect_args:n {#1 }
6653    \let\omdoc@sect@name\l__document_structure_sect_name_str
6654    \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
6655    \if@mainmatter% numbering not overridden by frontmatter, etc.
6656       \bool_if:NTF \l__document_structure_sect_num_bool {
```

238

```
6657        \sfragment@num{#2}{#3}
6658      }{
6659        \sfragment@nonum{#2}{#3}
6660      }
6661      \def\current@section@level{\omdoc@sect@name}
6662    \else
6663      \sfragment@nonum{#2}{#3}
6664    \fi
6665 }% 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.

```
6666 \newcommand\sfragment@redefine@addtocontents[1]{%
6667 %\edef\__document_structureimport{#1}%
6668 %\@for\@I:=\__document_structureimport\do{%
6669 %\edef\@path{\csname module@\@I  @path\endcsname}%
6670 %\@ifundefined{tf@toc}\relax%
6671 %      {\protected@write\tf@toc{}{\string\@requiremodules{\@path}}}}
6672 %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
6673 %\def\addcontentsline##1##2##3{%
6674 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}
6675 %\else% hyperref.sty not loaded
6676 %\def\addcontentsline##1##2##3{%
6677 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}{
6678 %\fi
6679 }% hypreref.sty loaded?
```

now the `sfragment` environment itself. This takes care of the table of contents via the helper macro above and then selects the appropriate sectioning command from `article.cls`. It also registeres the current level of sfragments in the `\sfragment@level` counter.

```
6680 \newenvironment{sfragment}[2][]% keys, title
6681 {
6682    \__document_structure_sfragment_args:n { #1 }%\sref@target%
```

If the `loadmodules` key is set on `\begin{sfragment}`, we redefine the `\addcontetsline` macro that determines how the sectioning commands below construct the entries for the table of contents.

```
6683    \stex_csl_to_imports:No \usemodule \l__document_structure_sfragment_imports_tl
6684
6685    \bool_if:NT \l__document_structure_sfragment_loadmodules_bool {
6686      \sfragment@redefine@addtocontents{
6687        %\@ifundefined{module@id}\used@modules%
6688        %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}
6689      }
6690    }
```

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

```
6691
6692    \stex_document_title:n { #2 }
6693
6694    \int_incr:N\l_document_structure_section_level_int
6695    \ifcase\l_document_structure_section_level_int
6696      \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
6697      \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
```

```
6698      \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
6699      \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
6700      \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
6701      \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#
6702      \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{paragr
6703    \fi
6704    \at@begin@sfragment[#1]\l_document_structure_section_level_int{#2}
6705    \str_if_empty:NF \l__document_structure_sfragment_id_str {
6706      \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
6707    }
6708 }% for customization
6709 {}
```

and finally, we localize the sections

```
6710 \newcommand\omdoc@part@kw{Part}
6711 \newcommand\omdoc@chapter@kw{Chapter}
6712 \newcommand\omdoc@section@kw{Section}
6713 \newcommand\omdoc@subsection@kw{Subsection}
6714 \newcommand\omdoc@subsubsection@kw{Subsubsection}
6715 \newcommand\omdoc@paragraph@kw{paragraph}
6716 \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

```
6717 \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).

```
6718 \cs_if_exist:NTF\frontmatter{
6719    \let\__document_structure_orig_frontmatter\frontmatter
6720    \let\frontmatter\relax
6721 }{
6722    \tl_set:Nn\__document_structure_orig_frontmatter{
6723      \clearpage
6724      \@mainmatterfalse
6725      \pagenumbering{roman}
6726    }
6727 }
6728 \cs_if_exist:NTF\backmatter{
6729    \let\__document_structure_orig_backmatter\backmatter
6730    \let\backmatter\relax
6731 }{
6732    \tl_set:Nn\__document_structure_orig_backmatter{
6733      \clearpage
6734      \@mainmatterfalse
```

```
6735        \pagenumbering{roman}
6736    }
6737 }
```

Using these, we can now define the `frontmatter` and `backmatter` environments

frontmatter   we use the `\orig@frontmatter` macro defined above and `\mainmatter` if it exists, oth-
erwise we define it.

```
6738 \newenvironment{frontmatter}{
6739    \__document_structure_orig_frontmatter
6740 }{
6741    \cs_if_exist:NTF\mainmatter{
6742        \mainmatter
6743    }{
6744        \clearpage
6745        \@mainmattertrue
6746        \pagenumbering{arabic}
6747    }
6748 }
```

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

```
6749 \newenvironment{backmatter}{
6750    \__document_structure_orig_backmatter
6751 }{
6752    \cs_if_exist:NTF\mainmatter{
6753        \mainmatter
6754    }{
6755        \clearpage
6756        \@mainmattertrue
6757        \pagenumbering{arabic}
6758    }
6759 }
```

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

```
6760 \@mainmattertrue\pagenumbering{arabic}
```

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

```
6761 \def \c__document_structure_document_str{document}
6762 \newcommand\afterprematurestop{}
6763 \def\prematurestop@endsfragment{
6764    \unless\ifx\@currenvir\c__document_structure_document_str
6765        \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafte
6766        \expandafter\prematurestop@endsfragment
6767    \fi
6768 }
6769 \providecommand\prematurestop{
6770    \message{Stopping~sTeX~processing~prematurely}
6771    \prematurestop@endsfragment
6772    \afterprematurestop
6773    \end{document}
6774 }
```

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

37.4 Global Variables

\setSGvar   set a global variable

```
6775 \RequirePackage{etoolbox}
6776 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}
```

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

\useSGvar   use a global variable

```
6777 \newrobustcmd\useSGvar[1]{%
6778   \@ifundefined{sTeX@Gvar@#1}
6779   {\PackageError{document-structure}
6780     {The sTeX Global variable #1 is undefined}
6781     {set it with \protect\setSGvar}}
6782 \@nameuse{sTeX@Gvar@#1}}
```

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

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

```
6783 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
6784   \@ifundefined{sTeX@Gvar@#1}
6785   {\PackageError{document-structure}
6786     {The sTeX Global variable #1 is undefined}
6787     {set it with \protect\setSGvar}}
6788   {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}
```

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

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.

```
6789  ⟨∗cls⟩
6790  ⟨@@=notesslides⟩
6791  \ProvidesExplClass{notesslides}{2022/02/28}{3.1.0}{notesslides Class}
6792  \RequirePackage{l3keys2e}
6793
6794  \keys_define:nn{notesslides / cls}{
6795    class    .str_set_x:N = \c__notesslides_class_str,
6796    notes    .bool_set:N  = \c__notesslides_notes_bool ,
6797    slides   .code:n       = { \bool_set_false:N \c__notesslides_notes_bool },
6798    docopt   .str_set_x:N = \c__notesslides_docopt_str,
6799    unknown  .code:n       = {
6800      \PassOptionsToPackage{\CurrentOption}{document-structure}
6801      \PassOptionsToClass{\CurrentOption}{beamer}
6802      \PassOptionsToPackage{\CurrentOption}{notesslides}
6803      \PassOptionsToPackage{\CurrentOption}{stex}
6804    }
6805  }
6806  \ProcessKeysOptions{ notesslides / cls }
6807
6808  \str_if_empty:NF \c__notesslides_class_str {
6809    \PassOptionsToPackage{class=\c__notesslides_class_str}{document-structure}
6810  }
6811
6812  \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{book}{
6813    \PassOptionsToPackage{defaulttopsect=part}{notesslides}
6814  }
6815  \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{report}{
6816    \PassOptionsToPackage{defaulttopsect=part}{notesslides}
6817  }
6818
6819  \RequirePackage{stex}
```

```
6820   \stex_html_backend:T {
6821     \bool_set_true:N\c__notesslides_notes_bool
6822   }
6823
6824   \bool_if:NTF \c__notesslides_notes_bool {
6825     \PassOptionsToPackage{notes=true}{notesslides}
6826   }{
6827     \PassOptionsToPackage{notes=false}{notesslides}
6828   }
6829   ⟨/cls⟩
```

