The sTeX3 Package [*]

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

2022-04-08

Abstract

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

- 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 2022-04-08)

i

Contents

Part I
Manual

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

> Boxes like this one explain how some sTeX concept relates to the Mmt/OMDoc system, philosophy or language.

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 to use semantic annotations in LaTeX documents,

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

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

Chapter 2

Quickstart

2.1 Setup

2.1.1 The sTeX IDE

TODO: VSCode Plugin

2.1.2 Manual Setup

Foregoing on the sTeX IDE, we will need several pieces of software; namely:

- **The sTeX-Package** available here.

  sTeX is also available on CTAN and in TeXLive.

- 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).

- **The Mmt System** available here[1]. 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.

---

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

2.2 A First sTeX Document

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

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

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

Compiling this document with `pdflatex` should yield the output

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

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

Theorem 0.2. The geometric series converges towards 1.

Feel free to 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 now:

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

smodule First, we open a new *module* called `GeometricSeries`. This module is assigned a *globally unique* identifier (URI), which (depending on your pdf viewer) 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` in the `smglom/calculus`-archive, and `realarith` in the `smglom/arithmetics`-archive. 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 desired module available. Additionally, they "export" these symbols to all further modules which include the *current* module – i.e. if in some future module we would put **\importmodule{GeometricSeries}**, we would also have *\infinitesum* etc. at our disposal.

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

```
\symdef{GeometricSeries}[name=geometric-series]{\comp{S}}
```

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

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

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

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

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

```
The \definame{geometricSeries} 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. 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.

\definame
\definiendum
The sdefinition-environment provides two additional macros, \definame and \definiendum which behave similar to \symname and \symref, but explicitly mark the symbols as *being defined* in this environment, to allow for special highlighting.

```
\[\defeq{\geometricSeries}{\definiens{
    \infinitesum{\svar{n}}{1}{
        \realdivide[frac]{1}{
            \realpower{2}{\svar{n}}
    }}
}}.\]
```
The next snippet – set in a math environment – uses several semantic macros imported from (or recursively via) series and realarithmetics, such as \defeq, \infinitesum, etc. In math mode, using a semantic macro inserts its (default) definition. A semantic macro can have several notations – in that case, we can explicitly choose a specific notation by providing its identifier as an optional argument; e.g. \realdivide[frac]{a}{b} will use the explicit notation named frac of the semantic macro \realdivide, which yields $\frac{a}{b}$ instead of a/b.

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

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

2.2.1 OMDoc/xhtml Conversion

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

TODO VSCode Plugin

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

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

...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>
```

---

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

...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:

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

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

- 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.

- 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.

- 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 syntax of OPENMATH.

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 three 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.

3.2.2   The Structure of sTeX Archives

An sTeX archive `group/name` needs to be 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 this 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 "encyclopedic" symbol documentations

- `/source/def/` – definitions

- `/source/ex/` – examples

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

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

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

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

- `/source/pic/` – image files.

3.2.3 MANIFEST.MF-Files

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

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.

In the majority of cases \inputref is likely to be preferred over \mhinput.

\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.

\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`.

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.

Will throw 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.

3.3 Module, Symbol and Notation Declarations

3.3.1 The `smodule`-Environment

smodule A new module is declared using the basic syntax

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

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

The `smodule`-environment takes several optional 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.

An SᴛᴇX module corresponds to an Mᴍᴛ/OMDᴏᴄ *theory*. As such it gets assigned a module URI (*universal resource identifier*) of the form `<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.

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

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

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

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

Example 3

Input:

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

Output:

> Given a foo, we can...

.

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

Example 4

Input:

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

Output:

> this is a symbol taking two arguments.

.

`\notation` In that case, 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→ Mмт/OMDoc as `OMA`-terms with head `<OMS name="...?binarysymbol"/>`.
> ↝T↝ Semantic macros with no arguments correspond to `OMS` directly.

`\comp` Unfortunately, we have no highlighting whatsoever now. That is because we need to tell STEX explicitly which parts of the notation are *notation components* which *should* be highlighted. 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`.

16

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 maningful 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.

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

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

Operator Notations

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

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

Example 8

Input:

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

Output:

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

.

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

3.3.3   Argument Types

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

b-Type Arguments

A `b`-type 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→ `b`-type arguments behave exactly like `i`-type arguments within TeX, but applica-
> —M→ tions of binding operators, i.e. symbols with `b`-type arguments, are translated to
> ⤳T⤳ `OMBIND`-terms in OMDoc/MMT, rather than `OMA`.

Fo 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.

a-Type Arguments

`a`-type 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. `a`-type 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 `a`-type 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 `a`-type 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, assoiative 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 $a \neq b \wedge a \neq c \wedge a \neq d \wedge b \neq c \wedge b \neq d \wedge c \neq d$

B-Type Arguments

Finally, B-type 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
> —M→ OMDoc/Mmt constants.
> ⤳T⤳ Correspondingly, the name "type" should be taken with a grain of salt, since
>   OMDoc/Mmt– being foundation-independent – does not a priori implement a
>   fixed typing system.

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

Example 13

Input:

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

Output:

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

.

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

Example 14

Input:

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

Output:

The successor operation $\mathbb{N}{\rightarrow}\mathbb{N}$ 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 in theory straight-forward:

Example 15

Input:

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

Output:

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

.

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

Example 16

Input:

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

Output:

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

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

Example 17

Input:

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

Output:

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

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

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

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

Example 18

Input:

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

Output:

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

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

`\infprec`
`\neginfprec`

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

More precisely, each notation takes

1. One *operator precedence* and

2. one *argument precedence* for each argument.

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

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

When SsTeX 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. SsTeX starts out with $p_d =$ `\infprec`.

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

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

4. Next, SsTeX 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 SsTeX again inserts no parentheses.

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

7. Next, SsTeX 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 SsTeX 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.

<u>`\svar`</u> 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.

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:

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

·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

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 $lcm(a, b)$ in english, but is called *kleinstes gemeinsames Vielfaches* in german and consequently denoted as $kgV(a, b)$ 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[Some/Archive]{path?ModuleName}` is only allowed within an `smodule`-environment and makes the symbols declared therein available. Additionally the content of `ModuleName` will be exported if the current module is imported somewhere else via `\importmodule`.

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

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

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

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

- If `\begin{smodule}{Foo}` occurs in a file `/path/to/file/Foo[.⟨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

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

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 ⟨$M, \circ, e$⟩ with $\circ : M \times M \to M$ and $e \in M$ such that...

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

- A *partial order* is a structure ⟨$S, \leq$⟩ 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, ratherer, structure *signatures*) in order to talk about (concrete or variable) *particular* monoids, topological spaces, partial orders etc.

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

How this works is again best demonstrated by example:

Example 24

Input:

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

Output:

A monoid is...

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

Example 25

Input:

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

Output:

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

.

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

Example 26

Input:

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

Output:

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

.

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

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

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

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

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

\varinstantiate   The answer is that we can also instantiate a `mathstructure` with a *variable*. The syntax of \varianstantiate 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:

31

```
1  \varinstantiate{varM}{}{monoid}{M}
2
3  A \symname{monoid} is a structure
4  $\varM!:=\mathstruct{\varM{universe},\varM{op}!,\varM{unit}}$
5  such that
6  $\varM{op}!:\funtype{\varM{universe},\varM{universe}}{\varM{universe}}$
7   and...
8
9  \varinstantiate{varMb}{universe = Int}{monoid}{M_2}
10
11  \noindent Let $\varMb!:=\mathstruct{\varMb{universe},\varMb{op}!,\varMb{unit}}$
12  a \symname{monoid} on $\Int$...
```

Output:

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

.

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

3.4.4   The `copymodule` Environment

TODO: explain

Given modules:

Example 28

Input:

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

Output:

.

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

Example 29

Input:

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

Output:

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

.

TODO: explain donotclone

3.4.5 The `interpretmodule` Environment

TODO: explain

Example 30

Input:

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

Output:

.

3.5 Primitive Symbols (The sTeX Metatheory)

TODO: metatheory documentation

Chapter 4

Using sTeX Symbols

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

4.1 `\symref` and its variants

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

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

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

> **Example 31**
> Input:
>
> ```
> 1  \symdef{Nat}[
> 2      name=natural-number,
> 3      type=\set
> 4  ]{\comp{\mathbb{N}}}
> 5
> 6  A \symname{Nat} is...
> ```
>
> Output:
>
> > A natural number is...

.

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

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

Example 32
Input:

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

Output:

```
  Natural numbers are...
```

.

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

4.2   Marking Up Text and On-the-Fly Notations

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

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

Example 33
Input:

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

Output:

```
  The sum of n and m is...
```

36

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

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

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

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

Example 34

Input:

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

Output:

> Addition is...

.

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

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

Example 35

Input:

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

Output:

> adding k yields...

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

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

Example 36

Input:

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

Output:

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

.

4.3   Referencing Symbols and Statements

TODO: references documentation

Chapter 5

sTeX Statements

5.1 Definitions, Theorems, Examples, Paragraphs

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

The corresponding environments for that are:

- `sdefinition` for definitions,

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

- `sexample` for examples, and

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

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

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

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

Example 37

Input:

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

Output:

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

.

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

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

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

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

Example 38

Input:

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

Output:

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

.

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

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

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

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

5.2   Proofs

TODO

---

[2]Of course, SТEX 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

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 suppposed to look like.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Chapter 7

Additional Packages

TODO: tikzinput documentation

7.1 Modular Document Structuring

TODO: document-structure documentation

7.2 Slides and Course Notes

TODO: notesslides documentation

7.3 Homework, Problems and Exams

TODO: problem documentation
TODO: hwexam documentation

Part II
Documentation

Chapter 8

sTEX-Basics

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

8.1 Macros and Environments

\sTeX Both print this sTEX logo.
\stex

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

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

8.1.1 HTML Annotations

\if@latexml LaTeX2e conditional for LATEXML

\latexml_if_p: ⋆ LaTeX3 conditionals for LATEXML.
\latexml_if:TF ⋆

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

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

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

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

Annotates the HTML generated by ⟨*content*⟩ with

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

`\stex_annotate_invisible:n` adds the attributes

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

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

`stex_annotate_env`	`\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.

\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
\mhinput

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

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

\addmhbibresource

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

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

\libinput

\libinput{⟨*filename*⟩}

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

\libusepackage

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

Like \libinput, but looks for .sty-files and calls \usepackage[\meta{args}]\Arg{filename} instead of \input.
  Throws an error, if none or more than one suitable package file is found.

\mhgraphics
\cmhgraphics

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

\lstinputmhlisting
\clstinputmhlisting

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

Chapter 10

sTEX-References

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

10.1   Macros and Environments

\STEXreftitle

\STEXreftitle{⟨*some title*⟩}

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

\stex_get_document_uri:

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

\l_stex_current_docns_str

Stores its result in \l_stex_current_docns_str

\stex_get_document_url:

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

\l_stex_current_docurl_str

Stores its result in \l_stex_current_docurl_str

10.1.1   Setting Reference Targets

\stex_ref_new_doc_target:n

\stex_ref_new_doc_target:n{⟨*id*⟩}

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

\stex_ref_new_sym_target:n

\stex_ref_new_sym_target:n{⟨*uri*⟩}

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

10.1.2 Using References

$\overline{\texttt{\textbackslash sref}}$ `\sref[⟨opt-args⟩]{⟨id⟩}`

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

$\overline{\texttt{\textbackslash srefsym}}$ `\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*⟩.

$\overline{\texttt{\textbackslash srefsymuri}}$ `\srefsymuri{⟨URI⟩}{⟨text⟩}`

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

Chapter 11

sTEX-Modules

This sub package contains code related to Modules

11.1 Macros and Environments

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

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

name The *name* of the module,

ns the *namespace* in field `ns`,

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

lang the module's *language*,

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

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

meta the metatheory of the module.

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

`\c_stex_module_<URI>_constants`

The names of all constants declared in the module

`\c_stex_module_<URI>_constants`

The full URIs of all modules imported in this module

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

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

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

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

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

`\stex_add_to_current_module:n`
`\STEXexport`

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

`\stex_add_constant_to_current_module:n`

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

`\stex_add_import_to_current_module:n`

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

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

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

\stex_modules_current_namespace:

Computes the current namespace as follows:

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

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

11.1.1 The `smodule` environment

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

\STEXModule `\STEXModule {⟨fragment⟩}`

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

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

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

Chapter 12

sTeX-Module Inheritance

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

12.1 Macros and Environments

12.1.1 SMS Mode

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

`\g_stex_smsmode_allowedmacros_tl`

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

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

`\g_stex_smsmode_allowedmacros_escape_tl`

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

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

`\g_stex_smsmode_allowedenvs_seq`

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

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

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

Tests whether SMS mode is currently active.

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

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

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

12.1.2 Imports and Inheritance

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

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

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

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

\stex_import_module_uri:nn

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

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

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

1. If ⟨archive-ID⟩ is empty:

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

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

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

2. Otherwise:

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

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

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

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

\l_stex_import_name_str
\l_stex_import_archive_str
\l_stex_import_path_str
\l_stex_import_ns_str

stores the result in these four variables.

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

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

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

Chapter 13

sTeX-Symbols

Code related to symbol declarations and notations

13.1 Macros and Environments

`\symdecl`

`\symdecl{⟨macroname⟩}[⟨args⟩]`

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

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

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

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

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

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

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

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

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

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

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

- `name` (string),

- `module` (string),

- `notations` (sequence of strings; initially empty),

- `local` (boolean),

- `type` (token list),

- `args` (string of `is`, `as` and `bs`),

- `arity` (integer string),

- `assocs` (integer string; number of associative arguments),

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

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

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

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

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

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

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

- `symbol` (URI string),

- `language` (string),

- `variant` (string),

- `opprec` (integer string),

- `argprecs` (sequence of integer strings)

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

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

Chapter 14

sTEX-Terms

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

14.1 Macros and Environments

\STEXsymbol

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

\symref

\symref{⟨*symbol*⟩}{⟨*text*⟩}

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

\stex_invoke_symbol:n

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

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

_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

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

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

Contents

17.1 Introduction

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

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

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

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

---

[2]EDNOTE: talk a bit more about proofs and their structure,... maybe copy from OMDoc spec.

17.2 The User Interface

17.2.1 Package Options

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

17.2.2 Proofs and Proof steps

sproof The `proof` environment is the main container for proofs. It takes an optional `KeyVal` argument that allows to specify the `id` (identifier) and `for` (for which assertion is this a proof) keys. The regular argument of the `proof` environment contains an introductory comment, that may be used to announce the proof style. The `proof` environment contains a sequence of `\step`, `proofcomment`, and `pfcases` environments that are used to markup the proof steps. The `proof` environment has a variant `Proof`, which does not use the proof end marker. This is convenient, if a proof ends in a case distinction, which brings

sProof it's own proof end marker with it. The `Proof` environment is a variant of `proof` that does not mark the end of a proof with a little box; presumably, since one of the subproofs already has one and then a box supplied by the outer proof would generate an otherwise

\spfidea empty line. 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 `KeyVal` argument as `sproof` and another one: a natural language text that sketches the proof.

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

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

17.2.3 Justifications

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

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

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

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

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

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

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

1.3. $n > 1$:

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

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

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

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

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

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

□

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

17.2.4 Proof Structure

subproof

The `pfcases` environment is used to mark up a subproof. This environment takes an optional `KeyVal` argument for semantic annotations and a second argument that allows

method

to specify an introductory comment (just like in the `proof` environment). The `method` key can be used to give the name of the proof method executed to make this subproof.

spfcases

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

spfcase

The content of a `pfcases` environment are a sequence of case proofs marked up in the `pfcase` environment, which takes an optional `KeyVal` argument for semantic annotations. The second argument is used to specify the the description of the case under consideration. The content of a `pfcase` environment is the same as that of a `proof`, i.e.

\spfcasesketch

`step`s, `proofcomment`s, and `pfcases` environments. `\spfcasesketch` is a variant of the `spfcase` environment that takes the same arguments, but instead of the `spfsteps` in the body uses a third argument for a proof sketch.

sproofcomment

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

17.2.5 Proof End Markers

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

\sproofend

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

\sProofEndSymbol

`\sProofEndSymbol` configuration macro (e.g. by specifying `\sProofEndSymbol{q.e.d}`).

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

17.2.6 Configuration of the Presentation

EdN:3

Finally, we provide configuration hooks in Figure 1 for the keywords in proofs. These are mainly intended for package authors building on `statements`, e.g. for multi-language support.[3] The proof step labels can be customized via the `\pstlabelstyle` macro:

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

Figure 1: Configuration Hooks for Semantic Proof Markup

\pstlabelstyle

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

---

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

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

17.3   Limitations

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

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

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

Chapter 18

sTEX-Metatheory

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

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

18.1 Symbols

Part III

Extensions

Chapter 19

Tikzinput

19.1 Macros and Environments

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

Chapter 20

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

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

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

20.1 Introduction

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

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

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

source and the formatter does the copying during document formatting/presentation.[4]

20.2   The User Interface

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

20.2.1   Package and Class Options

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

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

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

20.2.2   Document Structure

document
\documentkeys
id

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

sfragment

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

id
creators
contributors
short
loadmodules

tional metadata argument has the keys `id` for an identifier, `creators` and `contributors` for the Dublin Core metadata [DCM03]; see [Koh20a] for details of the format. The `short` allows to give a short title for the generated section. If the title contains semantic macros, they need to be protected by `\protect`, and we need to give the `loadmodules` key it needs no value. For instance we would have

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

---

[4]EdNote: integrate with latexml's XMRef in the Math mode.

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

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

blindfragment

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

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

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

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

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

---

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

20.2.3 Ignoring Inputs

<div style="float:left">ignore
showignores</div>

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

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

narrative/content markup In sTeX we mark up narrative-structured documents. In the generated OMDoc documents we want to be able to cache content objects that are not directly visible. For instance in the `statements` package [Koh20d] we use the `\inlinedef` macro to mark up phrase-level definitions, which verbalize more formal definitions. The latter can be hidden by an ignore and referenced by the `verbalizes` key in `\inlinedef`.

<div style="float:left">\prematurestop

\afterprematurestop</div>

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

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

20.2.4 Structure Sharing

<div style="float:left">\STRlabel
\STRcopy</div>

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

<div style="float:left">\STRsemantics</div>

The `\STRlabel` macro has a variant `\STRsemantics`, where the label argument is optional, and which takes a third argument, which is ignored in LaTeX. This allows to specify the meaning of the content (whatever that may mean) in cases, where the source document is not formatted for presentation, but is transformed into some content markup format.[5]

EdN:5

20.2.5 Global Variables

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

<div style="float:left">\setSGvar
\useSGvar
\ifSGvar</div>

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

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

---

[5]EdNote: document LMID und LMXREf here if we decide to keep them.

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

20.2.6   Colors

\blue
\red
...
\black

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

20.3   Limitations

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

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

Chapter 21

NotesSlides – Slides and Course Notes

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

21.1 Introduction

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

21.2 The User Interface

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

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

21.2.1 Package Options

EdN:6   The `notesslides` class takes a variety of class options:[6]

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

81

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

- `showmeta`. If this is set, then the metadata keys are shown (see [Koh20b] for details and customization options).

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

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

21.2.2   Notes and Slides

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

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

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

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

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

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

Example 4: A typical Course Notes File

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

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

---

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

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

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

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

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

If we want to transclude a the contents of a file as a note, we can use a new variant
`\inputref*` of the `\inputref` macro from [KGA20]: `\inputref*{foo}` is equivalent to `\begin{note}\inputref{foo}\end{note}`.

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

21.2.3 Header and Footer Lines of the Slides

The default logo provided by the `notesslides` package is the sTEX logo it can be cus-
`\setslidelogo` tomized using `\setslidelogo{⟨logo name⟩}`.

The default footer line of the `notesslides` package mentions copyright and licensing. In the `beamer` class, `\source` stores the author's name as the copyright holder . By default it is *Michael Kohlhase* in the `notesslides` package since he is the main user and designer
`\setsource` of this package. `\setsource{⟨name⟩}` can change the writer's name. For licensing, we use the Creative Commons Attribuition-ShareAlike license by default to strengthen the public domain. If package `hyperref` is loaded, then we can attach a hyperlink to the
`\setlicensing` license logo. `\setlicensing[⟨url⟩]{⟨logo name⟩}` is used for customization, where ⟨*url*⟩ is optional.

21.2.4 Frame Images

Sometimes, we want to integrate slides as images after all – e.g. because we already have a PowerPoint presentation, to which we want to add sTEXnotes. In this case we
`\frameimage` can use `\frameimage[⟨opt⟩]{⟨path⟩}`, where ⟨*opt*⟩ are the options of `\includegraphics` from the `graphicx` package [CR99] and ⟨*path*⟩ is the file path (extension can be left off like in `\includegraphics`). We have added the `label` key that allows to give a frame
EdN:7 label that can be referenced like a regular `beamer` frame.[7]
`\mhframeimage` The `\mhframeimage` macro is a variant of `\frameimage` with repository support. Instead of writing

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

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

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

---

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

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

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

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

21.2.5   Colors and Highlighting

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

21.2.6   Front Matter, Titles, etc.

21.2.7   Excursions

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

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

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

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

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

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

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

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

21.2.8 Miscellaneous

21.3 Limitations

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

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

Chapter 22

`problem.sty`: An Infrastructure for formatting Problems

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

22.1 Introduction

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

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

22.2 The User Interface

22.2.1 Package Options

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

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

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

---

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

22.2.2 Problems and Solutions

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

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

Example 5: A marked up Problem

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

Problem 0.1 (Fitting Elefants)
How many Elefants can you fit into a Volkswagen beetle?

Hint: Think positively, this is simple!

Note: Justify your answer

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

Example 6: The Formatted Problem from Figure 5

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

may arise in grading.

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

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

22.2.3 Multiple Choice Blocks

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

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

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

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

See Figure **??** for an example

22.2.4 Including Problems

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

22.2.5 Reporting Metadata

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

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

22.3 Limitations

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

1. none reported yet

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

Problem 0.2 (Functions)

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

1. def

2. function

3. fun

4. public static void

Problem 0.3 (Functions)

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

1. def
   !

2. function
   that is for C and C++

3. fun
   that is for Standard ML

4. public static void
   that is for Java

Example 7: A Problem with a multiple choice block

Chapter 23

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

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

Contents

23.1 Introduction

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

23.2 The User Interface

23.2.1 Package and Class Options

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

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

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

23.2.2 Assignments

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

23.2.3 Typesetting Exams

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

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

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

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

23.2.4 Including Assignments

\inputassignment   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
number   in the included file). The keys number, title, type, given, and due are just as for the
title   assignment environment and (if given) overwrite the ones specified in the assignment
type   environment in the included file.
given
due

23.3 Limitations

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

1. none reported yet.

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

formats to

| Name: | Matriculation Number: |

320101 General Computer Science (Fall 2010)

2022-04-08

You have one hour (sharp) for the test;
Write the solutions to the sheet.
The estimated time for solving this exam is 58 minutes, leaving you 2 minutes for revising your exam.
You can reach 30 points if you solve all problems. You will only need 27 points for a perfect score, i.e. 3 points are bonus points.

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

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

prob.	0.1	0.2	0.3	1.1	2.1	2.2	2.3	3.1	3.2	3.3	Sum	grade
						To be used for grading, do not write here						
total				4	4	6	6	4	4	2	30	
reached												

good luck

Example 8: A generated test heading.

Part IV

Implementation

Chapter 24

sTeX -Basics Implementation

24.1 The sTeXDocument Class

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

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

24.2 Preliminaries

```
19  ⟨∗package⟩
20
21  %%%%%%%%%%%%  basics.dtx  %%%%%%%%%%%%
22
23  \RequirePackage{expl3,l3keys2e,ltxcmds,standalone}
24  \ProvidesExplPackage{stex}{2022/03/03}{3.1.0}{sTeX package}
25
26  \message{^^J
```

```
27  ******************************^^J
28  *~This~is~sTeX~version~3.1.0~*^^J
29  ******************************^^J
30  ^^J}
31
32  %\RequirePackage{morewrites}
33  %\RequirePackage{amsmath}
34
```

Package options:

```
35  \keys_define:nn { stex } {
36    debug     .clist_set:N  = \c_stex_debug_clist ,
37    lang      .clist_set:N  = \c_stex_languages_clist ,
38    mathhub   .tl_set_x:N   = \mathhub ,
39    usesms    .bool_set:N   = \c_stex_persist_mode_bool ,
40    writesms  .bool_set:N   = \c_stex_persist_write_mode_bool ,
41    image     .bool_set:N   = \c_tikzinput_image_bool,
42    unknown   .code:n       = {}
43  }
44  \ProcessKeysOptions { stex }
```

\stex The sTeXlogo:
\sTeX

```
45  \RequirePackage{xspace}
46  \protected\def\stex{
47    \@ifundefined{texorpdfstring}{\let\texorpdfstring\@firstoftwo}{}
48    \texorpdfstring{\raisebox{-.5ex}S\kern-.5ex\TeX}{sTeX}\xspace
49  }
50  \let\sTeX\stex
```

(*End definition for* \stex *and* \sTeX. *These functions are documented on page* *46.*)

24.3   Messages and logging

```
51  ⟨@@=stex_log⟩
```

Warnings and error messages

```
52  \msg_new:nnn{stex}{error/unknownlanguage}{
53    Unknown~language:~#1
54  }
55  \msg_new:nnn{stex}{warning/nomathhub}{
56    MATHHUB~system~variable~not~found~and~no~
57    \detokenize{\mathhub}-value~set!
58  }
59  \msg_new:nnn{stex}{error/deactivated-macro}{
60    The~\detokenize{#1}~command~is~only~allowed~in~#2!
61  }
```

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

```
62  \cs_new_protected:Nn \stex_debug:nn {
63    \clist_if_in:NnTF \c_stex_debug_clist { all } {
64      \msg_set:nnn{stex}{debug / #1}{
65        \\Debug~#1:~#2\\
66      }
67      \msg_none:nn{stex}{debug / #1}
68    }{
```

```
69        \clist_if_in:NnT \c_stex_debug_clist { #1 } {
70          \msg_set:nnn{stex}{debug / #1}{
71            \\Debug~#1:~#2\\
72          }
73          \msg_none:nn{stex}{debug / #1}
74        }
75      }
76    }
```

(*End definition for* `\stex_debug:nn`. *This function is documented on page 46.*)

Redirecting messages:

```
77 \clist_if_in:NnTF \c_stex_debug_clist {all} {
78     \msg_redirect_module:nnn{ stex }{ none }{ term }
79 }{
80   \clist_map_inline:Nn \c_stex_debug_clist {
81     \msg_redirect_name:nnn{ stex }{ debug / ##1 }{ term }
82   }
83 }
84
85 \stex_debug:nn{log}{debug~mode~on}
```

24.4   HTML Annotations

```
86 ⟨@@=stex_annotate⟩
```

`\l_stex_html_arg_tl`
`\c_stex_html_emptyarg_tl`

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

```
87 \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`

```
88 \cs_new_protected:Nn \_stex_html_checkempty:n {
89   \tl_set:Nn \l_stex_html_arg_tl { #1 }
90   \tl_if_empty:NT \l_stex_html_arg_tl {
91     \tl_set_eq:NN \l_stex_html_arg_tl \c_stex_html_emptyarg_tl
92   }
93 }
```

(*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

```
94 \bool_new:N \_stex_html_do_output_bool
95 \bool_set_true:N \_stex_html_do_output_bool
96
97 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
98   \bool_if:nTF \_stex_html_do_output_bool
99     \prg_return_true: \prg_return_false:
100 }
```

(*End definition for* `\stex_if_do_html:TF`. *This function is documented on page 46.*)

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

```
101 \cs_new_protected:Nn \stex_suppress_html:n {
102   \exp_args:Nne \use:nn {
103     \bool_set_false:N \_stex_html_do_output_bool
104     #1
105   }{
106     \stex_if_do_html:T {
107       \bool_set_true:N \_stex_html_do_output_bool
108     }
109   }
110 }
```

(*End definition for* `\stex_suppress_html:n`*. This function is documented on page* *46.*)

`\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.

```
111 \tl_if_exist:NF\stex@backend{
112   \ifcsname if@rustex\endcsname
113     \def\stex@backend{rustex}
114   \else
115     \ifcsname if@latexml\endcsname
116       \def\stex@backend{latexml}
117     \else
118       \def\stex@backend{pdflatex}
119     \fi
120   \fi
121 }
122 \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* *47.*)

24.5   Babel Languages

```
123 ⟨@@=stex_language⟩
```

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

```
124 \prop_const_from_keyval:Nn \c_stex_languages_prop {
125   en = english ,
126   de = ngerman ,
127   ar = arabic ,
128   bg = bulgarian ,
129   ru = russian ,
130   fi = finnish ,
131   ro = romanian ,
132   tr = turkish ,
133   fr = french
134 }
135
136 \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop {
137   english   = en ,
138   ngerman   = de ,
```

```
139   arabic   = ar ,
140   bulgarian = bg ,
141   russian   = ru ,
142   finnish   = fi ,
143   romanian  = ro ,
144   turkish   = tr ,
145   french    = fr
146 }
147 % todo: chinese simplified (zhs)
148 %        chinese traditional (zht)
```

(*End definition for* \c_stex_languages_prop *and* \c_stex_language_abbrevs_prop. *These variables are documented on page 47.*)

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

```
149 \clist_if_empty:NF \c_stex_languages_clist {
150   \clist_clear:N \l_tmpa_clist
151   \clist_map_inline:Nn \c_stex_languages_clist {
152     \prop_get:NnNTF \c_stex_languages_prop { #1 } \l_tmpa_str {
153       \clist_put_right:No \l_tmpa_clist \l_tmpa_str
154     } {
155       \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
156     }
157   }
158   \stex_debug:nn{lang} {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
159   \RequirePackage[\clist_use:Nn \l_tmpa_clist,]{babel}
160 }
161
162 \AtBeginDocument{
163   \stex_html_backend:T {
164     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
165     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
166     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
167     \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
168     \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
169       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
170       \stex_debug:nn{basics} {Language~\l_tmpa_str~
171         inferred~from~file~name}
172       \stex_annotate_invisible:nnn{language}{ \l_tmpa_str }{}
173     }
174   }
175 }
```

24.6   Persistence

```
176 ⟨@@=stex_persist⟩
177 \bool_if:NTF \c_stex_persist_mode_bool {
178   \def \stex_persist:n #1 {}
179   \def \stex_persist:x #1 {}
180 }{
181   \bool_if:NTF \c_stex_persist_write_mode_bool {
182   \iow_new:N \c__stex_persist_iow
183   \iow_open:Nn \c__stex_persist_iow{\jobname.sms}
184   \AtEndDocument{
```

99

```
185      \iow_close:N \c__stex_persist_iow
186    }
187    \cs_new_protected:Nn \stex_persist:n {
188      \tl_set:Nn \l_tmpa_tl { #1 }
189      \regex_replace_all:nnN { \cP\# } { \cO\# } \l_tmpa_tl
190      \exp_args:NNo \iow_now:Nn \c__stex_persist_iow \l_tmpa_tl
191    }
192    \cs_generate_variant:Nn \stex_persist:n {x}
193  }{
194      \def \stex_persist:n #1 {}
195      \def \stex_persist:x #1 {}
196  }
197 }
```

24.7 Auxiliary Methods

```
198 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
199    \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
200    \def#1{
201      \msg_error:nnnn{stex}{error/deactivated-macro}{\detokenize{#1}}{#2}
202    }
203 }
```

(*End definition for* \stex_deactivate_macro:Nn. *This function is documented on page 47.*)

```
204 \cs_new_protected:Nn \stex_reactivate_macro:N {
205    \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname
206 }
```

(*End definition for* \stex_reactivate_macro:N. *This function is documented on page 47.*)

```
207 \protected\def\ignorespacesandpars{
208    \begingroup\catcode13=10\relax
209    \@ifnextchar\par{
210      \endgroup\expandafter\ignorespacesandpars\@gobble
211    }{
212      \endgroup
213    }
214 }
215
216 \cs_new_protected:Nn \stex_copy_control_sequence:NNN {
217    \tl_set:Nx \_tmp_args_tl {\cs_argument_spec:N #2}
218    \exp_args:NNo \tl_remove_all:Nn \_tmp_args_tl \c_hash_str
219    \int_set:Nn \l_tmpa_int {\tl_count:N \_tmp_args_tl}
220
221    \tl_clear:N \_tmp_args_tl
222    \int_step_inline:nn \l_tmpa_int {
223      \tl_put_right:Nx \_tmp_args_tl {{\exp_not:n{####}\exp_not:n{##1}}}
224    }
225
226    \tl_set:Nn #3 {\cs_generate_from_arg_count:NNnn #1 \cs_set:Npn}
```

```
227    \tl_put_right:Nx #3 { {\int_use:N \l_tmpa_int}{
228        \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
229        \exp_after:wN\exp_after:wN\exp_after:wN {
230          \exp_after:wN #2 \_tmp_args_tl
231        }
232    }}
233 }
234 \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {cNN}
235 \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {NcN}
236 \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {ccN}
```

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

\MMTrule
```
237 \NewDocumentCommand \MMTrule {m m}{
238    \seq_set_split:Nnn \l_tmpa_seq , {#2}
239    \int_zero:N \l_tmpa_int
240    \stex_annotate_invisible:nnn{mmtrule}{scala://#1}{
241      $\seq_map_inline:Nn \l_tmpa_seq {
242        \int_incr:N \l_tmpa_int
243        \stex_annotate:nnn{arg}{i\int_use:N \l_tmpa_int}{##1}
244      }$
245    }
246 }
247
248 \NewDocumentCommand \MMTinclude {m}{
249    \stex_annotate_invisible:nnn{import}{#1}{}
250 }
251 ⟨/package⟩
```

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

Chapter 25

STEX
-MathHub Implementation

```
252 ⟨∗package⟩
253
254 %%%%%%%%%%%%   mathhub.dtx   %%%%%%%%%%%%
255
256 ⟨@@=stex_path⟩
```

Warnings and error messages
```
257 \msg_new:nnn{stex}{error/norepository}{
258   No~archive~#1~found~in~#2
259 }
260 \msg_new:nnn{stex}{error/notinarchive}{
261   Not~currently~in~an~archive,~but~\detokenize{#1}~
262   needs~one!
263 }
264 \msg_new:nnn{stex}{error/nofile}{
265   \detokenize{#1}~could~not~find~file~#2
266 }
267 \msg_new:nnn{stex}{error/twofiles}{
268   \detokenize{#1}~found~two~candidates~for~#2
269 }
```

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

```
270 \cs_new_protected:Nn \stex_path_from_string:Nn {
271   \str_set:Nx \l_tmpa_str { #2 }
272   \str_if_empty:NTF \l_tmpa_str {
273     \seq_clear:N #1
274   }{
275     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
276     \sys_if_platform_windows:T{
277       \seq_clear:N \l_tmpa_tl
```

```
278          \seq_map_inline:Nn #1 {
279            \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
280            \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
281          }
282          \seq_set_eq:NN #1 \l_tmpa_tl
283        }
284        \stex_path_canonicalize:N #1
285      }
286    }
287
```

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

<div style="color:red">\stex_path_to_string:NN</div>
<div style="color:red">\stex_path_to_string:N</div>

```
288    \cs_new_protected:Nn \stex_path_to_string:NN {
289      \exp_args:NNe \str_set:Nn #2 { \seq_use:Nn #1 / }
290    }
291
292    \cs_new:Nn \stex_path_to_string:N {
293      \seq_use:Nn #1 /
294    }
```

(*End definition for* `\stex_path_to_string:NN` *and* `\stex_path_to_string:N`*. These functions are documented on page* *48.*)

<div style="color:red">\c__stex_path_dot_str</div>
<div style="color:red">\c__stex_path_up_str</div>

. and .., respectively.

```
295    \str_const:Nn \c__stex_path_dot_str {.}
296    \str_const:Nn \c__stex_path_up_str {..}
```

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

<div style="color:red">\stex_path_canonicalize:N</div> Canonicalizes the path provided; in particular, resolves . and .. path segments.

```
297    \cs_new_protected:Nn \stex_path_canonicalize:N {
298      \seq_if_empty:NF #1 {
299        \seq_clear:N \l_tmpa_seq
300        \seq_get_left:NN #1 \l_tmpa_tl
301        \str_if_empty:NT \l_tmpa_tl {
302          \seq_put_right:Nn \l_tmpa_seq {}
303        }
304        \seq_map_inline:Nn #1 {
305          \str_set:Nn \l_tmpa_tl { ##1 }
306          \str_if_eq:NNF \l_tmpa_tl \c__stex_path_dot_str {
307            \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
308              \seq_if_empty:NTF \l_tmpa_seq {
309                \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
310                  \c__stex_path_up_str
311                }
312              }{
313                \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
314                \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
315                  \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
316                    \c__stex_path_up_str
317                  }
318                }{
```

```
319                        \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
320                    }
321                }
322            }{
323                \str_if_empty:NF \l_tmpa_tl {
324                    \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
325                }
326            }
327        }
328    }
329    \seq_gset_eq:NN #1 \l_tmpa_seq
330    }
331 }
```

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

`\stex_path_if_absolute_p:N`
`\stex_path_if_absolute:NTF`

```
332 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
333    \seq_if_empty:NTF #1 {
334        \prg_return_false:
335    }{
336        \seq_get_left:NN #1 \l_tmpa_tl
337        \sys_if_platform_windows:TF{
338            \str_if_in:NnTF \l_tmpa_tl {:}{
339                \prg_return_true:
340            }{
341                \prg_return_false:
342            }
343        }{
344            \str_if_empty:NTF \l_tmpa_tl {
345                \prg_return_true:
346            }{
347                \prg_return_false:
348            }
349        }
350    }
351 }
```

(End definition for `\stex_path_if_absolute:NTF`. *This function is documented on page 48.)*

25.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```
352 \str_new:N\l_stex_kpsewhich_return_str
353 \cs_new_protected:Nn \stex_kpsewhich:n {
354    \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
355    \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
356    \tl_trim_spaces:N \l_stex_kpsewhich_return_str
357 }
```

(End definition for `\stex_kpsewhich:n`. *This function is documented on page 48.)*

We determine the PWD

```
358 \sys_if_platform_windows:TF{
359   \begingroup\escapechar=-1\catcode`\\=12
360   \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
361   \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
362   \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_
363 }{
364   \stex_kpsewhich:n{-var-value~PWD}
365 }
366
367 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
368 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
369 \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 48.*)

25.3 File Hooks and Tracking

```
370 ⟨@@=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 SIEX-purposes.

keeps track of file changes

```
371 \seq_gclear_new:N\g__stex_files_stack
```

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

```
372 \str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
373 \stex_path_from_string:Nn \c_stex_mainfile_seq
374   \c_stex_mainfile_str
```

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

```
375 \seq_gclear_new:N\g_stex_currentfile_seq
```

(*End definition for* `\g_stex_currentfile_seq`. *This variable is documented on page 49.*)

```
376 \cs_new_protected:Nn \stex_filestack_push:n {
377   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
378   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
379     \stex_path_from_string:Nn\g_stex_currentfile_seq{
380       \c_stex_pwd_str/#1
381     }
382   }
383   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
384   \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
385 }
```

105

(End definition for `\stex_filestack_push:n`*. This function is documented on page [49](#).)*

\stex_filestack_pop:

```
386 \cs_new_protected:Nn \stex_filestack_pop: {
387   \seq_if_empty:NF\g__stex_files_stack{
388     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
389   }
390   \seq_if_empty:NTF\g__stex_files_stack{
391     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
392   }{
393     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
394     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
395   }
396 }
```

(End definition for `\stex_filestack_pop:`*. This function is documented on page [49](#).)*

Hooks for the current file:

```
397 \AddToHook{file/before}{
398   \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
399 }
400 \AddToHook{file/after}{
401   \stex_filestack_pop:
402 }
```

25.4   MathHub Repositories

```
403 ⟨@@=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.

```
404 \str_if_empty:NTF\mathhub{
405   \sys_if_platform_windows:TF{
406     \begingroup\escapechar=-1\catcode`\\=12
407     \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str MATHHUB\c_percent_str}
408     \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
409     \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_ste
410   }{
411     \stex_kpsewhich:n{-var-value~MATHHUB}
412   }
413   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
414
415   \str_if_empty:NT \c_stex_mathhub_str {
416     \sys_if_platform_windows:TF{
417       \begingroup\escapechar=-1\catcode`\\=12
418       \exp_args:Nx\stex_kpsewhich:n{-var-value~HOME}
419       \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
420       \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_s
421     }{
422       \stex_kpsewhich:n{-var-value~HOME}
423     }
424     \ior_open:NnT \l_tmpa_ior{\l_stex_kpsewhich_return_str / .stex / mathhub.path}{
425       \begingroup\escapechar=-1\catcode`\\=12
426       \ior_str_get:NN \l_tmpa_ior \l_tmpa_str
```

```
427        \sys_if_platform_windows:T{
428          \exp_args:NNx\str_replace_all:Nnn\l_tmpa_str{\c_backslash_str}/
429        }
430        \str_gset_eq:NN \c_stex_mathhub_str\l_tmpa_str
431        \endgroup
432        \ior_close:N \l_tmpa_ior
433      }
434    }
435    \str_if_empty:NTF\c_stex_mathhub_str{
436      \msg_warning:nn{stex}{warning/nomathhub}
437    }{
438      \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
439      \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
440    }
441  }{
442    \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
443    \stex_path_if_absolute:NF \c_stex_mathhub_seq {
444      \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
445        \c_stex_pwd_str/\mathhub
446      }
447    }
448    \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
449    \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
450  }
```

(*End definition for* \mathhub, \c_stex_mathhub_seq, *and* \c_stex_mathhub_str. *These variables are documented on page* *49*.)

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

```
451  \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
452    \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
453      \str_set:Nx \l_tmpa_str { #1 }
454      \prop_new:c { c_stex_mathhub_#1_manifest_prop }
455      \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
456      \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
457      \__stex_mathhub_find_manifest:N \l_tmpa_seq
458      \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
459        \msg_error:nnxx{stex}{error/norepository}{#1}{
460          \stex_path_to_string:N \c_stex_mathhub_str
461        }
462      } {
463        \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
464      }
465    }
466  }
```

(*End definition for* __stex_mathhub_do_manifest:n.)

\l__stex_mathhub_manifest_file_seq

```
467  \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`:

```
468 \cs_new_protected:Nn \__stex_mathhub_find_manifest:N {
469   \seq_set_eq:NN\l_tmpa_seq #1
470   \bool_set_true:N\l_tmpa_bool
471   \bool_while_do:Nn \l_tmpa_bool {
472     \seq_if_empty:NTF \l_tmpa_seq {
473       \bool_set_false:N\l_tmpa_bool
474     }{
475       \file_if_exist:nTF{
476         \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
477       }{
478         \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
479         \bool_set_false:N\l_tmpa_bool
480       }{
481         \file_if_exist:nTF{
482           \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
483         }{
484           \seq_put_right:Nn\l_tmpa_seq{META-INF}
485           \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
486           \bool_set_false:N\l_tmpa_bool
487         }{
488           \file_if_exist:nTF{
489             \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
490           }{
491             \seq_put_right:Nn\l_tmpa_seq{meta-inf}
492             \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
493             \bool_set_false:N\l_tmpa_bool
494           }{
495             \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
496           }
497         }
498       }
499     }
500   }
501   \seq_set_eq:NN\l__stex_mathhub_manifest_file_seq\l_tmpa_seq
502 }
```

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

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

```
503 \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:

```
504 \cs_new_protected:Nn \__stex_mathhub_parse_manifest:n {
505   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
506   \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
507   \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
508     \str_set:Nn \l_tmpa_str {##1}
509     \exp_args:NNoo \seq_set_split:Nnn
510         \l_tmpb_seq \c_colon_str \l_tmpa_str
511     \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
```

```
512        \exp_args:NNe \str_set:Nn \l_tmpb_tl {
513          \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
514        }
515        \exp_args:No \str_case:nnTF \l_tmpa_tl {
516          {id} {
517            \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
518              { id } \l_tmpb_tl
519          }
520          {narration-base} {
521            \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
522              { narr } \l_tmpb_tl
523          }
524          {url-base} {
525            \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
526              { docurl } \l_tmpb_tl
527          }
528          {source-base} {
529            \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
530              { ns } \l_tmpb_tl
531          }
532          {ns} {
533            \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
534              { ns } \l_tmpb_tl
535          }
536          {dependencies} {
537            \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
538              { deps } \l_tmpb_tl
539          }
540        }{}{}
541      }{}
542    }
543    \ior_close:N \c__stex_mathhub_manifest_ior
544    \stex_persist:x {
545      \prop_set_from_keyval:cn{ c_stex_mathhub_#1_manifest_prop }{
546        \exp_after:wN \prop_to_keyval:N \csname c_stex_mathhub_#1_manifest_prop\endcsname
547      }
548    }
549 }
```

(*End definition for* __stex_mathhub_parse_manifest:n.)

\stex_set_current_repository:n

```
550 \cs_new_protected:Nn \stex_set_current_repository:n {
551   \stex_require_repository:n { #1 }
552   \prop_set_eq:Nc \l_stex_current_repository_prop {
553     c_stex_mathhub_#1_manifest_prop
554   }
555 }
```

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

\stex_require_repository:n

```
556 \cs_new_protected:Nn \stex_require_repository:n {
557   \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
558     \stex_debug:nn{mathhub}{Opening~archive:~#1}
```

```
559        \__stex_mathhub_do_manifest:n { #1 }
560      }
561 }
```

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

Current MathHub repository

```
562 %\prop_new:N \l_stex_current_repository_prop
563 \bool_if:NF \c_stex_persist_mode_bool {
564    \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
565    \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
566      \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
567    } {
568      \__stex_mathhub_parse_manifest:n { main }
569      \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
570        \l_tmpa_str
571      \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str _manifest_prop }
572        \c_stex_mathhub_main_manifest_prop
573      \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
574      \stex_debug:nn{mathhub}{Current~repository:~
575        \prop_item:Nn \l_stex_current_repository_prop {id}
576      }
577    }
578 }
```

(End definition for `\l_stex_current_repository_prop`*. This variable is documented on page 49.)*

`\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.

```
579 \cs_new_protected:Nn \stex_in_repository:nn {
580    \str_set:Nx \l_tmpa_str { #1 }
581    \cs_set:Npn \l_tmpa_cs ##1 { #2 }
582    \str_if_empty:NTF \l_tmpa_str {
583      \prop_if_exist:NTF \l_stex_current_repository_prop {
584        \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
585        \exp_args:Ne \l_tmpa_cs{
586          \prop_item:Nn \l_stex_current_repository_prop { id }
587        }
588      }{
589        \l_tmpa_cs{}
590      }
591    }{
592      \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}
593      \stex_require_repository:n \l_tmpa_str
594      \str_set:Nx \l_tmpa_str { #1 }
595      \exp_args:Nne \use:nn {
596        \stex_set_current_repository:n \l_tmpa_str
597        \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
598      }{
599        \stex_debug:nn{mathhub}{switching~back~to:~
600          \prop_if_exist:NTF \l_stex_current_repository_prop {
601            \prop_item:Nn \l_stex_current_repository_prop { id }:~
602            \meaning\l_stex_current_repository_prop
603          }{
```

```
604            no~repository
605          }
606        }
607        \prop_if_exist:NTF \l_stex_current_repository_prop {
608          \stex_set_current_repository:n {
609            \prop_item:Nn \l_stex_current_repository_prop { id }
610          }
611        }{
612          \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
613        }
614      }
615    }
616  }
```

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

25.5   Using Content in Archives

```
617  \def \mhpath #1 #2 {
618    \exp_args:Ne \tl_if_empty:nTF{#1}{
619      \c_stex_mathhub_str /
620        \prop_item:Nn \l_stex_current_repository_prop { id }
621        / source / #2
622    }{
623      \c_stex_mathhub_str / #1 / source / #2
624    }
625  }
```

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

```
626  \newif \ifinputref \inputreffalse
627
628  \cs_new_protected:Nn \__stex_mathhub_mhinput:nn {
629    \stex_in_repository:nn {#1} {
630      \ifinputref
631        \input{ \c_stex_mathhub_str / ##1 / source / #2 }
632      \else
633        \inputreftrue
634        \input{ \c_stex_mathhub_str / ##1 / source / #2 }
635        \inputreffalse
636      \fi
637    }
638  }
639  \NewDocumentCommand \mhinput { O{} m}{
640    \__stex_mathhub_mhinput:nn{ #1 }{ #2 }
641  }
642
643  \cs_new_protected:Nn \__stex_mathhub_inputref:nn {
644    \stex_in_repository:nn {#1} {
645      \stex_html_backend:TF {
646        \str_clear:N \l_tmpa_str
```

```
647        \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
648          \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
649        }
650        \stex_annotate_invisible:nnn{inputref}{
651          \l_tmpa_str / #2
652        }{}
653      }{
654        \begingroup
655          \inputreftrue
656          \tl_if_empty:nTF{ ##1 }{
657            \input{#2}
658          }{
659            \input{ \c_stex_mathhub_str / ##1 / source / #2 }
660          }
661        \endgroup
662      }
663    }
664  }
665  \NewDocumentCommand \inputref { O{} m }{
666    \__stex_mathhub_inputref:nn{ #1 }{ #2 }
667  }
```

(*End definition for* \inputref *and* \mhinput. *These functions are documented on page 50.*)

```
668  \cs_new_protected:Nn \__stex_mathhub_mhbibresource:nn {
669    \stex_in_repository:nn {#1} {
670      \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
671    }
672  }
673  \newcommand\addmhbibresource[2][]{
674    \__stex_mathhub_mhbibresource:nn{ #1 }{ #2 }
675  }
```

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

```
676  \cs_new_protected:Npn \libinput #1 {
677    \prop_if_exist:NF \l_stex_current_repository_prop {
678      \msg_error:nnn{stex}{error/notinarchive}\libinput
679    }
680    \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
681      \msg_error:nnn{stex}{error/notinarchive}\libinput
682    }
683    \seq_clear:N \l__stex_mathhub_libinput_files_seq
684    \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
685    \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
686
687    \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
688      \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #1.tex}
689      \IfFileExists{ \l_tmpa_str }{
690        \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
691      }{}
692      \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
693      \seq_put_right:No \l_tmpa_seq \l_tmpa_str
```

```
694    }
695
696    \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}
697    \IfFileExists{ \l_tmpa_str }{
698      \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
699    }{}
700
701    \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
702      \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
703    }{
704      \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
705        \input{ ##1 }
706      }
707    }
708 }
```

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

```
709 \NewDocumentCommand \libusepackage {O{} m} {
710    \prop_if_exist:NF \l_stex_current_repository_prop {
711      \msg_error:nnn{stex}{error/notinarchive}\libusepackage
712    }
713    \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
714      \msg_error:nnn{stex}{error/notinarchive}\libusepackage
715    }
716    \seq_clear:N \l__stex_mathhub_libinput_files_seq
717    \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
718    \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
719
720    \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
721      \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #2}
722      \IfFileExists{ \l_tmpa_str.sty }{
723        \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
724      }{}
725      \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
726      \seq_put_right:No \l_tmpa_seq \l_tmpa_str
727    }
728
729    \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2}
730    \IfFileExists{ \l_tmpa_str.sty }{
731      \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
732    }{}
733
734    \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
735      \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
736    }{
737      \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
738        \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
739          \usepackage[#1]{ ##1 }
740        }
741      }{
742        \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
743      }
```

113

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

`\mhgraphics`
`\cmhgraphics`

746
747 `\AddToHook{begindocument}{`
748 `\ltx@ifpackageloaded{graphicx}{`
749 `    \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}`
750 `    \newcommand\mhgraphics[2][]{%`
751 `      \def\Gin@mhrepos{}\setkeys{Gin}{#1}%`
752 `      \includegraphics[#1]{\mhpath\Gin@mhrepos{#2}}}`
753 `    \newcommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}`
754 `  }{}`

(*End definition for* `\mhgraphics` *and* `\cmhgraphics`. *These functions are documented on page 50.*)

`\lstinputmhlisting`
`\clstinputmhlisting`

755 `\ltx@ifpackageloaded{listings}{`
756 `    \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}`
757 `    \newcommand\lstinputmhlisting[2][]{%`
758 `      \def\lst@mhrepos{}\setkeys{lst}{#1}%`
759 `      \lstinputlisting[#1]{\mhpath\lst@mhrepos{#2}}}`
760 `    \newcommand\clstinputmhlisting[2][]{\begin{center}\lstinputmhlisting[#1]{#2}\end{center}`
761 `  }{}`
762 `}`
763
764 `⟨/package⟩`

(*End definition for* `\lstinputmhlisting` *and* `\clstinputmhlisting`. *These functions are documented on page 50.*)

Chapter 26

sTEX
-References Implementation

```
765 ⟨*package⟩
766
767 %%%%%%%%%%%%    references.dtx    %%%%%%%%%%%%
768
769 ⟨@@=stex_refs⟩
```

Warnings and error messages

```
770
```

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

```
771 %\iow_new:N \c__stex_refs_refs_iow
772 \AtBeginDocument{
773 %  \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
774 }
775 \AtEndDocument{
776 %  \iow_close:N \c__stex_refs_refs_iow
777 }
```

\STEXreftitle

```
778 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
779
780 \NewDocumentCommand \STEXreftitle { m } {
781   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
782 }
```

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

26.1 Document URIs and URLs

\l_stex_current_docns_str

```
783 \str_new:N \l_stex_current_docns_str
```

(*End definition for* \l_stex_current_docns_str*. This variable is documented on page 51.*)

```
784 \cs_new_protected:Nn \stex_get_document_uri: {
785   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
786   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
787   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
788   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
789   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
790
791   \str_clear:N \l_tmpa_str
792   \prop_if_exist:NT \l_stex_current_repository_prop {
793     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
794       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
795     }
796   }
797
798   \str_if_empty:NTF \l_tmpa_str {
799     \str_set:Nx \l_stex_current_docns_str {
800       file:/\stex_path_to_string:N \l_tmpa_seq
801     }
802   }{
803     \bool_set_true:N \l_tmpa_bool
804     \bool_while_do:Nn \l_tmpa_bool {
805       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
806       \exp_args:No \str_case:nnTF { \l_tmpb_str } {
807         {source} { \bool_set_false:N \l_tmpa_bool }
808       }{}{
809         \seq_if_empty:NT \l_tmpa_seq {
810           \bool_set_false:N \l_tmpa_bool
811         }
812       }
813     }
814
815     \seq_if_empty:NTF \l_tmpa_seq {
816       \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
817     }{
818       \str_set:Nx \l_stex_current_docns_str {
819         \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
820       }
821     }
822   }
823 }
```

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

```
824 \str_new:N \l_stex_current_docurl_str
```

(*End definition for* \l_stex_current_docurl_str. *This variable is documented on page* *51.*)

```
825 \cs_new_protected:Nn \stex_get_document_url: {
826   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
827   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
828   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
```

```
829    \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
830    \seq_put_right:No \l_tmpa_seq \l_tmpb_str
831
832    \str_clear:N \l_tmpa_str
833    \prop_if_exist:NT \l_stex_current_repository_prop {
834      \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
835        \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
836          \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
837        }
838      }
839    }
840
841    \str_if_empty:NTF \l_tmpa_str {
842      \str_set:Nx \l_stex_current_docurl_str {
843        file:/\stex_path_to_string:N \l_tmpa_seq
844      }
845    }{
846      \bool_set_true:N \l_tmpa_bool
847      \bool_while_do:Nn \l_tmpa_bool {
848        \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
849        \exp_args:No \str_case:nnTF { \l_tmpb_str } {
850          {source} { \bool_set_false:N \l_tmpa_bool }
851        }{}{
852          \seq_if_empty:NT \l_tmpa_seq {
853            \bool_set_false:N \l_tmpa_bool
854          }
855        }
856      }
857
858      \seq_if_empty:NTF \l_tmpa_seq {
859        \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
860      }{
861        \str_set:Nx \l_stex_current_docurl_str {
862          \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
863        }
864      }
865    }
866  }
```

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

26.2   Setting Reference Targets

```
867  \str_const:Nn \c__stex_refs_url_str{URL}
868  \str_const:Nn \c__stex_refs_ref_str{REF}
869  \str_new:N \l__stex_refs_curr_label_str
870  % @currentlabel -> number
871  % @currentlabelname -> title
872  % @currentHref -> name.number <- id of some kind
873  % \theH# -> \arabic{section}
874  % \the#  -> number
875  % \hyper@makecurrent{#}
876  \int_new:N \l__stex_refs_unnamed_counter_int
```

```
877 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
878   \stex_get_document_uri:
879   \str_clear:N \l__stex_refs_curr_label_str
880   \str_set:Nx \l_tmpa_str { #1 }
881   \str_if_empty:NT \l_tmpa_str {
882     \int_incr:N \l__stex_refs_unnamed_counter_int
883     \str_set:Nx \l_tmpa_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
884   }
885   \str_set:Nx \l__stex_refs_curr_label_str {
886     \l_stex_current_docns_str?\l_tmpa_str
887   }
888   \seq_if_exist:cF{g__stex_refs_labels_\l_tmpa_str _seq}{
889     \seq_new:c {g__stex_refs_labels_\l_tmpa_str _seq}
890   }
891   \seq_if_in:coF{g__stex_refs_labels_\l_tmpa_str _seq}\l__stex_refs_curr_label_str {
892     \seq_gput_right:co{g__stex_refs_labels_\l_tmpa_str _seq}\l__stex_refs_curr_label_str
893   }
894   \stex_if_smsmode:TF {
895     \stex_get_document_url:
896     \str_gset_eq:cN {sref_url_\l__stex_refs_curr_label_str _str}\l_stex_current_docurl_str
897     \str_gset_eq:cN {sref_\l__stex_refs_curr_label_str _type}\c__stex_refs_url_str
898   }{
899     %\iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=~\expandafter\unexpanded\expandafter{
900     \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
901     \immediate\write\@auxout{\stexauxadddocref{\l_stex_current_docns_str}{\l_tmpa_str}}
902     \str_gset:cx {sref_\l__stex_refs_curr_label_str _type}\c__stex_refs_ref_str
903   }
904 }
```

(*End definition for* \stex_ref_new_doc_target:n. *This function is documented on page 51.*)

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

```
905 \cs_new_protected:Npn \stexauxadddocref #1 #2 {
906   \str_set:Nn \l_tmpa_str {#1?#2}
907   \str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
908   \seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
909     \seq_new:c {g__stex_refs_labels_#2_seq}
910   }
911   \seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
912     \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
913   }
914 }
```

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

```
915 \AtEndDocument{
916   \def\stexauxadddocref#1 #2 {}{}
917 }
```

```
918 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
919   \stex_if_smsmode:TF {
920     \str_if_exist:cF{sref_sym_#1_type}{
921       \stex_get_document_url:
922       \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str
```

```
923        \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
924      }
925    }{
926      \str_if_empty:NF \l__stex_refs_curr_label_str {
927        \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
928        \immediate\write\@auxout{
929          \exp_not:N\expandafter\def\exp_not:N\csname \exp_not:N\detokenize{sref_sym_#1_label_
930            \l__stex_refs_curr_label_str
931          }
932        }
933      }
934    }
935 }
```

(End definition for \stex_ref_new_sym_target:n. This function is documented on page 51.)

26.3   Using References

```
936 \str_new:N \l__stex_refs_indocument_str
```

\sref   Optional arguments:

```
937
938 \keys_define:nn { stex / sref } {
939    linktext      .tl_set:N  = \l__stex_refs_linktext_tl ,
940    fallback      .tl_set:N  = \l__stex_refs_fallback_tl ,
941    pre           .tl_set:N  = \l__stex_refs_pre_tl ,
942    post          .tl_set:N  = \l__stex_refs_post_tl ,
943 }
944 \cs_new_protected:Nn \__stex_refs_args:n {
945    \tl_clear:N \l__stex_refs_linktext_tl
946    \tl_clear:N \l__stex_refs_fallback_tl
947    \tl_clear:N \l__stex_refs_pre_tl
948    \tl_clear:N \l__stex_refs_post_tl
949    \str_clear:N \l__stex_refs_repo_str
950    \keys_set:nn { stex / sref } { #1 }
951 }
```

The actual macro:

```
952 \NewDocumentCommand \sref { O{} m}{
953    \__stex_refs_args:n { #1 }
954    \str_if_empty:NTF \l__stex_refs_indocument_str {
955      \str_set:Nx \l_tmpa_str { #2 }
956      \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
957      \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
958        \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
959          \seq_get_left:cNF {g__stex_refs_labels_\l_tmpa_str _seq} \l_tmpa_str {
960            \str_clear:N \l_tmpa_str
961          }
962        }{
963          \str_clear:N \l_tmpa_str
964        }
965      }{
966        \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
967        \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
```

```
968          \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
969          \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
970            \str_set_eq:NN \l_tmpc_str \l_tmpa_str
971            \str_clear:N \l_tmpa_str
972            \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str _seq} {
973              \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
974                \str_range:nnn { ##1 }{ -\l_tmpa_int}{ -1 }
975              }{
976                \seq_map_break:n {
977                  \str_set:Nn \l_tmpa_str { ##1 }
978                }
979              }
980            }
981          }{
982            \str_clear:N \l_tmpa_str
983          }
984        }
985        \str_if_empty:NTF \l_tmpa_str {
986          \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_lin
987        }{
988          \str_if_eq:cNTF {sref_\l_tmpa_str _type} \c__stex_refs_ref_str {
989            \tl_if_empty:NTF \l__stex_refs_linktext_tl {
990              \cs_if_exist:cTF{autoref}{
991                \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
992              }{
993                \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
994              }
995            }{
996              \ltx@ifpackageloaded{hyperref}{
997                \hyperref[sref_\l_tmpa_str]\l__stex_refs_linktext_tl
998              }{
999                \l__stex_refs_linktext_tl
1000             }
1001           }
1002         }{
1003           \ltx@ifpackageloaded{hyperref}{
1004             \href{\use:c{sref_url_\l_tmpa_str _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_t
1005           }{
1006             \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs
1007           }
1008         }
1009       }
1010     }{
1011       % TODO
1012     }
1013 }
```

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

```
1014 \NewDocumentCommand \srefsym { O{} m}{
1015   \stex_get_symbol:n { #2 }
1016   \__stex_refs_sym_aux:nn{#1}{\l_stex_get_symbol_uri_str}
1017 }
```

120

```
1018
1019 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1020   \str_if_exist:cTF {sref_sym_#2 _label_str }{
1021     \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1022   }{
1023     \__stex_refs_args:n { #1 }
1024     \str_if_empty:NTF \l__stex_refs_indocument_str {
1025       \tl_if_exist:cTF{sref_sym_#2 _type}{
1026         % doc uri in \l_tmpb_str
1027         \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1028         \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1029           % reference
1030           \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1031             \cs_if_exist:cTF{autoref}{
1032               \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1033             }{
1034               \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1035             }
1036           }{
1037             \ltx@ifpackageloaded{hyperref}{
1038               \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1039             }{
1040               \l__stex_refs_linktext_tl
1041             }
1042           }
1043         }{
1044           % URL
1045           \ltx@ifpackageloaded{hyperref}{
1046             \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl \
1047           }{
1048             \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_re
1049           }
1050         }
1051       }{
1052         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_l
1053       }
1054     }{
1055       % TODO
1056     }
1057   }
1058 }
```

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

\srefsymuri

```
1059 \cs_new_protected:Npn \srefsymuri #1 #2 {
1060   \__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1061 }
```

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

```
1062 ⟨/package⟩
```

Chapter 27

sTEX
-Modules Implementation

```
1063 ⟨∗package⟩
1064
1065 %%%%%%%%%%%%    modules.dtx    %%%%%%%%%%%%%
1066
1067 ⟨@@=stex_modules⟩
```

Warnings and error messages

```
1068 \msg_new:nnn{stex}{error/unknownmodule}{
1069    No~module~#1~found
1070 }
1071 \msg_new:nnn{stex}{error/syntax}{
1072    Syntax~error:~#1
1073 }
1074 \msg_new:nnn{stex}{error/siglanguage}{
1075    Module~#1~declares~signature~#2,~but~does~not~
1076    declare~its~language
1077 }
1078 \msg_new:nnn{stex}{warning/deprecated}{
1079    #1~is~deprecated;~please~use~#2~instead!
1080 }
1081
1082 \msg_new:nnn{stex}{error/conflictingmodules}{
1083    Conflicting~imports~for~module~#1
1084 }
```

`\l_stex_current_module_str`   The current module:

```
1085 \str_new:N \l_stex_current_module_str
```

(*End definition for* `\l_stex_current_module_str`. *This variable is documented on page 54.*)

`\l_stex_all_modules_seq`   Stores all available modules

```
1086 \seq_new:N \l_stex_all_modules_seq
```

(*End definition for* `\l_stex_all_modules_seq`. *This variable is documented on page 54.*)

```
1087 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1088     \str_if_empty:NTF \l_stex_current_module_str
1089         \prg_return_false: \prg_return_true:
1090 }
```

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

```
1091 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1092     \prop_if_exist:cTF { c_stex_module_#1_prop }
1093         \prg_return_true: \prg_return_false:
1094 }
```

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

Only allowed within modules:

```
1095 \cs_new_protected:Nn \stex_execute_in_module:n { \stex_if_in_module:T {
1096     \stex_add_to_current_module:n { #1 }
1097     \stex_do_up_to_module:n { #1 }
1098 }}
1099 \cs_generate_variant:Nn \stex_execute_in_module:n {x}
1100
1101 \cs_new_protected:Nn \stex_add_to_current_module:n {
1102     \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1103 }
1104 \cs_generate_variant:Nn \stex_add_to_current_module:n {x}
1105 \cs_new_protected:Npn \STEXexport {
1106     \begingroup
1107     \newlinechar=-1\relax
1108     \endlinechar=-1\relax
1109     %\catcode'\ = 9\relax
1110     \expandafter\endgroup\__stex_modules_export:n
1111 }
1112 \cs_new_protected:Nn \__stex_modules_export:n {
1113     \ignorespaces #1
1114     \stex_add_to_current_module:n { \ignorespaces #1 }
1115     \stex_smsmode_do:
1116 }
1117 \stex_deactivate_macro:Nn \STEXexport {module~environments}
```

(*End definition for* `\stex_add_to_current_module:n` *and* `\STEXexport`*. These functions are documented on page* *54.*)

```
1118 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1119     \str_set:Nx \l_tmpa_str { #1 }
1120     \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
1121 }
```

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

```
1122 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1123   \str_set:Nx \l_tmpa_str { #1 }
1124   \exp_args:Nno
1125   \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1126     \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1127   }
1128 }
```

(*End definition for* \stex_add_import_to_current_module:n. *This function is documented on page* *54.*)

```
1129 \cs_new_protected:Nn \stex_collect_imports:n {
1130   \seq_clear:N \l_stex_collect_imports_seq
1131   \__stex_modules_collect_imports:n {#1}
1132 }
1133 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1134   \seq_map_inline:cn {c_stex_module_#1_imports} {
1135     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1136       \__stex_modules_collect_imports:n { ##1 }
1137     }
1138   }
1139   \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1140     \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1141   }
1142 }
```

(*End definition for* \stex_collect_imports:n. *This function is documented on page* *54.*)

```
1143 \int_new:N \l__stex_modules_group_depth_int
1144 \cs_new_protected:Nn \stex_do_up_to_module:n {
1145   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1146     #1
1147   }{
1148     #1
1149     \expandafter \tl_gset:Nn
1150     \csname l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1151     \expandafter\expandafter\expandafter\endcsname
1152     \expandafter\expandafter\expandafter { \csname
1153       l__stex_modules_aftergroup_\l_stex_current_module_str _tl\endcsname #1 }
1154     \aftergroup\__stex_modules_aftergroup_do:
1155   }
1156 }
1157 \cs_generate_variant:Nn \stex_do_up_to_module:n {x}
1158 \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1159   \stex_debug:nn{aftergroup}{\cs_meaning:c{
1160     l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1161   }}
1162   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1163     \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1164     \tl_gclear:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1165   }{
1166     \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
```

124

```
1167        \aftergroup\__stex_modules_aftergroup_do:
1168      }
1169 }
1170 \cs_new_protected:Nn \_stex_reset_up_to_module:n {
1171      \expandafter\let\csname l__stex_modules_aftergroup_#1_tl\endcsname\undefined
1172 }
```

(*End definition for* \stex_do_up_to_module:n. *This function is documented on page* *54*.)

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

```
1173
```

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

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

```
1174 \str_new:N \l_stex_module_ns_str
1175 \str_new:N \l_stex_module_subpath_str
1176 \cs_new_protected:Nn \__stex_modules_compute_namespace:nN {
1177      \seq_set_eq:NN \l_tmpa_seq #2
1178      % split off file extension
1179      \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str % <- filename
1180      \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1181      \seq_get_left:NN \l_tmpb_seq \l_tmpb_str % <- filename without suffixes
1182      \seq_put_right:No \l_tmpa_seq \l_tmpb_str % <- file path including name without suffixes
1183
1184      \bool_set_true:N \l_tmpa_bool
1185      \bool_while_do:Nn \l_tmpa_bool {
1186        \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1187        \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1188          {source} { \bool_set_false:N \l_tmpa_bool }
1189        }{}{
1190          \seq_if_empty:NT \l_tmpa_seq {
1191            \bool_set_false:N \l_tmpa_bool
1192          }
1193        }
1194      }
1195
1196      \stex_path_to_string:NN \l_tmpa_seq \l_stex_module_subpath_str
1197      % \l_tmpa_seq <- sub-path relative to archive
1198      \str_if_empty:NTF \l_stex_module_subpath_str {
1199        \str_set:Nx \l_stex_module_ns_str {#1}
1200      }{
1201        \str_set:Nx \l_stex_module_ns_str {
1202          #1/\l_stex_module_subpath_str
1203        }
1204      }
1205 }
1206
1207 \cs_new_protected:Nn \stex_modules_current_namespace: {
1208      \str_clear:N \l_stex_module_subpath_str
1209      \prop_if_exist:NTF \l_stex_current_repository_prop {
1210        \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
```

```
1211        \__stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1212   }{
1213     % split off file extension
1214     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1215     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1216     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1217     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1218     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1219     \str_set:Nx \l_stex_module_ns_str {
1220       file:/\stex_path_to_string:N \l_tmpa_seq
1221     }
1222   }
1223 }
```

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

27.1   The `smodule` environment

`smodule` arguments:

```
1224 \keys_define:nn { stex / module } {
1225   title         .tl_set:N    = \smoduletitle ,
1226   type          .str_set_x:N = \smoduletype ,
1227   id            .str_set_x:N = \smoduleid ,
1228   deprecate     .str_set_x:N = \l_stex_module_deprecate_str ,
1229   ns            .str_set_x:N = \l_stex_module_ns_str ,
1230   lang          .str_set_x:N = \l_stex_module_lang_str ,
1231   sig           .str_set_x:N = \l_stex_module_sig_str ,
1232   creators      .str_set_x:N = \l_stex_module_creators_str ,
1233   contributors  .str_set_x:N = \l_stex_module_contributors_str ,
1234   meta          .str_set_x:N = \l_stex_module_meta_str ,
1235   srccite       .str_set_x:N = \l_stex_module_srccite_str
1236 }
1237
1238 \cs_new_protected:Nn \__stex_modules_args:n {
1239   \str_clear:N \smoduletitle
1240   \str_clear:N \smoduletype
1241   \str_clear:N \smoduleid
1242   \str_clear:N \l_stex_module_ns_str
1243   \str_clear:N \l_stex_module_deprecate_str
1244   \str_clear:N \l_stex_module_lang_str
1245   \str_clear:N \l_stex_module_sig_str
1246   \str_clear:N \l_stex_module_creators_str
1247   \str_clear:N \l_stex_module_contributors_str
1248   \str_clear:N \l_stex_module_meta_str
1249   \str_clear:N \l_stex_module_srccite_str
1250   \keys_set:nn { stex / module } { #1 }
1251 }
1252
1253 % module parameters here? In the body?
1254
```

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

```
1255 \cs_new_protected:Nn \stex_module_setup:nn {
```

```
1256    \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1257    \str_set:Nx \l_stex_module_name_str { #2 }
1258    \__stex_modules_args:n { #1 }
```

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

```
1259    \stex_if_in_module:TF {
1260      % Nested module
1261      \prop_get:cnN {c_stex_module_\l_stex_current_module_str _prop}
1262        { ns } \l_stex_module_ns_str
1263      \str_set:Nx \l_stex_module_name_str {
1264        \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
1265          { name } / \l_stex_module_name_str
1266      }
1267    }{
1268      % not nested:
1269      \str_if_empty:NT \l_stex_module_ns_str {
1270        \stex_modules_current_namespace:
1271        \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1272          / {\l_stex_module_ns_str}
1273        \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1274        \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1275          \str_set:Nx \l_stex_module_ns_str {
1276            \stex_path_to_string:N \l_tmpa_seq
1277          }
1278        }
1279      }
1280    }
```

Next, we determine the language of the module:

```
1281    \str_if_empty:NT \l_stex_module_lang_str {
1282      \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1283      \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1284      \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1285      \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
1286        \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
1287          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
1288        }
1289      }
1290      \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1291      \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
1292        \seq_pop_right:NN \l_tmpa_seq \l_stex_module_lang_str
1293        \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1294          inferred~from~file~name}
1295      }
1296    }
1297
1298    \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
1299      \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1300        \l_tmpa_str {
1301        \ltx@ifpackageloaded{babel}{
1302          \exp_args:Nx \selectlanguage { \l_tmpa_str }
1303        }{}
1304        } {
```

```
1305        \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
1306      }
1307  }}
```

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

```
1308    \str_if_empty:NTF \l_stex_module_sig_str {
1309      \exp_args:Nnx \prop_gset_from_keyval:cn {
1310        c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1311      } {
1312        name      = \l_stex_module_name_str ,
1313        ns        = \l_stex_module_ns_str ,
1314        file      = \exp_not:o { \g_stex_currentfile_seq } ,
1315        lang      = \l_stex_module_lang_str ,
1316        sig       = \l_stex_module_sig_str ,
1317        deprecate = \l_stex_module_deprecate_str ,
1318        meta      = \l_stex_module_meta_str
1319      }
1320      \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1321      \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1322      \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _copymodules}
1323      \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1324      \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
```

We load the metatheory:

```
1325      \str_if_empty:NT \l_stex_module_meta_str {
1326        \str_set:Nx \l_stex_module_meta_str {
1327          \c_stex_metatheory_ns_str ? Metatheory
1328        }
1329      }
1330      \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1331        \bool_set_true:N \l_stex_in_meta_bool
1332        \exp_args:Nx \stex_add_to_current_module:n {
1333          \bool_set_true:N \l_stex_in_meta_bool
1334          \stex_activate_module:n {\l_stex_module_meta_str}
1335          \bool_set_false:N \l_stex_in_meta_bool
1336        }
1337        \stex_activate_module:n {\l_stex_module_meta_str}
1338        \bool_set_false:N \l_stex_in_meta_bool
1339      }
1340    }{
1341      \str_if_empty:NT \l_stex_module_lang_str {
1342        \msg_error:nnxx{stex}{error/siglanguage}{
1343          \l_stex_module_ns_str?\l_stex_module_name_str
1344        }{\l_stex_module_sig_str}
1345      }
1346      \stex_debug:nn{modules}{Signature~\l_stex_module_sig_str~for~\l_stex_module_ns_str?\l_st
1347      \stex_if_module_exists:nTF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1348        \stex_debug:nn{modules}{(already exists)}
1349      }{
1350        \stex_debug:nn{modules}{(needs loading)}
1351        \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1352        \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1353        \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
```

```
1354          \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1355          \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1356          \str_set:Nx \l_tmpa_str {
1357            \stex_path_to_string:N \l_tmpa_seq /
1358            \l_tmpa_str . \l_stex_module_sig_str .tex
1359          }
1360          \IfFileExists \l_tmpa_str {
1361            \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1362              \str_clear:N \l_stex_current_module_str
1363              \seq_clear:N \l_stex_all_modules_seq
1364              \stex_debug:nn{modules}{Loading~signature}
1365            }
1366          }{
1367            \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1368          }
1369        }
1370        \stex_if_smsmode:F {
1371          \stex_activate_module:n {
1372            \l_stex_module_ns_str ? \l_stex_module_name_str
1373          }
1374        }
1375        \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1376      }
1377      \str_if_empty:NF \l_stex_module_deprecate_str {
1378        \msg_warning:nnxx{stex}{warning/deprecated}{
1379          Module~\l_stex_current_module_str
1380        }{
1381          \l_stex_module_deprecate_str
1382        }
1383      }
1384      \seq_put_right:Nx \l_stex_all_modules_seq {
1385        \l_stex_module_ns_str ? \l_stex_module_name_str
1386      }
1387      \tl_clear:c{l__stex_modules_aftergroup_\l_stex_module_ns_str ? \l_stex_module_name_str _tl
1388    }
```

(*End definition for* \stex_module_setup:nn*. This function is documented on page* *55*.)

smodule   The module environment.