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

```
6830   ⟨*package⟩
6831   \ProvidesExplPackage{notesslides}{2022/02/28}{3.1.0}{notesslides Package}
6832   \RequirePackage{l3keys2e}
6833
6834   \keys_define:nn{notesslides / pkg}{
6835     topsect         .str_set_x:N  = \c__notesslides_topsect_str,
6836     defaulttopsect  .str_set_x:N  = \c__notesslides_defaulttopsec_str,
6837     notes           .bool_set:N   = \c__notesslides_notes_bool ,
6838     slides          .code:n       = { \bool_set_false:N \c__notesslides_notes_bool },
6839     sectocframes    .bool_set:N   = \c__notesslides_sectocframes_bool ,
6840     frameimages     .bool_set:N   = \c__notesslides_frameimages_bool ,
6841     fiboxed         .bool_set:N   = \c__notesslides_fiboxed_bool ,
6842     noproblems      .bool_set:N   = \c__notesslides_noproblems_bool,
6843     unknown         .code:n       = {
6844       \PassOptionsToClass{\CurrentOption}{stex}
6845       \PassOptionsToClass{\CurrentOption}{tikzinput}
6846     }
6847   }
6848   \ProcessKeysOptions{ notesslides / pkg }
6849
6850   \RequirePackage{stex}
6851   \stex_html_backend:T {
6852     \bool_set_true:N\c__notesslides_notes_bool
6853   }
6854
6855   \newif\ifnotes
6856   \bool_if:NTF \c__notesslides_notes_bool {
6857     \notestrue
6858   }{
6859     \notesfalse
6860   }
6861
```

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

```
6862   \str_if_empty:NTF \c__notesslides_topsect_str {
6863     \str_set_eq:NN \__notesslidestopsect \c__notesslides_defaulttopsec_str
6864   }{
6865     \str_set_eq:NN \__notesslidestopsect \c__notesslides_topsect_str
6866   }
6867   \PassOptionsToPackage{topsect=\__notesslidestopsect}{document-structure}
6868   ⟨/package⟩
```

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

```
6869 ⟨*cls⟩
6870 \bool_if:NTF \c__notesslides_notes_bool {
6871   \str_if_empty:NT \c__notesslides_class_str {
6872     \str_set:Nn \c__notesslides_class_str {article}
6873   }
6874   \exp_after:wN\LoadClass\exp_after:wN[\c__notesslides_docopt_str]
6875     {\c__notesslides_class_str}
6876 }{
6877   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
6878   \newcounter{Item}
6879   \newcounter{paragraph}
6880   \newcounter{subparagraph}
6881   \newcounter{Hfootnote}
6882 }
6883 \RequirePackage{document-structure}
```

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

```
6884 \RequirePackage{notesslides}
6885 ⟨/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 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.

```
6886 ⟨*package⟩
6887 \bool_if:NT \c__notesslides_notes_bool {
6888   \RequirePackage{a4wide}
6889   \RequirePackage{marginnote}
6890   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
6891   \RequirePackage{mdframed}
6892   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
6893   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
6894 }
6895 \RequirePackage{stex-tikzinput}
6896 \RequirePackage{etoolbox}
6897 \RequirePackage{amssymb}
6898 \RequirePackage{amsmath}
6899 \RequirePackage{comment}
6900 \RequirePackage{textcomp}
6901 \RequirePackage{url}
6902 \RequirePackage{graphicx}
6903 \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`.[18]

```
6904 \bool_if:NT \c__notesslides_notes_bool {
6905   \renewcommand\usetheme[2][]{\usepackage[#1]{beamernotestheme#2}}
6906 }
6907
6908
6909 \NewDocumentCommand \libusetheme {O{} m} {
6910   \bool_if:NTF \c__notesslides_notes_bool {
6911     \libusepackage[#1]{beamernotestheme#2}
6912   }{
6913     \libusepackage[#1]{beamertheme#2}
6914   }
6915 }
```

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

```
6916 \newcounter{slide}
6917 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
6918 \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.

```
6919 \bool_if:NTF \c__notesslides_notes_bool {
6920   \renewenvironment{note}{\ignorespaces}{}
6921 }{
6922   \excludecomment{note}
6923 }
```

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.

```
6924 \bool_if:NT \c__notesslides_notes_bool {
6925   \newlength{\slideframewidth}
6926   \setlength{\slideframewidth}{1.5pt}
```

frame We first define the keys.

```
6927   \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
6928     \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
6929       \bool_set_true:N #1
6930     }{
6931       \bool_set_false:N #1
6932     }
6933   }
6934   \keys_define:nn{notesslides / frame}{
6935     label                 .str_set_x:N  = \l__notesslides_frame_label_str,
6936     allowframebreaks    .code:n       = {
6937       \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
6938     },
6939     allowdisplaybreaks  .code:n       = {
```

---

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

```
6940        \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
6941      },
6942      fragile              .code:n      = {
6943        \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
6944      },
6945      shrink               .code:n      = {
6946        \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
6947      },
6948      squeeze              .code:n      = {
6949        \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
6950      },
6951      t                    .code:n      = {
6952        \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
6953      },
6954    }
6955    \cs_new_protected:Nn \__notesslides_frame_args:n {
6956      \str_clear:N \l__notesslides_frame_label_str
6957      \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
6958      \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
6959      \bool_set_true:N \l__notesslides_frame_fragile_bool
6960      \bool_set_true:N \l__notesslides_frame_shrink_bool
6961      \bool_set_true:N \l__notesslides_frame_squeeze_bool
6962      \bool_set_true:N \l__notesslides_frame_t_bool
6963      \keys_set:nn { notesslides / frame }{ #1 }
6964    }
```

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

```
6965      \renewenvironment{frame}[1][]{
6966        \__notesslides_frame_args:n{#1}
6967        \sffamily
6968        \stepcounter{slide}
6969        \def\@currentlabel{\theslide}
6970        \str_if_empty:NF \l__notesslides_frame_label_str {
6971          \label{\l__notesslides_frame_label_str}
6972        }
```

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

```
6973        \def\itemize@level{outer}
6974        \def\itemize@outer{outer}
6975        \def\itemize@inner{inner}
6976        \renewcommand\newpage{\addtocounter{framenumber}{1}}
6977        \newcommand\metakeys@show@keys[2]{\marginnote{{\scriptsize ##2}}}
6978        \renewenvironment{itemize}{
6979          \ifx\itemize@level\itemize@outer
6980            \def\itemize@label{$\rhd$}
6981          \fi
6982          \ifx\itemize@level\itemize@inner
6983            \def\itemize@label{$\scriptstyle\rhd$}
6984          \fi
6985          \begin{list}
6986          {\itemize@label}
6987          {\setlength{\labelsep}{.3em}
6988           \setlength{\labelwidth}{.5em}
6989           \setlength{\leftmargin}{1.5em}
6990          }
```

```
6991        \edef\itemize@level{\itemize@inner}
6992      }{
6993        \end{list}
6994      }
```

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

```
6995        \stex_html_backend:TF {
6996          \begin{stex_annotate_env}{frame}{}\vbox\bgroup
6997        }{
6998          \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwid
6999        }
7000      }{
7001        \stex_html_backend:TF {
7002          \miko@slidelabel\egroup\end{stex_annotate_env}
7003        }{\medskip\miko@slidelabel\end{mdframed}}
7004      }
```

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

\frametitle

```
7005      \renewcommand{\frametitle}[1]{
7006        \stex_document_title:n { #1 }
7007        {\Large\bf\sf\color{blue}{#1}}\medskip
7008      }
7009  }
```

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

\pause   [19]

```
7010  \bool_if:NT \c__notesslides_notes_bool {
7011    \newcommand\pause{}
7012  }
```

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

nparagraph

```
7013  \bool_if:NTF \c__notesslides_notes_bool {
7014    \newenvironment{nparagraph}[1][]{\begin{sparagraph}[#1]}{\end{sparagraph}}
7015  }{
7016    \excludecomment{nparagraph}
7017  }
```

nfragment

```
7018  \bool_if:NTF \c__notesslides_notes_bool {
7019    \newenvironment{nfragment}[2][]{\begin{sfragment}[#1]{#2}}{\end{sfragment}}
7020  }{
7021    \excludecomment{nfragment}
7022  }
```

ndefinition

```
7023  \bool_if:NTF \c__notesslides_notes_bool {
7024    \newenvironment{ndefinition}[1][]{\begin{sdefinition}[#1]}{\end{sdefinition}}
7025  }{
7026    \excludecomment{ndefinition}
7027  }
```

---

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

```
7028  \bool_if:NTF \c__notesslides_notes_bool {
7029    \newenvironment{nassertion}[1][]{\begin{sassertion}[#1]}{\end{sassertion}}
7030  }{
7031    \excludecomment{nassertion}
7032  }
```

```
7033  \bool_if:NTF \c__notesslides_notes_bool {
7034    \newenvironment{nproof}[2][]{\begin{sproof}[#1]{#2}}{\end{sproof}}
7035  }{
7036    \excludecomment{nproof}
7037  }
```

```
7038  \bool_if:NTF \c__notesslides_notes_bool {
7039    \newenvironment{nexample}[1][]{\begin{sexample}[#1]}{\end{sexample}}
7040  }{
7041    \excludecomment{nexample}
7042  }
```

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

```
7043  \def\inputref@preskip{\smallskip}
7044  \def\inputref@postskip{\medskip}
```

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

\inputref*

```
7045  \let\orig@inputref\inputref
7046  \def\inputref{\@ifstar\ninputref\orig@inputref}
7047  \newcommand\ninputref[2][]{
7048    \bool_if:NT \c__notesslides_notes_bool {
7049      \orig@inputref[#1]{#2}
7050    }
7051  }
```

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

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*⟩}.