__stex_modules_begin_module: implements \begin{smodule}

```
1389  \cs_new_protected:Nn \__stex_modules_begin_module: {
1390    \stex_reactivate_macro:N \STEXexport
1391    \stex_reactivate_macro:N \importmodule
1392    \stex_reactivate_macro:N \symdecl
1393    \stex_reactivate_macro:N \notation
1394    \stex_reactivate_macro:N \symdef
1395
1396    \stex_debug:nn{modules}{
1397      New~module:\\
1398      Namespace:~\l_stex_module_ns_str\\
1399      Name:~\l_stex_module_name_str\\
1400      Language:~\l_stex_module_lang_str\\
1401      Signature:~\l_stex_module_sig_str\\
```

129

```
1402       Metatheory:~\l_stex_module_meta_str\\
1403       File:~\stex_path_to_string:N \g_stex_currentfile_seq
1404    }
1405
1406    \stex_if_do_html:T{
1407      \begin{stex_annotate_env} {theory} {
1408        \l_stex_module_ns_str ? \l_stex_module_name_str
1409      }
1410
1411      \stex_annotate_invisible:nnn{header}{} {
1412        \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1413        \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1414        \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1415          \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1416        }
1417        \str_if_empty:NF \smoduletype {
1418          \stex_annotate:nnn{type}{\smoduletype}{}
1419        }
1420      }
1421    }
1422    % TODO: Inherit metatheory for nested modules?
1423 }
1424 \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}

```
1425 \cs_new_protected:Nn \__stex_modules_end_module: {
1426    \stex_debug:nn{modules}{Closing~module~\prop_item:cn {c_stex_module_\l_stex_current_module
1427    \_stex_reset_up_to_module:n \l_stex_current_module_str
1428    \stex_if_smsmode:T {
1429      \stex_persist:x {
1430        \prop_set_from_keyval:cn{c_stex_module_\l_stex_current_module_str _prop}{
1431          \exp_after:wN \prop_to_keyval:N \csname c_stex_module_\l_stex_current_module_str _pr
1432        }
1433        \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _constants}{
1434          \seq_use:cn{c_stex_module_\l_stex_current_module_str _constants},
1435        }
1436        \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _imports}{
1437          \seq_use:cn{c_stex_module_\l_stex_current_module_str _imports},
1438        }
1439        \tl_set:cn {c_stex_module_\l_stex_current_module_str _code}
1440      }
1441      \exp_after:wN \let \exp_after:wN \l_tmpa_tl \csname c_stex_module_\l_stex_current_module
1442      \exp_after:wN \stex_persist:n \exp_after:wN { \exp_after:wN { \l_tmpa_tl } }
1443    }
1444 }
```

(*End definition for* __stex_modules_end_module:.)

The core environment

```
1445 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1446 \NewDocumentEnvironment { smodule } { O{} m } {
1447    \stex_module_setup:nn{#1}{#2}
1448    \par
```

130

```
1449    \stex_if_smsmode:F{
1450      \tl_clear:N \l_tmpa_tl
1451      \clist_map_inline:Nn \smoduletype {
1452        \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1453          \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1454        }
1455      }
1456      \tl_if_empty:NTF \l_tmpa_tl {
1457        \__stex_modules_smodule_start:
1458      }{
1459        \l_tmpa_tl
1460      }
1461    }
1462    \__stex_modules_begin_module:
1463    \str_if_empty:NF \smoduleid {
1464      \stex_ref_new_doc_target:n \smoduleid
1465    }
1466    \stex_smsmode_do:
1467  } {
1468    \__stex_modules_end_module:
1469    \stex_if_smsmode:F {
1470      \end{stex_annotate_env}
1471      \clist_set:No \l_tmpa_clist \smoduletype
1472      \tl_clear:N \l_tmpa_tl
1473      \clist_map_inline:Nn \l_tmpa_clist {
1474        \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1475          \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1476        }
1477      }
1478      \tl_if_empty:NTF \l_tmpa_tl {
1479        \__stex_modules_smodule_end:
1480      }{
1481        \l_tmpa_tl
1482      }
1483    }
1484  }
```

\stexpatchmodule

```
1485 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1486 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1487
1488 \newcommand\stexpatchmodule[3][] {
1489     \str_set:Nx \l_tmpa_str{ #1 }
1490     \str_if_empty:NTF \l_tmpa_str {
1491        \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1492        \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1493     }{
1494        \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1495        \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1496     }
1497 }
```

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

27.2 Invoking modules

```
1498 \NewDocumentCommand \STEXModule { m } {
1499   \exp_args:NNx \str_set:Nn \l_tmpa_str { # 1 }
1500   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1501   \tl_set:Nn \l_tmpa_tl {
1502     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1503   }
1504   \seq_map_inline:Nn \l_stex_all_modules_seq {
1505     \str_set:Nn \l_tmpb_str { ##1 }
1506     \str_if_eq:eeT { \l_tmpa_str } {
1507       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1508     } {
1509       \seq_map_break:n {
1510         \tl_set:Nn \l_tmpa_tl {
1511           \stex_invoke_module:n { ##1 }
1512         }
1513       }
1514     }
1515   }
1516   \l_tmpa_tl
1517 }
1518
1519 \cs_new_protected:Nn \stex_invoke_module:n {
1520   \stex_debug:nn{modules}{Invoking~module~#1}
1521   \peek_charcode_remove:NTF ! {
1522     \__stex_modules_invoke_uri:nN { #1 }
1523   } {
1524     \peek_charcode_remove:NTF ? {
1525       \__stex_modules_invoke_symbol:nn { #1 }
1526     } {
1527       \msg_error:nnx{stex}{error/syntax}{
1528         ?~or~!~expected~after~
1529         \c_backslash_str STEXModule{#1}
1530       }
1531     }
1532   }
1533 }
1534
1535 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1536   \str_set:Nn #2 { #1 }
1537 }
1538
1539 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1540   \stex_invoke_symbol:n{#1?#2}
1541 }
```

(*End definition for* \STEXModule *and* \stex_invoke_module:n*. These functions are documented on page* *55.*)

```
1542 \bool_new:N \l_stex_in_meta_bool
1543 \bool_set_false:N \l_stex_in_meta_bool
```

```
1544 \cs_new_protected:Nn \stex_activate_module:n {
1545   \stex_debug:nn{modules}{Activating~module~#1}
1546   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1547     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1548     \use:c{ c_stex_module_#1_code }
1549   }
1550 }
```

(*End definition for* `\stex_activate_module:n`*. This function is documented on page* *56**.*)

```
1551 ⟨/package⟩
```

Chapter 28

sTEX
-Module Inheritance
Implementation

₁₅₅₂ ⟨∗package⟩
₁₅₅₃
₁₅₅₄ %%%%%%%%%%%%   inheritance.dtx   %%%%%%%%%%%%
₁₅₅₅

28.1   SMS Mode

₁₅₅₆ ⟨@@=stex_smsmode⟩

₁₅₅₇ \tl_new:N \g_stex_smsmode_allowedmacros_tl
₁₅₅₈ \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
₁₅₅₉ \seq_new:N \g_stex_smsmode_allowedenvs_seq
₁₅₆₀
₁₅₆₁ \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
₁₅₆₂   \makeatletter
₁₅₆₃   \makeatother
₁₅₆₄   \ExplSyntaxOn
₁₅₆₅   \ExplSyntaxOff
₁₅₆₆   \rustexBREAK
₁₅₆₇ }
₁₅₆₈
₁₅₆₉ \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
₁₅₇₀   \symdef
₁₅₇₁   \importmodule
₁₅₇₂   \notation
₁₅₇₃   \symdecl
₁₅₇₄   \STEXexport
₁₅₇₅   \inlineass
₁₅₇₆   \inlinedef
₁₅₇₇   \inlineex
₁₅₇₈   \endinput
₁₅₇₉   \setnotation

```
1580    \copynotation
1581    \assign
1582    \renamedecl
1583    \donotcopy
1584    \instantiate
1585 }
1586
1587 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1588    \tl_to_str:n {
1589      smodule,
1590      copymodule,
1591      interpretmodule,
1592      sdefinition,
1593      sexample,
1594      sassertion,
1595      sparagraph,
1596      mathstructure
1597    }
1598 }
```

(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 57.)

\stex_if_smsmode_p:
\stex_if_smsmode:*TF*

```
1599 \bool_new:N \g__stex_smsmode_bool
1600 \bool_set_false:N \g__stex_smsmode_bool
1601 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1602    \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1603 }
```

(End definition for \stex_if_smsmode:TF. This function is documented on page 57.)

__stex_smsmode_in_smsmode:nn

```
1604 \cs_new_protected:Nn \__stex_smsmode_in_smsmode:nn { \stex_suppress_html:n {
1605    \vbox_set:Nn \l_tmpa_box {
1606      \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1607      \bool_gset_true:N \g__stex_smsmode_bool
1608      #2
1609      \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1610    }
1611    \box_clear:N \l_tmpa_box
1612 } }
```

(End definition for __stex_smsmode_in_smsmode:nn.)

\stex_file_in_smsmode:nn

```
1613 \quark_new:N \q__stex_smsmode_break
1614
1615 \NewDocumentCommand \__stex_smsmode_importmodule: { O{} m } {
1616    \seq_gput_right:Nn \l__stex_smsmode_importmodules_seq {{#1}{#2}}
1617    \stex_smsmode_do:
1618 }
1619
1620 \cs_new_protected:Nn \__stex_smsmode_module:nn {
1621    \__stex_modules_args:n{#1}
```

```
1622    \stex_if_in_module:F {
1623      \str_if_empty:NF \l_stex_module_sig_str {
1624        \stex_modules_current_namespace:
1625        \str_set:Nx \l_stex_module_name_str { #2 }
1626        \stex_if_module_exists:nF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1627          \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1628          \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1629          \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1630          \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1631          \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1632          \str_set:Nx \l_tmpa_str {
1633            \stex_path_to_string:N \l_tmpa_seq /
1634            \l_tmpa_str . \l_stex_module_sig_str .tex
1635          }
1636          \IfFileExists \l_tmpa_str {
1637            \exp_args:NNx \seq_gput_right:Nn \l__stex_smsmode_sigmodules_seq \l_tmpa_str
1638          }{
1639            \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1640          }
1641        }
1642      }
1643    }
1644  }
1645
1646  \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1647    \stex_filestack_push:n{#1}
1648    \seq_gclear:N \l__stex_smsmode_importmodules_seq
1649    \seq_gclear:N \l__stex_smsmode_sigmodules_seq
1650    % ----- new ---------------------------
1651    \__stex_smsmode_in_smsmode:nn{#1}{
1652      \let\importmodule\__stex_smsmode_importmodule:
1653      \let\stex_module_setup:nn\__stex_smsmode_module:nn
1654      \let\__stex_modules_begin_module:\relax
1655      \let\__stex_modules_end_module:\relax
1656      \seq_clear:N \g_stex_smsmode_allowedenvs_seq
1657      \exp_args:NNx \seq_put_right:Nn \g_stex_smsmode_allowedenvs_seq {\tl_to_str:n{smodule}}
1658      \tl_clear:N \g_stex_smsmode_allowedmacros_tl
1659      \tl_clear:N \g_stex_smsmode_allowedmacros_escape_tl
1660      \tl_put_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\importmodule}
1661      \everyeof{\q__stex_smsmode_break\noexpand}
1662      \expandafter\expandafter\expandafter
1663      \stex_smsmode_do:
1664      \csname @ @ input\endcsname "#1"\relax
1665
1666      \seq_map_inline:Nn \l__stex_smsmode_sigmodules_seq {
1667        \stex_filestack_push:n{##1}
1668        \expandafter\expandafter\expandafter
1669        \stex_smsmode_do:
1670        \csname @ @ input\endcsname "##1"\relax
1671        \stex_filestack_pop:
1672      }
1673    }
1674    % ----- new ---------------------------
1675    \__stex_smsmode_in_smsmode:nn{#1} {
```

```
1676        #2
1677        % ----- new ---------------------------
1678        \begingroup
1679        %\stex_debug:nn{smsmode}{Here:~\seq_use:Nn\l__stex_smsmode_importmodules_seq, }
1680        \seq_map_inline:Nn \l__stex_smsmode_importmodules_seq {
1681          \stex_import_module_uri:nn ##1
1682          \stex_import_require_module:nnnn
1683            \l_stex_import_ns_str
1684            \l_stex_import_archive_str
1685            \l_stex_import_path_str
1686            \l_stex_import_name_str
1687        }
1688        \endgroup
1689        \stex_debug:nn{smsmode}{Actually~loading~file~#1}
1690        % ----- new ---------------------------
1691        \everyeof{\q__stex_smsmode_break\noexpand}
1692        \expandafter\expandafter\expandafter
1693        \stex_smsmode_do:
1694        \csname @ @ input\endcsname "#1"\relax
1695      }
1696      \stex_filestack_pop:
1697  }
```

(*End definition for* `\stex_file_in_smsmode:nn`. *This function is documented on page 58.*)

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

```
1698  \cs_new_protected:Npn \stex_smsmode_do: {
1699    \stex_if_smsmode:T {
1700      \__stex_smsmode_do:w
1701    }
1702  }
1703  \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1704    \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 }}{
1705      \expandafter\if\expandafter\relax\noexpand#1
1706        \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1707      \else\expandafter\__stex_smsmode_do:w\fi
1708    }{
1709      \__stex_smsmode_do:w %#1
1710    }
1711  }
1712  \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1713    \cs_if_eq:NNF #1 \q__stex_smsmode_break {
1714      \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1715        #1\__stex_smsmode_do:w
1716      }{
1717        \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1718          #1
1719        }{
1720          \cs_if_eq:NNTF \begin #1 {
1721            \__stex_smsmode_check_begin:n
1722          }{
1723            \cs_if_eq:NNTF \end #1 {
1724              \__stex_smsmode_check_end:n
```

137

```
1725              }{
1726                \__stex_smsmode_do:w
1727              }
1728            }
1729          }
1730        }
1731      }
1732    }
1733
1734    \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1735      \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1736        \begin{#1}
1737      }{
1738        \__stex_smsmode_do:w
1739      }
1740    }
1741    \cs_new_protected:Nn \__stex_smsmode_check_end:n {
1742      \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1743        \end{#1}\__stex_smsmode_do:w
1744      }{
1745        \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1746      }
1747    }
```

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

28.2   Inheritance

```
1748  ⟨@@=stex_importmodule⟩
```

```
1749  \cs_new_protected:Nn \stex_import_module_uri:nn {
1750    \str_set:Nx \l_stex_import_archive_str { #1 }
1751    \str_set:Nn \l_stex_import_path_str { #2 }
1752
1753    \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1754    \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1755    \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1756
1757    \stex_modules_current_namespace:
1758    \bool_lazy_all:nTF {
1759      {\str_if_empty_p:N \l_stex_import_archive_str}
1760      {\str_if_empty_p:N \l_stex_import_path_str}
1761      {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1762    }{
1763      \str_set_eq:NN \l_stex_import_path_str \l_stex_module_subpath_str
1764      \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1765    }{
1766      \str_if_empty:NT \l_stex_import_archive_str {
1767        \prop_if_exist:NT \l_stex_current_repository_prop {
1768          \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1769        }
1770      }
1771      \str_if_empty:NTF \l_stex_import_archive_str {
```

```
1772        \str_if_empty:NF \l_stex_import_path_str {
1773          \str_set:Nx \l_stex_import_ns_str {
1774            \l_stex_module_ns_str / \l_stex_import_path_str
1775          }
1776        }
1777      }{
1778        \stex_require_repository:n \l_stex_import_archive_str
1779        \prop_get:cnN { c_stex_mathhub_\l_stex_import_archive_str _manifest_prop } { ns }
1780          \l_stex_import_ns_str
1781        \str_if_empty:NF \l_stex_import_path_str {
1782          \str_set:Nx \l_stex_import_ns_str {
1783            \l_stex_import_ns_str / \l_stex_import_path_str
1784          }
1785        }
1786      }
1787    }
1788  }
```

(*End definition for* `\stex_import_module_uri:nn`. *This function is documented on page 59.*)

`\l_stex_import_name_str` Store the return values of `\stex_import_module_uri:nn`.
`\l_stex_import_archive_str`
`\l_stex_import_path_str`
`\l_stex_import_ns_str`

```
1789 \str_new:N \l_stex_import_name_str
1790 \str_new:N \l_stex_import_archive_str
1791 \str_new:N \l_stex_import_path_str
1792 \str_new:N \l_stex_import_ns_str
```

(*End definition for* `\l_stex_import_name_str` *and others. These variables are documented on page 59.*)

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

```
1793 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1794    \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1795
1796      %\stex_debug:nn{requiremodule}{Here:\\~~1:~#1\\~~2:~#2\\~~3:~#3\\~~4:~#4}
1797
1798      \exp_args:NNxx \seq_set_split:Nnn \l_tmpa_seq {\tl_to_str:n{/}} {#4}
1799      \seq_get_left:NN \l_tmpa_seq \l_tmpc_str
1800
1801      %\stex_debug:nn{requiremodule}{Top~module:\l_tmpc_str}
1802
1803      % archive
1804      \str_set:Nx \l_tmpa_str { #2 }
1805      \str_if_empty:NTF \l_tmpa_str {
1806        \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1807      } {
1808        \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1809        \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1810        \seq_put_right:Nn \l_tmpa_seq { source }
1811      }
1812
1813      % path
1814      \str_set:Nx \l_tmpb_str { #3 }
1815      \str_if_empty:NTF \l_tmpb_str {
1816        \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / \l_tmpc_str }
1817
```

```
1818        \ltx@ifpackageloaded{babel} {
1819          \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1820              { \languagename } \l_tmpb_str {
1821                \msg_error:nnx{stex}{error/unknownlanguage}{\languagename}
1822              }
1823        } {
1824          \str_clear:N \l_tmpb_str
1825        }
1826
1827        %\stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1828        \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1829          \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1830        }{
1831          %\stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1832          \IfFileExists{ \l_tmpa_str.tex }{
1833            \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1834          }{
1835            % try english as default
1836            %\stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1837            \IfFileExists{ \l_tmpa_str.en.tex }{
1838              \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1839            }{
1840              \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1841            }
1842          }
1843        }
1844
1845      } {
1846        \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1847        \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1848
1849        \ltx@ifpackageloaded{babel} {
1850          \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1851              { \languagename } \l_tmpb_str {
1852                \msg_error:nnx{stex}{error/unknownlanguage}{\languagename}
1853              }
1854        } {
1855          \str_clear:N \l_tmpb_str
1856        }
1857
1858        \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1859
1860        %\stex_debug:nn{modules}{Checking~\l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex}
1861        \IfFileExists{ \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }{
1862          \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.te
1863        }{
1864          %\stex_debug:nn{modules}{Checking~\l_tmpa_str/\l_tmpc_str.tex}
1865          \IfFileExists{ \l_tmpa_str/\l_tmpc_str.tex }{
1866            \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.tex }
1867          }{
1868            % try english as default
1869            %\stex_debug:nn{modules}{Checking~\l_tmpa_str/\l_tmpc_str.en.tex}
1870            \IfFileExists{ \l_tmpa_str/\l_tmpc_str.en.tex }{
1871              \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.en.tex }
```

140

```
1872            }{
1873              %\stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1874              \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1875                \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1876              }{
1877                %\stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1878                \IfFileExists{ \l_tmpa_str.tex }{
1879                  \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1880                }{
1881                  % try english as default
1882                  %\stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1883                  \IfFileExists{ \l_tmpa_str.en.tex }{
1884                    \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1885                  }{
1886                    \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1887                  }
1888                }
1889              }
1890            }
1891          }
1892        }
1893      }
1894
1895      \str_if_eq:eeF{\g__stex_importmodule_file_str}{\seq_use:Nn \g_stex_currentfile_seq /}{
1896        \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
1897          \seq_clear:N \l_stex_all_modules_seq
1898          \str_clear:N \l_stex_current_module_str
1899          \str_set:Nx \l_tmpb_str { #2 }
1900          \str_if_empty:NF \l_tmpb_str {
1901            \stex_set_current_repository:n { #2 }
1902          }
1903          \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1904        }
1905
1906        \stex_if_module_exists:nF { #1 ? #4 } {
1907          \msg_error:nnx{stex}{error/unknownmodule}{
1908            #1?#4~(in~file~\g__stex_importmodule_file_str)
1909          }
1910        }
1911      }
1912
1913    }
1914    \stex_activate_module:n { #1 ? #4 }
1915  }
```

(*End definition for* \stex_import_require_module:nnnn. *This function is documented on page 59.*)

```
1916  \NewDocumentCommand \importmodule { O{} m } {
1917    \stex_import_module_uri:nn { #1 } { #2 }
1918    \stex_debug:nn{modules}{Importing~module:~
1919      \l_stex_import_ns_str ? \l_stex_import_name_str
1920    }
1921    \stex_import_require_module:nnnn
```

141

```
1922    { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1923    { \l_stex_import_path_str } { \l_stex_import_name_str }
1924    \stex_if_smsmode:F {
1925      \stex_annotate_invisible:nnn
1926        {import} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1927    }
1928    \exp_args:Nx \stex_add_to_current_module:n {
1929      \stex_import_require_module:nnnn
1930      { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1931      { \l_stex_import_path_str } { \l_stex_import_name_str }
1932    }
1933    \exp_args:Nx \stex_add_import_to_current_module:n {
1934      \l_stex_import_ns_str ? \l_stex_import_name_str
1935    }
1936    \stex_smsmode_do:
1937    \ignorespacesandpars
1938 }
1939 \stex_deactivate_macro:Nn \importmodule {module~environments}
```

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

\usemodule

```
1940 \NewDocumentCommand \usemodule { O{} m } {
1941    \stex_if_smsmode:F {
1942      \stex_import_module_uri:nn { #1 } { #2 }
1943      \stex_import_require_module:nnnn
1944      { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1945      { \l_stex_import_path_str } { \l_stex_import_name_str }
1946      \stex_annotate_invisible:nnn
1947        {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1948    }
1949    \stex_smsmode_do:
1950    \ignorespacesandpars
1951 }
```

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

```
1952 \cs_new_protected:Nn \stex_csl_to_imports:Nn {
1953    \tl_if_empty:nF{#2}{
1954      \clist_set:Nn \l_tmpa_clist {#2}
1955      \clist_map_inline:Nn \l_tmpa_clist {
1956        \tl_if_head_eq_charcode:nNTF {##1}[{
1957          #1 ##1
1958        }{
1959          #1{##1}
1960        }
1961      }
1962    }
1963 }
1964 \cs_generate_variant:Nn \stex_csl_to_imports:Nn {No}
1965
1966
1967 ⟨/package⟩
```

142

Chapter 29

sTeX
-Symbols Implementation

```
1968 ⟨∗package⟩
1969
1970 %%%%%%%%%%%%   symbols.dtx   %%%%%%%%%%%%
1971
```

Warnings and error messages

```
1972 \msg_new:nnn{stex}{error/wrongargs}{
1973   args~value~in~symbol~declaration~for~#1~
1974   needs~to~be~i,~a,~b~or~B,~but~#2~given
1975 }
1976 \msg_new:nnn{stex}{error/unknownsymbol}{
1977   No~symbol~#1~found!
1978 }
1979 \msg_new:nnn{stex}{error/seqlength}{
1980   Expected~#1~arguments;~got~#2!
1981 }
1982 \msg_new:nnn{stex}{error/unknownnotation}{
1983   Unknown~notation~#1~for~#2!
1984 }
```

29.1   Symbol Declarations

```
1985 ⟨@@=stex_symdecl⟩
```

\stex_all_symbols:n   Map over all available symbols

```
1986 \cs_new_protected:Nn \stex_all_symbols:n {
1987   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
1988   \seq_map_inline:Nn \l_stex_all_modules_seq {
1989     \seq_map_inline:cn{c_stex_module_##1_constants}{
1990       \__stex_symdecl_all_symbols_cs{##1?####1}
1991     }
1992   }
1993 }
```

(*End definition for* \stex_all_symbols:n. *This function is documented on page 61.*)

```
1994 \NewDocumentCommand \STEXsymbol { m } {
1995   \stex_get_symbol:n { #1 }
1996   \exp_args:No
1997   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1998 }
```

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

symdecl arguments:

```
1999 \keys_define:nn { stex / symdecl } {
2000   name        .str_set_x:N  = \l_stex_symdecl_name_str ,
2001   local       .bool_set:N   = \l_stex_symdecl_local_bool ,
2002   args        .str_set_x:N  = \l_stex_symdecl_args_str ,
2003   type        .tl_set:N     = \l_stex_symdecl_type_tl ,
2004   deprecate   .str_set_x:N  = \l_stex_symdecl_deprecate_str ,
2005   align       .str_set:N    = \l_stex_symdecl_align_str , % TODO(?)
2006   gfc         .str_set:N    = \l_stex_symdecl_gfc_str , % TODO(?)
2007   specializes .str_set:N    = \l_stex_symdecl_specializes_str , % TODO(?)
2008   def         .tl_set:N     = \l_stex_symdecl_definiens_tl ,
2009   assoc       .choices:nn   =
2010       {bin,binl,binr,pre,conj,pwconj}
2011       {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}
2012 }
2013
2014 \bool_new:N \l_stex_symdecl_make_macro_bool
2015
2016 \cs_new_protected:Nn \__stex_symdecl_args:n {
2017   \str_clear:N \l_stex_symdecl_name_str
2018   \str_clear:N \l_stex_symdecl_args_str
2019   \str_clear:N \l_stex_symdecl_deprecate_str
2020   \str_clear:N \l_stex_symdecl_assoctype_str
2021   \bool_set_false:N \l_stex_symdecl_local_bool
2022   \tl_clear:N \l_stex_symdecl_type_tl
2023   \tl_clear:N \l_stex_symdecl_definiens_tl
2024
2025   \keys_set:nn { stex / symdecl } { #1 }
2026 }
```

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

```
2027
2028 \NewDocumentCommand \symdecl { s m O{}} {
2029   \__stex_symdecl_args:n { #3 }
2030   \IfBooleanTF #1 {
2031     \bool_set_false:N \l_stex_symdecl_make_macro_bool
2032   } {
2033     \bool_set_true:N \l_stex_symdecl_make_macro_bool
2034   }
2035   \stex_symdecl_do:n { #2 }
2036   \stex_smsmode_do:
2037 }
2038
2039 \cs_new_protected:Nn \stex_symdecl_do:nn {
```

144

```
2040      \__stex_symdecl_args:n{#1}
2041      \bool_set_false:N \l_stex_symdecl_make_macro_bool
2042      \stex_symdecl_do:n{#2}
2043    }
2044
2045    \stex_deactivate_macro:Nn \symdecl {module~environments}
```

(*End definition for* \symdecl. *This function is documented on page* *60.*)

\stex_symdecl_do:n

```
2046    \cs_new_protected:Nn \stex_symdecl_do:n {
2047      \stex_if_in_module:F {
2048        % TODO throw error? some default namespace?
2049      }
2050
2051      \str_if_empty:NT \l_stex_symdecl_name_str {
2052        \str_set:Nx \l_stex_symdecl_name_str { #1 }
2053      }
2054
2055      \prop_if_exist:cT { l_stex_symdecl_
2056          \l_stex_current_module_str ?
2057          \l_stex_symdecl_name_str
2058        _prop
2059      }{
2060        % TODO throw error (beware of circular dependencies)
2061      }
2062
2063      \prop_clear:N \l_tmpa_prop
2064      \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
2065      \seq_clear:N \l_tmpa_seq
2066      \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
2067      \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
2068
2069      \str_if_empty:NT \l_stex_symdecl_deprecate_str {
2070        \str_if_empty:NF \l_stex_module_deprecate_str {
2071          \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
2072        }
2073      }
2074      \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
2075
2076      \exp_args:No \stex_add_constant_to_current_module:n {
2077        \l_stex_symdecl_name_str
2078      }
2079
2080      % arity/args
2081      \int_zero:N \l_tmpb_int
2082
2083      \bool_set_true:N \l_tmpa_bool
2084      \str_map_inline:Nn \l_stex_symdecl_args_str {
2085        \token_case_meaning:NnF ##1 {
2086          0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2087          {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2088          {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2089          {\tl_to_str:n a} {
```

145

```
2090        \bool_set_false:N \l_tmpa_bool
2091        \int_incr:N \l_tmpb_int
2092      }
2093     {\tl_to_str:n B} {
2094        \bool_set_false:N \l_tmpa_bool
2095        \int_incr:N \l_tmpb_int
2096      }
2097    }{
2098      \msg_error:nnxx{stex}{error/wrongargs}{
2099        \l_stex_current_module_str ?
2100        \l_stex_symdecl_name_str
2101      }{##1}
2102    }
2103  }
2104  \bool_if:NTF \l_tmpa_bool {
2105    % possibly numeric
2106    \str_if_empty:NTF \l_stex_symdecl_args_str {
2107      \prop_put:Nnn \l_tmpa_prop { args } {}
2108      \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2109    }{
2110      \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
2111      \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2112      \str_clear:N \l_tmpa_str
2113      \int_step_inline:nn \l_tmpa_int {
2114        \str_put_right:Nn \l_tmpa_str i
2115      }
2116      \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
2117    }
2118  } {
2119    \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
2120    \prop_put:Nnx \l_tmpa_prop { arity }
2121      { \str_count:N \l_stex_symdecl_args_str }
2122  }
2123  \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2124
2125  \tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
2126    \prop_put:Nnx \l_tmpa_prop { defined }{ false }
2127  }{
2128    \prop_put:Nnx \l_tmpa_prop { defined }{ true }
2129  }
2130
2131  % semantic macro
2132
2133  \bool_if:NT \l_stex_symdecl_make_macro_bool {
2134    \exp_args:Nx \stex_do_up_to_module:n {
2135      \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2136        \l_stex_current_module_str ? \l_stex_symdecl_name_str
2137      }}
2138    }
2139  }
2140
2141  \stex_debug:nn{symbols}{New~symbol:~
2142    \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2143    Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
```

146

```
2144     Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
2145     Definiens:~\exp_not:o {\l_stex_symdecl_definiens_tl}
2146   }
2147
2148   % circular dependencies require this:
2149   \stex_if_do_html:T {
2150     \stex_annotate_invisible:nnn {symdecl} {
2151       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2152     } {
2153       \tl_if_empty:NF \l_stex_symdecl_type_tl {
2154         \stex_annotate_invisible:nnn{type}{}{$\l_stex_symdecl_type_tl$}
2155       }
2156       \stex_annotate_invisible:nnn{args}{}{
2157         \prop_item:Nn \l_tmpa_prop { args }
2158       }
2159       \stex_annotate_invisible:nnn{macroname}{#1}{}
2160       \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2161         \stex_annotate_invisible:nnn{definiens}{}
2162           {$\l_stex_symdecl_definiens_tl$}
2163       }
2164       \str_if_empty:NF \l_stex_symdecl_assoctype_str {
2165         \stex_annotate_invisible:nnn{assoctype}{\l_stex_symdecl_assoctype_str}{}
2166       }
2167     }
2168   }
2169   \prop_if_exist:cF {
2170     l_stex_symdecl_
2171     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2172     _prop
2173   } {
2174     \bool_if:NTF \l_stex_symdecl_local_bool \stex_do_up_to_module:x \stex_execute_in_module:
2175       \__stex_symdecl_restore_symbol:nnnnnnn
2176         {\l_stex_symdecl_name_str}
2177         { \prop_item:Nn \l_tmpa_prop {args} }
2178         { \prop_item:Nn \l_tmpa_prop {arity} }
2179         { \prop_item:Nn \l_tmpa_prop {assocs} }
2180         { \prop_item:Nn \l_tmpa_prop {defined} }
2181         {\bool_if:NT \l_stex_symdecl_make_macro_bool {#1} }
2182         {\l_stex_current_module_str}
2183     }
2184   }
2185 }
2186 \cs_new_protected:Nn \__stex_symdecl_restore_symbol:nnnnnnn {
2187   \prop_clear:N \l_tmpa_prop
2188   \prop_put:Nnn \l_tmpa_prop { module } { #7 }
2189   \prop_put:Nnn \l_tmpa_prop { name } { #1}
2190   \prop_put:Nnn \l_tmpa_prop { args } {#2}
2191   \prop_put:Nnn \l_tmpa_prop { arity } { #3 }
2192   \prop_put:Nnn \l_tmpa_prop { assocs } { #4 }
2193   \prop_put:Nnn \l_tmpa_prop { defined } { #5 }
2194   \tl_if_empty:nF{#6}{
2195     \tl_set:cx{#6}{\stex_invoke_symbol:n{\detokenize{#7 ? #1}}}
2196   }
2197   \prop_set_eq:cN{l_stex_symdecl_ \detokenize{#7 ? #1} _prop}\l_tmpa_prop
```

147

```
2198      \seq_clear:c{l_stex_symdecl_ \detokenize{#7 ? #1} _notations}
2199  }
```

(*End definition for* \stex_symdecl_do:n. *This function is documented on page* *61.*)

\stex_get_symbol:n