```
7052  \newlength{\slidelogoheight}
7053
7054  \bool_if:NTF \c__notesslides_notes_bool {
7055    \setlength{\slidelogoheight}{.4cm}
7056  }{
7057    \setlength{\slidelogoheight}{1cm}
7058  }
7059  \newsavebox{\slidelogo}
```

249

```
7060    \sbox{\slidelogo}{\sTeX}
7061    \newrobustcmd{\setslidelogo}[1]{
7062        \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
7063    }
```

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

`\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.

```
7064    \def\source{Michael Kohlhase}% customize locally
7065    \newrobustcmd{\setsource}[1]{\def\source{#1}}
```

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

`\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.

```
7066    \def\copyrightnotice{\footnotesize\copyright :\hspace{.3ex}{\source}}
7067    \newsavebox{\cclogo}
7068    \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{stex-cc_somerights}}
7069    \newif\ifcchref\cchreffalse
7070    \AtBeginDocument{
7071        \@ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
7072    }
7073    \def\licensing{
7074        \ifcchref
7075            \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
7076        \else
7077            {\usebox{\cclogo}}
7078        \fi
7079    }
7080    \newrobustcmd{\setlicensing}[2][]{
7081        \def\@url{#1}
7082        \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
7083        \ifx\@url\@empty
7084            \def\licensing{{\usebox{\cclogo}}}
7085        \else
7086            \def\licensing{
7087                \ifcchref
7088                \href{#1}{\usebox{\cclogo}}
7089                \else
7090                {\usebox{\cclogo}}
7091                \fi
7092            }
7093        \fi
7094    }
```

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

EdN:20   `\slidelabel`   Now, we set up the slide label for the `article` mode.[20]

```
7095    \newrobustcmd\miko@slidelabel{
7096        \vbox to \slidelogoheight{
```

---

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

250

```
7097        \vss\hbox to \slidewidth
7098        {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
7099    }
7100 }
```

(*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.

```
7101 \def\Gin@mhrepos{}
7102 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7103 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
7104 \newrobustcmd\frameimage[2][]{
7105    \stepcounter{slide}
7106    \bool_if:NT \c__notesslides_frameimages_bool {
7107       \def\Gin@ewidth{}\setkeys{Gin}{#1}
7108       \bool_if:NF \c__notesslides_notes_bool { \vfill }
7109       \begin{center}
7110          \bool_if:NTF \c__notesslides_fiboxed_bool {
7111             \fbox{
7112                \ifx\Gin@ewidth\@empty
7113                   \ifx\Gin@mhrepos\@empty
7114                      \mhgraphics[width=\slidewidth,#1]{#2}
7115                   \else
7116                      \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7117                   \fi
7118                \else% Gin@ewidth empty
7119                   \ifx\Gin@mhrepos\@empty
7120                      \mhgraphics[#1]{#2}
7121                   \else
7122                      \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7123                   \fi
7124                \fi% Gin@ewidth empty
7125             }
7126          }{
7127             \ifx\Gin@ewidth\@empty
7128                \ifx\Gin@mhrepos\@empty
7129                   \mhgraphics[width=\slidewidth,#1]{#2}
7130                \else
7131                   \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7132                \fi
7133                \ifx\Gin@mhrepos\@empty
7134                   \mhgraphics[#1]{#2}
7135                \else
7136                   \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7137                \fi
7138             \fi% Gin@ewidth empty
7139          }
7140       \end{center}
7141       \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
7142       \bool_if:NF \c__notesslides_notes_bool { \vfill }
```

```
7143    }
7144 } % ifmks@sty@frameimages
```

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

38.5   Colors and Highlighting

We first specify sans serif fonts as the default.

```
7145 \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

```
7146 \AddToHook{begindocument}{
7147    \definecolor{green}{rgb}{0,.5,0}
7148    \definecolor{purple}{cmyk}{.3,1,0,.17}
7149 }
```

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`.

```
7150 % \def\STpresent#1{\textcolor{blue}{#1}}
7151 \def\defemph#1{{\textcolor{magenta}{#1}}}
7152 \def\symrefemph#1{{\textcolor{cyan}{#1}}}
7153 \def\compemph#1{{\textcolor{blue}{#1}}}
7154 \def\titleemph#1{{\textcolor{blue}{#1}}}
7155 \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.

```
7156 \pgfdeclareimage[width=.8em]{miko@small@dbend}{stex-dangerous-bend}
7157 \def\smalltextwarning{
7158    \pgfuseimage{miko@small@dbend}
7159    \xspace
7160 }
7161 \pgfdeclareimage[width=1.2em]{miko@dbend}{stex-dangerous-bend}
7162 \newrobustcmd\textwarning{
7163    \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
7164    \xspace
7165 }
7166 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{stex-dangerous-bend}
7167 \newrobustcmd\bigtextwarning{
7168    \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
7169    \xspace
7170 }
```

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

```
7171 \newrobustcmd\putgraphicsat[3]{
7172    \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
7173 }
7174 \newrobustcmd\putat[2]{
7175    \begin{picture}(0,0)\put(#1){#2}\end{picture}
7176 }
```

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`.

```
7177 \bool_if:NT \c__notesslides_sectocframes_bool {
7178   \str_if_eq:VnTF \__notesslidestopsect{part}{
7179     \newcounter{chapter}\counterwithin*{section}{chapter}
7180   }{
7181     \str_if_eq:VnT\__notesslidestopsect{chapter}{
7182       \newcounter{chapter}\counterwithin*{section}{chapter}
7183     }
7184   }
7185 }
```

\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

```
7186 \def\part@prefix{}
7187 \@ifpackageloaded{document-structure}{}{
7188   \str_case:VnF \__notesslidestopsect {
7189     {part}{
7190       \int_set:Nn \l_document_structure_section_level_int {0}
7191       \def\thesection{\arabic{chapter}.\arabic{section}}
7192       \def\part@prefix{\arabic{chapter}.}
7193     }
7194     {chapter}{
7195       \int_set:Nn \l_document_structure_section_level_int {1}
7196       \def\thesection{\arabic{chapter}.\arabic{section}}
7197       \def\part@prefix{\arabic{chapter}.}
7198     }
7199   }{
7200     \int_set:Nn \l_document_structure_section_level_int {2}
7201     \def\part@prefix{}
7202   }
7203 }
7204
7205 \bool_if:NF \c__notesslides_notes_bool { % only in slides
```

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

The new counters are used in the `sfragment` environment that choses the LaTeX sectioning macros according to `\section@level`.

sfragment

```
7206 \renewenvironment{sfragment}[2][]{
7207   \__document_structure_sfragment_args:n { #1 }
7208   \int_incr:N \l_document_structure_section_level_int
7209   \bool_if:NT \c__notesslides_sectocframes_bool {
7210     \stepcounter{slide}
7211     \begin{frame}[noframenumbering]
7212     \vfill\Large\centering
7213     \red{
7214       \ifcase\l_document_structure_section_level_int\or
```

```
7215        \stepcounter{part}
7216        \def\__notesslideslabel{{\omdoc@part@kw}~\Roman{part}}
7217        \def\currentsectionlevel{\omdoc@part@kw}
7218      \or
7219        \stepcounter{chapter}
7220        \def\__notesslideslabel{{\omdoc@chapter@kw}~\arabic{chapter}}
7221        \def\currentsectionlevel{\omdoc@chapter@kw}
7222      \or
7223        \stepcounter{section}
7224        \def\__notesslideslabel{\part@prefix\arabic{section}}
7225        \def\currentsectionlevel{\omdoc@section@kw}
7226      \or
7227        \stepcounter{subsection}
7228        \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
7229        \def\currentsectionlevel{\omdoc@subsection@kw}
7230      \or
7231        \stepcounter{subsubsection}
7232        \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
7233        \def\currentsectionlevel{\omdoc@subsubsection@kw}
7234      \or
7235        \stepcounter{paragraph}
7236        \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
7237        \def\currentsectionlevel{\omdoc@paragraph@kw}
7238      \else
7239        \def\__notesslideslabel{}
7240        \def\currentsectionlevel{\omdoc@paragraph@kw}
7241      \fi% end ifcase
7242      \__notesslideslabel%\sref@label@id\__notesslideslabel
7243      \quad #2%
7244    }%
7245    \vfill%
7246    \end{frame}%
7247  }
7248  \str_if_empty:NF \l__document_structure_sfragment_id_str {
7249    \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
7250  }
7251  }{}
7252 }
```

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

```
7253 \def\inserttheorembodyfont{\normalfont}
7254 %\bool_if:NF \c__notesslides_notes_bool {
7255 %  \defbeamertemplate{theorem begin}{miko}
7256 %  {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
7257 %    \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
7258 %    \inserttheorempunctuation\inserttheorembodyfont\xspace}
7259 %  \defbeamertemplate{theorem end}{miko}{}
```

and we set it as the default one.

```
7260 %  \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.

```
7261 %  \expandafter\def\csname Parent2\endcsname{}
```

```
7262 %}
7263
7264 \AddToHook{begindocument}{ % this does not work for some reasone
7265   \setbeamertemplate{theorems}[ams style]
7266 }
7267 \bool_if:NT \c__notesslides_notes_bool {
7268   \renewenvironment{columns}[1][]{%
7269     \par\noindent%
7270     \begin{minipage}%
7271     \slidewidth\centering\leavevmode%
7272   }{%
7273     \end{minipage}\par\noindent%
7274   }%
7275   \newsavebox\columnbox%
7276   \renewenvironment<>{column}[2][]{%
7277     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
7278   }{%
7279     \end{minipage}\end{lrbox}\usebox\columnbox%
7280   }%
7281 }
7282 \bool_if:NTF \c__notesslides_noproblems_bool {
7283   \newenvironment{problems}{}{}
7284 }{
7285   \excludecomment{problems}
7286 }
```

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.

```
7287 \gdef\printexcursions{}
7288 \newcommand\excursionref[2]{% label, text
7289   \bool_if:NT \c__notesslides_notes_bool {
7290     \begin{sparagraph}[title=Excursion]
7291       #2 \sref[fallback=the appendix]{#1}.
7292     \end{sparagraph}
7293   }
7294 }
7295 \newcommand\activate@excursion[2][]{
7296   \gappto\printexcursions{\inputref[#1]{#2}}
7297 }
7298 \newcommand\excursion[4][]{% repos, label, path, text
7299   \bool_if:NT \c__notesslides_notes_bool {
7300     \activate@excursion[#1]{#3}\excursionref{#2}{#4}
7301   }
7302 }
```

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

\excursiongroup

```
7303 \keys_define:nn{notesslides / excursiongroup }{
```

```
7304    id        .str_set_x:N  = \l__notesslides_excursion_id_str,
7305    intro     .tl_set:N     = \l__notesslides_excursion_intro_tl,
7306    mhrepos   .str_set_x:N  = \l__notesslides_excursion_mhrepos_str
7307 }
7308 \cs_new_protected:Nn \__notesslides_excursion_args:n {
7309    \tl_clear:N \l__notesslides_excursion_intro_tl
7310    \str_clear:N \l__notesslides_excursion_id_str
7311    \str_clear:N \l__notesslides_excursion_mhrepos_str
7312    \keys_set:nn {notesslides / excursiongroup }{ #1 }
7313 }
7314 \newcommand\excursiongroup[1][]{
7315    \__notesslides_excursion_args:n{ #1 }
7316    \ifdefempty\printexcursions{}% only if there are excursions
7317    {\begin{note}
7318      \begin{sfragment}[#1]{Excursions}%
7319        \ifdefempty\l__notesslides_excursion_intro_tl{}{
7320          \inputref[\l__notesslides_excursion_mhrepos_str]{
7321            \l__notesslides_excursion_intro_tl
7322          }
7323        }
7324        \printexcursions%
7325      \end{sfragment}
7326    \end{note}}
7327 }
7328 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
7329 ⟨/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.

```
7330  ⟨∗package⟩
7331  ⟨@@=problems⟩
7332  \ProvidesExplPackage{problem}{2022/02/26}{3.0.1}{Semantic Markup for Problems}
7333  \RequirePackage{l3keys2e,stex}
7334
7335  \keys_define:nn { problem / pkg }{
7336    notes     .default:n   = { true },
7337    notes     .bool_set:N  = \c__problems_notes_bool,
7338    gnotes    .default:n   = { true },
7339    gnotes    .bool_set:N  = \c__problems_gnotes_bool,
7340    hints     .default:n   = { true },
7341    hints     .bool_set:N  = \c__problems_hints_bool,
7342    solutions .default:n   = { true },
7343    solutions .bool_set:N  = \c__problems_solutions_bool,
7344    pts       .default:n   = { true },
7345    pts       .bool_set:N  = \c__problems_pts_bool,
7346    min       .default:n   = { true },
7347    min       .bool_set:N  = \c__problems_min_bool,
7348    boxed     .default:n   = { true },
7349    boxed     .bool_set:N  = \c__problems_boxed_bool,
7350    unknown   .code:n      = {}
7351  }
7352  \newif\ifsolutions
7353
7354  \ProcessKeysOptions{ problem / pkg }
7355  \bool_if:NTF \c__problems_solutions_bool {
7356    \solutionstrue
7357  }{
7358    \solutionsfalse
7359  }
```

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

```
7360  \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.

```
7361  \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.

```
7362  \def\prob@problem@kw{Problem}
7363  \def\prob@solution@kw{Solution}
7364  \def\prob@hint@kw{Hint}
7365  \def\prob@note@kw{Note}
7366  \def\prob@gnote@kw{Grading}
7367  \def\prob@pt@kw{pt}
7368  \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
7369  \AddToHook{begindocument}{
7370    \ltx@ifpackageloaded{babel}{
7371        \makeatletter
7372        \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7373        \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7374          \input{problem-ngerman.ldf}
7375        }
7376        \clist_if_in:NnT \l_tmpa_clist {finnish}{
7377          \input{problem-finnish.ldf}
7378        }
7379        \clist_if_in:NnT \l_tmpa_clist {french}{
7380          \input{problem-french.ldf}
7381        }
7382        \clist_if_in:NnT \l_tmpa_clist {russian}{
7383          \input{problem-russian.ldf}
7384        }
7385        \makeatother
7386    }{}
7387  }
```

39.2   Problems and Solutions

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

```
7388  \keys_define:nn{ problem / problem }{
7389    id      .str_set_x:N  = \l__problems_prob_id_str,
7390    pts     .tl_set:N     = \l__problems_prob_pts_tl,
7391    min     .tl_set:N     = \l__problems_prob_min_tl,
7392    title   .tl_set:N     = \l__problems_prob_title_tl,
7393    type    .tl_set:N     = \l__problems_prob_type_tl,
7394    imports .tl_set:N     = \l__problems_prob_imports_tl,
7395    name    .str_set_x:N  = \l__problems_prob_name_str,
7396    refnum  .int_set:N    = \l__problems_prob_refnum_int
```

```
7397 }
7398 \cs_new_protected:Nn \__problems_prob_args:n {
7399   \str_clear:N \l__problems_prob_id_str
7400   \str_clear:N \l__problems_prob_name_str
7401   \tl_clear:N \l__problems_prob_pts_tl
7402   \tl_clear:N \l__problems_prob_min_tl
7403   \tl_clear:N \l__problems_prob_title_tl
7404   \tl_clear:N \l__problems_prob_type_tl
7405   \tl_clear:N \l__problems_prob_imports_tl
7406   \int_zero_new:N \l__problems_prob_refnum_int
7407   \keys_set:nn { problem / problem }{ #1 }
7408   \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
7409     \let\l__problems_prob_refnum_int\undefined
7410   }
7411 }
```

Then we set up a counter for problems.

\numberproblemsin

```
7412 \newcounter{problem}[section]
7413 \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.

```
7414 \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

```
7415 \newcommand\prob@number{
7416   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
7417     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
7418   }{
7419     \int_if_exist:NTF \l__problems_prob_refnum_int {
7420       \prob@label{\int_use:N \l__problems_prob_refnum_int }
7421     }{
7422       \prob@label\theproblem
7423     }
7424   }
7425 }
```

(*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.

```
7426 \newcommand\prob@title[3]{%
7427   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
7428     #2 \l__problems_inclprob_title_tl #3
7429   }{
7430     \tl_if_exist:NTF \l__problems_prob_title_tl {
7431       #2 \l__problems_prob_title_tl #3
7432     }{
7433       #1
```

259

```
7434              }
7435          }
7436    }
```

(*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.