```
2200  \str_new:N \l_stex_get_symbol_uri_str
2201
2202  \cs_new_protected:Nn \stex_get_symbol:n {
2203    \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2204      \tl_set:Nn \l_tmpa_tl { #1 }
2205      \__stex_symdecl_get_symbol_from_cs:
2206    }{
2207      % argument is a string
2208      % is it a command name?
2209      \cs_if_exist:cTF { #1 }{
2210        \cs_set_eq:Nc \l_tmpa_tl { #1 }
2211        \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2212        \str_if_empty:NTF \l_tmpa_str {
2213          \exp_args:Nx \cs_if_eq:NNTF {
2214            \tl_head:N \l_tmpa_tl
2215          } \stex_invoke_symbol:n {
2216            \__stex_symdecl_get_symbol_from_cs:
2217          }{
2218            \__stex_symdecl_get_symbol_from_string:n { #1 }
2219          }
2220        } {
2221          \__stex_symdecl_get_symbol_from_string:n { #1 }
2222        }
2223      }{
2224        % argument is not a command name
2225        \__stex_symdecl_get_symbol_from_string:n { #1 }
2226        % \l_stex_all_symbols_seq
2227      }
2228    }
2229    \str_if_eq:eeF {
2230      \prop_item:cn {
2231        l_stex_symdecl_\l_stex_get_symbol_uri_str _prop
2232      }{ deprecate }
2233    }{}{
2234      \msg_warning:nnxx{stex}{warning/deprecated}{
2235        Symbol~\l_stex_get_symbol_uri_str
2236      }{
2237        \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{ deprecate }
2238      }
2239    }
2240  }
2241
2242  \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2243    \tl_set:Nn \l_tmpa_tl {
2244      \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2245    }
2246    \str_set:Nn \l_tmpa_str { #1 }
2247    \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
```

```
2248
2249    \stex_all_symbols:n {
2250      \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2251        \seq_map_break:n{\seq_map_break:n{
2252          \tl_set:Nn \l_tmpa_tl {
2253            \str_set:Nn \l_stex_get_symbol_uri_str { ##1 }
2254          }
2255        }}
2256      }
2257    }
2258
2259    \l_tmpa_tl
2260  }
2261
2262  \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
2263    \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2264      { \tl_tail:N \l_tmpa_tl }
2265    \tl_if_single:NTF \l_tmpa_tl {
2266      \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2267        \exp_after:wN \str_set:Nn \exp_after:wN
2268          \l_stex_get_symbol_uri_str \l_tmpa_tl
2269      }{
2270        % TODO
2271        % tail is not a single group
2272      }
2273    }{
2274      % TODO
2275      % tail is not a single group
2276    }
2277  }
```

(*End definition for* `\stex_get_symbol:n`. *This function is documented on page* *61*.)

29.2 Notations

2278 ⟨@@=stex_notation⟩

notation arguments:

```
2279  \keys_define:nn { stex / notation } {
2280  %  lang    .tl_set_x:N  = \l__stex_notation_lang_str ,
2281    variant .tl_set_x:N  = \l__stex_notation_variant_str ,
2282    prec    .str_set_x:N = \l__stex_notation_prec_str ,
2283    op      .tl_set:N    = \l__stex_notation_op_tl ,
2284    primary .bool_set:N  = \l__stex_notation_primary_bool ,
2285    primary .default:n   = {true} ,
2286    unknown .code:n      = \str_set:Nx
2287        \l__stex_notation_variant_str \l_keys_key_str
2288  }
2289
2290  \cs_new_protected:Nn \_stex_notation_args:n {
2291  %  \str_clear:N \l__stex_notation_lang_str
2292    \str_clear:N \l__stex_notation_variant_str
2293    \str_clear:N \l__stex_notation_prec_str
2294    \tl_clear:N \l__stex_notation_op_tl
```

```
2295    \bool_set_false:N \l__stex_notation_primary_bool
2296
2297    \keys_set:nn { stex / notation } { #1 }
2298 }
```

\notation

```
2299 \NewDocumentCommand \notation { s m O{}} {
2300    \_stex_notation_args:n { #3 }
2301    \tl_clear:N \l_stex_symdecl_definiens_tl
2302    \stex_get_symbol:n { #2 }
2303    \tl_set:Nn \l_stex_notation_after_do_tl {
2304       \__stex_notation_final:
2305       \IfBooleanTF#1{
2306          \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2307       }{}
2308       \stex_smsmode_do:\ignorespacesandpars
2309    }
2310    \stex_notation_do:nnnnn
2311       { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } }
2312       { \prop_item:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } }
2313       { \l__stex_notation_variant_str }
2314       { \l__stex_notation_prec_str}
2315 }
2316 \stex_deactivate_macro:Nn \notation {module~environments}
```

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

\stex_notation_do:nnnnn

```
2317 \seq_new:N \l__stex_notation_precedences_seq
2318 \tl_new:N \l__stex_notation_opprec_tl
2319 \int_new:N \l__stex_notation_currarg_int
2320 \tl_new:N \stex_symbol_after_invokation_tl
2321
2322 \cs_new_protected:Nn \stex_notation_do:nnnnn {
2323    \let\l_stex_current_symbol_str\relax
2324    \seq_clear:N \l__stex_notation_precedences_seq
2325    \tl_clear:N \l__stex_notation_opprec_tl
2326    \str_set:Nx \l__stex_notation_args_str { #1 }
2327    \str_set:Nx \l__stex_notation_arity_str { #2 }
2328    \str_set:Nx \l__stex_notation_suffix_str { #3 }
2329    \str_set:Nx \l__stex_notation_prec_str { #4 }
2330
2331    % precedences
2332    \str_if_empty:NTF \l__stex_notation_prec_str {
2333       \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2334          \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2335       }{
2336          \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2337       }
2338    } {
2339       \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2340          \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2341          \int_step_inline:nn { \l__stex_notation_arity_str } {
2342             \exp_args:NNo
2343             \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
```

150

```
2344        }
2345      }{
2346        \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2347        \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2348          \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2349          \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2350            \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2351              \l_tmpa_seq {\tl_to_str:n{x} } { \l_tmpa_str }
2352            \seq_map_inline:Nn \l_tmpa_seq {
2353              \seq_put_right:Nn \l_tmpb_seq { ##1 }
2354            }
2355          }
2356        }{
2357          \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2358            \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2359          }{
2360            \tl_set:No \l__stex_notation_opprec_tl { 0 }
2361          }
2362        }
2363      }
2364    }
2365
2366    \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2367    \int_step_inline:nn { \l__stex_notation_arity_str } {
2368      \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
2369        \exp_args:NNo
2370        \seq_put_right:No \l__stex_notation_precedences_seq {
2371          \l__stex_notation_opprec_tl
2372        }
2373      }
2374    }
2375    \tl_clear:N \l_stex_notation_dummyargs_tl
2376
2377    \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2378      \exp_args:NNe
2379      \cs_set:Npn \l_stex_notation_macrocode_cs {
2380        \_stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2381          { \l__stex_notation_suffix_str }
2382          { \l__stex_notation_opprec_tl }
2383          { \exp_not:n { #5 } }
2384      }
2385      \l_stex_notation_after_do_tl
2386    }{
2387      \str_if_in:NnTF \l__stex_notation_args_str b {
2388        \exp_args:Nne \use:nn
2389        {
2390        \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2391        \cs_set:Npn \l__stex_notation_arity_str } { {
2392        \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2393          { \l__stex_notation_suffix_str }
2394          { \l__stex_notation_opprec_tl }
2395          { \exp_not:n { #5 } }
2396      }}
2397    }{
```

```
2398        \str_if_in:NnTF \l__stex_notation_args_str B {
2399          \exp_args:Nne \use:nn
2400          {
2401          \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2402          \cs_set:Npn \l__stex_notation_arity_str } { {
2403            \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2404              { \l__stex_notation_suffix_str }
2405              { \l__stex_notation_opprec_tl }
2406              { \exp_not:n { #5 } }
2407          } }
2408        }{
2409          \exp_args:Nne \use:nn
2410          {
2411          \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2412          \cs_set:Npn \l__stex_notation_arity_str } { {
2413            \_stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2414              { \l__stex_notation_suffix_str }
2415              { \l__stex_notation_opprec_tl }
2416              { \exp_not:n { #5 } }
2417          } }
2418        }
2419      }
2420
2421      \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2422      \int_zero:N \l__stex_notation_currarg_int
2423      \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2424      \__stex_notation_arguments:
2425    }
2426 }
```

(*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

```
2427 \cs_new_protected:Nn \__stex_notation_arguments: {
2428    \int_incr:N \l__stex_notation_currarg_int
2429    \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2430      \l_stex_notation_after_do_tl
2431    }{
2432      \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2433      \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2434      \str_if_eq:VnTF \l_tmpa_str a {
2435        \__stex_notation_argument_assoc:nn{a}
2436      }{
2437        \str_if_eq:VnTF \l_tmpa_str B {
2438          \__stex_notation_argument_assoc:nn{B}
2439        }{
2440          \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpb_str
2441          \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2442            { \_stex_term_math_arg:nnn
2443              { \l_tmpa_str\int_use:N \l__stex_notation_currarg_int }
2444              { \l_tmpb_str }
2445              { ####\int_use:N \l__stex_notation_currarg_int }
2446          }
2447        }
```

152

```
2448            \__stex_notation_arguments:
2449          }
2450        }
2451      }
2452    }
```

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

`\__stex_notation_argument_assoc:nn`

```
2453  \cs_new_protected:Nn \__stex_notation_argument_assoc:nn {
2454
2455    \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2456      {\l__stex_notation_arity_str}{
2457      #2
2458    }
2459    \int_zero:N \l_tmpa_int
2460    \tl_clear:N \l_tmpa_tl
2461    \str_map_inline:Nn \l__stex_notation_args_str {
2462      \int_incr:N \l_tmpa_int
2463      \tl_put_right:Nx \l_tmpa_tl {
2464        \str_if_eq:nnTF {##1}{a}{ {} }{
2465          \str_if_eq:nnTF {##1}{B}{ {} }{
2466            {\_stex_term_arg:nn{##1\int_use:N \l_tmpa_int}{############### \int_use:N \l_tmpa
2467          }
2468        }
2469      }
2470    }
2471    \exp_after:wN\exp_after:wN\exp_after:wN \def
2472    \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2473    \exp_after:wN\exp_after:wN\exp_after:wN ##
2474    \exp_after:wN\exp_after:wN\exp_after:wN 1
2475    \exp_after:wN\exp_after:wN\exp_after:wN ##
2476    \exp_after:wN\exp_after:wN\exp_after:wN 2
2477    \exp_after:wN\exp_after:wN\exp_after:wN {
2478      \exp_after:wN \exp_after:wN \exp_after:wN
2479      \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2480        \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2481      }
2482    }
2483
2484    \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2485    \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
2486      \_stex_term_math_assoc_arg:nnnn
2487        { #1\int_use:N \l__stex_notation_currarg_int }
2488        { \l_tmpa_str }
2489        { ####\int_use:N \l__stex_notation_currarg_int }
2490        { \l_tmpa_cs {####1} {####2} }
2491    } }
2492    \__stex_notation_arguments:
2493  }
```

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

`\__stex_notation_final:` Called after processing all notation arguments

```
2494 \cs_new_protected:Nn \__stex_notation_restore_notation:nnnnn {
2495   \cs_generate_from_arg_count:cNnn{stex_notation_\detokenize{#1} \c_hash_str \detokenize{#2}
2496   \cs_set_nopar:Npn {#3}{#4}
2497   \tl_if_empty:nF {#5}{
2498     \tl_set:cn{stex_op_notation_\detokenize{#1} \c_hash_str \detokenize{#2}_cs}{ \comp{ #5 }
2499   }
2500   \seq_if_exist:cT { l_stex_symdecl_\detokenize{#1} _notations }{
2501     \seq_put_right:cx { l_stex_symdecl_\detokenize{#1} _notations } { \detokenize{#2} }
2502   }
2503 }
2504
2505 \cs_new_protected:Nn \__stex_notation_final: {
2506
2507   \stex_execute_in_module:x {
2508     \__stex_notation_restore_notation:nnnnn
2509       {\l_stex_get_symbol_uri_str}
2510       {\l__stex_notation_suffix_str}
2511       {\l__stex_notation_arity_str}
2512       {
2513         \exp_after:wN \exp_after:wN \exp_after:wN
2514         \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2515         { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2516       }
2517       {\exp_args:No \exp_not:n \l__stex_notation_op_tl }
2518   }
2519
2520   \stex_debug:nn{symbols}{
2521     Notation~\l__stex_notation_suffix_str
2522     ~for~\l_stex_get_symbol_uri_str^^J
2523     Operator~precedence:~\l__stex_notation_opprec_tl^^J
2524     Argument~precedences:~
2525       \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2526     Notation: \cs_meaning:c {
2527       stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2528       \l__stex_notation_suffix_str
2529       _cs
2530     }
2531   }
2532     % HTML annotations
2533   \stex_if_do_html:T {
2534     \stex_annotate_invisible:nnn { notation }
2535     { \l_stex_get_symbol_uri_str } {
2536       \stex_annotate_invisible:nnn { notationfragment }
2537         { \l__stex_notation_suffix_str }{}
2538       \stex_annotate_invisible:nnn { precedence }
2539         { \l__stex_notation_prec_str }{}
2540
2541       \int_zero:N \l_tmpa_int
2542       \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2543       \tl_clear:N \l_tmpa_tl
2544       \int_step_inline:nn { \l__stex_notation_arity_str }{
2545         \int_incr:N \l_tmpa_int
2546         \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2547         \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_rem
```

154

```
2548          \str_if_eq:VnTF \l_tmpb_str a {
2549            \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2550              \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2551              \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2552            } }
2553          }{
2554            \str_if_eq:VnTF \l_tmpb_str B {
2555              \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2556                \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2557                \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2558              } }
2559            }{
2560              \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2561                \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
2562              } }
2563            }
2564          }
2565        }
2566        \stex_annotate_invisible:nnn { notationcomp }{}{
2567          \str_set:Nx \l_stex_current_symbol_str {\l_stex_get_symbol_uri_str }
2568          $ \exp_args:Nno \use:nn { \use:c {
2569            stex_notation_ \l_stex_current_symbol_str
2570            \c_hash_str \l__stex_notation_suffix_str _cs
2571          } } { \l_tmpa_tl } $
2572        }
2573      }
2574    }
2575 }
```

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

\setnotation

```
2576 \keys_define:nn { stex / setnotation } {
2577 %  lang     .tl_set_x:N  = \l__stex_notation_lang_str ,
2578   variant .tl_set_x:N  = \l__stex_notation_variant_str ,
2579   unknown .code:n      = \str_set:Nx
2580       \l__stex_notation_variant_str \l_keys_key_str
2581 }
2582
2583 \cs_new_protected:Nn \_stex_setnotation_args:n {
2584 % \str_clear:N \l__stex_notation_lang_str
2585   \str_clear:N \l__stex_notation_variant_str
2586   \keys_set:nn { stex / setnotation } { #1 }
2587 }
2588
2589 \cs_new_protected:Nn \__stex_notation_setnotation:nn {
2590   \seq_if_exist:cT{l_stex_symdecl_#1_notations}{
2591     \seq_remove_all:cn { l_stex_symdecl_#1 _notations }{ #2 }
2592     \seq_put_left:cn { l_stex_symdecl_#1 _notations }{ #2 }
2593   }
2594 }
2595
2596 \cs_new_protected:Nn \stex_setnotation:n {
2597   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1 _notations }
```

```
2598       { \l__stex_notation_variant_str }{
2599         \stex_execute_in_module:x{ \__stex_notation_setnotation:nn {#1}{\l__stex_notation_vari
2600         \stex_debug:nn {notations}{
2601           Setting~default~notation~
2602           {\l__stex_notation_variant_str }~for~
2603           #1 \\
2604           \expandafter\meaning\csname
2605           l_stex_symdecl_#1 _notations\endcsname
2606         }
2607       }{
2608         \msg_error:nnxx{stex}{unknownnotation}{\l__stex_notation_variant_str}{#1}
2609       }
2610 }
2611
2612 \NewDocumentCommand \setnotation {m m} {
2613   \stex_get_symbol:n { #1 }
2614   \_stex_setnotation_args:n { #2 }
2615   \stex_setnotation:n{\l_stex_get_symbol_uri_str}
2616   \stex_smsmode_do:\ignorespacesandpars
2617 }
2618
2619 \cs_new_protected:Nn \stex_copy_notations:nn {
2620   \stex_debug:nn {notations}{
2621     Copying~notations~from~#2~to~#1\\
2622     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
2623   }
2624   \tl_clear:N \l_tmpa_tl
2625   \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2626     \tl_put_right:Nn \l_tmpa_tl { {## ##1} }
2627   }
2628   \seq_map_inline:cn {l_stex_symdecl_#2_notations}{
2629     \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2630     \edef \l_tmpa_tl {
2631       \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2632       \exp_after:wN\exp_after:wN\exp_after:wN {
2633         \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2634       }
2635     }
2636
2637     \stex_execute_in_module:x {
2638       \__stex_notation_restore_notation:nnnnn
2639         {#1}{##1}
2640         { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }
2641         { \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl} }
2642         {
2643           \cs_if_exist:cT{stex_op_notation_ #2\c_hash_str ##1 _cs}{
2644             \exp_args:NNo\exp_args:No\exp_not:n{\csname stex_op_notation_ #2\c_hash_str ##1
2645           }
2646         }
2647     }
2648   }
2649 }
2650
2651 \NewDocumentCommand \copynotation {m m} {
```

156

```
2652      \stex_get_symbol:n { #1 }
2653      \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2654      \stex_get_symbol:n { #2 }
2655      \exp_args:Noo
2656      \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2657      \stex_smsmode_do:\ignorespacesandpars
2658    }
2659
```

(*End definition for* \setnotation. *This function is documented on page 18.*)

\symdef

```
2660    \keys_define:nn { stex / symdef } {
2661      name     .str_set_x:N = \l_stex_symdecl_name_str ,
2662      local    .bool_set:N  = \l_stex_symdecl_local_bool ,
2663      args     .str_set_x:N = \l_stex_symdecl_args_str ,
2664      type     .tl_set:N    = \l_stex_symdecl_type_tl ,
2665      def      .tl_set:N    = \l_stex_symdecl_definiens_tl ,
2666      op       .tl_set:N    = \l__stex_notation_op_tl ,
2667  % lang      .str_set_x:N = \l__stex_notation_lang_str ,
2668      variant .str_set_x:N = \l__stex_notation_variant_str ,
2669      prec     .str_set_x:N = \l__stex_notation_prec_str ,
2670      assoc    .choices:nn  =
2671          {bin,binl,binr,pre,conj,pwconj}
2672          {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
2673      unknown .code:n      = \str_set:Nx
2674          \l__stex_notation_variant_str \l_keys_key_str
2675    }
2676
2677    \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2678      \str_clear:N \l_stex_symdecl_name_str
2679      \str_clear:N \l_stex_symdecl_args_str
2680      \str_clear:N \l_stex_symdecl_assoctype_str
2681      \bool_set_false:N \l_stex_symdecl_local_bool
2682      \tl_clear:N \l_stex_symdecl_type_tl
2683      \tl_clear:N \l_stex_symdecl_definiens_tl
2684  % \str_clear:N \l__stex_notation_lang_str
2685      \str_clear:N \l__stex_notation_variant_str
2686      \str_clear:N \l__stex_notation_prec_str
2687      \tl_clear:N \l__stex_notation_op_tl
2688
2689      \keys_set:nn { stex / symdef } { #1 }
2690    }
2691
2692    \NewDocumentCommand \symdef { m O{} } {
2693      \__stex_notation_symdef_args:n { #2 }
2694      \bool_set_true:N \l_stex_symdecl_make_macro_bool
2695      \stex_symdecl_do:n { #1 }
2696      \tl_set:Nn \l_stex_notation_after_do_tl {
2697        \__stex_notation_final:
2698        \stex_smsmode_do:\ignorespacesandpars
2699      }
2700      \str_set:Nx \l_stex_get_symbol_uri_str {
2701        \l_stex_current_module_str ? \l_stex_symdecl_name_str
```

```
2702      }
2703    \exp_args:Nx \stex_notation_do:nnnnn
2704      { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } }
2705      { \prop_item:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } }
2706      { \l__stex_notation_variant_str }
2707      { \l__stex_notation_prec_str}
2708 }
2709 \stex_deactivate_macro:Nn \symdef {module~environments}
```

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

29.3   Variables

```
2710 ⟨@@=stex_variables⟩
2711
2712 \keys_define:nn { stex / vardef } {
2713    name    .str_set_x:N = \l__stex_variables_name_str ,
2714    args    .str_set_x:N = \l__stex_variables_args_str ,
2715    type    .tl_set:N    = \l__stex_variables_type_tl ,
2716    def     .tl_set:N    = \l__stex_variables_def_tl ,
2717    op      .tl_set:N    = \l__stex_variables_op_tl ,
2718    prec    .str_set_x:N = \l__stex_variables_prec_str ,
2719    assoc   .choices:nn  =
2720        {bin,binl,binr,pre,conj,pwconj}
2721        {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
2722    bind    .choices:nn  =
2723        {forall,exists}
2724        {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2725 }
2726
2727 \cs_new_protected:Nn \__stex_variables_args:n {
2728    \str_clear:N \l__stex_variables_name_str
2729    \str_clear:N \l__stex_variables_args_str
2730    \str_clear:N \l__stex_variables_prec_str
2731    \str_clear:N \l__stex_variables_assoctype_str
2732    \str_clear:N \l__stex_variables_bind_str
2733    \tl_clear:N \l__stex_variables_type_tl
2734    \tl_clear:N \l__stex_variables_def_tl
2735    \tl_clear:N \l__stex_variables_op_tl
2736
2737    \keys_set:nn { stex / vardef } { #1 }
2738 }
2739
2740 \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{}} {
2741    \__stex_variables_args:n {#2}
2742    \str_if_empty:NT \l__stex_variables_name_str {
2743        \str_set:Nx \l__stex_variables_name_str { #1 }
2744    }
2745    \prop_clear:N \l_tmpa_prop
2746    \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
2747
2748    \int_zero:N \l_tmpb_int
2749    \bool_set_true:N \l_tmpa_bool
2750    \str_map_inline:Nn \l__stex_variables_args_str {
```

```
2751    \token_case_meaning:NnF ##1 {
2752      0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2753      {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2754      {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2755      {\tl_to_str:n a} {
2756        \bool_set_false:N \l_tmpa_bool
2757        \int_incr:N \l_tmpb_int
2758      }
2759      {\tl_to_str:n B} {
2760        \bool_set_false:N \l_tmpa_bool
2761        \int_incr:N \l_tmpb_int
2762      }
2763    }{
2764      \msg_error:nnxx{stex}{error/wrongargs}{
2765        variable~\l__stex_variables_name_str
2766      }{##1}
2767    }
2768  }
2769  \bool_if:NTF \l_tmpa_bool {
2770    % possibly numeric
2771    \str_if_empty:NTF \l__stex_variables_args_str {
2772      \prop_put:Nnn \l_tmpa_prop { args } {}
2773      \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2774    }{
2775      \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
2776      \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2777      \str_clear:N \l_tmpa_str
2778      \int_step_inline:nn \l_tmpa_int {
2779        \str_put_right:Nn \l_tmpa_str i
2780      }
2781      \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
2782      \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2783    }
2784  } {
2785    \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2786    \prop_put:Nnx \l_tmpa_prop { arity }
2787      { \str_count:N \l__stex_variables_args_str }
2788  }
2789  \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2790  \tl_set:cx { #1 }{ \stex_invoke_variable:n { \l__stex_variables_name_str } }

2792  \prop_set_eq:cN { l_stex_variable_\l__stex_variables_name_str _prop} \l_tmpa_prop

2794  \tl_if_empty:NF \l__stex_variables_op_tl {
2795    \cs_set:cpx {
2796      stex_var_op_notation_ \l__stex_variables_name_str _cs
2797    } { \exp_not:N\comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } } }
2798  }

2800    \tl_set:Nn \l_stex_notation_after_do_tl {
2801      \exp_args:Nne \use:nn {
2802        \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str _cs }
2803          \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } } }
2804      } {{
```

```
2805        \exp_after:wN \exp_after:wN \exp_after:wN
2806        \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2807        { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_symb
2808      }}
2809      \stex_if_do_html:T {
2810        \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
2811          \stex_annotate_invisible:nnn { precedence }
2812            { \l__stex_variables_prec_str }{}
2813          \tl_if_empty:NF \l__stex_variables_type_tl {\stex_annotate_invisible:nnn{type}{}{$\l
2814          \stex_annotate_invisible:nnn{args}{}{ \l__stex_variables_args_str }
2815          \stex_annotate_invisible:nnn{macroname}{#1}{}
2816          \tl_if_empty:NF \l__stex_variables_def_tl {
2817            \stex_annotate_invisible:nnn{definiens}{}
2818              {$\l__stex_variables_def_tl$}
2819          }
2820          \str_if_empty:NF \l__stex_variables_assoctype_str {
2821            \stex_annotate_invisible:nnn{assoctype}{\l__stex_variables_assoctype_str}{}
2822          }
2823          \str_if_empty:NF \l__stex_variables_bind_str {
2824            \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{}
2825          }
2826          \int_zero:N \l_tmpa_int
2827          \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
2828          \tl_clear:N \l_tmpa_tl
2829          \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
2830            \int_incr:N \l_tmpa_int
2831            \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
2832            \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables
2833            \str_if_eq:VnTF \l_tmpb_str a {
2834              \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2835                \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2836                \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2837              } }
2838            }{
2839              \str_if_eq:VnTF \l_tmpb_str B {
2840                \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2841                  \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2842                  \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2843                } }
2844              }{
2845                \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2846                  \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
2847                } }
2848              }
2849            }
2850          }
2851          \stex_annotate_invisible:nnn { notationcomp }{}{
2852            \str_set:Nx \l_stex_current_symbol_str {var://\l__stex_variables_name_str }
2853            $ \exp_args:Nno \use:nn { \use:c {
2854              stex_var_notation_\l__stex_variables_name_str _cs
2855            } } { \l_tmpa_tl } $
2856          }
2857        }
2858      }\ignorespacesandpars
```

160

```
2859    }
2860
2861    \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
2862 }
2863
2864 \cs_new:Nn \_stex_reset:N {
2865    \tl_if_exist:NTF #1 {
2866       \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
2867    }{
2868       \let \exp_not:N #1 \exp_not:N \undefined
2869    }
2870 }
2871
2872 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
2873    \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
2874    \exp_args:Nnx \use:nn {
2875       % TODO
2876       \stex_annotate_invisible:nnn {vardecl}{\clist_use:Nn\l__stex_variables_names,}{
2877          #2
2878       }
2879    }{
2880       \_stex_reset:N \varnot
2881       \_stex_reset:N \vartype
2882       \_stex_reset:N \vardefi
2883    }
2884 }
2885
2886 \NewDocumentCommand \vardef { s } {
2887    \IfBooleanTF#1 {
2888       \__stex_variables_do_complex:nn
2889    }{
2890       \__stex_variables_do_simple:nnn
2891    }
2892 }
2893
2894 \NewDocumentCommand \svar { O{} m }{
2895    \tl_if_empty:nTF {#1}{
2896       \str_set:Nn \l_tmpa_str { #2 }
2897    }{
2898       \str_set:Nn \l_tmpa_str { #1 }
2899    }
2900    \_stex_term_omv:nn {
2901       var://\l_tmpa_str
2902    }{
2903       \exp_args:Nnx \use:nn {
2904          \def\comp{\_varcomp}
2905          \str_set:Nx \l_stex_current_symbol_str { var://\l_tmpa_str }
2906          \comp{ #2 }
2907       }{
2908          \_stex_reset:N \comp
2909          \_stex_reset:N \l_stex_current_symbol_str
2910       }
2911    }
2912 }
```

```latex
2913
2914
2915
2916  \keys_define:nn { stex / varseq } {
2917    name    .str_set_x:N  = \l__stex_variables_name_str ,
2918    args    .int_set:N    = \l__stex_variables_args_int ,
2919    type    .tl_set:N     = \l__stex_variables_type_tl  ,
2920    mid     .tl_set:N     = \l__stex_variables_mid_tl   ,
2921    bind    .choices:nn   =
2922        {forall,exists}
2923        {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2924  }
2925
2926  \cs_new_protected:Nn \__stex_variables_seq_args:n {
2927    \str_clear:N \l__stex_variables_name_str
2928    \int_set:Nn \l__stex_variables_args_int 1
2929    \tl_clear:N \l__stex_variables_type_tl
2930    \str_clear:N \l__stex_variables_bind_str
2931
2932    \keys_set:nn { stex / varseq } { #1 }
2933  }
2934
2935  \NewDocumentCommand \varseq {m O{} m m m}{
2936    \__stex_variables_seq_args:n { #2 }
2937    \str_if_empty:NT \l__stex_variables_name_str {
2938      \str_set:Nx \l__stex_variables_name_str { #1 }
2939    }
2940    \prop_clear:N \l_tmpa_prop
2941    \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
2942
2943    \seq_set_from_clist:Nn \l_tmpa_seq {#3}
2944    \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
2945      \msg_error:nnxx{stex}{error/seqlength}
2946        {\int_use:N \l__stex_variables_args_int}
2947        {\seq_count:N \l_tmpa_seq}
2948    }
2949    \seq_set_from_clist:Nn \l_tmpb_seq {#4}
2950    \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
2951      \msg_error:nnxx{stex}{error/seqlength}
2952        {\int_use:N \l__stex_variables_args_int}
2953        {\seq_count:N \l_tmpb_seq}
2954    }
2955    \prop_put:Nnn \l_tmpa_prop {starts} {#3}
2956    \prop_put:Nnn \l_tmpa_prop {ends} {#4}
2957
2958    \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
2959      \cs_set:Npn  {\int_use:N \l__stex_variables_args_int} { #5 }
2960
2961    \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
2962    \int_step_inline:nn \l__stex_variables_args_int {
2963      \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
2964    }
2965    \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
2966    \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
```

162

```
2967    \tl_if_empty:NF \l__stex_variables_mid_tl {
2968      \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
2969      \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
2970    }
2971    \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
2972    \int_step_inline:nn \l__stex_variables_args_int {
2973      \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
2974    }
2975    \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
2976    \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
2977
2978
2979    \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl
2980
2981    \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}
2982
2983    \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
2984
2985    \int_step_inline:nn \l__stex_variables_args_int {
2986      \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
2987        \_stex_term_math_arg:nnn{i##1}{0}{\exp_not:n{####}##1}
2988      }}
2989    }
2990
2991    \tl_set:Nx \l_tmpa_tl {
2992      \_stex_term_math_oma:nnnn { varseq://\l__stex_variables_name_str}{}{0}{
2993        \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
2994      }
2995    }
2996
2997    \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \stex_symbol_after_invokation_tl} }
2998
2999    \exp_args:Nno \use:nn {
3000    \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
3001      \cs_set:Npn {\int_use:N \l__stex_variables_args_int}}{\l_tmpa_tl}
3002
3003    \stex_debug:nn{sequences}{New~Sequence:~
3004      \expandafter\meaning\csname stex_varseq_\l__stex_variables_name_str _cs\endcsname\\~\\
3005      \prop_to_keyval:N \l_tmpa_prop
3006    }
3007    \stex_if_do_html:T{\stex_annotate_invisible:nnn{varseq}{\l__stex_variables_name_str}{
3008      \tl_if_empty:NF \l__stex_variables_type_tl {
3009        \stex_annotate:nnn {type}{}{$\seqtype\l__stex_variables_type_tl$}
3010      }
3011      \stex_annotate:nnn {args}{\int_use:N \l__stex_variables_args_int}{}
3012      \str_if_empty:NF \l__stex_variables_bind_str {
3013        \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{}
3014      }
3015    }}
3016
3017    \prop_set_eq:cN {stex_varseq_\l__stex_variables_name_str _prop}\l_tmpa_prop
3018    \ignorespacesandpars
3019 }
3020
```

3021 ⟨/package⟩

Chapter 30

STEX -Terms Implementation

```
3022 ⟨*package⟩
3023
3024 %%%%%%%%%%%%    terms.dtx    %%%%%%%%%%%%
3025
3026 ⟨@@=stex_terms⟩
```

Warnings and error messages

```
3027 \msg_new:nnn{stex}{error/nonotation}{
3028   Symbol~#1~invoked,~but~has~no~notation#2!
3029 }
3030 \msg_new:nnn{stex}{error/notationarg}{
3031   Error~in~parsing~notation~#1
3032 }
3033 \msg_new:nnn{stex}{error/noop}{
3034   Symbol~#1~has~no~operator~notation~for~notation~#2
3035 }
3036 \msg_new:nnn{stex}{error/notallowed}{
3037   Symbol~invokation~#1~not~allowed~in~notation~component~of~#2
3038 }
3039 \msg_new:nnn{stex}{error/doubleargument}{
3040   Argument~#1~of~symbol~#2~already~assigned
3041 }
3042 \msg_new:nnn{stex}{error/overarity}{
3043   Argument~#1~invalid~for~symbol~#2~with~arity~#3
3044 }
3045
```

30.1   Symbol Invocations

`\stex_invoke_symbol:n`   Invokes a semantic macro

```
3046
3047
3048 \bool_new:N \l_stex_allow_semantic_bool
3049 \bool_set_true:N \l_stex_allow_semantic_bool
3050
```

```
3051 \cs_new_protected:Nn \stex_invoke_symbol:n {
3052   \bool_if:NTF \l_stex_allow_semantic_bool {
3053     \str_if_eq:eeF {
3054       \prop_item:cn {
3055         l_stex_symdecl_#1_prop
3056       }{ deprecate }
3057     }{}{
3058       \msg_warning:nnxx{stex}{warning/deprecated}{
3059         Symbol~#1
3060       }{
3061         \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
3062       }
3063     }
3064     \if_mode_math:
3065       \exp_after:wN \__stex_terms_invoke_math:n
3066     \else:
3067       \exp_after:wN \__stex_terms_invoke_text:n
3068     \fi: { #1 }
3069   }{
3070     \msg_error:nnxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
3071   }
3072 }
3073
3074 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
3075   \peek_charcode_remove:NTF ! {
3076     \__stex_terms_invoke_op_custom:nn {#1}
3077   }{
3078     \__stex_terms_invoke_custom:nn {#1}
3079   }
3080 }
3081
3082 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
3083   \peek_charcode_remove:NTF ! {
3084     % operator
3085     \peek_charcode_remove:NTF * {
3086       % custom op
3087       \__stex_terms_invoke_op_custom:nn {#1}
3088     }{
3089       % op notation
3090       \peek_charcode:NTF [ {
3091         \__stex_terms_invoke_op_notation:nw {#1}
3092       }{
3093         \__stex_terms_invoke_op_notation:nw {#1}[]
3094       }
3095     }
3096   }{
3097     \peek_charcode_remove:NTF * {
3098       \__stex_terms_invoke_custom:nn {#1}
3099       % custom
3100     }{
3101       % normal
3102       \peek_charcode:NTF [ {
3103         \__stex_terms_invoke_notation:nw {#1}
3104       }{
```

```
3105        \__stex_terms_invoke_notation:nw {#1}[]
3106      }
3107    }
3108  }
3109 }
3110

3111
3112 \cs_new_protected:Nn \__stex_terms_invoke_op_custom:nn {
3113   \exp_args:Nnx \use:nn {
3114     \def\comp{\_comp}
3115     \str_set:Nn \l_stex_current_symbol_str { #1 }
3116     \bool_set_false:N \l_stex_allow_semantic_bool
3117     \_stex_term_oms:nnn {#1}{#1 \c_hash_str CUSTOM-}{
3118       \comp{ #2 }
3119     }
3120   }{
3121     \_stex_reset:N \comp
3122     \_stex_reset:N \l_stex_current_symbol_str
3123     \bool_set_true:N \l_stex_allow_semantic_bool
3124   }
3125 }
3126
3127 \keys_define:nn { stex / terms } {
3128 %  lang     .tl_set_x:N = \l_stex_notation_lang_str ,
3129   variant .tl_set_x:N = \l_stex_notation_variant_str ,
3130   unknown .code:n     = \str_set:Nx
3131       \l_stex_notation_variant_str \l_keys_key_str
3132 }
3133
3134 \cs_new_protected:Nn \__stex_terms_args:n {
3135  % \str_clear:N \l_stex_notation_lang_str
3136   \str_clear:N \l_stex_notation_variant_str
3137
3138   \keys_set:nn { stex / terms } { #1 }
3139 }
3140
3141 \cs_new_protected:Nn \stex_find_notation:nn {
3142   \__stex_terms_args:n { #2 }
3143   \seq_if_empty:cTF {
3144     l_stex_symdecl_ #1 _notations
3145   } {
3146     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
3147   } {
3148     \str_if_empty:NTF \l_stex_notation_variant_str {
3149       \seq_get_left:cN {l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
3150     }{
3151       \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
3152         \l_stex_notation_variant_str
3153       }{
3154       % \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3155       }{
3156         \msg_error:nnxx{stex}{error/nonotation}{#1}{
3157           ~\l_stex_notation_variant_str
3158         }
```

```
3159              }
3160            }
3161         }
3162    }

3164    \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3165      \exp_args:Nnx \use:nn {
3166        \def\comp{\_comp}
3167        \str_set:Nn \l_stex_current_symbol_str { #1 }
3168        \stex_find_notation:nn { #1 }{ #2 }
3169        \bool_set_false:N \l_stex_allow_semantic_bool
3170        \cs_if_exist:cTF {
3171          stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3172        }{
3173          \_stex_term_oms:nnn { #1 }{
3174            #1 \c_hash_str \l_stex_notation_variant_str
3175          }{
3176            \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3177          }
3178        }{
3179          \int_compare:nNnTF {\prop_item:cn {l_stex_symdecl_#1_prop}{arity}} = 0{
3180            \cs_if_exist:cTF {
3181              stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3182            }{
3183              \tl_set:Nx \stex_symbol_after_invokation_tl {
3184                \_stex_reset:N \comp
3185                \_stex_reset:N \stex_symbol_after_invokation_tl
3186                \_stex_reset:N \l_stex_current_symbol_str
3187                \bool_set_true:N \l_stex_allow_semantic_bool
3188              }
3189              \def\comp{\_comp}
3190              \str_set:Nn \l_stex_current_symbol_str { #1 }
3191              \bool_set_false:N \l_stex_allow_semantic_bool
3192              \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3193            }{
3194              \msg_error:nnxx{stex}{error/nonotation}{#1}{
3195                ~\l_stex_notation_variant_str
3196              }
3197            }
3198          }{
3199            \msg_error:nnxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3200          }
3201        }
3202      }{
3203        \_stex_reset:N \comp
3204        \_stex_reset:N \l_stex_current_symbol_str
3205        \bool_set_true:N \l_stex_allow_semantic_bool
3206      }
3207    }

3209    \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {
3210      \stex_find_notation:nn { #1 }{ #2 }
3211      \cs_if_exist:cTF {
3212        stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
```

```
3213     }{
3214        \tl_set:Nx \stex_symbol_after_invokation_tl {
3215          \_stex_reset:N \comp
3216          \_stex_reset:N \stex_symbol_after_invokation_tl
3217          \_stex_reset:N \l_stex_current_symbol_str
3218          \bool_set_true:N \l_stex_allow_semantic_bool
3219        }
3220        \def\comp{\_comp}
3221        \str_set:Nn \l_stex_current_symbol_str { #1 }
3222        \bool_set_false:N \l_stex_allow_semantic_bool
3223        \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3224     }{
3225        \msg_error:nnxx{stex}{error/nonotation}{#1}{
3226          ~\l_stex_notation_variant_str
3227        }
3228     }
3229  }
3230
3231  \prop_new:N \l__stex_terms_custom_args_prop
3232
3233  \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3234     \exp_args:Nnx \use:nn {
3235        \bool_set_false:N \l_stex_allow_semantic_bool
3236        \def\comp{\_comp}
3237        \str_set:Nn \l_stex_current_symbol_str { # 1 }
3238        \prop_clear:N \l__stex_terms_custom_args_prop
3239        \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3240        \prop_get:cnN {
3241          l_stex_symdecl_#1 _prop
3242        }{ args } \l_tmpa_str
3243        \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3244        \tl_set:Nn \arg { \__stex_terms_arg: }
3245        \str_if_empty:NTF \l_tmpa_str {
3246          \_stex_term_oms:nnn {#1}{#1\c_hash_str CUSTOM-}{#2}
3247        }{
3248          \str_if_in:NnTF \l_tmpa_str b {
3249            \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3250          }{
3251            \str_if_in:NnTF \l_tmpa_str B {
3252              \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3253            }{
3254              \_stex_term_oma:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3255            }
3256          }
3257        }
3258        % TODO check that all arguments exist
3259     }{
3260        \_stex_reset:N \l_stex_current_symbol_str
3261        \_stex_reset:N \arg
3262        \_stex_reset:N \comp
3263        \_stex_reset:N \l__stex_terms_custom_args_prop
3264        \bool_set_true:N \l_stex_allow_semantic_bool
3265     }
3266  }
```

```
3267
3268  \NewDocumentCommand \__stex_terms_arg: { s O{} m }{
3269    \tl_if_empty:nTF {#2}{
3270      \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
3271      \bool_set_true:N \l_tmpa_bool
3272      \bool_do_while:Nn \l_tmpa_bool {
3273        \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int
3274          \int_incr:N \l_tmpa_int
3275        }{
3276          \bool_set_false:N \l_tmpa_bool
3277        }
3278      }
3279    }{
3280      \int_set:Nn \l_tmpa_int { #2 }
3281    }
3282    \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
3283    \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
3284      \msg_error:nnxxx{stex}{error/overarity}
3285        {\int_use:N \l_tmpa_int}
3286        {\l_stex_current_symbol_str}
3287        {\str_count:N \l_tmpa_str}
3288    }
3289    \str_set:Nx \l_tmpa_str {\str_item:Nn \l_tmpa_str \l_tmpa_int}
3290    \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
3291      \bool_lazy_any:nF {
3292        {\str_if_eq_p:Vn \l_tmpa_str {a}}
3293        {\str_if_eq_p:Vn \l_tmpa_str {B}}
3294      }{
3295        \msg_error:nnxx{stex}{error/doubleargument}
3296          {\int_use:N \l_tmpa_int}
3297          {\l_stex_current_symbol_str}
3298      }
3299    }
3300    \exp_args:NNx \prop_put:Nnn \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {#3}
3301    \bool_set_true:N \l_stex_allow_semantic_bool
3302    \IfBooleanTF#1{
3303      \stex_annotate_invisible:n { %TODO
3304        \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{#3}
3305      }
3306    }{ %TODO
3307      \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{#3}
3308    }
3309    \bool_set_false:N \l_stex_allow_semantic_bool
3310  }
3311
3312
3313  \cs_new_protected:Nn \_stex_term_arg:nn {
3314    \bool_set_true:N \l_stex_allow_semantic_bool
3315    \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3316    \bool_set_false:N \l_stex_allow_semantic_bool
3317  }
3318
3319  \cs_new_protected:Nn \_stex_term_math_arg:nnn {
3320    \exp_args:Nnx \use:nn
```

```
3321        { \int_set:Nn \l__stex_terms_downprec { #2 }
3322            \_stex_term_arg:nn { #1 }{ #3 }
3323        }
3324        { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3325  }
```

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

`\_stex_term_math_assoc_arg:nnnn`

```
3326  \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
3327    \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3328    \tl_set:Nn \l_tmpb_tl {\_stex_term_math_arg:nnn{#1}{#2}}
3329    \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 }}{
3330      \expandafter\if\expandafter\relax\noexpand#3
3331        \expandafter\__stex_terms_math_assoc_arg_maybe_sequence:N\expandafter#3
3332      \else\expandafter\__stex_terms_math_assoc_arg_simple:nn
3333      \expandafter{\expandafter}\expandafter#3\fi
3334    }{
3335      \__stex_terms_math_assoc_arg_simple:nn{#1}{#3}
3336    }
3337  }
3338
3339  \cs_new_protected:Nn \__stex_terms_math_assoc_arg_maybe_sequence:N {
3340    \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
3341    \str_if_empty:NTF \l_tmpa_str {
3342      \exp_args:Nx \cs_if_eq:NNTF {
3343        \tl_head:N #1
3344      } \stex_invoke_sequence:n {
3345        \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
3346        \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
3347        \tl_set:Nx \l_tmpa_tl {\prop_item:cn {stex_varseq_\l_tmpa_str _prop}{notation}}
3348        \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
3349        \tl_set:Nx \l_tmpa_tl {{\exp_not:N \exp_not:n{
3350          \exp_not:n{\exp_args:Nnx \use:nn} {
3351            \exp_not:n {
3352              \def\comp{\_varcomp}
3353              \str_set:Nn \l_stex_current_symbol_str
3354            } {varseq://\l_tmpa_str}
3355            \exp_not:n{ ##1 }
3356          }{
3357            \exp_not:n {
3358              \_stex_reset:N \comp
3359              \_stex_reset:N \l_stex_current_symbol_str
3360            }
3361          }
3362        }}}
3363        \exp_args:Nno \use:nn {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
3364        \seq_reverse:N \l_tmpa_seq
3365        \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
3366        \seq_map_inline:Nn \l_tmpa_seq {
3367          \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3368            \exp_args:Nno
3369            \l_tmpa_cs { ##1 } \l_tmpa_tl
3370          }
```

```
3371        }
3372        \tl_set:Nx \l_tmpa_tl {
3373          \_stex_term_omv:nn {varseq://\l_tmpa_str}{
3374            \exp_args:No \exp_not:n \l_tmpa_tl
3375          }
3376        }
3377        \exp_args:No\l_tmpb_tl\l_tmpa_tl
3378      }{
3379        \__stex_terms_math_assoc_arg_simple:nn{} { #1 }
3380      }
3381    } {
3382      \__stex_terms_math_assoc_arg_simple:nn{} { #1 }
3383    }
3384
3385 }
3386
3387 \cs_new_protected:Nn \__stex_terms_math_assoc_arg_simple:nn {
3388    \clist_set:Nn \l_tmpa_clist{ #2 }
3389    \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3390      \tl_set:Nn \l_tmpa_tl { #2 }
3391    }{
3392      \clist_reverse:N \l_tmpa_clist
3393      \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3394      \tl_set:Nx \l_tmpa_tl { \_stex_term_arg:nn{A#1}{
3395        \exp_args:No \exp_not:n \l_tmpa_tl
3396      }}
3397      \clist_map_inline:Nn \l_tmpa_clist {
3398        \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3399          \exp_args:Nno
3400          \l_tmpa_cs { \_stex_term_arg:nn{A#1}{##1} } \l_tmpa_tl
3401        }
3402      }
3403    }
3404    \exp_args:No\l_tmpb_tl\l_tmpa_tl
3405 }
```

(*End definition for* _stex_term_math_assoc_arg:nnnn. *This function is documented on page 62.*)

30.2 Terms

Precedences:

```
3406 \tl_const:Nx \infprec {\int_use:N \c_max_int}
3407 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
3408 \int_new:N \l__stex_terms_downprec
3409 \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 63.*)

Bracketing:

```
3410 \tl_set:Nn \l__stex_terms_left_bracket_str (
3411 \tl_set:Nn \l__stex_terms_right_bracket_str )
```

__stex_terms_maybe_brackets:nn   Compares precedences and insert brackets accordingly

```
3412 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
3413   \bool_if:NTF \l__stex_terms_brackets_done_bool {
3414     \bool_set_false:N \l__stex_terms_brackets_done_bool
3415     #2
3416   } {
3417     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
3418       \bool_if:NTF \l_stex_inparray_bool { #2 }{
3419         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
3420         \dobrackets { #2 }
3421       }
3422     }{ #2 }
3423   }
3424 }
```

\dobrackets

```
3425 \bool_new:N \l__stex_terms_brackets_done_bool
3426 %\RequirePackage{scalerel}
3427 \cs_new_protected:Npn \dobrackets #1 {
3428   %\ThisStyle{\if D\m@switch
3429   %   \exp_args:Nnx \use:nn
3430   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
3431   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
3432   %  \else
3433     \exp_args:Nnx \use:nn
3434     {
3435       \bool_set_true:N \l__stex_terms_brackets_done_bool
3436       \int_set:Nn \l__stex_terms_downprec \infprec
3437       \l__stex_terms_left_bracket_str
3438       #1
3439     }
3440     {
3441       \bool_set_false:N \l__stex_terms_brackets_done_bool
3442       \l__stex_terms_right_bracket_str
3443       \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3444     }
3445   %\fi}
3446 }
```

\withbrackets

```
3447 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
3448   \exp_args:Nnx \use:nn
3449   {
3450     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
3451     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
3452     #3
3453   }
3454   {
```

```
3455    \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
3456        {\l__stex_terms_left_bracket_str}
3457    \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
3458        {\l__stex_terms_right_bracket_str}
3459    }
3460 }
```

(*End definition for* \withbrackets. *This function is documented on page 63.*)

\STEXinvisible

```
3461 \cs_new_protected:Npn \STEXinvisible #1 {
3462    \stex_annotate_invisible:n { #1 }
3463 }
```

(*End definition for* \STEXinvisible. *This function is documented on page 63.*)

OMDoc terms:

_stex_term_math_oms:nnnn

```
3464 \cs_new_protected:Nn \_stex_term_oms:nnn {
3465    \stex_annotate:nnn{ OMID }{ #2 }{
3466        #3
3467    }
3468 }
3469
3470 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
3471    \__stex_terms_maybe_brackets:nn { #3 }{
3472        \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3473    }
3474 }
```

(*End definition for* _stex_term_math_oms:nnnn. *This function is documented on page 62.*)

_stex_term_math_omv:nn

```
3475 \cs_new_protected:Nn \_stex_term_omv:nn {
3476    \stex_annotate:nnn{ OMV }{ #1 }{
3477        #2
3478    }
3479 }
```

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

_stex_term_math_oma:nnnn

```
3480 \cs_new_protected:Nn \_stex_term_oma:nnn {
3481    \stex_annotate:nnn{ OMA }{ #2 }{
3482        #3
3483    }
3484 }
3485
3486 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
3487    \__stex_terms_maybe_brackets:nn { #3 }{
3488        \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3489    }
3490 }
```

(*End definition for* _stex_term_math_oma:nnnn. *This function is documented on page 62.*)

```
3491 \cs_new_protected:Nn \_stex_term_ombind:nnn {
3492   \stex_annotate:nnn{ OMBIND }{ #2 }{
3493     #3
3494   }
3495 }
3496
3497 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
3498   \__stex_terms_maybe_brackets:nn { #3 }{
3499     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3500   }
3501 }
```

(End definition for _stex_term_math_omb:nnnn*. This function is documented on page 62.)*

```
3502 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3503
3504 \keys_define:nn { stex / symname } {
3505   pre      .tl_set_x:N    = \l__stex_terms_pre_tl ,
3506   post     .tl_set_x:N    = \l__stex_terms_post_tl ,
3507   root     .tl_set_x:N    = \l__stex_terms_root_tl
3508 }
3509
3510 \cs_new_protected:Nn \stex_symname_args:n {
3511   \tl_clear:N \l__stex_terms_post_tl
3512   \tl_clear:N \l__stex_terms_pre_tl
3513   \tl_clear:N \l__stex_terms_root_str
3514   \keys_set:nn { stex / symname } { #1 }
3515 }
3516
3517 \NewDocumentCommand \symref { m m }{
3518   \let\compemph_uri_prev:\compemph@uri
3519   \let\compemph@uri\symrefemph@uri
3520   \STEXsymbol{#1}!{ #2 }
3521   \let\compemph@uri\compemph_uri_prev:
3522 }
3523
3524 \NewDocumentCommand \synonym { O{} m m}{
3525   \stex_symname_args:n { #1 }
3526   \let\compemph_uri_prev:\compemph@uri
3527   \let\compemph@uri\symrefemph@uri
3528   % TODO
3529   \STEXsymbol{#2}!{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
3530   \let\compemph@uri\compemph_uri_prev:
3531 }
3532
3533 \NewDocumentCommand \symname { O{} m }{
3534   \stex_symname_args:n { #1 }
3535   \stex_get_symbol:n { #2 }
3536   \str_set:Nx \l_tmpa_str {
3537     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3538   }
3539   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
```

175

```
3540
3541    \let\compemph_uri_prev:\compemph@uri
3542    \let\compemph@uri\symrefemph@uri
3543    \exp_args:NNx \use:nn
3544    \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!{
3545      \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
3546    } }
3547    \let\compemph@uri\compemph_uri_prev:
3548  }
3549
3550  \NewDocumentCommand \Symname { O{} m }{
3551    \stex_symname_args:n { #1 }
3552    \stex_get_symbol:n { #2 }
3553    \str_set:Nx \l_tmpa_str {
3554      \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3555    }
3556    \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3557    \let\compemph_uri_prev:\compemph@uri
3558    \let\compemph@uri\symrefemph@uri
3559    \exp_args:NNx \use:nn
3560    \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!{
3561      \exp_after:wN \stex_capitalize:n \l_tmpa_str
3562        \l__stex_terms_post_tl
3563    } }
3564    \let\compemph@uri\compemph_uri_prev:
3565  }
```

(End definition for `\symref` *and* `\symname`*. These functions are documented on page 62.)*

30.3 Notation Components

```
3566  ⟨@@=stex_notationcomps⟩
```

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

```
3567  \cs_new_protected:Npn \_comp #1 {
3568    \str_if_empty:NF \l_stex_current_symbol_str {
3569      \stex_html_backend:TF {
3570        \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
3571      }{
3572        \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3573      }
3574    }
3575  }
3576
3577  \cs_new_protected:Npn \_varcomp #1 {
3578    \str_if_empty:NF \l_stex_current_symbol_str {
3579      \stex_html_backend:TF {
3580        \stex_annotate:nnn { varcomp }{ \l_stex_current_symbol_str }{ #1 }
3581      }{
3582        \exp_args:Nnx \varemph@uri { #1 } { \l_stex_current_symbol_str }
3583      }
3584    }
3585  }
3586
```

```
3587  \def\comp{\_comp}

3588
3589  \cs_new_protected:Npn \compemph@uri #1 #2 {
3590      \compemph{ #1 }
3591  }

3592

3593
3594  \cs_new_protected:Npn \compemph #1 {
3595      #1
3596  }

3597
3598  \cs_new_protected:Npn \defemph@uri #1 #2 {
3599      \defemph{#1}
3600  }

3601
3602  \cs_new_protected:Npn \defemph #1 {
3603      \textbf{#1}
3604  }

3605
3606  \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3607      \symrefemph{#1}
3608  }

3609
3610  \cs_new_protected:Npn \symrefemph #1 {
3611      \textbf{#1}
3612  }

3613
3614  \cs_new_protected:Npn \varemph@uri #1 #2 {
3615      \varemph{#1}
3616  }

3617
3618  \cs_new_protected:Npn \varemph #1 {
3619      #1
3620  }
```

(*End definition for* `\comp` *and others. These functions are documented on page 63.*)

`\ellipses`

```
3621  \NewDocumentCommand \ellipses {} { \ldots }
```

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

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

```
3622  \bool_new:N \l_stex_inparray_bool
3623  \bool_set_false:N \l_stex_inparray_bool
3624  \NewDocumentCommand \parray { m m } {
3625      \begingroup
3626      \bool_set_true:N \l_stex_inparray_bool
3627      \begin{array}{#1}
3628          #2
3629      \end{array}
3630      \endgroup
3631  }

3632
3633  \NewDocumentCommand \prmatrix { m } {
```

```
3634      \begingroup
3635      \bool_set_true:N \l_stex_inparray_bool
3636      \begin{matrix}
3637        #1
3638      \end{matrix}
3639      \endgroup
3640  }
3641
3642  \def \maybephline {
3643      \bool_if:NT \l_stex_inparray_bool {\hline}
3644  }
3645
3646  \def \parrayline #1 #2 {
3647      #1 #2 \bool_if:NT \l_stex_inparray_bool {\\}
3648  }
3649
3650  \def \pmrow #1 { \parrayline{}{ #1 } }
3651
3652  \def \parraylineh #1 #2 {
3653      #1 #2 \bool_if:NT \l_stex_inparray_bool {\\\hline}
3654  }
3655
3656  \def \parraycell #1 {
3657      #1 \bool_if:NT \l_stex_inparray_bool {&}
3658  }
```

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

30.4 Variables

```
3659  ⟨@@=stex_variables⟩
```

\stex_invoke_variable:n   Invokes a variable

```
3660  \cs_new_protected:Nn \stex_invoke_variable:n {
3661      \if_mode_math:
3662          \exp_after:wN \__stex_variables_invoke_math:n
3663      \else:
3664          \exp_after:wN \__stex_variables_invoke_text:n
3665      \fi: {#1}
3666  }
3667
3668  \cs_new_protected:Nn \__stex_variables_invoke_text:n {
3669      %TODO
3670  }
3671
3672
3673  \cs_new_protected:Nn \__stex_variables_invoke_math:n {
3674      \peek_charcode_remove:NTF ! {
3675          \peek_charcode_remove:NTF ! {
3676              \peek_charcode:NTF [ {
3677                  \__stex_variables_invoke_op_custom:nw
3678              }{
3679                  % TODO throw error
3680              }
```

```
3681       }{
3682         \__stex_variables_invoke_op:n { #1 }
3683       }
3684     }{
3685       \peek_charcode_remove:NTF * {
3686         \__stex_variables_invoke_text:n { #1 }
3687       }{
3688         \__stex_variables_invoke_math_ii:n { #1 }
3689       }
3690     }
3691   }
3692
3693   \cs_new_protected:Nn \__stex_variables_invoke_op:n {
3694     \cs_if_exist:cTF {
3695       stex_var_op_notation_ #1 _cs
3696     }{
3697       \exp_args:Nnx \use:nn {
3698         \def\comp{\_varcomp}
3699         \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3700         \_stex_term_omv:nn { var://#1 }{
3701           \use:c{stex_var_op_notation_ #1 _cs }
3702         }
3703       }{
3704         \_stex_reset:N \comp
3705         \_stex_reset:N \l_stex_current_symbol_str
3706       }
3707     }{
3708       \int_compare:nNnTF {\prop_item:cn {l_stex_variable_#1_prop}{arity}}} = 0{
3709         \__stex_variables_invoke_math_ii:n {#1}
3710       }{
3711         \msg_error:nnxx{stex}{error/noop}{variable~#1}{}
3712       }
3713     }
3714   }
3715
3716   \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n  #1 {
3717     \cs_if_exist:cTF {
3718       stex_var_notation_#1_cs
3719     }{
3720       \tl_set:Nx \stex_symbol_after_invokation_tl {
3721         \_stex_reset:N \comp
3722         \_stex_reset:N \stex_symbol_after_invokation_tl
3723         \_stex_reset:N \l_stex_current_symbol_str
3724         \bool_set_true:N \l_stex_allow_semantic_bool
3725       }
3726       \def\comp{\_varcomp}
3727       \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3728       \bool_set_false:N \l_stex_allow_semantic_bool
3729       \use:c{stex_var_notation_#1_cs}
3730     }{
3731       \msg_error:nnxx{stex}{error/nonotation}{variable~#1}{s}
3732     }
3733   }
```

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

179

30.5 Sequences

```
3734 ⟨@@=stex_sequences⟩
3735
3736 \cs_new_protected:Nn \stex_invoke_sequence:n {
3737   \peek_charcode_remove:NTF ! {
3738     \_stex_term_omv:nn {varseq://#1}{
3739       \exp_args:Nnx \use:nn {
3740         \def\comp{\_varcomp}
3741         \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3742         \prop_item:cn{stex_varseq_#1_prop}{notation}
3743       }{
3744         \_stex_reset:N \comp
3745         \_stex_reset:N \l_stex_current_symbol_str
3746       }
3747     }
3748   }{
3749     \bool_set_false:N \l_stex_allow_semantic_bool
3750     \def\comp{\_varcomp}
3751     \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3752     \tl_set:Nx \stex_symbol_after_invokation_tl {
3753       \_stex_reset:N \comp
3754       \_stex_reset:N \stex_symbol_after_invokation_tl
3755       \_stex_reset:N \l_stex_current_symbol_str
3756       \bool_set_true:N \l_stex_allow_semantic_bool
3757     }
3758     \use:c { stex_varseq_#1_cs }
3759   }
3760 }
3761 ⟨/package⟩
```

Chapter 31

sTEX -Structural Features Implementation

```
3762 ⟨∗package⟩
3763
3764 %%%%%%%%%%%%    features.dtx    %%%%%%%%%%%%
3765
```

Warnings and error messages
```
3766 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3767   Symbol~#1~can~not~be~assigned~in~copymodule~#2
3768 }
3769 \msg_new:nnn{stex}{error/interpretmodule/nodefiniens}{
3770   Symbol~#1~not~assigned~in~interpretmodule~#2
3771 }
3772
3773 \msg_new:nnn{stex}{error/unknownstructure}{
3774   No~structure~#1~found!
3775 }
3776
3777 \msg_new:nnn{stex}{error/unknownfield}{
3778   No~field~#1~in~instance~#2~found!\\#3
3779 }
3780
3781 \msg_new:nnn{stex}{error/keyval}{
3782   Invalid~key=value~pair:#1
3783 }
3784 \msg_new:nnn{stex}{error/instantiate/missing}{
3785   Assignments~missing~in~instantiate:~#1
3786 }
3787 \msg_new:nnn{stex}{error/incompatible}{
3788   Incompatible~signature:~#1~(#2)~and~#3~(#4)
3789 }
3790
```

31.1 Imports with modification

```
3791  ⟨@@=stex_copymodule⟩
3792  \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
3793    \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3794      \tl_set:Nn \l_tmpa_tl { #1 }
3795      \__stex_copymodule_get_symbol_from_cs:
3796    }{
3797      % argument is a string
3798      % is it a command name?
3799      \cs_if_exist:cTF { #1 }{
3800        \cs_set_eq:Nc \l_tmpa_tl { #1 }
3801        \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3802        \str_if_empty:NTF \l_tmpa_str {
3803          \exp_args:Nx \cs_if_eq:NNTF {
3804            \tl_head:N \l_tmpa_tl
3805          } \stex_invoke_symbol:n {
3806            \__stex_copymodule_get_symbol_from_cs:n{ #2 }
3807          }{
3808            \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3809          }
3810        } {
3811          \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3812        }
3813      }{
3814        % argument is not a command name
3815        \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3816        % \l_stex_all_symbols_seq
3817      }
3818    }
3819  }
3820
3821  \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
3822    \str_set:Nn \l_tmpa_str { #1 }
3823    \bool_set_false:N \l_tmpa_bool
3824    \bool_if:NF \l_tmpa_bool {
3825      \tl_set:Nn \l_tmpa_tl {
3826        \msg_error:nnn{stex}{error/unknownsymbol}{#1}
3827      }
3828    \str_set:Nn \l_tmpa_str { #1 }
3829    \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3830    \seq_map_inline:Nn #2 {
3831      \str_set:Nn \l_tmpb_str { ##1 }
3832      \str_if_eq:eeT { \l_tmpa_str } {
3833        \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3834      } {
3835        \seq_map_break:n {
3836          \tl_set:Nn \l_tmpa_tl {
3837            \str_set:Nn \l_stex_get_symbol_uri_str {
3838              ##1
3839            }
3840          }
3841        }
3842      }
```

```
3843        }
3844      \l_tmpa_tl
3845    }
3846  }
3847
3848  \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
3849    \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3850      { \tl_tail:N \l_tmpa_tl }
3851    \tl_if_single:NTF \l_tmpa_tl {
3852      \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3853        \exp_after:wN \str_set:Nn \exp_after:wN
3854          \l_stex_get_symbol_uri_str \l_tmpa_tl
3855        \__stex_copymodule_get_symbol_check:n { #1 }
3856      }{
3857        % TODO
3858        % tail is not a single group
3859      }
3860    }{
3861      % TODO
3862      % tail is not a single group
3863    }
3864  }
3865
3866  \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
3867    \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
3868      \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3869        :~\seq_use:Nn #1 {,~}
3870      }
3871    }
3872  }
3873
3874  \cs_new_protected:Nn \stex_copymodule_start:nnnn {
3875    % import module
3876    \stex_import_module_uri:nn { #1 } { #2 }
3877    \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3878    \stex_import_require_module:nnnn
3879      { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3880      { \l_stex_import_path_str } { \l_stex_import_name_str }
3881
3882    \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3883    \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
3884
3885    % fields
3886    \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
3887    \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
3888      \seq_map_inline:cn {c_stex_module_##1_constants}{
3889        \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
3890          ##1 ? ####1
3891        }
3892      }
3893    }
3894
3895    % setup prop
3896    \seq_clear:N \l_tmpa_seq
```

```
3897    \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3898      name     = \l_stex_current_copymodule_name_str ,
3899      module   = \l_stex_current_module_str ,
3900      from     = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3901      includes = \l_tmpa_seq %,
3902    %  fields    = \l_tmpa_seq
3903    }
3904    \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3905      as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3906      \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_se
3907    \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,
3908
3909    \stex_if_do_html:T {
3910      \begin{stex_annotate_env} {#4} {
3911        \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3912      }
3913      \stex_annotate_invisible:nnn{domain}{\l_stex_import_ns_str ?\l_stex_import_name_str}{}
3914    }
3915 }
3916
3917 \cs_new_protected:Nn \stex_copymodule_end:n {
3918    % apply to every field
3919    \def \l_tmpa_cs ##1 ##2 {#1}
3920
3921    \tl_clear:N \__stex_copymodule_module_tl
3922    \tl_clear:N \__stex_copymodule_exec_tl
3923
3924    %\prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3925    \seq_clear:N \__stex_copymodule_fields_seq
3926
3927    \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
3928      \seq_map_inline:cn {c_stex_module_##1_constants}{
3929
3930        \tl_clear:N \__stex_copymodule_curr_symbol_tl % <- wrap in current symbol html
3931        \l_tmpa_cs{##1}{####1}
3932
3933        \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
3934          \str_set_eq:Nc \__stex_copymodule_curr_name_str {l__stex_copymodule_copymodule_##1?#
3935          \stex_if_do_html:T {
3936            \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
3937              \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?###
3938            }
3939          }
3940        }{
3941          \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str /
3942        }
3943
3944        \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
3945        \prop_put:Nnx \l_tmpa_prop { name } \__stex_copymodule_curr_name_str
3946        \prop_put:Nnx \l_tmpa_prop { module } \l_stex_current_module_str
3947
3948        \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
3949          \stex_if_do_html:T {
3950            \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
```

```
3951            $\stex_annotate_invisible:nnn{definiens}{}{\exp_after:wN \exp_not:N\csname l__st
3952          }
3953        }
3954        \prop_put:Nnn \l_tmpa_prop { defined } { true }
3955      }
3956
3957      \stex_add_constant_to_current_module:n \__stex_copymodule_curr_name_str
3958      \tl_put_right:Nx \__stex_copymodule_module_tl {
3959        \seq_clear:c {l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_n
3960        \prop_set_from_keyval:cn {
3961          l_stex_symdecl_\l_stex_current_module_str ? \__stex_copymodule_curr_name_str _prop
3962        }{
3963          \prop_to_keyval:N \l_tmpa_prop
3964        }
3965      }
3966
3967      \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
3968        \stex_if_do_html:T {
3969          \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
3970            \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1
3971          }
3972        }
3973        \tl_put_right:Nx \__stex_copymodule_module_tl {
3974          \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
3975            \stex_invoke_symbol:n {
3976              \l_stex_current_module_str ? \__stex_copymodule_curr_name_str
3977            }
3978          }
3979        }
3980      }
3981
3982      \seq_put_right:Nx \__stex_copymodule_fields_seq {\l_stex_current_module_str ? \__stex_
3983
3984      \tl_put_right:Nx \__stex_copymodule_exec_tl {
3985        \stex_copy_notations:nn {\l_stex_current_module_str ? \__stex_copymodule_curr_name_s
3986      }
3987
3988      \tl_put_right:Nx \__stex_copymodule_exec_tl {
3989        \stex_if_do_html:TF{
3990          \stex_annotate_invisible:nnn{assignment} {##1?####1} { \exp_after:wN \exp_not:n \e
3991        }{
3992          \exp_after:wN \exp_not:n \exp_after:wN {\__stex_copymodule_curr_symbol_tl}
3993        }
3994      }
3995    }
3996  }
3997
3998
3999  \prop_put:Nno \l_stex_current_copymodule_prop {fields} \__stex_copymodule_fields_seq
4000  \tl_put_left:Nx \__stex_copymodule_module_tl {
4001    \prop_set_from_keyval:cn {
4002      l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _pro
4003    }{
4004      \prop_to_keyval:N \l_stex_current_copymodule_prop
```

```
4005        }
4006    }
4007
4008    \seq_gput_right:cx{c_stex_module_\l_stex_current_module_str _copymodules}{
4009      \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4010    }
4011
4012    \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4013    \stex_debug:nn{copymodule}{result:\meaning \__stex_copymodule_module_tl}
4014    \stex_debug:nn{copymodule}{output:\meaning \__stex_copymodule_exec_tl}
4015
4016    \__stex_copymodule_exec_tl
4017    \stex_if_do_html:T {
4018      \end{stex_annotate_env}
4019    }
4020 }
4021
4022 \NewDocumentEnvironment {copymodule} { O{} m m}{
4023    \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ copymodule }
4024    \stex_deactivate_macro:Nn \symdecl {module~environments}
4025    \stex_deactivate_macro:Nn \symdef {module~environments}
4026    \stex_deactivate_macro:Nn \notation {module~environments}
4027    \stex_reactivate_macro:N \assign
4028    \stex_reactivate_macro:N \renamedecl
4029    \stex_reactivate_macro:N \donotcopy
4030    \stex_smsmode_do:
4031 }{
4032    \stex_copymodule_end:n {}
4033 }
4034
4035 \NewDocumentEnvironment {interpretmodule} { O{} m m}{
4036    \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ interpretmodule }
4037    \stex_deactivate_macro:Nn \symdecl {module~environments}
4038    \stex_deactivate_macro:Nn \symdef {module~environments}
4039    \stex_deactivate_macro:Nn \notation {module~environments}
4040    \stex_reactivate_macro:N \assign
4041    \stex_reactivate_macro:N \renamedecl
4042    \stex_reactivate_macro:N \donotcopy
4043    \stex_smsmode_do:
4044 }{
4045    \stex_copymodule_end:n {
4046      \tl_if_exist:cF {
4047        l__stex_copymodule_copymodule_##1?##2_def_tl
4048      }{
4049        \str_if_eq:eeF {
4050          \prop_item:cn{
4051            l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
4052        }{ true }{
4053          \msg_error:nnxx{stex}{error/interpretmodule/nodefiniens}{
4054            ##1?##2
4055          }{\l_stex_current_copymodule_name_str}
4056        }
4057      }
4058    }
```

186

```
4059 }
4060
4061 \iffalse \begin{stex_annotate_env} \fi
4062 \NewDocumentEnvironment {realization} { O{} m}{
4063   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #2 }{ realize }
4064   \stex_deactivate_macro:Nn \symdecl {module~environments}
4065   \stex_deactivate_macro:Nn \symdef {module~environments}
4066   \stex_deactivate_macro:Nn \notation {module~environments}
4067   \stex_reactivate_macro:N \donotcopy
4068   \stex_reactivate_macro:N \assign
4069   \stex_smsmode_do:
4070 }{
4071   \stex_import_module_uri:nn { #1 } { #2 }
4072   \tl_clear:N \__stex_copymodule_exec_tl
4073   \tl_set:Nx \__stex_copymodule_module_tl {
4074     \stex_import_require_module:nnnn
4075       { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4076       { \l_stex_import_path_str } { \l_stex_import_name_str }
4077   }
4078
4079   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4080     \seq_map_inline:cn {c_stex_module_##1_constants}{
4081       \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str / #
4082       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
4083         \stex_if_do_html:T {
4084           \tl_put_right:Nx \__stex_copymodule_exec_tl {
4085             \stex_annotate_invisible:nnn{assignment} {##1?####1} {
4086               $\stex_annotate_invisible:nnn{definiens}{}{\exp_after:wN \exp_not:N\csname l__
4087             }
4088           }
4089         }
4090         \tl_put_right:Nx \__stex_copymodule_module_tl {
4091           \prop_put:cnn {l_stex_symdecl_##1?####1_prop}{ defined }{ true }
4092         }
4093       }
4094   }}
4095
4096   \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4097
4098   \__stex_copymodule_exec_tl
4099   \stex_if_do_html:T {\end{stex_annotate_env}}
4100 }
4101
4102 \NewDocumentCommand \donotcopy { m }{
4103   \str_clear:N \l_stex_import_name_str
4104   \str_set:Nn \l_tmpa_str { #1 }
4105   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4106   \seq_map_inline:Nn \l_stex_all_modules_seq {
4107     \str_set:Nn \l_tmpb_str { ##1 }
4108     \str_if_eq:eeT { \l_tmpa_str } {
4109       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4110     } {
4111       \seq_map_break:n {
4112         \stex_if_do_html:T {
```

```
4113          \stex_if_smsmode:F {
4114            \stex_annotate_invisible:nnn{donotcopy}{##1}{
4115              \stex_annotate:nnn{domain}{##1}{}
4116            }
4117          }
4118        }
4119        \str_set_eq:NN \l_stex_import_name_str \l_tmpb_str
4120      }
4121    }
4122    \seq_map_inline:cn {c_stex_module_##1_copymodules}{
4123      \str_set:Nn \l_tmpb_str { ####1 }
4124      \str_if_eq:eeT { \l_tmpa_str } {
4125        \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4126      } {
4127        \seq_map_break:n {\seq_map_break:n {
4128          \stex_if_do_html:T {
4129            \stex_if_smsmode:F {
4130              \stex_annotate_invisible:nnn{donotcopy}{####1}{
4131                \stex_annotate:nnn{domain}{
4132                  \prop_item:cn {l_stex_copymodule_ ####1 _prop}{module}
4133                }{}
4134              }
4135            }
4136          }
4137          \str_set:Nx \l_stex_import_name_str {
4138            \prop_item:cn {l_stex_copymodule_ ####1 _prop}{module}
4139          }
4140        }}
4141      }
4142    }
4143  }
4144  \str_if_empty:NTF \l_stex_import_name_str {
4145    % TODO throw error
4146  }{
4147    \stex_collect_imports:n {\l_stex_import_name_str }
4148    \seq_map_inline:Nn \l_stex_collect_imports_seq {
4149      \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
4150      \seq_map_inline:cn {c_stex_module_##1_constants}{
4151        \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? ####1 }
4152        \bool_lazy_any:nT {
4153          { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_name_str}}
4154          { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_macroname_str}}
4155          { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_def_tl}}
4156        }{
4157          % TODO throw error
4158        }
4159      }
4160    }
4161    \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
4162    \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_name_str }
4163    \prop_put:Nno \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
4164  }
4165  \stex_smsmode_do:
4166 }
```

```
4167
4168 \NewDocumentCommand \assign { m m }{
4169   \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
4170   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
4171   \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _def_tl}{#2}
4172   \stex_smsmode_do:
4173 }
4174
4175 \keys_define:nn { stex / renamedecl } {
4176   name        .str_set_x:N = \l_stex_renamedecl_name_str
4177 }
4178 \cs_new_protected:Nn \__stex_copymodule_renamedecl_args:n {
4179   \str_clear:N \l_stex_renamedecl_name_str
4180   \keys_set:nn { stex / renamedecl } { #1 }
4181 }
4182
4183 \NewDocumentCommand \renamedecl { O{} m m }{
4184   \__stex_copymodule_renamedecl_args:n { #1 }
4185   \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
4186   \stex_debug:nn{renamedecl}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
4187   \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
4188   \str_if_empty:NTF \l_stex_renamedecl_name_str {
4189     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4190       \l_stex_get_symbol_uri_str
4191     } }
4192   } {
4193     \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex
4194     \stex_debug:nn{renamedecl}{@~\l_stex_current_module_str ? \l_stex_renamedecl_name_str}
4195     \prop_set_eq:cc {l_stex_symdecl_
4196       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4197       _prop
4198     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
4199     \seq_set_eq:cc {l_stex_symdecl_
4200       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4201       _notations
4202     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
4203     \prop_put:cnx {l_stex_symdecl_
4204       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4205       _prop
4206     }{ name }{ \l_stex_renamedecl_name_str }
4207     \prop_put:cnx {l_stex_symdecl_
4208       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4209       _prop
4210     }{ module }{ \l_stex_current_module_str }
4211     \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
4212       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4213     }
4214     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4215       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4216     } }
4217   }
4218   \stex_smsmode_do:
4219 }
4220
```

```
4221  \stex_deactivate_macro:Nn \assign {copymodules}
4222  \stex_deactivate_macro:Nn \renamedecl {copymodules}
4223  \stex_deactivate_macro:Nn \donotcopy {copymodules}

4224

4225
```