```
7437  \def\prob@heading{
7438    {\prob@problem@kw}\ \prob@number\prob@title{~}{~(}{)\strut}
7439    %\sref@label@id{\prob@problem@kw~\prob@number}{}
7440  }
```

(*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

```
7441  \newenvironment{sproblem}[1][]{
7442    \__problems_prob_args:n{#1}%\sref@target%
7443    \@in@omtexttrue% we are in a statement (for inline definitions)
7444    \stepcounter{problem}\record@problem
7445    \def\current@section@level{\prob@problem@kw}
7446
7447    \str_if_empty:NT \l__problems_prob_name_str {
7448      \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
7449      \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
7450      \seq_get_left:NN \l_tmpa_seq \l__problems_prob_name_str
7451    }
7452
7453    \stex_if_do_html:T{
7454      \tl_if_empty:NF \l__problems_prob_title_tl {
7455        \exp_args:No \stex_document_title:n \l__problems_prob_title_tl
7456      }
7457    }
7458
7459    \exp_args:Nno\stex_module_setup:nn{type=problem}\l__problems_prob_name_str
7460
7461    \stex_reactivate_macro:N \STEXexport
7462    \stex_reactivate_macro:N \importmodule
7463    \stex_reactivate_macro:N \symdecl
7464    \stex_reactivate_macro:N \notation
7465    \stex_reactivate_macro:N \symdef
7466
7467    \stex_if_do_html:T{
7468      \begin{stex_annotate_env} {problem} {
7469        \l_stex_module_ns_str ? \l_stex_module_name_str
7470      }
7471
7472      \stex_annotate_invisible:nnn{header}{} {
7473        \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
```

```
7474      \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
7475      \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
7476        \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
7477      }
7478    }
7479  }

7480
7481  \stex_csl_to_imports:No \importmodule \l__problems_prob_imports_tl

7482
7483
7484  \tl_if_exist:NTF \l__problems_inclprob_type_tl {
7485    \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
7486  }{
7487    \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
7488  }
7489  \str_if_exist:NTF \l__problems_inclprob_id_str {
7490    \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
7491  }{
7492    \str_set_eq:NN \sproblemid \l__problems_prob_id_str
7493  }

7494
7495
7496  \stex_if_smsmode:F {
7497    \clist_set:No \l_tmpa_clist \sproblemtype
7498    \tl_clear:N \l_tmpa_tl
7499    \clist_map_inline:Nn \l_tmpa_clist {
7500      \tl_if_exist:cT {__problems_sproblem_##1_start:}{
7501        \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_start:}}
7502      }
7503    }
7504    \tl_if_empty:NTF \l_tmpa_tl {
7505      \__problems_sproblem_start:
7506    }{
7507      \l_tmpa_tl
7508    }
7509  }
7510  \stex_ref_new_doc_target:n \sproblemid
7511  \stex_smsmode_do:
7512 }{
7513    \__stex_modules_end_module:
7514    \stex_if_smsmode:F{
7515      \clist_set:No \l_tmpa_clist \sproblemtype
7516      \tl_clear:N \l_tmpa_tl
7517      \clist_map_inline:Nn \l_tmpa_clist {
7518        \tl_if_exist:cT {__problems_sproblem_##1_end:}{
7519          \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_end:}}
7520        }
7521      }
7522      \tl_if_empty:NTF \l_tmpa_tl {
7523        \__problems_sproblem_end:
7524      }{
7525        \l_tmpa_tl
7526      }
7527    }
```

261

```
7528     \stex_if_do_html:T{
7529        \end{stex_annotate_env}
7530     }
7531
7532     \smallskip
7533 }
7534
7535 \seq_put_right:Nx\g_stex_smsmode_allowedenvs_seq{\tl_to_str:n{sproblem}}
7536
7537
7538
7539 \cs_new_protected:Nn \__problems_sproblem_start: {
7540     \par\noindent\textbf\prob@heading\show@pts\show@min\\\ignorespacesandpars
7541 }
7542 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
7543
7544 \newcommand\stexpatchproblem[3][] {
7545     \str_set:Nx \l_tmpa_str{ #1 }
7546     \str_if_empty:NTF \l_tmpa_str {
7547        \tl_set:Nn \__problems_sproblem_start: { #2 }
7548        \tl_set:Nn \__problems_sproblem_end: { #3 }
7549     }{
7550        \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
7551        \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
7552     }
7553 }
7554
7555
7556 \bool_if:NT \c__problems_boxed_bool {
7557     \surroundwithmdframed{problem}
7558 }
```

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

```
7559 \def\record@problem{
7560     \protected@write\@auxout{}
7561     {
7562        \string\@problem{\prob@number}
7563        {
7564           \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7565              \l__problems_inclprob_pts_tl
7566           }{
7567              \l__problems_prob_pts_tl
7568           }
7569        }%
7570        {
7571           \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7572              \l__problems_inclprob_min_tl
7573           }{
7574              \l__problems_prob_min_tl
7575           }
7576        }
7577     }
7578 }
```

(*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).

```
7579 \def\@problem#1#2#3{}
```

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

solution The `solution` environment is similar to the `problem` environment, only that it is independent of the boxed mode. It also has it's own keys that we need to define first.

```
7580 \keys_define:nn { problem / solution }{
7581   id           .str_set_x:N = \l__problems_solution_id_str ,
7582   for          .tl_set:N    = \l__problems_solution_for_tl ,
7583   height       .dim_set:N   = \l__problems_solution_height_dim ,
7584   creators     .clist_set:N = \l__problems_solution_creators_clist ,
7585   contributors .clist_set:N = \l__problems_solution_contributors_clist ,
7586   srccite      .tl_set:N    = \l__problems_solution_srccite_tl
7587 }
7588 \cs_new_protected:Nn \__problems_solution_args:n {
7589   \str_clear:N \l__problems_solution_id_str
7590   \tl_clear:N \l__problems_solution_for_tl
7591   \tl_clear:N \l__problems_solution_srccite_tl
7592   \clist_clear:N \l__problems_solution_creators_clist
7593   \clist_clear:N \l__problems_solution_contributors_clist
7594   \dim_zero:N \l__problems_solution_height_dim
7595   \keys_set:nn { problem / solution }{ #1 }
7596 }
```

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

```
7597 \newcommand\@startsolution[1][]{
7598   \__problems_solution_args:n { #1 }
7599   \@in@omtexttrue% we are in a statement.
7600   \bool_if:NF \c__problems_boxed_bool { \hrule }
7601   \smallskip\noindent
7602   {\textbf\prob@solution@kw :\enspace}
7603   \begin{small}
7604   \def\current@section@level{\prob@solution@kw}
7605   \ignorespacesandpars
7606 }
```

\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.

```
7607 \box_new:N \l__problems_solution_box
7608 \newenvironment{solution}[1][]{
7609   \stex_html_backend:TF{
7610     \stex_if_do_html:T{
7611       \begin{stex_annotate_env}{solution}{}
7612     }
7613   }{
7614     \setbox\l__problems_solution_box\vbox\bgroup
7615       \par\smallskip\hrule\smallskip
7616       \noindent\textbf{Solution:}~
7617   }
7618 }{
7619   \stex_html_backend:TF{
```

```
7620        \stex_if_do_html:T{
7621          \end{stex_annotate_env}
7622        }
7623    }{
7624      \smallskip\hrule
7625      \egroup
7626      \bool_if:NT \c__problems_solutions_bool {
7627        \box\l__problems_solution_box
7628      }
7629    }
7630  }
7631
7632  \newcommand\startsolutions{
7633    \bool_set_true:N \c__problems_solutions_bool
7634  % \specialcomment{solution}{\@startsolution}{
7635  %   \bool_if:NF \c__problems_boxed_bool {
7636  %     \hrule\medskip
7637  %   }
7638  %   \end{small}%
7639  % }
7640  % \bool_if:NT \c__problems_boxed_bool {
7641  %   \surroundwithmdframed{solution}
7642  % }
7643  }
```

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

```
7644  \newcommand\stopsolutions{\bool_set_false:N \c__problems_solutions_bool}%\excludecomment{sol
```

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

so it only remains to start/stop solutions depending on what option was specified.