31.2   The feature environment

structural@feature

```
4226  ⟨@@=stex_features⟩

4227

4228  \NewDocumentEnvironment{structural_feature_module}{ m m m }{
4229    \stex_if_in_module:F {
4230      \msg_set:nnn{stex}{error/nomodule}{
4231        Structural~Feature~has~to~occur~in~a~module:\\
4232        Feature~#2~of~type~#1\\
4233        In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
4234      }
4235      \msg_error:nn{stex}{error/nomodule}
4236    }

4237

4238    \str_set_eq:NN \l_tmpa_str \l_stex_current_module_str

4239

4240    \stex_module_setup:nn{meta=NONE}{#2 - #1}

4241

4242    \stex_if_do_html:T {
4243      \begin{stex_annotate_env}{ feature:#1 }{\l_tmpa_str ? #2 - #1}
4244        \stex_annotate_invisible:nnn{header}{}{ #3 }
4245    }
4246  }{
4247    \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
4248    \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
4249    \stex_debug:nn{features}{
4250      Feature: \l_stex_last_feature_str
4251    }
4252    \stex_if_do_html:T {
4253      \end{stex_annotate_env}
4254    }
4255  }
```

31.3   Structure

structure

```
4256  ⟨@@=stex_structures⟩
4257  \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
4258    \prop_if_exist:cF {c_stex_module_\l_stex_current_module_str _structures}{
4259      \prop_new:c {c_stex_module_\l_stex_current_module_str _structures}
4260    }
4261    \prop_gput:cxx{c_stex_module_\l_stex_current_module_str _structures}
4262      {#1}{#2}
4263  }

4264
```

```
4265 \keys_define:nn { stex / features / structure } {
4266   name            .str_set_x:N  = \l__stex_structures_name_str ,
4267 }
4268
4269 \cs_new_protected:Nn \__stex_structures_structure_args:n {
4270   \str_clear:N \l__stex_structures_name_str
4271   \keys_set:nn { stex / features / structure } { #1 }
4272 }
4273
4274 \NewDocumentEnvironment{mathstructure}{m O{}}{
4275   \__stex_structures_structure_args:n { #2 }
4276   \str_if_empty:NT \l__stex_structures_name_str {
4277     \str_set:Nx \l__stex_structures_name_str { #1 }
4278   }
4279   \stex_suppress_html:n {
4280     \exp_args:Nx \stex_symdecl_do:nn {
4281       name = \l__stex_structures_name_str ,
4282       def  = {\STEXsymbol{module-type}{
4283         \_stex_term_math_oms:nnnn {
4284           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4285             { ns } ?
4286             \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4287               { name } / \l__stex_structures_name_str - structure
4288         }{}{0}{}
4289       }}
4290     }{ #1 }
4291   }
4292   \exp_args:Nnnx
4293   \begin{structural_feature_module}{ structure }
4294     { \l__stex_structures_name_str }{}
4295   \stex_smsmode_do:
4296 }{
4297   \end{structural_feature_module}
4298   \_stex_reset_up_to_module:n \l_stex_last_feature_str
4299   \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
4300   \seq_clear:N \l_tmpa_seq
4301   \seq_map_inline:Nn \l_stex_collect_imports_seq {
4302     \seq_map_inline:cn{c_stex_module_##1_constants}{
4303       \seq_put_right:Nn \l_tmpa_seq { ##1 ? ####1 }
4304     }
4305   }
4306   \exp_args:Nnno
4307   \prop_gput:cnn {c_stex_module_ \l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
4308   \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
4309   \stex_add_structure_to_current_module:nn
4310     \l__stex_structures_name_str
4311     \l_stex_last_feature_str
4312
4313   \stex_execute_in_module:x {
4314     \tl_set:cn { #1 }{
4315       \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structure
4316     }
4317   }
4318 }
```

191

```
4319
4320  \cs_new:Nn \stex_invoke_structure:nn {
4321    \stex_invoke_symbol:n { #1?#2 }
4322  }
4323
4324  \cs_new_protected:Nn \stex_get_structure:n {
4325    \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4326      \tl_set:Nn \l_tmpa_tl { #1 }
4327      \__stex_structures_get_from_cs:
4328    }{
4329      \cs_if_exist:cTF { #1 }{
4330        \cs_set_eq:Nc \l_tmpa_cs { #1 }
4331        \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
4332        \str_if_empty:NTF \l_tmpa_str {
4333          \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs} \stex_invoke_structure:nn {
4334            \__stex_structures_get_from_cs:
4335          }{
4336            \__stex_structures_get_from_string:n { #1 }
4337          }
4338        }{
4339          \__stex_structures_get_from_string:n { #1 }
4340        }
4341      }{
4342        \__stex_structures_get_from_string:n { #1 }
4343      }
4344    }
4345  }
4346
4347  \cs_new_protected:Nn \__stex_structures_get_from_cs: {
4348    \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4349      { \tl_tail:N \l_tmpa_tl }
4350    \str_set:Nx \l_tmpa_str {
4351      \exp_after:wN \use_i:nn \l_tmpa_tl
4352    }
4353    \str_set:Nx \l_tmpb_str {
4354      \exp_after:wN \use_ii:nn \l_tmpa_tl
4355    }
4356    \str_set:Nx \l_stex_get_structure_str {
4357      \l_tmpa_str ? \l_tmpb_str
4358    }
4359    \str_set:Nx \l_stex_get_structure_module_str {
4360      \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
4361    }
4362  }
4363
4364  \cs_new_protected:Nn \__stex_structures_get_from_string:n {
4365    \tl_set:Nn \l_tmpa_tl {
4366      \msg_error:nnn{stex}{error/unknownstructure}{#1}
4367    }
4368    \str_set:Nn \l_tmpa_str { #1 }
4369    \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4370
4371    \seq_map_inline:Nn \l_stex_all_modules_seq {
4372      \prop_if_exist:cT {c_stex_module_##1_structures} {
```

```
4373        \prop_map_inline:cn {c_stex_module_##1_structures} {
4374          \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
4375            \prop_map_break:n{\seq_map_break:n{
4376              \tl_set:Nn \l_tmpa_tl {
4377                \str_set:Nn \l_stex_get_structure_str {##1?####1}
4378                \str_set:Nn \l_stex_get_structure_module_str {####2}
4379              }
4380            }}
4381          }
4382        }
4383      }
4384    }
4385    \l_tmpa_tl
4386 }
```

```
4387
4388 \keys_define:nn { stex / instantiate } {
4389    name        .str_set_x:N  = \l__stex_structures_name_str
4390 }
4391 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
4392    \str_clear:N \l__stex_structures_name_str
4393    \keys_set:nn { stex / instantiate } { #1 }
4394 }
4395
4396 \NewDocumentCommand \instantiate {m O{} m m m}{
4397    \begingroup
4398      \stex_get_structure:n {#4}
4399      \__stex_structures_instantiate_args:n { #2 }
4400      \str_if_empty:NT \l__stex_structures_name_str {
4401        \str_set:Nn \l__stex_structures_name_str { #1 }
4402      }
4403      \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4404      \seq_clear:N \l__stex_structures_fields_seq
4405      \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4406      \seq_map_inline:Nn \l_stex_collect_imports_seq {
4407        \seq_map_inline:cn {c_stex_module_##1_constants}{
4408          \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4409        }
4410      }
4411
4412      \tl_if_empty:nF{#3}{
4413        \seq_set_split:Nnn \l_tmpa_seq , {#3}
4414        \prop_clear:N \l_tmpa_prop
4415        \seq_map_inline:Nn \l_tmpa_seq {
4416          \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4417          \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4418            \msg_error:nnn{stex}{error/keyval}{##1}
4419          }
4420          \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
4421          \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4422          \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
4423          \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}
4424          \exp_args:Nxx \str_if_eq:nnF
```

```
4425            {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4426            {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
4427            \msg_error:nnxxxx{stex}{error/incompatible}
4428              {\l__stex_structures_dom_str}
4429              {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4430              {\l_stex_get_symbol_uri_str}
4431              {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
4432          }
4433          \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str
4434        }
4435      }
4436
4437      \seq_map_inline:Nn \l__stex_structures_fields_seq {
4438        \str_set:Nx \l_tmpa_str {field:\l__stex_structures_name_str . \prop_item:cn {l_stex_sy
4439        \stex_debug:nn{instantiate}{Field~\l_tmpa_str :~##1}
4440
4441        \stex_add_constant_to_current_module:n {\l_tmpa_str}
4442        \stex_execute_in_module:x {
4443          \prop_set_from_keyval:cn { l_stex_symdecl_ \l_stex_current_module_str?\l_tmpa_str _p
4444            name  = \l_tmpa_str ,
4445            args  = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4446            arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4447            assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4448          }
4449          \seq_clear:c {l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _notations}
4450        }
4451
4452        \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4453          \stex_find_notation:nn{##1}{}
4454          \stex_execute_in_module:x {
4455            \seq_put_right:cn {l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _notation
4456          }
4457
4458          \stex_copy_control_sequence:ccN
4459            {stex_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_notation_
4460            {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4461            \l_tmpa_tl
4462          \exp_args:No \stex_execute_in_module:n \l_tmpa_tl
4463
4464
4465          \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
4466            \tl_set_eq:Nc \l_tmpa_cs {stex_op_notation_##1\c_hash_str \l_stex_notation_variant
4467            \stex_execute_in_module:x {
4468              \tl_set:cn
4469              {stex_op_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_nota
4470              { \exp_args:No \exp_not:n \l_tmpa_cs}
4471            }
4472          }
4473
4474        }
4475
4476        \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\l_stex_cur
4477      }
4478
```

```
4479      \stex_execute_in_module:x {
4480        \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
4481          domain = \l_stex_get_structure_module_str ,
4482          \prop_to_keyval:N \l_tmpa_prop
4483        }
4484        \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
4485      }
4486      \stex_debug:nn{instantiate}{
4487        Instance~\l_stex_current_module_str?\l__stex_structures_name_str \\
4488        \prop_to_keyval:N \l_tmpa_prop
4489      }
4490      \exp_args:Nxx \stex_symdecl_do:nn {
4491        type={\STEXsymbol{module-type}{
4492          \__stex_term_math_oms:nnnn {
4493            \l_stex_get_structure_module_str
4494          }{}{0}{}
4495        }}
4496      }{\l__stex_structures_name_str}
4497 %      {
4498        \str_set:Nx \l_stex_get_symbol_uri_str {\l_stex_current_module_str?\l__stex_structures
4499        \tl_set:Nn \l_stex_notation_after_do_tl {\__stex_notation_final:}
4500        \stex_notation_do:nnnnn{}{0}{}{}{\comp{#5}}
4501 %    }
4502      %\exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}
4503    \endgroup
4504    \stex_smsmode_do:\ignorespacesandpars
4505 }
4506
4507 \cs_new_protected:Nn \stex_symbol_or_var:n {
4508    \cs_if_exist:cTF{#1}{
4509      \cs_set_eq:Nc \l_tmpa_tl { #1 }
4510      \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4511      \str_if_empty:NTF \l_tmpa_str {
4512        \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
4513          \stex_invoke_variable:n {
4514            \bool_set_true:N \l_stex_symbol_or_var_bool
4515            \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
4516            \str_set:Nx \l_stex_get_symbol_uri_str {
4517              \exp_after:wN \use:n \l_tmpa_tl
4518            }
4519          }{
4520            \bool_set_false:N \l_stex_symbol_or_var_bool
4521            \stex_get_symbol:n{#1}
4522          }
4523      }{
4524        \__stex_structures_symbolorvar_from_string:n{ #1 }
4525      }
4526    }{
4527      \__stex_structures_symbolorvar_from_string:n{ #1 }
4528    }
4529 }
4530
4531 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
4532    \prop_if_exist:cTF {l_stex_variable_#1 _prop}{
```

195

```
4533       \bool_set_true:N \l_stex_symbol_or_var_bool
4534       \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
4535    }{
4536       \bool_set_false:N \l_stex_symbol_or_var_bool
4537       \stex_get_symbol:n{#1}
4538    }
4539 }
4540
4541 \keys_define:nn { stex / varinstantiate } {
4542    name         .str_set_x:N  = \l__stex_structures_name_str,
4543    bind         .choices:nn   =
4544       {forall,exists}
4545       {\str_set:Nx \l__stex_structures_bind_str {\l_keys_choice_tl}}
4546
4547 }
4548 \cs_new_protected:Nn \__stex_structures_varinstantiate_args:n {
4549    \str_clear:N \l__stex_structures_name_str
4550    \str_clear:N \l__stex_structures_bind_str
4551    \keys_set:nn { stex / varinstantiate } { #1 }
4552 }
4553
4554 \NewDocumentCommand \varinstantiate {m O{} m m m}{
4555    \begingroup
4556       \stex_get_structure:n {#4}
4557       \__stex_structures_varinstantiate_args:n { #2 }
4558       \str_if_empty:NT \l__stex_structures_name_str {
4559          \str_set:Nn \l__stex_structures_name_str { #1 }
4560       }
4561       \stex_if_do_html:TF{
4562          \stex_annotate:nnn{varinstance}{\l__stex_structures_name_str}
4563       }{\use:n}
4564       {
4565          \stex_if_do_html:T{
4566             \stex_annotate:nnn{domain}{\l_stex_get_structure_module_str}{}
4567          }
4568          \seq_clear:N \l__stex_structures_fields_seq
4569          \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4570          \seq_map_inline:Nn \l_stex_collect_imports_seq {
4571             \seq_map_inline:cn {c_stex_module_##1_constants}{
4572                \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4573             }
4574          }
4575          \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4576          \prop_clear:N \l_tmpa_prop
4577          \tl_if_empty:nF {#3} {
4578             \seq_set_split:Nnn \l_tmpa_seq , {#3}
4579             \seq_map_inline:Nn \l_tmpa_seq {
4580                \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4581                \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4582                   \msg_error:nnn{stex}{error/keyval}{##1}
4583                }
4584                \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_stru
4585                \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4586                \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol
```

196

```
4587            \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
4588            \stex_if_do_html:T{
4589              \stex_annotate:nnn{assign}{\l__stex_structures_dom_str,\l_stex_get_symbol_uri_st
4590            }
4591            \bool_if:NTF \l_stex_symbol_or_var_bool {
4592              \exp_args:Nxx \str_if_eq:nnF
4593                {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4594                {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}{
4595              \msg_error:nnxxxx{stex}{error/incompatible}
4596                {\l__stex_structures_dom_str}
4597                {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4598                {\l_stex_get_symbol_uri_str}
4599                {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}
4600            }
4601            \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n
4602          }{
4603              \exp_args:Nxx \str_if_eq:nnF
4604                {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4605                {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
4606              \msg_error:nnxxxx{stex}{error/incompatible}
4607                {\l__stex_structures_dom_str}
4608                {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4609                {\l_stex_get_symbol_uri_str}
4610                {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
4611            }
4612            \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n {
4613          }
4614        }
4615      }
4616      \tl_gclear:N \g__stex_structures_aftergroup_tl
4617      \seq_map_inline:Nn \l__stex_structures_fields_seq {
4618        \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdec
4619        \stex_debug:nn{varinstantiate}{Field~\l_tmpa_str :~##1}
4620        \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4621          \stex_find_notation:nn{##1}{}
4622          \cs_gset_eq:cc{g__stex_structures_tmpa_\l_tmpa_str _cs}
4623            {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4624          \stex_debug:nn{varinstantiate}{Notation:~\cs_meaning:c{g__stex_structures_tmpa_\l_
4625          \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
4626            \cs_gset_eq:cc {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
4627              {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4628            \stex_debug:nn{varinstantiate}{Operator~Notation:~\cs_meaning:c{g__stex_struct
4629          }
4630        }
4631
4632        \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4633          \prop_set_from_keyval:cn { l_stex_variable_ \l_tmpa_str _prop}{
4634            name  = \l_tmpa_str ,
4635            args  = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4636            arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4637            assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4638          }
4639          \cs_set_eq:cc {stex_var_notation_\l_tmpa_str _cs}
4640            {g__stex_structures_tmpa_\l_tmpa_str _cs}
```

197

```
4641              \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str _cs}
4642                {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
4643            }
4644            \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\stex_inv
4645          }
4646          \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4647            \prop_set_from_keyval:cn {l_stex_varinstance_\l__stex_structures_name_str _prop }{
4648              domain = \l_stex_get_structure_module_str ,
4649              \prop_to_keyval:N \l_tmpa_prop
4650            }
4651            \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
4652            \tl_set:cn {l_stex_varinstance_\l__stex_structures_name_str _op_tl}{
4653              \exp_args:Nnx \exp_not:N \use:nn {
4654                \str_set:Nn \exp_not:N \l_stex_current_symbol_str {var://\l__stex_structures_nam
4655                \_stex_term_omv:nn {var://\l__stex_structures_name_str}{
4656                  \exp_not:n{
4657                    \_varcomp{#5}
4658                  }
4659                }
4660              }{
4661                \exp_not:n{\_stex_reset:N \l_stex_current_symbol_str}
4662              }
4663            }
4664          }
4665        }
4666        \stex_debug:nn{varinstantiate}{\expandafter\detokenize\expandafter{\g__stex_structures_a
4667        \aftergroup\g__stex_structures_aftergroup_tl
4668    \endgroup
4669    \stex_smsmode_do:\ignorespacesandpars
4670 }

4672 \cs_new_protected:Nn \stex_invoke_instance:n {
4673    \peek_charcode_remove:NTF ! {
4674      \stex_invoke_symbol:n{#1}
4675    }{
4676      \_stex_invoke_instance:nn {#1}
4677    }
4678 }


4681 \cs_new_protected:Nn \stex_invoke_varinstance:n {
4682    \peek_charcode_remove:NTF ! {
4683      \exp_args:Nnx \use:nn {
4684        \def\comp{\_varcomp}
4685        \use:c{l_stex_varinstance_#1_op_tl}
4686      }{
4687        \_stex_reset:N \comp
4688      }
4689    }{
4690      \_stex_invoke_varinstance:nn {#1}
4691    }
4692 }

4694 \cs_new_protected:Nn \_stex_invoke_instance:nn {
```

198

```
4695      \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
4696        \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
4697      }{
4698        \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
4699        \msg_error:nnxxx{stex}{error/unknownfield}{#2}{#1}{
4700          \prop_to_keyval:N \l_tmpa_prop
4701        }
4702      }
4703 }
4704
4705 \cs_new_protected:Nn \_stex_invoke_varinstance:nn {
4706      \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
4707        \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
4708        \l_tmpa_tl
4709      }{
4710        \msg_error:nnnnn{stex}{error/unknownfield}{#2}{#1}{}
4711      }
4712 }
```

(End definition for \instantiate*. This function is documented on page 31.)*

\stex_invoke_structure:nnn

```
4713 % #1: URI of the instance
4714 % #2: URI of the instantiated module
4715 \cs_new_protected:Nn \stex_invoke_structure:nnn {
4716      \tl_if_empty:nTF{ #3 }{
4717        \prop_set_eq:Nc \l__stex_structures_structure_prop {
4718          c_stex_feature_ #2 _prop
4719        }
4720        \tl_clear:N \l_tmpa_tl
4721        \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
4722        \seq_map_inline:Nn \l_tmpa_seq {
4723          \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
4724          \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
4725          \cs_if_exist:cT {
4726            stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
4727          }{
4728            \tl_if_empty:NF \l_tmpa_tl {
4729              \tl_put_right:Nn \l_tmpa_tl {,}
4730            }
4731            \tl_put_right:Nx \l_tmpa_tl {
4732              \stex_invoke_symbol:n {#1/\l_tmpa_str}!
4733            }
4734          }
4735        }
4736        \exp_args:No \mathstruct \l_tmpa_tl
4737      }{
4738        \stex_invoke_symbol:n{#1/#3}
4739      }
4740 }
```

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

```
4741 ⟨/package⟩
```

199

Chapter 32

sTeX
-Statements Implementation

```
4742 ⟨∗package⟩
4743
4744 %%%%%%%%%%%%   features.dtx   %%%%%%%%%%%%
4745
4746 ⟨@@=stex_statements⟩
```

Warnings and error messages

```
4747
```

\titleemph

```
4748 \def\titleemph#1{\textbf{#1}}
```

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

32.1   Definitions

definiendum

```
4749 \keys_define:nn {stex / definiendum }{
4750   pre     .tl_set:N    = \l__stex_statements_definiendum_pre_tl,
4751   post    .tl_set:N    = \l__stex_statements_definiendum_post_tl,
4752   root    .str_set_x:N = \l__stex_statements_definiendum_root_str,
4753   gfa     .str_set_x:N = \l__stex_statements_definiendum_gfa_str
4754 }
4755 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
4756   \str_clear:N \l__stex_statements_definiendum_root_str
4757   \tl_clear:N \l__stex_statements_definiendum_post_tl
4758   \str_clear:N \l__stex_statements_definiendum_gfa_str
4759   \keys_set:nn { stex / definiendum }{ #1 }
4760 }
4761 \NewDocumentCommand \definiendum { O{} m m} {
4762   \__stex_statements_definiendum_args:n { #1 }
4763   \stex_get_symbol:n { #2 }
4764   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4765   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
4766     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

```
4767        \tl_set:Nn \l_tmpa_tl { #3 }
4768      } {
4769        \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
4770        \tl_set:Nn \l_tmpa_tl {
4771          \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
4772        }
4773      }
4774    } {
4775      \tl_set:Nn \l_tmpa_tl { #3 }
4776    }
4777
4778    % TODO root
4779    \stex_html_backend:TF {
4780      \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
4781    } {
4782      \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
4783    }
4784 }
4785 \stex_deactivate_macro:Nn \definiendum {definition~environments}
```

(*End definition for* definiendum. *This function is documented on page 40.*)

definame

```
4786
4787 \NewDocumentCommand \definame { O{} m } {
4788    \__stex_statements_definiendum_args:n { #1 }
4789    % TODO: root
4790    \stex_get_symbol:n { #2 }
4791    \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4792    \str_set:Nx \l_tmpa_str {
4793      \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4794    }
4795    \str_replace_all:Nnn \l_tmpa_str {-} {~}
4796    \stex_html_backend:TF {
4797      \stex_if_do_html:T {
4798        \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4799          \l_tmpa_str\l__stex_statements_definiendum_post_tl
4800        }
4801      }
4802    } {
4803      \exp_args:Nnx \defemph@uri {
4804        \l_tmpa_str\l__stex_statements_definiendum_post_tl
4805      } { \l_stex_get_symbol_uri_str }
4806    }
4807 }
4808 \stex_deactivate_macro:Nn \definame {definition~environments}
4809
4810 \NewDocumentCommand \Definame { O{} m } {
4811    \__stex_statements_definiendum_args:n { #1 }
4812    \stex_get_symbol:n { #2 }
4813    \str_set:Nx \l_tmpa_str {
4814      \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4815    }
4816    \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
```

```
4817    \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4818    \stex_html_backend:TF {
4819      \stex_if_do_html:T {
4820        \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4821          \l_tmpa_str\l__stex_statements_definiendum_post_tl
4822        }
4823      }
4824    } {
4825      \exp_args:Nnx \defemph@uri {
4826        \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
4827      } { \l_stex_get_symbol_uri_str }
4828    }
4829 }
4830 \stex_deactivate_macro:Nn \Definame {definition~environments}
4831
4832 \NewDocumentCommand \premise { m }{
4833    \stex_annotate:nnn{ premise }{}{ #1 }
4834 }
4835 \NewDocumentCommand \conclusion { m }{
4836    \stex_annotate:nnn{ conclusion }{}{ #1 }
4837 }
4838 \NewDocumentCommand \definiens { O{} m }{
4839    \str_clear:N \l_stex_get_symbol_uri_str
4840    \tl_if_empty:nF {#1} {
4841      \stex_get_symbol:n { #1 }
4842    }
4843    \str_if_empty:NT \l_stex_get_symbol_uri_str {
4844      \int_compare:nNnTF {\clist_count:N \l__stex_statements_sdefinition_for_clist} = 1 {
4845        \str_set:Nx \l_stex_get_symbol_uri_str {\clist_item:Nn \l__stex_statements_sdefinition
4846      }{
4847        % TODO throw error
4848      }
4849    }
4850    \str_if_eq:eeT {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{module}}
4851      {\l_stex_current_module_str}{
4852        \str_if_eq:eeF {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defin
4853        {true}{
4854          \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
4855          \exp_args:Nx \stex_add_to_current_module:n {
4856            \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
4857          }
4858        }
4859      }
4860    \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
4861 }
4862
4863 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
4864 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
4865 \stex_deactivate_macro:Nn \definiens {definition~environments}
4866
```

(*End definition for* `definame`*. This function is documented on page [40](#).)

sdefinition

```
4867
4868  \keys_define:nn {stex / sdefinition }{
4869    type     .str_set_x:N  = \sdefinitiontype,
4870    id       .str_set_x:N  = \sdefinitionid,
4871    name     .str_set_x:N  = \sdefinitionname,
4872    for      .clist_set:N  = \l__stex_statements_sdefinition_for_clist ,
4873    title    .tl_set:N     = \sdefinitiontitle
4874  }
4875  \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
4876    \str_clear:N \sdefinitiontype
4877    \str_clear:N \sdefinitionid
4878    \str_clear:N \sdefinitionname
4879    \clist_clear:N \l__stex_statements_sdefinition_for_clist
4880    \tl_clear:N \sdefinitiontitle
4881    \keys_set:nn { stex / sdefinition }{ #1 }
4882  }
4883
4884  \NewDocumentEnvironment{sdefinition}{O{}}{
4885    \__stex_statements_sdefinition_args:n{ #1 }
4886    \stex_reactivate_macro:N \definiendum
4887    \stex_reactivate_macro:N \definame
4888    \stex_reactivate_macro:N \Definame
4889    \stex_reactivate_macro:N \premise
4890    \stex_reactivate_macro:N \definiens
4891    \stex_if_smsmode:F{
4892      \seq_clear:N \l_tmpa_seq
4893      \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4894        \tl_if_empty:nF{ ##1 }{
4895          \stex_get_symbol:n { ##1 }
4896          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4897            \l_stex_get_symbol_uri_str
4898          }
4899        }
4900      }
4901      \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpa_seq
4902      \exp_args:Nnnx
4903      \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {,}}
4904      \str_if_empty:NF \sdefinitiontype {
4905        \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
4906      }
4907      \str_if_empty:NF \sdefinitionname {
4908        \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
4909      }
4910      \clist_set:No \l_tmpa_clist \sdefinitiontype
4911      \tl_clear:N \l_tmpa_tl
4912      \clist_map_inline:Nn \l_tmpa_clist {
4913        \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
4914          \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
4915        }
4916      }
4917      \tl_if_empty:NTF \l_tmpa_tl {
4918        \__stex_statements_sdefinition_start:
4919      }{
4920        \l_tmpa_tl
```

```
4921        }
4922      }
4923      \stex_ref_new_doc_target:n \sdefinitionid
4924      \stex_smsmode_do:
4925  }{
4926      \stex_suppress_html:n {
4927        \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
4928      }
4929      \stex_if_smsmode:F {
4930        \clist_set:No \l_tmpa_clist \sdefinitiontype
4931        \tl_clear:N \l_tmpa_tl
4932        \clist_map_inline:Nn \l_tmpa_clist {
4933          \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
4934            \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
4935          }
4936        }
4937        \tl_if_empty:NTF \l_tmpa_tl {
4938          \__stex_statements_sdefinition_end:
4939        }{
4940          \l_tmpa_tl
4941        }
4942        \end{stex_annotate_env}
4943      }
4944  }
```

```
4945  \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
4946      \par\noindent\titleemph{Definition\tl_if_empty:NF \sdefinitiontitle {
4947        ~(\sdefinitiontitle)
4948      }~}
4949  }
4950  \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\par\medskip}
4951
4952  \newcommand\stexpatchdefinition[3][] {
4953      \str_set:Nx \l_tmpa_str{ #1 }
4954      \str_if_empty:NTF \l_tmpa_str {
4955        \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
4956        \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
4957      }{
4958        \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2
4959        \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
4960      }
4961  }
```

(*End definition for* \stexpatchdefinition. *This function is documented on page 42.*)

inline:

```
4962  \keys_define:nn {stex / inlinedef }{
4963    type    .str_set_x:N  = \sdefinitiontype,
4964    id      .str_set_x:N  = \sdefinitionid,
4965    for     .clist_set:N  = \l__stex_statements_sdefinition_for_clist ,
4966    name    .str_set_x:N  = \sdefinitionname
4967  }
4968  \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
```

204

```
4969    \str_clear:N \sdefinitiontype
4970    \str_clear:N \sdefinitionid
4971    \str_clear:N \sdefinitionname
4972    \clist_clear:N \l__stex_statements_sdefinition_for_clist
4973    \keys_set:nn { stex / inlinedef }{ #1 }
4974 }
4975 \NewDocumentCommand \inlinedef { O{} m } {
4976    \begingroup
4977    \__stex_statements_inlinedef_args:n{ #1 }
4978    \stex_reactivate_macro:N \definiendum
4979    \stex_reactivate_macro:N \definame
4980    \stex_reactivate_macro:N \Definame
4981    \stex_reactivate_macro:N \premise
4982    \stex_reactivate_macro:N \definiens
4983    \stex_ref_new_doc_target:n \sdefinitionid
4984    \stex_if_smsmode:TF{\stex_suppress_html:n {
4985       \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
4986    }}{
4987       \seq_clear:N \l_tmpa_seq
4988       \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4989          \tl_if_empty:nF{ ##1 }{
4990             \stex_get_symbol:n { ##1 }
4991             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4992                \l_stex_get_symbol_uri_str
4993             }
4994          }
4995       }
4996       \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpa_seq
4997       \exp_args:Nnx
4998       \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
4999          \str_if_empty:NF \sdefinitiontype {
5000             \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5001          }
5002          #2
5003          \str_if_empty:NF \sdefinitionname {
5004             \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sdefinitionname}}
5005             \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5006          }
5007       }
5008    }
5009    \endgroup
5010    \stex_smsmode_do:
5011 }
```

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

32.2 Assertions

sassertion

```
5012
5013 \keys_define:nn {stex / sassertion }{
5014    type    .str_set_x:N  = \sassertiontype,
5015    id      .str_set_x:N  = \sassertionid,
```

```
5016   title    .tl_set:N    = \sassertiontitle ,
5017   for      .clist_set:N = \l__stex_statements_sassertion_for_clist ,
5018   name     .str_set_x:N = \sassertionname
5019 }
5020 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
5021   \str_clear:N \sassertiontype
5022   \str_clear:N \sassertionid
5023   \str_clear:N \sassertionname
5024   \clist_clear:N \l__stex_statements_sassertion_for_clist
5025   \tl_clear:N \sassertiontitle
5026   \keys_set:nn { stex / sassertion }{ #1 }
5027 }
5028
5029 %\tl_new:N \g__stex_statements_aftergroup_tl
5030
5031 \NewDocumentEnvironment{sassertion}{O{}}{
5032   \__stex_statements_sassertion_args:n{ #1 }
5033   \stex_reactivate_macro:N \premise
5034   \stex_reactivate_macro:N \conclusion
5035   \stex_if_smsmode:F {
5036     \seq_clear:N \l_tmpa_seq
5037     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5038       \tl_if_empty:nF{ ##1 }{
5039         \stex_get_symbol:n { ##1 }
5040         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5041           \l_stex_get_symbol_uri_str
5042         }
5043       }
5044     }
5045     \exp_args:Nnnx
5046     \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}
5047     \str_if_empty:NF \sassertiontype {
5048       \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
5049     }
5050     \str_if_empty:NF \sassertionname {
5051       \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
5052     }
5053     \clist_set:No \l_tmpa_clist \sassertiontype
5054     \tl_clear:N \l_tmpa_tl
5055     \clist_map_inline:Nn \l_tmpa_clist {
5056       \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
5057         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
5058       }
5059     }
5060     \tl_if_empty:NTF \l_tmpa_tl {
5061       \__stex_statements_sassertion_start:
5062     }{
5063       \l_tmpa_tl
5064     }
5065   }
5066   \str_if_empty:NTF \sassertionid {
5067     \str_if_empty:NF \sassertionname {
5068       \stex_ref_new_doc_target:n {}
5069     }
```

206

```
5070      } {
5071         \stex_ref_new_doc_target:n \sassertionid
5072      }
5073      \stex_smsmode_do:
5074  }{
5075     \str_if_empty:NF \sassertionname {
5076        \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5077        \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5078     }
5079     \stex_if_smsmode:F {
5080        \clist_set:No \l_tmpa_clist \sassertiontype
5081        \tl_clear:N \l_tmpa_tl
5082        \clist_map_inline:Nn \l_tmpa_clist {
5083           \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
5084              \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
5085           }
5086        }
5087        \tl_if_empty:NTF \l_tmpa_tl {
5088           \__stex_statements_sassertion_end:
5089        }{
5090           \l_tmpa_tl
5091        }
5092        \end{stex_annotate_env}
5093     }
5094  }
```

\stexpatchassertion

```
5095
5096  \cs_new_protected:Nn \__stex_statements_sassertion_start: {
5097     \par\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {
5098        (\sassertiontitle)
5099     }~}
5100  }
5101  \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
5102
5103  \newcommand\stexpatchassertion[3][] {
5104     \str_set:Nx \l_tmpa_str{ #1 }
5105     \str_if_empty:NTF \l_tmpa_str {
5106        \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
5107        \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
5108     }{
5109        \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
5110        \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3
5111     }
5112  }
```

(*End definition for* \stexpatchassertion. *This function is documented on page 42.*)

\inlineass inline:

```
5113  \keys_define:nn {stex / inlineass }{
5114     type    .str_set_x:N  = \sassertiontype,
5115     id      .str_set_x:N  = \sassertionid,
5116     for     .clist_set:N  = \l__stex_statements_sassertion_for_clist ,
5117     name    .str_set_x:N  = \sassertionname
```

207

```
5118 }
5119 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
5120   \str_clear:N \sassertiontype
5121   \str_clear:N \sassertionid
5122   \str_clear:N \sassertionname
5123   \clist_clear:N \l__stex_statements_sassertion_for_clist
5124   \keys_set:nn { stex / inlineass }{ #1 }
5125 }
5126 \NewDocumentCommand \inlineass { O{} m } {
5127   \begingroup
5128   \stex_reactivate_macro:N \premise
5129   \stex_reactivate_macro:N \conclusion
5130   \__stex_statements_inlineass_args:n{ #1 }
5131   \str_if_empty:NTF \sassertionid {
5132     \str_if_empty:NF \sassertionname {
5133       \stex_ref_new_doc_target:n {}
5134     }
5135   } {
5136     \stex_ref_new_doc_target:n \sassertionid
5137   }
5138
5139   \stex_if_smsmode:TF{
5140     \str_if_empty:NF \sassertionname {
5141       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5142       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5143     }
5144   }{
5145     \seq_clear:N \l_tmpa_seq
5146     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5147       \tl_if_empty:nF{ ##1 }{
5148         \stex_get_symbol:n { ##1 }
5149         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5150           \l_stex_get_symbol_uri_str
5151         }
5152       }
5153     }
5154     \exp_args:Nnx
5155     \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{
5156       \str_if_empty:NF \sassertiontype {
5157         \stex_annotate_invisible:nnn{typestrings}{\sassertiontype}{}
5158       }
5159       #2
5160       \str_if_empty:NF \sassertionname {
5161         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5162         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5163         \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
5164       }
5165     }
5166   }
5167   \endgroup
5168   \stex_smsmode_do:
5169 }
```

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

208

32.3   Examples

```
5170
5171 \keys_define:nn {stex / sexample }{
5172   type     .str_set_x:N  = \exampletype,
5173   id       .str_set_x:N  = \sexampleid,
5174   title    .tl_set:N     = \sexampletitle,
5175   name     .str_set_x:N  = \sexamplename ,
5176   for      .clist_set:N  = \l__stex_statements_sexample_for_clist,
5177 }
5178 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
5179   \str_clear:N \sexampletype
5180   \str_clear:N \sexampleid
5181   \str_clear:N \sexamplename
5182   \tl_clear:N \sexampletitle
5183   \clist_clear:N \l__stex_statements_sexample_for_clist
5184   \keys_set:nn { stex / sexample }{ #1 }
5185 }
5186
5187 \NewDocumentEnvironment{sexample}{O{}}{
5188   \__stex_statements_sexample_args:n{ #1 }
5189   \stex_reactivate_macro:N \premise
5190   \stex_reactivate_macro:N \conclusion
5191   \stex_if_smsmode:F {
5192     \seq_clear:N \l_tmpa_seq
5193     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5194       \tl_if_empty:nF{ ##1 }{
5195         \stex_get_symbol:n { ##1 }
5196         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5197           \l_stex_get_symbol_uri_str
5198         }
5199       }
5200     }
5201     \exp_args:Nnnx
5202     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
5203     \str_if_empty:NF \sexampletype {
5204       \stex_annotate_invisible:nnn{typestrings}{\sexampletype}{}
5205     }
5206     \str_if_empty:NF \sexamplename {
5207       \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}
5208     }
5209     \clist_set:No \l_tmpa_clist \sexampletype
5210     \tl_clear:N \l_tmpa_tl
5211     \clist_map_inline:Nn \l_tmpa_clist {
5212       \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
5213         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
5214       }
5215     }
5216     \tl_if_empty:NTF \l_tmpa_tl {
5217       \__stex_statements_sexample_start:
5218     }{
5219       \l_tmpa_tl
5220     }
```

```
5221       }
5222     \str_if_empty:NF \sexampleid {
5223       \stex_ref_new_doc_target:n \sexampleid
5224     }
5225     \stex_smsmode_do:
5226 }{
5227     \str_if_empty:NF \sexamplename {
5228       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5229     }
5230     \stex_if_smsmode:F {
5231       \clist_set:No \l_tmpa_clist \sexampletype
5232       \tl_clear:N \l_tmpa_tl
5233       \clist_map_inline:Nn \l_tmpa_clist {
5234         \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
5235           \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
5236         }
5237       }
5238       \tl_if_empty:NTF \l_tmpa_tl {
5239         \__stex_statements_sexample_end:
5240       }{
5241         \l_tmpa_tl
5242       }
5243       \end{stex_annotate_env}
5244     }
5245 }
```

\stexpatchexample

```
5246
5247 \cs_new_protected:Nn \__stex_statements_sexample_start: {
5248     \par\noindent\titleemph{Example~\tl_if_empty:NF \sexampletitle {
5249       (\sexampletitle)
5250     }~}
5251 }
5252 \cs_new_protected:Nn \__stex_statements_sexample_end: {\par\medskip}
5253
5254 \newcommand\stexpatchexample[3][] {
5255     \str_set:Nx \l_tmpa_str{ #1 }
5256     \str_if_empty:NTF \l_tmpa_str {
5257       \tl_set:Nn \__stex_statements_sexample_start: { #2 }
5258       \tl_set:Nn \__stex_statements_sexample_end: { #3 }
5259     }{
5260       \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
5261       \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
5262     }
5263 }
```

(End definition for \stexpatchexample*. This function is documented on page 42.)*

\inlineex inline:

```
5264 \keys_define:nn {stex / inlineex }{
5265   type    .str_set_x:N  = \sexampletype,
5266   id      .str_set_x:N  = \sexampleid,
5267   for     .clist_set:N  = \l__stex_statements_sexample_for_clist ,
5268   name    .str_set_x:N  = \sexamplename
```

```
5269 }
5270 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
5271   \str_clear:N \sexampletype
5272   \str_clear:N \sexampleid
5273   \str_clear:N \sexamplename
5274   \clist_clear:N \l__stex_statements_sexample_for_clist
5275   \keys_set:nn { stex / inlineex }{ #1 }
5276 }
5277 \NewDocumentCommand \inlineex { O{} m } {
5278   \begingroup
5279   \stex_reactivate_macro:N \premise
5280   \stex_reactivate_macro:N \conclusion
5281   \__stex_statements_inlineex_args:n{ #1 }
5282   \str_if_empty:NF \sexampleid {
5283     \stex_ref_new_doc_target:n \sexampleid
5284   }
5285   \stex_if_smsmode:TF{
5286     \str_if_empty:NF \sexamplename {
5287       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\examplename}}
5288     }
5289   }{
5290     \seq_clear:N \l_tmpa_seq
5291     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5292       \tl_if_empty:nF{ ##1 }{
5293         \stex_get_symbol:n { ##1 }
5294         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5295           \l_stex_get_symbol_uri_str
5296         }
5297       }
5298     }
5299     \exp_args:Nnx
5300     \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpa_seq {,}}{
5301       \str_if_empty:NF \sexampletype {
5302         \stex_annotate_invisible:nnn{typestrings}{\sexampletype}{}
5303       }
5304       #2
5305       \str_if_empty:NF \sexamplename {
5306         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5307         \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}
5308       }
5309     }
5310   }
5311   \endgroup
5312   \stex_smsmode_do:
5313 }
```

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

32.4 Logical Paragraphs

sparagraph

```
5314 \keys_define:nn { stex / sparagraph} {
5315   id       .str_set_x:N  = \sparagraphid ,
```

```
5316    title    .tl_set:N     = \l_stex_sparagraph_title_tl ,
5317    type     .str_set_x:N  = \sparagraphtype ,
5318    for      .clist_set:N  = \l__stex_statements_sparagraph_for_clist ,
5319    from     .tl_set:N     = \sparagraphfrom ,
5320    to       .tl_set:N     = \sparagraphto ,
5321    start    .tl_set:N     = \l_stex_sparagraph_start_tl ,
5322    name     .str_set:N    = \sparagraphname ,
5323    imports .tl_set:N      = \l__stex_statements_sparagraph_imports_tl
5324 }
5325
5326 \cs_new_protected:Nn \stex_sparagraph_args:n {
5327   \tl_clear:N \l_stex_sparagraph_title_tl
5328   \tl_clear:N \sparagraphfrom
5329   \tl_clear:N \sparagraphto
5330   \tl_clear:N \l_stex_sparagraph_start_tl
5331   \tl_clear:N \l__stex_statements_sparagraph_imports_tl
5332   \str_clear:N \sparagraphid
5333   \str_clear:N \sparagraphtype
5334   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5335   \str_clear:N \sparagraphname
5336   \keys_set:nn { stex / sparagraph }{ #1 }
5337 }
5338 \newif\if@in@omtext\@in@omtextfalse
5339
5340 \NewDocumentEnvironment {sparagraph} { O{} } {
5341   \stex_sparagraph_args:n { #1 }
5342   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5343     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
5344   }{
5345     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
5346   }
5347   \@in@omtexttrue
5348   \stex_if_smsmode:F {
5349     \seq_clear:N \l_tmpa_seq
5350     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5351       \tl_if_empty:nF{ ##1 }{
5352         \stex_get_symbol:n { ##1 }
5353         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5354           \l_stex_get_symbol_uri_str
5355         }
5356       }
5357     }
5358     \exp_args:Nnnx
5359     \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
5360     \str_if_empty:NF \sparagraphtype {
5361       \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
5362     }
5363     \str_if_empty:NF \sparagraphfrom {
5364       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5365     }
5366     \str_if_empty:NF \sparagraphto {
5367       \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5368     }
5369     \str_if_empty:NF \sparagraphname {
```

```
5370        \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
5371      }
5372      \clist_set:No \l_tmpa_clist \sparagraphtype
5373      \tl_clear:N \l_tmpa_tl
5374      \clist_map_inline:Nn \sparagraphtype {
5375        \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
5376          \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
5377        }
5378      }
5379      \stex_csl_to_imports:No \usemodule \l__stex_statements_sparagraph_imports_tl
5380      \tl_if_empty:NTF \l_tmpa_tl {
5381        \__stex_statements_sparagraph_start:
5382      }{
5383        \l_tmpa_tl
5384      }
5385    }
5386    \clist_set:No \l_tmpa_clist \sparagraphtype
5387    \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}
5388    {
5389      \stex_reactivate_macro:N \definiendum
5390      \stex_reactivate_macro:N \definame
5391      \stex_reactivate_macro:N \Definame
5392      \stex_reactivate_macro:N \premise
5393      \stex_reactivate_macro:N \definiens
5394    }
5395    \str_if_empty:NTF \sparagraphid {
5396      \str_if_empty:NTF \sparagraphname {
5397        \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5398          \stex_ref_new_doc_target:n {}
5399        }
5400      } {
5401        \stex_ref_new_doc_target:n {}
5402      }
5403    } {
5404      \stex_ref_new_doc_target:n \sparagraphid
5405    }
5406    \exp_args:NNx
5407    \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5408      \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5409        \tl_if_empty:nF{ ##1 }{
5410          \stex_get_symbol:n { ##1 }
5411          \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5412        }
5413      }
5414    }
5415    \stex_smsmode_do:
5416    \ignorespacesandpars
5417 }{
5418    \str_if_empty:NF \sparagraphname {
5419      \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5420      \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5421    }
5422    \stex_if_smsmode:F {
5423      \clist_set:No \l_tmpa_clist \sparagraphtype
```

213

```
5424        \tl_clear:N \l_tmpa_tl
5425        \clist_map_inline:Nn \l_tmpa_clist {
5426          \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
5427            \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}}
5428          }
5429        }
5430        \tl_if_empty:NTF \l_tmpa_tl {
5431          \__stex_statements_sparagraph_end:
5432        }{
5433          \l_tmpa_tl
5434        }
5435        \end{stex_annotate_env}
5436      }
5437 }
```

```
5438
5439 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
5440   \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5441     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
5442       \titleemph{\l_stex_sparagraph_title_tl}:~
5443     }
5444   }{
5445     \titleemph{\l_stex_sparagraph_start_tl}~
5446   }
5447 }
5448 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
5449
5450 \newcommand\stexpatchparagraph[3][] {
5451     \str_set:Nx \l_tmpa_str{ #1 }
5452     \str_if_empty:NTF \l_tmpa_str {
5453       \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
5454       \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
5455     }{
5456       \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
5457       \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
5458     }
5459 }
5460
5461 \keys_define:nn { stex / inlinepara} {
5462   id      .str_set_x:N   = \sparagraphid ,
5463   type    .str_set_x:N   = \sparagraphtype ,
5464   for     .clist_set:N   = \l__stex_statements_sparagraph_for_clist ,
5465   from    .tl_set:N      = \sparagraphfrom ,
5466   to      .tl_set:N      = \sparagraphto ,
5467   name    .str_set:N     = \sparagraphname
5468 }
5469 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
5470   \tl_clear:N \sparagraphfrom
5471   \tl_clear:N \sparagraphto
5472   \str_clear:N \sparagraphid
5473   \str_clear:N \sparagraphtype
5474   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5475   \str_clear:N \sparagraphname
```

```
5476      \keys_set:nn { stex / inlinepara }{ #1 }
5477  }
5478  \NewDocumentCommand \inlinepara { O{} m } {
5479      \begingroup
5480      \__stex_statements_inlinepara_args:n{ #1 }
5481      \clist_set:No \l_tmpa_clist \sparagraphtype
5482      \str_if_empty:NTF \sparagraphid {
5483          \str_if_empty:NTF \sparagraphname {
5484              \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5485                  \stex_ref_new_doc_target:n {}
5486              }
5487          } {
5488              \stex_ref_new_doc_target:n {}
5489          }
5490      } {
5491          \stex_ref_new_doc_target:n \sparagraphid
5492      }
5493      \stex_if_smsmode:TF{
5494          \str_if_empty:NF \sparagraphname {
5495              \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5496              \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5497          }
5498      }{
5499          \seq_clear:N \l_tmpa_seq
5500          \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5501              \tl_if_empty:nF{ ##1 }{
5502                  \stex_get_symbol:n { ##1 }
5503                  \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5504                      \l_stex_get_symbol_uri_str
5505                  }
5506              }
5507          }
5508          \exp_args:Nnx
5509          \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}{
5510              \str_if_empty:NF \sparagraphtype {
5511                  \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
5512              }
5513              \str_if_empty:NF \sparagraphfrom {
5514                  \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5515              }
5516              \str_if_empty:NF \sparagraphto {
5517                  \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5518              }
5519              \str_if_empty:NF \sparagraphname {
5520                  \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5521                  \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
5522                  \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5523              }
5524              \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5525                  \clist_map_inline:Nn \l_tmpa_seq {
5526                      \stex_ref_new_sym_target:n {##1}
5527                  }
5528              }
5529              #2
```

```
5530        }
5531     }
5532     \endgroup
5533     \stex_smsmode_do:
5534 }
5535
```

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

```
5536 ⟨/package⟩
```

Chapter 33

The Implementation

33.1 Package Options

We declare some switches which will modify the behavior according to the package options. Generally, an option xxx will just set the appropriate switches to true (otherwise they stay false).[8]

```
5537 ⟨∗package⟩
5538 ⟨@@=stex_sproof⟩
5539
5540 %%%%%%%%%%%%%    sproof.dtx    %%%%%%%%%%%%%
5541
```

33.2 Proofs

We first define some keys for the proof environment.

```
5542 \keys_define:nn { stex / spf } {
5543   id          .str_set_x:N  = \spfid,
5544   for         .clist_set:N  = \l__stex_sproof_spf_for_clist ,
5545   from        .tl_set:N     = \l__stex_sproof_spf_from_tl ,
5546   proofend    .tl_set:N     = \l__stex_sproof_spf_proofend_tl,
5547   type        .str_set_x:N  = \spftype,
5548   title       .tl_set:N     = \spftitle,
5549   continues   .tl_set:N     = \l__stex_sproof_spf_continues_tl,
5550   functions   .tl_set:N     = \l__stex_sproof_spf_functions_tl,
5551   method      .tl_set:N     = \l__stex_sproof_spf_method_tl
5552 }
5553 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
5554 \str_clear:N \spfid
5555 \tl_clear:N \l__stex_sproof_spf_for_tl
5556 \tl_clear:N \l__stex_sproof_spf_from_tl
5557 \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
5558 \str_clear:N \spftype
5559 \tl_clear:N \spftitle
5560 \tl_clear:N \l__stex_sproof_spf_continues_tl
5561 \tl_clear:N \l__stex_sproof_spf_functions_tl
```

---

[8]EDNOTE: need an implementation for LaTeXML

```
5562 \tl_clear:N \l__stex_sproof_spf_method_tl
5563   \bool_set_false:N \l__stex_sproof_inc_counter_bool
5564 \keys_set:nn { stex / spf }{ #1 }
5565 }
```

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

```
5566 \str_set:Nn\c__stex_sproof_flow_str{inline}
```

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

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

pst@with@label   This environment manages[7] the path labeling of the proof steps in the description environment of the outermost `proof` environment. The argument is the label prefix up to now; which we cache in `\pst@label` (we need evaluate it first, since are in the right place now!). Then we increment the proof depth which is stored in `\count10` (lower counters are used by TeX for page numbering) and initialize the next level counter `\count\count10` with 1. In the end call for this environment, we just decrease the proof depth counter by 1 again.

```
5567 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
5568 \cs_new_protected:Npn \sproofnumber {
5569   \int_set:Nn \l_tmpa_int {1}
5570   \bool_while_do:nn {
5571     \int_compare_p:nNn {
5572       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5573     } > 0
5574   }{
5575     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
5576     \int_incr:N \l_tmpa_int
5577   }
5578 }
5579 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
5580   \int_set:Nn \l_tmpa_int {1}
5581   \bool_while_do:nn {
5582     \int_compare_p:nNn {
5583       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5584     } > 0
5585   }{
5586     \int_incr:N \l_tmpa_int
5587   }
5588   \int_compare:nNnF \l_tmpa_int = 1 {
5589     \int_decr:N \l_tmpa_int
5590   }
5591   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
5592     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1
```

---

[7]This gets the labeling right but only works 8 levels deep

```
5593        }
5594    }
5595
5596    \cs_new_protected:Npn \__stex_sproof_add_counter: {
5597        \int_set:Nn \l_tmpa_int {1}
5598        \bool_while_do:nn {
5599            \int_compare_p:nNn {
5600                \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5601            } > 0
5602        }{
5603            \int_incr:N \l_tmpa_int
5604        }
5605        \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 1 }
5606    }
5607
5608    \cs_new_protected:Npn \__stex_sproof_remove_counter: {
5609        \int_set:Nn \l_tmpa_int {1}
5610        \bool_while_do:nn {
5611            \int_compare_p:nNn {
5612                \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5613            } > 0
5614        }{
5615            \int_incr:N \l_tmpa_int
5616        }
5617        \int_decr:N \l_tmpa_int
5618        \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 0 }
5619    }
```

\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

```
5620    \def\sproof@box{
5621        \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
5622    }
5623    \def\sproofend{
5624        \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
5625            \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
5626        }
5627    }
```

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

spf@*@kw

```
5628    \def\spf@proofsketch@kw{Proof~Sketch}
5629    \def\spf@proof@kw{Proof}
5630    \def\spf@step@kw{Step}
```

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

For the other languages, we set up triggers

```
5631    \AddToHook{begindocument}{
5632        \ltx@ifpackageloaded{babel}{
5633            \makeatletter
5634            \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5635            \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5636                \input{sproof-ngerman.ldf}
```

```
5637        }
5638      \clist_if_in:NnT \l_tmpa_clist {finnish}{
5639        \input{sproof-finnish.ldf}
5640      }
5641      \clist_if_in:NnT \l_tmpa_clist {french}{
5642        \input{sproof-french.ldf}
5643      }
5644      \clist_if_in:NnT \l_tmpa_clist {russian}{
5645        \input{sproof-russian.ldf}
5646      }
5647      \makeatother
5648    }{}
5649 }
```

spfsketch

```
5650 \newcommand\spfsketch[2][]{
5651    \begingroup
5652    \let \premise \stex_proof_premise:
5653    \__stex_sproof_spf_args:n{#1}
5654    \stex_if_smsmode:TF {
5655      \str_if_empty:NF \spfid {
5656        \stex_ref_new_doc_target:n \spfid
5657      }
5658    }{
5659      \seq_clear:N \l_tmpa_seq
5660      \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5661        \tl_if_empty:nF{ ##1 }{
5662          \stex_get_symbol:n { ##1 }
5663          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5664            \l_stex_get_symbol_uri_str
5665          }
5666        }
5667      }
5668      \exp_args:Nnx
5669      \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
5670        \str_if_empty:NF \spftype {
5671          \stex_annotate_invisible:nnn{type}{\spftype}{}
5672        }
5673        \clist_set:No \l_tmpa_clist \spftype
5674        \tl_set:Nn \l_tmpa_tl {
5675          \titleemph{
5676            \tl_if_empty:NTF \spftitle {
5677              \spf@proofsketch@kw
5678            }{
5679              \spftitle
5680            }
5681          }:~
5682        }
5683        \clist_map_inline:Nn \l_tmpa_clist {
5684          \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5685            \tl_clear:N \l_tmpa_tl
5686          }
5687        }
5688        \str_if_empty:NF \spfid {
```

```
5689        \stex_ref_new_doc_target:n \spfid
5690      }
5691      \l_tmpa_tl #2 \sproofend
5692    }
5693  }
5694  \endgroup
5695  \stex_smsmode_do:
5696 }
5697
```

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

spfeq   This is very similar to \spfsketch, but uses a computation array[9][10]

```
5698 \newenvironment{spfeq}[2][]{
5699   \__stex_sproof_spf_args:n{#1}
5700   \let \premise \stex_proof_premise:
5701   \stex_if_smsmode:TF {
5702     \str_if_empty:NF \spfid {
5703       \stex_ref_new_doc_target:n \spfid
5704     }
5705   }{
5706     \seq_clear:N \l_tmpa_seq
5707     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5708       \tl_if_empty:nF{ ##1 }{
5709         \stex_get_symbol:n { ##1 }
5710         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5711           \l_stex_get_symbol_uri_str
5712         }
5713       }
5714     }
5715     \exp_args:Nnnx
5716     \begin{stex_annotate_env}{spfeq}{\seq_use:Nn \l_tmpa_seq {,}}
5717     \str_if_empty:NF \spftype {
5718       \stex_annotate_invisible:nnn{type}{\spftype}{}
5719     }
5720
5721     \clist_set:No \l_tmpa_clist \spftype
5722     \tl_clear:N \l_tmpa_tl
5723     \clist_map_inline:Nn \l_tmpa_clist {
5724       \tl_if_exist:cT {__stex_sproof_spfeq_##1_start:}{
5725         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_start:}}
5726       }
5727       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5728         \tl_set:Nn \l_tmpa_tl {\use:n{}}
5729       }
5730     }
5731     \tl_if_empty:NTF \l_tmpa_tl {
5732       \__stex_sproof_spfeq_start:
5733     }{
5734       \l_tmpa_tl
5735     }{~#2}
```

---

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

[10]EDNOTE: document above

```
5736        \str_if_empty:NF \spfid {
5737            \stex_ref_new_doc_target:n \spfid
5738        }
5739        \begin{displaymath}\begin{array}{rcll}
5740    }
5741    \stex_smsmode_do:
5742 }{
5743    \stex_if_smsmode:F {
5744        \end{array}\end{displaymath}
5745        \clist_set:No \l_tmpa_clist \spftype
5746        \tl_clear:N \l_tmpa_tl
5747        \clist_map_inline:Nn \l_tmpa_clist {
5748            \tl_if_exist:cT {__stex_sproof_spfeq_##1_end:}{
5749                \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_end:}}
5750            }
5751        }
5752        \tl_if_empty:NTF \l_tmpa_tl {
5753            \__stex_sproof_spfeq_end:
5754        }{
5755            \l_tmpa_tl
5756        }
5757        \end{stex_annotate_env}
5758    }
5759 }
5760
5761 \cs_new_protected:Nn \__stex_sproof_spfeq_start: {
5762    \titleemph{
5763        \tl_if_empty:NTF \spftitle {
5764            \spf@proof@kw
5765        }{
5766            \spftitle
5767        }
5768    }:
5769 }
5770 \cs_new_protected:Nn \__stex_sproof_spfeq_end: {\sproofend}
5771
5772 \newcommand\stexpatchspfeq[3][] {
5773    \str_set:Nx \l_tmpa_str{ #1 }
5774    \str_if_empty:NTF \l_tmpa_str {
5775        \tl_set:Nn \__stex_sproof_spfeq_start: { #2 }
5776        \tl_set:Nn \__stex_sproof_spfeq_end: { #3 }
5777    }{
5778        \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_start:\endcsname{ #2 }
5779        \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_end:\endcsname{ #3 }
5780    }
5781 }
5782
```

(*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.

```
5783 \newenvironment{sproof}[2][]{
```

```
5784    \let \premise \stex_proof_premise:
5785    \intarray_gzero:N \l__stex_sproof_counter_intarray
5786    \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
5787    \__stex_sproof_spf_args:n{#1}
5788    \stex_if_smsmode:TF {
5789      \str_if_empty:NF \spfid {
5790        \stex_ref_new_doc_target:n \spfid
5791      }
5792    }{
5793      \seq_clear:N \l_tmpa_seq
5794      \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5795        \tl_if_empty:nF{ ##1 }{
5796          \stex_get_symbol:n { ##1 }
5797          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5798            \l_stex_get_symbol_uri_str
5799          }
5800        }
5801      }
5802      \exp_args:Nnnx
5803      \begin{stex_annotate_env}{sproof}{\seq_use:Nn \l_tmpa_seq {,}}
5804      \str_if_empty:NF \spftype {
5805        \stex_annotate_invisible:nnn{type}{\spftype}{}
5806      }
5807
5808      \clist_set:No \l_tmpa_clist \spftype
5809      \tl_clear:N \l_tmpa_tl
5810      \clist_map_inline:Nn \l_tmpa_clist {
5811        \tl_if_exist:cT {__stex_sproof_sproof_##1_start:}{
5812          \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_start:}}
5813        }
5814        \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5815          \tl_set:Nn \l_tmpa_tl {\use:n{}}
5816        }
5817      }
5818      \tl_if_empty:NTF \l_tmpa_tl {
5819        \__stex_sproof_sproof_start:
5820      }{
5821        \l_tmpa_tl
5822      }{~#2}
5823      \str_if_empty:NF \spfid {
5824        \stex_ref_new_doc_target:n \spfid
5825      }
5826      \begin{description}
5827    }
5828    \stex_smsmode_do:
5829  }{
5830    \stex_if_smsmode:F{
5831      \end{description}
5832      \clist_set:No \l_tmpa_clist \spftype
5833      \tl_clear:N \l_tmpa_tl
5834      \clist_map_inline:Nn \l_tmpa_clist {
5835        \tl_if_exist:cT {__stex_sproof_sproof_##1_end:}{
5836          \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_end:}}
5837        }
```

```
5838        }
5839        \tl_if_empty:NTF \l_tmpa_tl {
5840          \__stex_sproof_sproof_end:
5841        }{
5842          \l_tmpa_tl
5843        }
5844        \end{stex_annotate_env}
5845      }
5846    }
5847
5848    \cs_new_protected:Nn \__stex_sproof_sproof_start: {
5849      \par\noindent\titleemph{
5850        \tl_if_empty:NTF \spftype {
5851          \spf@proof@kw
5852        }{
5853          \spftype
5854        }
5855      }:
5856    }
5857    \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
5858
5859    \newcommand\stexpatchproof[3][] {
5860      \str_set:Nx \l_tmpa_str{ #1 }
5861      \str_if_empty:NTF \l_tmpa_str {
5862        \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
5863        \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
5864      }{
5865        \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
5866        \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
5867      }
5868    }
```

\spfidea

```
5869    \newcommand\spfidea[2][]{
5870      \__stex_sproof_spf_args:n{#1}
5871      \titleemph{
5872        \tl_if_empty:NTF \spftype {Proof~Idea}{
5873          \spftype
5874        }:
5875      }~#2
5876      \sproofend
5877    }
```

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

The next two environments (proof steps) and comments, are mostly semantical, they take KeyVal arguments that specify their semantic role. In draft mode, they read these values and show them. If the surrounding proof had display=flow, then no new \item is generated, otherwise it is. In any case, the proof step number (at the current level) is incremented.

spfstep

```
5878    \newenvironment{spfstep}[1][]{
5879      \__stex_sproof_spf_args:n{#1}
5880      \stex_if_smsmode:TF {
```

```
5881      \str_if_empty:NF \spfid {
5882        \stex_ref_new_doc_target:n \spfid
5883      }
5884    }{
5885      \@in@omtexttrue
5886      \seq_clear:N \l_tmpa_seq
5887      \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5888        \tl_if_empty:nF{ ##1 }{
5889          \stex_get_symbol:n { ##1 }
5890          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5891            \l_stex_get_symbol_uri_str
5892          }
5893        }
5894      }
5895      \exp_args:Nnnx
5896      \begin{stex_annotate_env}{spfstep}{\seq_use:Nn \l_tmpa_seq {,}}
5897      \str_if_empty:NF \spftype {
5898        \stex_annotate_invisible:nnn{type}{\spftype}{}
5899      }
5900      \clist_set:No \l_tmpa_clist \spftype
5901      \tl_set:Nn \l_tmpa_tl {
5902        \item[\sproofnumber]
5903        \bool_set_true:N \l__stex_sproof_inc_counter_bool
5904      }
5905      \clist_map_inline:Nn \l_tmpa_clist {
5906        \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5907          \tl_clear:N \l_tmpa_tl
5908        }
5909      }
5910      \l_tmpa_tl
5911      \tl_if_empty:NF \spftitle {
5912        {(\titleemph{\spftitle})\enspace}
5913      }
5914      \str_if_empty:NF \spfid {
5915        \stex_ref_new_doc_target:n \spfid
5916      }
5917    }
5918    \stex_smsmode_do:
5919    \ignorespacesandpars
5920  }{
5921    \bool_if:NT \l__stex_sproof_inc_counter_bool {
5922      \__stex_sproof_inc_counter:
5923    }
5924    \stex_if_smsmode:F {
5925      \end{stex_annotate_env}
5926    }
5927  }
```

sproofcomment

```
5928  \newenvironment{sproofcomment}[1][]{
5929    \__stex_sproof_spf_args:n{#1}
5930    \clist_set:No \l_tmpa_clist \spftype
5931    \tl_set:Nn \l_tmpa_tl {
5932      \item[\sproofnumber]
```

```
5933        \bool_set_true:N \l__stex_sproof_inc_counter_bool
5934      }
5935      \clist_map_inline:Nn \l_tmpa_clist {
5936        \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5937          \tl_clear:N \l_tmpa_tl
5938        }
5939      }
5940      \l_tmpa_tl
5941    }{
5942      \bool_if:NT \l__stex_sproof_inc_counter_bool {
5943        \__stex_sproof_inc_counter:
5944      }
5945    }
```

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.

```
5946 \newenvironment{subproof}[2][]{
5947    \__stex_sproof_spf_args:n{#1}
5948    \stex_if_smsmode:TF{
5949      \str_if_empty:NF \spfid {
5950        \stex_ref_new_doc_target:n \spfid
5951      }
5952    }{
5953      \seq_clear:N \l_tmpa_seq
5954      \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5955        \tl_if_empty:nF{ ##1 }{
5956          \stex_get_symbol:n { ##1 }
5957          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5958            \l_stex_get_symbol_uri_str
5959          }
5960        }
5961      }
5962      \exp_args:Nnnx
5963      \begin{stex_annotate_env}{subproof}{\seq_use:Nn \l_tmpa_seq {,}}
5964      \str_if_empty:NF \spftype {
5965        \stex_annotate_invisible:nnn{type}{\spftype}{}
5966      }
5967
5968      \clist_set:No \l_tmpa_clist \spftype
5969      \tl_set:Nn \l_tmpa_tl {
5970        \item[\sproofnumber]
5971        \bool_set_true:N \l__stex_sproof_inc_counter_bool
5972      }
5973      \clist_map_inline:Nn \l_tmpa_clist {
5974        \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5975          \tl_clear:N \l_tmpa_tl
5976        }
5977      }
5978      \l_tmpa_tl
5979      \tl_if_empty:NF \spftitle {
5980        {(\titleemph{\spftitle})\enspace}
5981      }
```

```
5982        {~#2}
5983        \str_if_empty:NF \spfid {
5984          \stex_ref_new_doc_target:n \spfid
5985        }
5986      }
5987      \__stex_sproof_add_counter:
5988      \stex_smsmode_do:
5989    }{
5990      \__stex_sproof_remove_counter:
5991      \bool_if:NT \l__stex_sproof_inc_counter_bool {
5992        \__stex_sproof_inc_counter:
5993      }
5994      \stex_if_smsmode:F{
5995        \end{stex_annotate_env}
5996      }
5997    }
```

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

```
5998    \newenvironment{spfcases}[2][]{
5999      \tl_if_empty:nTF{#1}{
6000        \begin{subproof}[method=by-cases]{#2}
6001      }{
6002        \begin{subproof}[#1,method=by-cases]{#2}
6003      }
6004    }{
6005      \end{subproof}
6006    }
```

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

```
6007    \newenvironment{spfcase}[2][]{
6008      \__stex_sproof_spf_args:n{#1}
6009      \stex_if_smsmode:TF {
6010        \str_if_empty:NF \spfid {
6011          \stex_ref_new_doc_target:n \spfid
6012        }
6013      }{
6014        \seq_clear:N \l_tmpa_seq
6015        \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6016          \tl_if_empty:nF{ ##1 }{
6017            \stex_get_symbol:n { ##1 }
6018            \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6019              \l_stex_get_symbol_uri_str
6020            }
6021          }
6022        }
6023        \exp_args:Nnnx
6024        \begin{stex_annotate_env}{spfcase}{\seq_use:Nn \l_tmpa_seq {,}}
6025        \str_if_empty:NF \spftype {
6026          \stex_annotate_invisible:nnn{type}{\spftype}{}
6027        }
6028        \clist_set:No \l_tmpa_clist \spftype
6029        \tl_set:Nn \l_tmpa_tl {
6030          \item[\sproofnumber]
```

```
6031        \bool_set_true:N \l__stex_sproof_inc_counter_bool
6032      }
6033      \clist_map_inline:Nn \l_tmpa_clist {
6034        \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6035          \tl_clear:N \l_tmpa_tl
6036        }
6037      }
6038      \l_tmpa_tl
6039      \tl_if_empty:nF{#2}{
6040        \titleemph{#2}:~
6041      }
6042    }
6043    \__stex_sproof_add_counter:
6044    \stex_smsmode_do:
6045 }{
6046    \__stex_sproof_remove_counter:
6047    \bool_if:NT \l__stex_sproof_inc_counter_bool {
6048      \__stex_sproof_inc_counter:
6049    }
6050    \stex_if_smsmode:F{
6051      \clist_set:No \l_tmpa_clist \spftype
6052      \tl_set:Nn \l_tmpa_tl{\sproofend}
6053      \clist_map_inline:Nn \l_tmpa_clist {
6054        \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6055          \tl_clear:N \l_tmpa_tl
6056        }
6057      }
6058      \l_tmpa_tl
6059      \end{stex_annotate_env}
6060    }
6061 }
```

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

```
6062 \newcommand\spfcasesketch[3][]{
6063    \begin{spfcase}[#1]{#2}#3\end{spfcase}
6064 }
```

33.3   Justifications

We define the actions that are undertaken, when the keys for justifications are encountered. Here this is very simple, we just define an internal macro with the value, so that we can use it later.