```
7645  \ifsolutions
7646    \startsolutions
7647  \else
7648    \stopsolutions
7649  \fi
```

exnote

```
7650  \bool_if:NTF \c__problems_notes_bool {
7651    \newenvironment{exnote}[1][]{
7652      \par\smallskip\hrule\smallskip
7653      \noindent\textbf{\prob@note@kw :~ }\small
7654    }{
7655      \smallskip\hrule
7656    }
7657  }{
7658    \excludecomment{exnote}
7659  }
```

hint

```
7660  \bool_if:NTF \c__problems_notes_bool {
7661    \newenvironment{hint}[1][]{
7662      \par\smallskip\hrule\smallskip
```

```
7663    \noindent\textbf{\prob@hint@kw :~ }\small
7664  }{
7665    \smallskip\hrule
7666  }
7667  \newenvironment{exhint}[1][]{
7668    \par\smallskip\hrule\smallskip
7669    \noindent\textbf{\prob@hint@kw :~ }\small
7670  }{
7671    \smallskip\hrule
7672  }
7673  }{
7674    \excludecomment{hint}
7675    \excludecomment{exhint}
7676  }
```

gnote

```
7677  \bool_if:NTF \c__problems_notes_bool {
7678    \newenvironment{gnote}[1][]{
7679      \par\smallskip\hrule\smallskip
7680      \noindent\textbf{\prob@gnote@kw :~ }\small
7681    }{
7682      \smallskip\hrule
7683    }
7684  }{
7685    \excludecomment{gnote}
7686  }
```

39.3   Multiple Choice Blocks

mcb [21]

```
7687  \newenvironment{mcb}{
7688    \begin{enumerate}
7689  }{
7690    \end{enumerate}
7691  }
```

we define the keys for the mcc macro

```
7692  \cs_new_protected:Nn \__problems_do_yes_param:Nn {
7693    \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
7694      \bool_set_true:N #1
7695    }{
7696      \bool_set_false:N #1
7697    }
7698  }
7699  \keys_define:nn { problem / mcc }{
7700    id        .str_set_x:N  = \l__problems_mcc_id_str ,
7701    feedback  .tl_set:N     = \l__problems_mcc_feedback_tl ,
7702    T         .default:n    = { false } ,
7703    T         .bool_set:N   = \l__problems_mcc_t_bool ,
7704    F         .default:n    = { false } ,
7705    F         .bool_set:N   = \l__problems_mcc_f_bool ,
```

---

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

```
7706    Ttext      .tl_set:N     = \l__problems_mcc_Ttext_str ,
7707    Ftext      .tl_set:N     = \l__problems_mcc_Ftext_str
7708  }
7709  \cs_new_protected:Nn \l__problems_mcc_args:n {
7710    \str_clear:N \l__problems_mcc_id_str
7711    \tl_clear:N \l__problems_mcc_feedback_tl
7712    \bool_set_false:N \l__problems_mcc_t_bool
7713    \bool_set_false:N \l__problems_mcc_f_bool
7714    \tl_clear:N \l__problems_mcc_Ttext_tl
7715    \tl_clear:N \l__problems_mcc_Ftext_tl
7716    \str_clear:N \l__problems_mcc_id_str
7717    \keys_set:nn { problem / mcc }{ #1 }
7718  }
```

\mcc

```
7719  \def\mccTrueText{\textbf{(true)~}}
7720  \def\mccFalseText{\textbf{(false)~}}
7721  \newcommand\mcc[2][]{
7722    \l__problems_mcc_args:n{ #1 }
7723    \item[$\Box$] #2
7724    \ifsolutions
7725      \\
7726      \bool_if:NT \l__problems_mcc_t_bool {
7727        \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccTrueText\l__problems_mcc_Ttext_tl
7728      }
7729      \bool_if:NT \l__problems_mcc_f_bool {
7730        \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccFalseText\l__problems_mcc_Ftext_tl
7731      }
7732      \tl_if_empty:NF \l__problems_mcc_feedback_tl {
7733        \emph{(\l__problems_mcc_feedback_tl)}
7734      }
7735    \fi
7736  } %solutions
```

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

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.

```
7737
7738  \keys_define:nn{ problem / inclproblem }{
7739    id       .str_set_x:N  = \l__problems_inclprob_id_str,
7740    pts      .tl_set:N     = \l__problems_inclprob_pts_tl,
7741    min      .tl_set:N     = \l__problems_inclprob_min_tl,
7742    title    .tl_set:N     = \l__problems_inclprob_title_tl,
7743    refnum   .int_set:N    = \l__problems_inclprob_refnum_int,
7744    type     .tl_set:N     = \l__problems_inclprob_type_tl,
7745    mhrepos  .str_set_x:N  = \l__problems_inclprob_mhrepos_str
7746  }
7747  \cs_new_protected:Nn \__problems_inclprob_args:n {
7748    \str_clear:N \l__problems_prob_id_str
```

```
7749    \tl_clear:N \l__problems_inclprob_pts_tl
7750    \tl_clear:N \l__problems_inclprob_min_tl
7751    \tl_clear:N \l__problems_inclprob_title_tl
7752    \tl_clear:N \l__problems_inclprob_type_tl
7753    \int_zero_new:N \l__problems_inclprob_refnum_int
7754    \str_clear:N \l__problems_inclprob_mhrepos_str
7755    \keys_set:nn { problem / inclproblem }{ #1 }
7756    \tl_if_empty:NT \l__problems_inclprob_pts_tl {
7757      \let\l__problems_inclprob_pts_tl\undefined
7758    }
7759    \tl_if_empty:NT \l__problems_inclprob_min_tl {
7760      \let\l__problems_inclprob_min_tl\undefined
7761    }
7762    \tl_if_empty:NT \l__problems_inclprob_title_tl {
7763      \let\l__problems_inclprob_title_tl\undefined
7764    }
7765    \tl_if_empty:NT \l__problems_inclprob_type_tl {
7766      \let\l__problems_inclprob_type_tl\undefined
7767    }
7768    \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
7769      \let\l__problems_inclprob_refnum_int\undefined
7770    }
7771  }
7772
7773  \cs_new_protected:Nn \__problems_inclprob_clear: {
7774    \let\l__problems_inclprob_id_str\undefined
7775    \let\l__problems_inclprob_pts_tl\undefined
7776    \let\l__problems_inclprob_min_tl\undefined
7777    \let\l__problems_inclprob_title_tl\undefined
7778    \let\l__problems_inclprob_type_tl\undefined
7779    \let\l__problems_inclprob_refnum_int\undefined
7780    \let\l__problems_inclprob_mhrepos_str\undefined
7781  }
7782  \__problems_inclprob_clear:
7783
7784  \newcommand\includeproblem[2][]{
7785    \__problems_inclprob_args:n{ #1 }
7786    \exp_args:No \stex_in_repository:nn\l__problems_inclprob_mhrepos_str{
7787      \stex_html_backend:TF {
7788        \str_clear:N \l_tmpa_str
7789        \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
7790          \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
7791        }
7792        \stex_annotate_invisible:nnn{includeproblem}{
7793          \l_tmpa_str / #2
7794        }{}
7795      }{
7796        \begingroup
7797          \inputreftrue
7798          \tl_if_empty:nTF{ ##1 }{
7799            \input{#2}
7800          }{
7801            \input{ \c_stex_mathhub_str / ##1 / source / #2 }
7802          }
```

```
7803        \endgroup
7804      }
7805    }
7806    \__problems_inclprob_clear:
7807  }
```

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

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.

```
7808  \AddToHook{enddocument}{
7809    \bool_if:NT \c__problems_pts_bool {
7810      \message{Total:~\arabic{pts}~points}
7811    }
7812    \bool_if:NT \c__problems_min_bool {
7813      \message{Total:~\arabic{min}~minutes}
7814    }
7815  }
```

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

```
7816  \def\pts#1{
7817    \bool_if:NT \c__problems_pts_bool {
7818      \marginpar{#1~\prob@pt@kw}
7819    }
7820  }
7821  \def\min#1{
7822    \bool_if:NT \c__problems_min_bool {
7823      \marginpar{#1~\prob@min@kw}
7824    }
7825  }
```

\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.

```
7826  \newcounter{pts}
7827  \def\show@pts{
7828    \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7829      \bool_if:NT \c__problems_pts_bool {
7830        \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
7831        \addtocounter{pts}{\l__problems_inclprob_pts_tl}
7832      }
7833    }{
7834      \tl_if_exist:NT \l__problems_prob_pts_tl {
7835        \bool_if:NT \c__problems_pts_bool {
7836          \tl_if_empty:NT\l__problems_prob_pts_tl{
7837            \tl_set:Nn \l__problems_prob_pts_tl {0}
7838          }
7839          \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
7840          \addtocounter{pts}{\l__problems_prob_pts_tl}
7841        }
7842      }
```

```
7843        }
7844    }
```

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

and now the same for the minutes

`\show@min`

```
7845   \newcounter{min}
7846   \def\show@min{
7847     \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7848       \bool_if:NT \c__problems_min_bool {
7849         \marginpar{\l__problems_inclprob_pts_tl\ min}
7850         \addtocounter{min}{\l__problems_inclprob_min_tl}
7851       }
7852     }{
7853       \tl_if_exist:NT \l__problems_prob_min_tl {
7854         \bool_if:NT \c__problems_min_bool {
7855           \tl_if_empty:NT\l__problems_prob_min_tl{
7856             \tl_set:Nn \l__problems_prob_min_tl {0}
7857           }
7858           \marginpar{\l__problems_prob_min_tl\ min}
7859           \addtocounter{min}{\l__problems_prob_min_tl}
7860         }
7861       }
7862     }
7863   }
7864 ⟨/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.

```
7865 ⟨∗package⟩
7866 \ProvidesExplPackage{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
7867 \RequirePackage{l3keys2e}
7868
7869 \newif\iftest\testfalse
7870 \DeclareOption{test}{\testtrue}
7871 \newif\ifmultiple\multiplefalse
7872 \DeclareOption{multiple}{\multipletrue}
7873 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
7874 \ProcessOptions
```

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

```
7875 \RequirePackage{keyval}[1997/11/10]
7876 \RequirePackage{problem}
```

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

```
7877 \newcommand\hwexam@assignment@kw{Assignment}
7878 \newcommand\hwexam@given@kw{Given}
7879 \newcommand\hwexam@due@kw{Due}
7880 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
7881 blank~for~extra~space}
7882 \def\hwexam@minutes@kw{minutes}
7883 \newcommand\correction@probs@kw{prob.}
7884 \newcommand\correction@pts@kw{total}
7885 \newcommand\correction@reached@kw{reached}
7886 \newcommand\correction@sum@kw{Sum}
7887 \newcommand\correction@grade@kw{grade}
7888 \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

```
7889 \AddToHook{begindocument}{
7890 \ltx@ifpackageloaded{babel}{
7891 \makeatletter
7892 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7893 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7894     \input{hwexam-ngerman.ldf}
7895 }
7896 \clist_if_in:NnT \l_tmpa_clist {finnish}{
7897     \input{hwexam-finnish.ldf}
7898 }
7899 \clist_if_in:NnT \l_tmpa_clist {french}{
7900     \input{hwexam-french.ldf}
7901 }
7902 \clist_if_in:NnT \l_tmpa_clist {russian}{
7903     \input{hwexam-russian.ldf}
7904 }
7905 \makeatother
7906 }{}
7907 }
7908
```

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.

```
7909 \newcounter{assignment}
7910 %\numberproblemsin{assignment}
```

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

```
7911 \keys_define:nn { hwexam / assignment } {
7912 id   .str_set_x:N = \l_@@_assign_id_str,
7913 number  .int_set:N  = \l_@@_assign_number_int,
7914 title  .tl_set:N  = \l_@@_assign_title_tl,
7915 type  .tl_set:N  = \l_@@_assign_type_tl,
7916 given .tl_set:N  = \l_@@_assign_given_tl,
7917 due .tl_set:N  = \l_@@_assign_due_tl,
7918 loadmodules .code:n  = {
7919 \bool_set_true:N \l_@@_assign_loadmodules_bool
7920 }
7921 }
7922 \cs_new_protected:Nn \_@@_assignment_args:n {
7923 \str_clear:N \l_@@_assign_id_str
7924 \int_set:Nn \l_@@_assign_number_int {-1}
7925 \tl_clear:N \l_@@_assign_title_tl
7926 \tl_clear:N \l_@@_assign_type_tl
7927 \tl_clear:N \l_@@_assign_given_tl
7928 \tl_clear:N \l_@@_assign_due_tl
7929 \bool_set_false:N \l_@@_assign_loadmodules_bool
7930 \keys_set:nn { hwexam / assignment }{ #1 }
7931 }
```

271

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.

```
7932  \newcommand\given@due[2]{
7933  \bool_lazy_all:nF {
7934  {\tl_if_empty_p:V \l_@@_inclassign_given_tl}
7935  {\tl_if_empty_p:V \l_@@_assign_given_tl}
7936  {\tl_if_empty_p:V \l_@@_inclassign_due_tl}
7937  {\tl_if_empty_p:V \l_@@_assign_due_tl}
7938  }{ #1 }
7939
7940  \tl_if_empty:NTF \l_@@_inclassign_given_tl {
7941  \tl_if_empty:NF \l_@@_assign_given_tl {
7942  \hwexam@given@kw\xspace\l_@@_assign_given_tl
7943  }
7944  }{
7945  \hwexam@given@kw\xspace\l_@@_inclassign_given_tl
7946  }
7947
7948  \bool_lazy_or:nnF {
7949  \bool_lazy_and_p:nn {
7950  \tl_if_empty_p:V \l_@@_inclassign_due_tl
7951  }{
7952  \tl_if_empty_p:V \l_@@_assign_due_tl
7953  }
7954  }{
7955  \bool_lazy_and_p:nn {
7956  \tl_if_empty_p:V \l_@@_inclassign_due_tl
7957  }{
7958  \tl_if_empty_p:V \l_@@_assign_due_tl
7959  }
7960  }{ ,~ }
7961
7962  \tl_if_empty:NTF \l_@@_inclassign_due_tl {
7963  \tl_if_empty:NF \l_@@_assign_due_tl {
7964  \hwexam@due@kw\xspace \l_@@_assign_due_tl
7965  }
7966  }{
7967  \hwexam@due@kw\xspace \l_@@_inclassign_due_tl
7968  }
7969
7970  \bool_lazy_all:nF {
7971  { \tl_if_empty_p:V \l_@@_inclassign_given_tl }
7972  { \tl_if_empty_p:V \l_@@_assign_given_tl }
7973  { \tl_if_empty_p:V \l_@@_inclassign_due_tl }
7974  { \tl_if_empty_p:V \l_@@_assign_due_tl }
7975  }{ #2 }
7976  }
```

\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.

```
7977 \newcommand\assignment@title[3]{
7978 \tl_if_empty:NTF \l_@@_inclassign_title_tl {
7979 \tl_if_empty:NTF \l_@@_assign_title_tl {
7980 #1
7981 }{
7982 #2\l_@@_assign_title_tl#3
7983 }
7984 }{
7985 #2\l_@@_inclassign_title_tl#3
7986 }
7987 }
```

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

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

```
7988 \newcommand\assignment@number{
7989 \int_compare:nNnTF \l_@@_inclassign_number_int = {-1} {
7990 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
7991 \arabic{assignment}
7992 } {
7993 \int_use:N \l_@@_assign_number_int
7994 }
7995 }{
7996 \int_use:N \l_@@_inclassign_number_int
7997 }
7998 }
```

(*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.

```
7999 \newenvironment{assignment}[1][]{
8000 \_@@_assignment_args:n { #1 }
8001 %\sref@target
8002 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8003 \global\stepcounter{assignment}
8004 }{
8005 \global\setcounter{assignment}{\int_use:N\l_@@_assign_number_int}
8006 }
8007 \setcounter{problem}{0}
8008 \renewcommand\prob@label[1]{\assignment@number.##1}
8009 \def\current@section@level{\document@hwexamtype}
8010 %\sref@label@id{\document@hwexamtype \thesection}
8011 \begin{@assignment}
8012 }{
8013 \end{@assignment}
8014 }
```

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

```
8015 \def\ass@title{
8016 {\protect\document@hwexamtype}~\arabic{assignment}
8017 \assignment@title{}{\;(}{)\;} -- \given@due{}{}
8018 }
8019 \ifmultiple
8020 \newenvironment{@assignment}{
8021 \bool_if:NTF \l_@@_assign_loadmodules_bool {
8022 \begin{sfragment}[loadmodules]{\ass@title}
8023 }{
8024 \begin{sfragment}{\ass@title}
8025 }
8026 }{
8027 \end{sfragment}
8028 }
```

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

```
8029 \else
8030 \newenvironment{@assignment}{
8031 \begin{center}\bf
8032 \Large\@title\strut\\
8033 \document@hwexamtype~\arabic{assignment}\assignment@title{\;}{:\;}{\\}
8034 \large\given@due{--\;}{\;--}
8035 \end{center}
8036 }{}
8037 \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.

```
8038 \keys_define:nn { hwexam / inclassignment } {
8039 %id   .str_set_x:N = \l_@@_assign_id_str,
8040 number  .int_set:N  = \l_@@_inclassign_number_int,
8041 title  .tl_set:N  = \l_@@_inclassign_title_tl,
8042 type  .tl_set:N  = \l_@@_inclassign_type_tl,
8043 given .tl_set:N  = \l_@@_inclassign_given_tl,
8044 due .tl_set:N  = \l_@@_inclassign_due_tl,
8045 mhrepos  .str_set_x:N = \l_@@_inclassign_mhrepos_str
8046 }
8047 \cs_new_protected:Nn \_@@_inclassignment_args:n {
8048 \int_set:Nn \l_@@_inclassign_number_int {-1}
8049 \tl_clear:N \l_@@_inclassign_title_tl
8050 \tl_clear:N \l_@@_inclassign_type_tl
8051 \tl_clear:N \l_@@_inclassign_given_tl
8052 \tl_clear:N \l_@@_inclassign_due_tl
8053 \str_clear:N \l_@@_inclassign_mhrepos_str
8054 \keys_set:nn { hwexam / inclassignment }{ #1 }
8055 }
8056 \_@@_inclassignment_args:n {}
8057
8058 \newcommand\inputassignment[2][]{
```

```
8059  \_@@_inclassignment_args:n { #1 }
8060  \str_if_empty:NTF \l_@@_inclassign_mhrepos_str {
8061  \input{#2}
8062  }{
8063  \stex_in_repository:nn{\l_@@_inclassign_mhrepos_str}{
8064  \input{\mhpath{\l_@@_inclassign_mhrepos_str}{#2}}
8065  }
8066  }
8067  \_@@_inclassignment_args:n {}
8068  }
8069  \newcommand\includeassignment[2][]{
8070  \newpage
8071  \inputassignment[#1]{#2}
8072  }
```

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

40.4   Typesetting Exams

\quizheading

```
8073  \ExplSyntaxOff
8074  \newcommand\quizheading[1]{%
8075  \def\@tas{#1}%
8076  \large\noindent NAME: \hspace{8cm}  MAILBOX:\\[2ex]%
8077  \ifx\@tas\@empty\else%
8078  \noindent TA:~\@for\@I:=\@tas\do{{\Large$\Box$}\@I\hspace*{1em}}\\[2ex]%
8079  \fi%
8080  }
8081  \ExplSyntaxOn
```

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

\testheading

```
8082
8083  \def\hwexamheader{\input{hwexam-default.header}}
8084
8085  \def\hwexamminutes{
8086  \tl_if_empty:NTF \testheading@duration {
8087  {\testheading@min}~\hwexam@minutes@kw
8088  }{
8089  \testheading@duration
8090  }
8091  }
8092
8093  \keys_define:nn { hwexam / testheading } {
8094  min   .tl_set:N  = \testheading@min,
8095  duration .tl_set:N  = \testheading@duration,
8096  reqpts .tl_set:N  = \testheading@reqpts,
8097  tools .tl_set:N  = \testheading@tools
8098  }
8099  \cs_new_protected:Nn \_@@_testheading_args:n {
8100  \tl_clear:N \testheading@min
8101  \tl_clear:N \testheading@duration
```

```
8102  \tl_clear:N \testheading@reqpts
8103  \tl_clear:N \testheading@tools
8104  \keys_set:nn { hwexam / testheading }{ #1 }
8105  }
8106  \newenvironment{testheading}[1][]{
8107  \_@@_testheading_args:n{ #1 }
8108  \newcount\check@time\check@time=\testheading@min
8109  \advance\check@time by -\theassignment@totalmin
8110  \newif\if@bonuspoints
8111  \tl_if_empty:NTF \testheading@reqpts {
8112  \@bonuspointsfalse
8113  }{
8114  \newcount\bonus@pts
8115  \bonus@pts=\theassignment@totalpts
8116  \advance\bonus@pts by -\testheading@reqpts
8117  \edef\bonus@pts{\the\bonus@pts}
8118  \@bonuspointstrue
8119  }
8120  \edef\check@time{\the\check@time}
8121
8122  \makeatletter\hwexamheader\makeatother
8123  }{
8124  \newpage
8125  }
```

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

`\testspace`

```
8126  \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}
```

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

`\testnewpage`

```
8127  \newcommand\testnewpage{\iftest\newpage\fi}
```

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

`\testemptypage`

```
8128  \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.

```
8129  ⟨@@=problems⟩
8130  \renewcommand\@problem[3]{
8131  \stepcounter{assignment@probs}
8132  \def\__problemspts{#2}
8133  \ifx\__problemspts\@empty\else
8134  \addtocounter{assignment@totalpts}{#2}
8135  \fi
8136  \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\f
8137  \xdef\correction@probs{\correction@probs & #1}%
8138  \xdef\correction@pts{\correction@pts & #2}
8139  \xdef\correction@reached{\correction@reached &}
```

276

8140 `}`
8141 ⟨@@=hwexam⟩

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

\correction@table   This macro generates the correction table

8142 `\newcounter{assignment@probs}`
8143 `\newcounter{assignment@totalpts}`
8144 `\newcounter{assignment@totalmin}`
8145 `\def\correction@probs{\correction@probs@kw}`
8146 `\def\correction@pts{\correction@pts@kw}`
8147 `\def\correction@reached{\correction@reached@kw}`
8148 `\stepcounter{assignment@probs}`
8149 `\newcommand\correction@table{`
8150 `\resizebox{\textwidth}{!}{%`
8151 `\begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%`
8152 `&\multicolumn{\theassignment@probs}{c||}%|`
8153 `{\footnotesize\correction@forgrading@kw} &\\\hline`
8154 `\correction@probs & \correction@sum@kw & \correction@grade@kw\\\hline`
8155 `\correction@pts &\theassignment@totalpts & \\\hline`
8156 `\correction@reached & & \\[.7cm]\hline`
8157 `\end{tabular}}}`
8158 ⟨/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}
```

277

Chapter 41

References

EdN:22

²²

[Bus+04]   Stephen Buswell et al. *The Open Math Standard, Version 2.0.* Tech. rep. The OpenMath Society, 2004. URL: http://www.openmath.org/standard/om20.

[CR99]   David Carlisle and Sebastian Rathz. *The* graphicxl *package.* Part of the TeX distribution. The Comprehensive TeX Archive Network. 1999. URL: https://www.tug.org/texlive/devsrc/Master/texmf-dist/doc/latex/graphics/graphicx.pdf.

[DCM03]   The DCMI Usage Board. *DCMI Metadata Terms.* DCMI Recommendation. Dublin Core Metadata Initiative, 2003. URL: http://dublincore.org/documents/dcmi-terms/.

[Koh06]   Michael Kohlhase. *OMDoc – An open markup format for mathematical documents [Version 1.2].* LNAI 4180. Springer Verlag, Aug. 2006. URL: http://omdoc.org/pubs/omdoc1.2.pdf.

[LMH]   *LMH Scripts.* URL: https://github.com/sLaTeX/lmhtools.

[MMT]   *MMT – Language and System for the Uniform Representation of Knowledge.* Project web site. URL: https://uniformal.github.io/ (visited on 01/15/2019).

[MRK18]   Dennis Müller, Florian Rabe, and Michael Kohlhase. "Theories as Types". In: *9th International Joint Conference on Automated Reasoning.* Ed. by Didier Galmiche, Stephan Schulz, and Roberto Sebastiani. Springer Verlag, 2018. URL: https://kwarc.info/kohlhase/papers/ijcar18-records.pdf.

[Rab15]   Florian Rabe. "The Future of Logic: Foundation-Independence". In: *Logica Universalis* 10.1 (2015). 10.1007/s11787-015-0132-x; Winner of the Contest "The Future of Logic" at the World Congress on Universal Logic, pp. 1–20.

[RK13]   Florian Rabe and Michael Kohlhase. "A Scalable Module System". In: *Information & Computation* 0.230 (2013), pp. 1–54. URL: https://kwarc.info/frabe/Research/mmt.pdf.

[RT]   *sLaTeX/RusTeX.* URL: https://github.com/sLaTeX/RusTeX (visited on 04/22/2022).

---

²²EDNOTE: we need an un-numbered version sfragment*

[SIa]   *sLaTeX/sTeX-IDE*. URL: https://github.com/slatex/sTeX-IDE (visited on 04/22/2022).

[SIb]   *sLaTeX/stexls-vscode-plugin*. URL: https://github.com/slatex/stexls-vscode-plugin (visited on 04/22/2022).

[SLS]   *sLaTeX/stexls*. URL: https://github.com/slatex/stexls (visited on 04/22/2022).

[ST]   *sTeX – An Infrastructure for Semantic Preloading of LaTeX Documents*. URL: https://ctan.org/pkg/stex (visited on 04/22/2022).

[sTeX]   *sTeX: A semantic Extension of TeX/LaTeX*. URL: https://github.com/sLaTeX/sTeX (visited on 05/11/2020).

[Tana]   Till Tantau. *beamer – A LaTeX class for producing presentations and slides*. URL: http://ctan.org/pkg/beamer (visited on 01/07/2014).

[Tanb]   Till Tantau. *User Guide to the Beamer Class*. URL: http://ctan.org/macros/latex/contrib/beamer/doc/beameruserguide.pdf.

[TL]   *TeX Live*. URL: http://www.tug.org/texlive/ (visited on 12/11/2012).