```
6065 \keys_define:nn { stex / just }{
6066    id       .str_set_x:N  = \l__stex_sproof_just_id_str,
6067    method   .tl_set:N     = \l__stex_sproof_just_method_tl,
6068    premises .tl_set:N     = \l__stex_sproof_just_premises_tl,
6069    args     .tl_set:N     = \l__stex_sproof_just_args_tl
6070 }
```

The next three environments and macros are purely semantic, so we ignore the keyval
EdN:11   arguments for now and only display the content.[11]

---

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

228

justification

> 6071 `\newenvironment{justification}[1][]{}{}`

\premise

> 6072 `\newcommand\stex_proof_premise:[2][]{#2}`

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

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

> 6073 `\newcommand\justarg[2][]{#2}`
> 6074 ⟨/package⟩

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

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

Chapter 34

STEX
-Others Implementation

```
6075 ⟨*package⟩
6076
6077 %%%%%%%%%%%%    others.dtx    %%%%%%%%%%%%
6078
6079 ⟨@@=stex_others⟩
```

Warnings and error messages
```
6080    % None
```

\MSC   Math subject classifier

```
6081 \NewDocumentCommand \MSC {m} {
6082    % TODO
6083 }
```

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

Patching tikzinput, if loaded

```
6084 \@ifpackageloaded{tikzinput}{
6085    \RequirePackage{stex-tikzinput}
6086 }{}
6087
6088 \bool_if:NT \c_stex_persist_mode_bool {
6089    \input{\jobname.sms}
6090    \prop_if_exist:NT\c_stex_mathhub_main_manifest_prop{
6091       \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
6092          \l_tmpa_str
6093       \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str _manifest_prop }
6094          \c_stex_mathhub_main_manifest_prop
6095       \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
6096    }
6097 }
6098 ⟨/package⟩
```

Chapter 35

sTeX
-Metatheory Implementation

```
6099  ⟨∗package⟩
6100  ⟨@@=stex_modules⟩
6101
6102  %%%%%%%%%%%%   metatheory.dtx   %%%%%%%%%%%%
6103
6104  \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX/meta}
6105  \begingroup
6106  \stex_module_setup:nn{
6107    ns=\c_stex_metatheory_ns_str,
6108    meta=NONE
6109  }{Metatheory}
6110  \stex_reactivate_macro:N \symdecl
6111  \stex_reactivate_macro:N \notation
6112  \stex_reactivate_macro:N \symdef
6113  \ExplSyntaxOff
6114  \csname stex_suppress_html:n\endcsname{
6115    % is-a (a:A, a \in A, a is an A, etc.)
6116    \symdecl{isa}[args=ai]
6117    \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
6118    \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
6119    \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
6120
6121    % bind (\forall, \Pi, \lambda etc.)
6122    \symdecl{bind}[args=Bi]
6123    \notation{bind}[forall]{\comp\forall #1.\;#2}{##1 \comp, ##2}
6124    \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6125    \notation{bind}[depfun]{\comp( #1 \comp)\;\to\;} #2}{##1 \comp, ##2}
6126
6127    % implicit bind
6128    \symdef{implicitbind}[args=Bi]{\comp\prod_{#1}#2}{##1\comp,##2}
6129
6130    % dummy variable
6131    \symdecl{dummyvar}
6132    \notation{dummyvar}[underscore]{\comp\_}
6133    \notation{dummyvar}[dot]{\comp\cdot}
```

```
6134    \notation{dummyvar}[dash]{\comp{{\rm --}}}

6135

6136    %fromto (function space, Hom-set, implication etc.)
6137    \symdecl{fromto}[args=ai]
6138    \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
6139    \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}

6140

6141    % mapto (lambda etc.)
6142    %\symdecl{mapto}[args=Bi]
6143    %\notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
6144    %\notation{mapto}[lambda]{\comp\lambda #1 \comp.\; #2}{#1 \comp, #2}
6145    %\notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.\; #2}{#1 \comp, #2}

6146

6147    % function/operator application
6148    \symdecl{apply}[args=ia]
6149    \notation{apply}[prec=0;0x\infprec,parens]{#1 \comp( #2 \comp)}{##1 \comp, ##2}
6150    \notation{apply}[prec=0;0x\infprec,lambda]{#1 \; #2 }{##1 \; ##2}

6151

6152    % collection of propositions/booleans/truth values
6153    \symdecl{prop}[name=proposition]
6154    \notation{prop}[prop]{\comp{{\rm prop}}}
6155    \notation{prop}[BOOL]{\comp{{\rm BOOL}}}

6156

6157    \symdecl{judgmentholds}[args=1]
6158    \notation{judgmentholds}[vdash,op=\vdash]{\comp\vdash\; #1}

6159

6160    % sequences
6161    \symdecl{seqtype}[args=1]
6162    \notation{seqtype}[kleene]{#1^{\comp\ast}}

6163

6164    \symdecl{seqexpr}[args=a]
6165    \notation{seqexpr}[angle,prec=nobrackets]{\comp\langle #1\comp\rangle}{##1\comp,##2}

6166

6167    \symdef{sequence-index}[args=2,li,prec=nobrackets]{{#1}_{#2}}
6168    \notation{sequence-index}[ui,prec=nobrackets]{{#1}^{#2}}

6169

6170    \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{,\ellipses}}{##1\comp,##2}
6171    \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{,\ellipses,}#2}{##1\comp,##2}
6172    \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{,\ellipses,}#2\comp{,\ellipses,}#

6173

6174    % letin (``let'', local definitions, variable substitution)
6175    \symdecl{letin}[args=bii]
6176    \notation{letin}[let]{\comp{{\rm let}}\;#1\comp{=}#2\;\comp{{\rm in}}\;#3}
6177    \notation{letin}[subst]{#3 \comp[ #1 \comp/ #2 \comp]}
6178    \notation{letin}[frac]{#3 \comp[ \frac{#2}{#1} \comp]}

6179

6180    % structures
6181    \symdecl*{module-type}[args=1]
6182    \notation{module-type}{\comp{\mathtt{MOD}} #1}
6183    \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
6184    \notation{mathstruct}[angle,prec=nobrackets]{\comp\langle #1 \comp\rangle}{##1 \comp, ##2}

6185

6186    % objects
6187    \symdecl{object}
```

232

```
6188    \notation{object}{\comp{\mathtt{OBJECT}}}

6189

6190  }
6191    \ExplSyntaxOn
6192    \stex_add_to_current_module:n{
6193      \let\nappa\apply
6194      \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
6195      \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
6196      \def\livar{\csname sequence-index\endcsname[li]}
6197      \def\uivar{\csname sequence-index\endcsname[ui]}
6198      \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
6199      \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
6200      \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
6201    }
6202  \__stex_modules_end_module:
6203  \endgroup

6204  ⟨/package⟩
```

Chapter 36

Tikzinput Implementation

```
6205 ⟨*package⟩
6206
6207 %%%%%%%%%%%%   tikzinput.dtx   %%%%%%%%%%%%
6208
6209 \ProvidesExplPackage{tikzinput}{2022/02/26}{3.0.1}{tikzinput package}
6210 \RequirePackage{l3keys2e}
6211
6212 \keys_define:nn { tikzinput } {
6213   image    .bool_set:N   = \c_tikzinput_image_bool,
6214   image    .default:n    = false ,
6215   unknown   .code:n       = {}
6216 }
6217
6218 \ProcessKeysOptions { tikzinput }
6219
6220 \bool_if:NTF \c_tikzinput_image_bool {
6221   \RequirePackage{graphicx}
6222
6223   \providecommand\usetikzlibrary[]{}
6224   \newcommand\tikzinput[2][]{\includegraphics[#1]{#2}}
6225 }{
6226   \RequirePackage{tikz}
6227   \RequirePackage{standalone}
6228
6229   \newcommand \tikzinput [2] [] {
6230     \setkeys{Gin}{#1}
6231     \ifx \Gin@ewidth \Gin@exclamation
6232       \ifx \Gin@eheight \Gin@exclamation
6233         \input { #2 }
6234       \else
6235         \resizebox{!}{ \Gin@eheight }{
6236           \input { #2 }
6237         }
6238       \fi
6239     \else
6240       \ifx \Gin@eheight \Gin@exclamation
6241         \resizebox{ \Gin@ewidth }{!}{
6242           \input { #2 }
```

```
6243            }
6244         \else
6245           \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
6246             \input { #2 }
6247           }
6248         \fi
6249       \fi
6250     }
6251 }
6252
6253 \newcommand \ctikzinput [2] [] {
6254    \begin{center}
6255      \tikzinput [#1] {#2}
6256    \end{center}
6257 }
6258
6259 \@ifpackageloaded{stex}{
6260    \RequirePackage{stex-tikzinput}
6261 }{}
6262
6263 ⟨/package⟩
6264 ⟨∗stex⟩
6265 \ProvidesExplPackage{stex-tikzinput}{2022/02/26}{3.0.1}{stex-tikzinput}
6266 \RequirePackage{stex}
6267 \RequirePackage{tikzinput}
6268
6269 \newcommand\mhtikzinput[2][]{%
6270    \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
6271    \stex_in_repository:nn\Gin@mhrepos{
6272      \tikzinput[#1]{\mhpath{##1}{#2}}
6273    }
6274 }
6275 \newcommand\cmhtikzinput[2][]{\begin{center}\mhtikzinput[#1]{#2}\end{center}}
6276 ⟨/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

6277 ⟨∗package⟩
6278 ⟨@@=document_structure⟩
6279 \ProvidesExplPackage{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure}
6280 \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).

```
6281
6282 \keys_define:nn{ document-structure }{
6283   class       .str_set_x:N  = \c_document_structure_class_str,
6284   topsect     .str_set_x:N  = \c_document_structure_topsect_str,,
6285   unknown     .code:n       = {
6286     \PassOptionsToClass{\CurrentOption}{stex}
6287     \PassOptionsToClass{\CurrentOption}{tikzinput}
6288   }
6289 %   showignores .bool_set:N   = \c_document_structure_showignores_bool,
6290 }
6291 \ProcessKeysOptions{ document-structure }
6292 \str_if_empty:NT \c_document_structure_class_str {
6293   \str_set:Nn \c_document_structure_class_str {article}
6294 }
6295 \str_if_empty:NT \c_document_structure_topsect_str {
6296   \str_set:Nn \c_document_structure_topsect_str {section}
6297 }
```

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

```
6298 \RequirePackage{xspace}
6299 \RequirePackage{comment}
6300 \RequirePackage{stex}
6301 \AddToHook{begindocument}{
```

```
6302  \ltx@ifpackageloaded{babel}{
6303      \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6304      \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6305          \makeatletter\input{document-structure-ngerman.ldf}\makeatother
6306      }
6307  }{}
6308  }
```

\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.

```
6309  \int_new:N \l_document_structure_section_level_int
6310  \str_case:VnF \c_document_structure_topsect_str {
6311    {part}{
6312        \int_set:Nn \l_document_structure_section_level_int {0}
6313    }
6314    {chapter}{
6315        \int_set:Nn \l_document_structure_section_level_int {1}
6316    }
6317  }{
6318    \str_case:VnF \c_document_structure_class_str {
6319      {book}{
6320          \int_set:Nn \l_document_structure_section_level_int {0}
6321      }
6322      {report}{
6323          \int_set:Nn \l_document_structure_section_level_int {0}
6324      }
6325    }{
6326      \int_set:Nn \l_document_structure_section_level_int {2}
6327    }
6328  }
```

37.2   Document Structure

The structure of the document is given by the `omgroup` environment just like in OMDoc. The hierarchy is adjusted automatically according to the LaTeX class in effect.

\currentsectionlevel   For the `\currentsectionlevel` and `\Currentsectionlevel` macros we use an internal macro `\current@section@level` that only contains the keyword (no markup). We initialize it with "document" as a default. In the generated OMDoc, we only generate a text

EdN:12   element of class `omdoc_currentsectionlevel`, wich will be instantiated by CSS later.[12]

```
6329  \def\current@section@level{document}%
6330  \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
6331  \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

\skipomgroup

```
6332  \cs_new_protected:Npn \skipomgroup {
```

---

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

```
6333    \ifcase\l_document_structure_section_level_int
6334    \or\stepcounter{part}
6335    \or\stepcounter{chapter}
6336    \or\stepcounter{section}
6337    \or\stepcounter{subsection}
6338    \or\stepcounter{subsubsection}
6339    \or\stepcounter{paragraph}
6340    \or\stepcounter{subparagraph}
6341    \fi
6342 }
```

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

blindfragment

```
6343 \newcommand\at@begin@blindomgroup[1]{}
6344 \newenvironment{blindfragment}
6345 {
6346    \int_incr:N\l_document_structure_section_level_int
6347    \at@begin@blindomgroup\l_document_structure_section_level_int
6348 }{}
```

\omgroup@nonum   convenience macro: \omgroup@nonum{⟨*level*⟩}{⟨*title*⟩} makes an unnumbered sectioning with title ⟨*title*⟩ at level ⟨*level*⟩.

```
6349 \newcommand\omgroup@nonum[2]{
6350    \ifx\hyper@anchor\@undefined\else\phantomsection\fi
6351    \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
6352 }
```

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

\omgroup@num   convenience macro: \omgroup@nonum{⟨*level*⟩}{⟨*title*⟩} makes numbered sectioning with title ⟨*title*⟩ at level ⟨*level*⟩. We have to check the short key was given in the omgroup environment and – if it is use it. But how to do that depends on whether the rdfmeta package has been loaded. In the end we call \sref@label@id to enable crossreferencing.

```
6353 \newcommand\omgroup@num[2]{
6354    \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
6355       \@nameuse{#1}{#2}
6356    }{
6357       \cs_if_exist:NTF\rdfmeta@sectioning{
6358          \@nameuse{rdfmeta@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
6359       }{
6360          \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
6361       }
6362    }
6363 %\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\omgroup@id
6364 }
```

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

sfragment

```
6365 \keys_define:nn { document-structure / omgroup }{
6366    id           .str_set_x:N = \l__document_structure_omgroup_id_str,
6367    date         .str_set_x:N = \l__document_structure_omgroup_date_str,
6368    creators     .clist_set:N = \l__document_structure_omgroup_creators_clist,
```

```
6369    contributors  .clist_set:N = \l__document_structure_omgroup_contributors_clist,
6370    srccite       .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
6371    type          .tl_set:N    = \l__document_structure_omgroup_type_tl,
6372    short         .tl_set:N    = \l__document_structure_omgroup_short_tl,
6373    display       .tl_set:N    = \l__document_structure_omgroup_display_tl,
6374    intro         .tl_set:N    = \l__document_structure_omgroup_intro_tl,
6375    imports       .tl_set:N    = \l__document_structure_omgroup_imports_tl,
6376    loadmodules   .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
6377  }
6378  \cs_new_protected:Nn \__document_structure_omgroup_args:n {
6379    \str_clear:N \l__document_structure_omgroup_id_str
6380    \str_clear:N \l__document_structure_omgroup_date_str
6381    \clist_clear:N \l__document_structure_omgroup_creators_clist
6382    \clist_clear:N \l__document_structure_omgroup_contributors_clist
6383    \tl_clear:N \l__document_structure_omgroup_srccite_tl
6384    \tl_clear:N \l__document_structure_omgroup_type_tl
6385    \tl_clear:N \l__document_structure_omgroup_short_tl
6386    \tl_clear:N \l__document_structure_omgroup_display_tl
6387    \tl_clear:N \l__document_structure_omgroup_imports_tl
6388    \tl_clear:N \l__document_structure_omgroup_intro_tl
6389    \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
6390    \keys_set:nn { document-structure / omgroup } { #1 }
6391  }
```

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

```
6392  \newif\if@mainmatter\@mainmattertrue
6393  \newcommand\at@begin@omgroup[3][]{}
```

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

```
6394  \keys_define:nn { document-structure / sectioning }{
6395    name    .str_set_x:N  = \l__document_structure_sect_name_str    ,
6396    ref     .str_set_x:N  = \l__document_structure_sect_ref_str     ,
6397    clear   .bool_set:N   = \l__document_structure_sect_clear_bool ,
6398    clear   .default:n    = {true}                       ,
6399    num     .bool_set:N   = \l__document_structure_sect_num_bool   ,
6400    num     .default:n    = {true}
6401  }
6402  \cs_new_protected:Nn \__document_structure_sect_args:n {
6403    \str_clear:N \l__document_structure_sect_name_str
6404    \str_clear:N \l__document_structure_sect_ref_str
6405    \bool_set_false:N \l__document_structure_sect_clear_bool
6406    \bool_set_false:N \l__document_structure_sect_num_bool
6407    \keys_set:nn { document-structure / sectioning } { #1 }
6408  }
6409  \newcommand\omdoc@sectioning[3][]{
6410    \__document_structure_sect_args:n {#1 }
6411    \let\omdoc@sect@name\l__document_structure_sect_name_str
6412    \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
6413    \if@mainmatter% numbering not overridden by frontmatter, etc.
6414      \bool_if:NTF \l__document_structure_sect_num_bool {
6415        \omgroup@num{#2}{#3}
```

239

```
6416       }{
6417         \omgroup@nonum{#2}{#3}
6418       }
6419       \def\current@section@level{\omdoc@sect@name}
6420     \else
6421       \omgroup@nonum{#2}{#3}
6422     \fi
6423 }% 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.

```
6424 \newcommand\omgroup@redefine@addtocontents[1]{%
6425 %\edef\__document_structureimport{#1}%
6426 %\@for\@I:=\__document_structureimport\do{%
6427 %\edef\@path{\csname module@\@I  @path\endcsname}%
6428 %\@ifundefined{tf@toc}\relax%
6429 %     {\protected@write\tf@toc{}{\string\@requiremodules{\@path}}}}}
6430 %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
6431 %\def\addcontentsline##1##2##3{%
6432 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}
6433 %\else% hyperref.sty not loaded
6434 %\def\addcontentsline##1##2##3{%
6435 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}{
6436 %\fi
6437 }% hypreref.sty loaded?
```

now the `omgroup` environment itself. This takes care of the table of contents via the helper macro above and then selects the appropriate sectioning command from `article.cls`. It also registeres the current level of omgroups in the `\omgroup@level` counter.

```
6438 \newenvironment{sfragment}[2][]% keys, title
6439 {
6440   \__document_structure_omgroup_args:n { #1 }%\sref@target%
```

If the `loadmodules` key is set on `\begin{sfragment}`, we redefine the `\addcontetsline` macro that determines how the sectioning commands below construct the entries for the table of contents.

```
6441     \stex_csl_to_imports:No \usemodule \l__document_structure_omgroup_imports_tl
6442
6443     \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
6444       \omgroup@redefine@addtocontents{
6445         %\@ifundefined{module@id}\used@modules%
6446         %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}
6447       }
6448     }
```

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

```
6449     \int_incr:N\l_document_structure_section_level_int
6450     \ifcase\l_document_structure_section_level_int
6451       \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
6452       \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
6453       \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
6454       \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
6455       \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
6456       \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#
6457       \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{paragr
```

```
6458    \fi
6459    \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
6460    \str_if_empty:NF \l__document_structure_omgroup_id_str {
6461      \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
6462    }
6463  }% for customization
6464  {}
```

and finally, we localize the sections

```
6465  \newcommand\omdoc@part@kw{Part}
6466  \newcommand\omdoc@chapter@kw{Chapter}
6467  \newcommand\omdoc@section@kw{Section}
6468  \newcommand\omdoc@subsection@kw{Subsection}
6469  \newcommand\omdoc@subsubsection@kw{Subsubsection}
6470  \newcommand\omdoc@paragraph@kw{paragraph}
6471  \newcommand\omdoc@subparagraph@kw{subparagraph}
```

37.3   Front and Backmatter

Index markup is provided by the `omtext` package [Koh20c], so in the `document-structure` package we only need to supply the corresponding `\printindex` command, if it is not already defined

`\printindex`

```
6472  \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` environments, we save their behavior (possibly defining it) in `orig@*matter` macros and make them undefined (so that we can define the environments).

```
6473  \cs_if_exist:NTF\frontmatter{
6474    \let\__document_structure_orig_frontmatter\frontmatter
6475    \let\frontmatter\relax
6476  }{
6477    \tl_set:Nn\__document_structure_orig_frontmatter{
6478      \clearpage
6479      \@mainmatterfalse
6480      \pagenumbering{roman}
6481    }
6482  }
6483  \cs_if_exist:NTF\backmatter{
6484    \let\__document_structure_orig_backmatter\backmatter
6485    \let\backmatter\relax
6486  }{
6487    \tl_set:Nn\__document_structure_orig_backmatter{
6488      \clearpage
6489      \@mainmatterfalse
6490      \pagenumbering{roman}
6491    }
6492  }
```

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, otherwise we define it.

```
6493 \newenvironment{frontmatter}{
6494   \__document_structure_orig_frontmatter
6495 }{
6496   \cs_if_exist:NTF\mainmatter{
6497     \mainmatter
6498   }{
6499     \clearpage
6500     \@mainmattertrue
6501     \pagenumbering{arabic}
6502   }
6503 }
```

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

```
6504 \newenvironment{backmatter}{
6505   \__document_structure_orig_backmatter
6506 }{
6507   \cs_if_exist:NTF\mainmatter{
6508     \mainmatter
6509   }{
6510     \clearpage
6511     \@mainmattertrue
6512     \pagenumbering{arabic}
6513   }
6514 }
```

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

```
6515 \@mainmattertrue\pagenumbering{arabic}
```

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

```
6516 \def \c__document_structure_document_str{document}
6517 \newcommand\afterprematurestop{}
6518 \def\prematurestop@endomgroup{
6519   \unless\ifx\@currenvir\c__document_structure_document_str
6520     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafte
6521     \expandafter\prematurestop@endomgroup
6522   \fi
6523 }
6524 \providecommand\prematurestop{
6525   \message{Stopping~sTeX~processing~prematurely}
6526   \prematurestop@endomgroup
6527   \afterprematurestop
6528   \end{document}
6529 }
```

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

37.4 Global Variables

\setSGvar set a global variable

```
6530 \RequirePackage{etoolbox}
6531 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}
```

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

\useSGvar use a global variable

```
6532 \newrobustcmd\useSGvar[1]{%
6533   \@ifundefined{sTeX@Gvar@#1}
6534   {\PackageError{document-structure}
6535     {The sTeX Global variable #1 is undefined}
6536     {set it with \protect\setSGvar}}
6537 \@nameuse{sTeX@Gvar@#1}}
```

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

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

```
6538 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
6539   \@ifundefined{sTeX@Gvar@#1}
6540   {\PackageError{document-structure}
6541     {The sTeX Global variable #1 is undefined}
6542     {set it with \protect\setSGvar}}
6543   {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}
```

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

Chapter 38

NotesSlides – Implementation

38.1 Class and Package Options

We define some Package Options and switches for the `notesslides` class and activate them by passing them on to `beamer.cls` and `omdoc.cls` and the `notesslides` package. We pass the `nontheorem` option to the `statements` package when we are not in notes mode, since the `beamer` package has its own (overlay-aware) theorem environments.

```
6544  ⟨∗cls⟩
6545  ⟨@@=notesslides⟩
6546  \ProvidesExplClass{notesslides}{2022/02/28}{3.1.0}{notesslides Class}
6547  \RequirePackage{l3keys2e}
6548
6549  \keys_define:nn{notesslides / cls}{
6550    class   .str_set_x:N = \c__notesslides_class_str,
6551    notes   .bool_set:N  = \c__notesslides_notes_bool ,
6552    slides  .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
6553    docopt  .str_set_x:N = \c__notesslides_docopt_str,
6554    unknown .code:n      = {
6555      \PassOptionsToPackage{\CurrentOption}{document-structure}
6556      \PassOptionsToClass{\CurrentOption}{beamer}
6557      \PassOptionsToPackage{\CurrentOption}{notesslides}
6558    }
6559  }
6560  \ProcessKeysOptions{ notesslides / cls }
6561
6562
6563  \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{book}{
6564    \PassOptionsToPackage{defaulttopsec=part}{notesslides}
6565  }
6566  \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{report}{
6567    \PassOptionsToPackage{defaulttopsec=part}{notesslides}
6568  }
6569
6570
6571
6572
6573  \bool_if:NTF \c__notesslides_notes_bool {
6574    \PassOptionsToPackage{notes=true}{notesslides}
```

```
6575  }{
6576      \PassOptionsToPackage{notes=false}{notesslides}
6577  }
6578  ⟨/cls⟩
```

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

```
6579  ⟨*package⟩
6580  \ProvidesExplPackage{notesslides}{2022/02/28}{3.1.0}{notesslides Package}
6581  \RequirePackage{l3keys2e}
6582
6583  \keys_define:nn{notesslides / pkg}{
6584      topsect         .str_set_x:N  = \c__notesslides_topsect_str,
6585      defaulttopsect  .str_set_x:N  = \c__notesslides_defaulttopsec_str,
6586      notes           .bool_set:N   = \c__notesslides_notes_bool ,
6587      slides          .code:n       = { \bool_set_false:N \c__notesslides_notes_bool },
6588      sectocframes    .bool_set:N   = \c__notesslides_sectocframes_bool ,
6589      frameimages     .bool_set:N   = \c__notesslides_frameimages_bool ,
6590      fiboxed         .bool_set:N   = \c__notesslides_fiboxed_bool ,
6591      noproblems      .bool_set:N   = \c__notesslides_noproblems_bool,
6592      unknown         .code:n       = {
6593        \PassOptionsToClass{\CurrentOption}{stex}
6594        \PassOptionsToClass{\CurrentOption}{tikzinput}
6595      }
6596  }
6597  \ProcessKeysOptions{ notesslides / pkg }
6598  \newif\ifnotes
6599  \bool_if:NTF \c__notesslides_notes_bool {
6600      \notestrue
6601  }{
6602      \notesfalse
6603  }
6604
```

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

```
6605  \str_if_empty:NTF \c__notesslides_topsect_str {
6606      \str_set_eq:NN \__notesslidestopsect \c__notesslides_defaulttopsec_str
6607  }{
6608      \str_set_eq:NN \__notesslidestopsect \c__notesslides_topsect_str
6609  }
6610  ⟨/package⟩
```

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

```
6611  ⟨*cls⟩
6612  \bool_if:NTF \c__notesslides_notes_bool {
6613      \str_if_empty:NT \c__notesslides_class_str {
6614        \str_set:Nn \c__notesslides_class_str {article}
6615      }
6616      \exp_after:wN\LoadClass\exp_after:wN[\c__notesslides_docopt_str]
6617        {\c__notesslides_class_str}
6618  }{
6619      \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
6620      \newcounter{Item}
6621      \newcounter{paragraph}
```

```
6622    \newcounter{subparagraph}
6623    \newcounter{Hfootnote}
6624 }
6625 \RequirePackage{document-structure}
```

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

```
6626 \RequirePackage{notesslides}
6627 ⟨/cls⟩
```

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

```
6628 ⟨*package⟩
6629 \bool_if:NT \c__notesslides_notes_bool {
6630    \RequirePackage{a4wide}
6631    \RequirePackage{marginnote}
6632    \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
6633    \RequirePackage{mdframed}
6634    \RequirePackage[noxcolor,noamsthm]{beamerarticle}
6635    \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
6636 }
6637 \RequirePackage{stex-tikzinput}
6638 \RequirePackage{etoolbox}
6639 \RequirePackage{amssymb}
6640 \RequirePackage{amsmath}
6641 \RequirePackage{comment}
6642 \RequirePackage{textcomp}
6643 \RequirePackage{url}
6644 \RequirePackage{graphicx}
6645 \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.[13]

```
6646 \bool_if:NT \c__notesslides_notes_bool {
6647    \renewcommand\usetheme[2][]{\usepackage[#1]{beamernotestheme#2}}
6648 }
6649
6650
6651 \NewDocumentCommand \libusetheme {O{} m} {
6652    \bool_if:NTF \c__notesslides_notes_bool {
6653       \libusepackage[#1]{beamernotestheme#2}
6654    }{
6655       \libusepackage[#1]{beamertheme#2}
6656    }
6657 }
```

---

[13]EDNOTE: MK: This is not ideal, but I am not sure that I want to be able to provide the full theme functionality there.

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

```
6658 \newcounter{slide}
6659 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
6660 \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.

```
6661 \bool_if:NTF \c__notesslides_notes_bool {
6662   \renewenvironment{note}{\ignorespaces}{}
6663 }{
6664   \excludecomment{note}
6665 }
```

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.

```
6666 \bool_if:NT \c__notesslides_notes_bool {
6667   \newlength{\slideframewidth}
6668   \setlength{\slideframewidth}{1.5pt}
```

frame   We first define the keys.

```
6669   \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
6670     \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
6671       \bool_set_true:N #1
6672     }{
6673       \bool_set_false:N #1
6674     }
6675   }
6676   \keys_define:nn{notesslides / frame}{
6677     label                 .str_set_x:N = \l__notesslides_frame_label_str,
6678     allowframebreaks   .code:n      = {
6679       \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
6680     },
6681     allowdisplaybreaks  .code:n      = {
6682       \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
6683     },
6684     fragile             .code:n      = {
6685       \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
6686     },
6687     shrink              .code:n      = {
6688       \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
6689     },
6690     squeeze             .code:n      = {
6691       \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
6692     },
6693     t                   .code:n      = {
6694       \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
6695     },
6696   }
6697   \cs_new_protected:Nn \__notesslides_frame_args:n {
6698     \str_clear:N \l__notesslides_frame_label_str
```

```
6699        \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
6700        \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
6701        \bool_set_true:N \l__notesslides_frame_fragile_bool
6702        \bool_set_true:N \l__notesslides_frame_shrink_bool
6703        \bool_set_true:N \l__notesslides_frame_squeeze_bool
6704        \bool_set_true:N \l__notesslides_frame_t_bool
6705        \keys_set:nn { notesslides / frame }{ #1 }
6706    }
```

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

```
6707    \renewenvironment{frame}[1][]{
6708        \__notesslides_frame_args:n{#1}
6709        \sffamily
6710        \stepcounter{slide}
6711        \def\@currentlabel{\theslide}
6712        \str_if_empty:NF \l__notesslides_frame_label_str {
6713            \label{\l__notesslides_frame_label_str}
6714        }
```

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

```
6715        \def\itemize@level{outer}
6716        \def\itemize@outer{outer}
6717        \def\itemize@inner{inner}
6718        \renewcommand\newpage{\addtocounter{framenumber}{1}}
6719        \newcommand\metakeys@show@keys[2]{\marginnote{{\scriptsize ##2}}}
6720        \renewenvironment{itemize}{
6721            \ifx\itemize@level\itemize@outer
6722                \def\itemize@label{$\rhd$}
6723            \fi
6724            \ifx\itemize@level\itemize@inner
6725                \def\itemize@label{$\scriptstyle\rhd$}
6726            \fi
6727            \begin{list}
6728            {\itemize@label}
6729            {\setlength{\labelsep}{.3em}
6730             \setlength{\labelwidth}{.5em}
6731             \setlength{\leftmargin}{1.5em}
6732            }
6733            \edef\itemize@level{\itemize@inner}
6734        }{
6735            \end{list}
6736        }
```

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

```
6737        \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth
6738    }{
6739        \medskip\miko@slidelabel\end{mdframed}
6740    }
```

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

\frametitle

```
6741        \renewcommand{\frametitle}[1]{{\Large\bf\sf\color{blue}{#1}}\medskip}
6742    }
```

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

248

\pause   ¹⁴

```
6743 \bool_if:NT \c__notesslides_notes_bool {
6744   \newcommand\pause{}
6745 }
```

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

nparagraph

```
6746 \bool_if:NTF \c__notesslides_notes_bool {
6747   \newenvironment{nparagraph}[1][]{\begin{sparagraph}[#1]}{\end{sparagraph}}
6748 }{
6749   \excludecomment{nparagraph}
6750 }
```

nfragment

```
6751 \bool_if:NTF \c__notesslides_notes_bool {
6752   \newenvironment{nfragment}[2][]{\begin{sfragment}[#1]{#2}}{\end{sfragment}}
6753 }{
6754   \excludecomment{nfragment}
6755 }
```

ndefinition

```
6756 \bool_if:NTF \c__notesslides_notes_bool {
6757   \newenvironment{ndefinition}[1][]{\begin{sdefinition}[#1]}{\end{sdefinition}}
6758 }{
6759   \excludecomment{ndefinition}
6760 }
```

nassertion

```
6761 \bool_if:NTF \c__notesslides_notes_bool {
6762   \newenvironment{nassertion}[1][]{\begin{sassertion}[#1]}{\end{sassertion}}
6763 }{
6764   \excludecomment{nassertion}
6765 }
```

nsproof

```
6766 \bool_if:NTF \c__notesslides_notes_bool {
6767   \newenvironment{nproof}[2][]{\begin{sproof}[#1]{#2}}{\end{sproof}}
6768 }{
6769   \excludecomment{nproof}
6770 }
```

nexample

```
6771 \bool_if:NTF \c__notesslides_notes_bool {
6772   \newenvironment{nexample}[1][]{\begin{sexample}[#1]}{\end{sexample}}
6773 }{
6774   \excludecomment{nexample}
6775 }
```

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

```
6776 \def\inputref@preskip{\smallskip}
6777 \def\inputref@postskip{\medskip}
```

---

¹⁴EDNOTE: MK: fake it in notes mode for now

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

\inputref*

```
6778 \let\orig@inputref\inputref
6779 \def\inputref{\@ifstar\ninputref\orig@inputref}
6780 \newcommand\ninputref[2][]{
6781   \bool_if:NT \c__notesslides_notes_bool {
6782     \orig@inputref[#1]{#2}
6783   }
6784 }
```

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

38.3   Header and Footer Lines

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

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

```
6785 \newlength{\slidelogoheight}
6786
6787 \bool_if:NTF \c__notesslides_notes_bool {
6788   \setlength{\slidelogoheight}{.4cm}
6789 }{
6790   \setlength{\slidelogoheight}{1cm}
6791 }
6792 \newsavebox{\slidelogo}
6793 \sbox{\slidelogo}{\sTeX}
6794 \newrobustcmd{\setslidelogo}[1]{
6795   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
6796 }
```

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

\setsource   \source stores the writer's name. By default it is *Michael Kohlhase* since he is the main user and designer of this package. \setsource{⟨*name*⟩} can change the writer's name.

```
6797 \def\source{Michael Kohlhase}% customize locally
6798 \newrobustcmd{\setsource}[1]{\def\source{#1}}
```

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

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

```
6799 \def\copyrightnotice{\footnotesize\copyright :\hspace{.3ex}{\source}}
6800 \newsavebox{\cclogo}
6801 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{stex-cc_somerights}}
6802 \newif\ifcchref\cchreffalse
6803 \AtBeginDocument{
6804   \@ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
6805 }
```

```
6806 \def\licensing{
6807   \ifccref
6808     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
6809   \else
6810     {\usebox{\cclogo}}
6811   \fi
6812 }
6813 \newrobustcmd{\setlicensing}[2][]{
6814   \def\@url{#1}
6815   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
6816   \ifx\@url\@empty
6817     \def\licensing{{\usebox{\cclogo}}}
6818   \else
6819     \def\licensing{
6820       \ifccref
6821       \href{#1}{\usebox{\cclogo}}
6822       \else
6823       {\usebox{\cclogo}}
6824       \fi
6825     }
6826   \fi
6827 }
```

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

`\slidelabel`   Now, we set up the slide label for the `article` mode.[15]

```
6828 \newrobustcmd\miko@slidelabel{
6829   \vbox to \slidelogoheight{
6830     \vss\hbox to \slidewidth
6831     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
6832   }
6833 }
```

(*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.

```
6834 \def\Gin@mhrepos{}
6835 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
6836 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
6837 \newrobustcmd\frameimage[2][]{
6838   \stepcounter{slide}
6839   \bool_if:NT \c__notesslides_frameimages_bool {
6840     \def\Gin@ewidth{}\setkeys{Gin}{#1}
6841     \bool_if:NF \c__notesslides_notes_bool { \vfill }
6842     \begin{center}
6843       \bool_if:NTF \c__notesslides_fiboxed_bool {
6844         \fbox{
6845           \ifx\Gin@ewidth\@empty
6846             \ifx\Gin@mhrepos\@empty
```

---

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

```
6847            \mhgraphics[width=\slidewidth,#1]{#2}
6848          \else
6849            \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
6850          \fi
6851        \else% Gin@ewidth empty
6852          \ifx\Gin@mhrepos\@empty
6853            \mhgraphics[#1]{#2}
6854          \else
6855            \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
6856          \fi
6857        \fi% Gin@ewidth empty
6858      }
6859    }{
6860      \ifx\Gin@ewidth\@empty
6861        \ifx\Gin@mhrepos\@empty
6862          \mhgraphics[width=\slidewidth,#1]{#2}
6863        \else
6864          \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
6865        \fi
6866        \ifx\Gin@mhrepos\@empty
6867          \mhgraphics[#1]{#2}
6868        \else
6869          \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
6870        \fi
6871      \fi% Gin@ewidth empty
6872    }
6873    \end{center}
6874    \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
6875    \bool_if:NF \c__notesslides_notes_bool { \vfill }
6876  }
6877 } % ifmks@sty@frameimages
```

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

38.5   Colors and Highlighting

We first specify sans serif fonts as the default.

```
6878 \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

```
6879 \AddToHook{begindocument}{
6880   \definecolor{green}{rgb}{0,.5,0}
6881   \definecolor{purple}{cmyk}{.3,1,0,.17}
6882 }
```

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.

```
6883 % \def\STpresent#1{\textcolor{blue}{#1}}
6884 \def\defemph#1{{\textcolor{magenta}{#1}}}
6885 \def\symrefemph#1{{\textcolor{cyan}{#1}}}
```

```
6886  \def\compemph#1{{\textcolor{blue}{#1}}}
6887  \def\titleemph#1{{\textcolor{blue}{#1}}}
6888  \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.

```
6889  \pgfdeclareimage[width=.8em]{miko@small@dbend}{stex-dangerous-bend}
6890  \def\smalltextwarning{
6891    \pgfuseimage{miko@small@dbend}
6892    \xspace
6893  }
6894  \pgfdeclareimage[width=1.2em]{miko@dbend}{stex-dangerous-bend}
6895  \newrobustcmd\textwarning{
6896    \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
6897    \xspace
6898  }
6899  \pgfdeclareimage[width=2.5em]{miko@big@dbend}{stex-dangerous-bend}
6900  \newrobustcmd\bigtextwarning{
6901    \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
6902    \xspace
6903  }
```

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

```
6904  \newrobustcmd\putgraphicsat[3]{
6905    \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
6906  }
6907  \newrobustcmd\putat[2]{
6908    \begin{picture}(0,0)\put(#1){#2}\end{picture}
6909  }
```

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`.

```
6910  \bool_if:NT \c__notesslides_sectocframes_bool {
6911    \str_if_eq:VnTF \__notesslidestopsect{part}{
6912      \newcounter{chapter}\counterwithin*{section}{chapter}
6913    }{
6914      \str_if_eq:VnT\__notesslidestopsect{chapter}{
6915        \newcounter{chapter}\counterwithin*{section}{chapter}
6916      }
6917    }
6918  }
```

\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

```
6919  \def\part@prefix{}
6920  \@ifpackageloaded{document-structure}{}{
6921    \str_case:VnF \__notesslidestopsect {
```

253

```
6922        {part}{
6923          \int_set:Nn \l_document_structure_section_level_int {0}
6924          \def\thesection{\arabic{chapter}.\arabic{section}}
6925          \def\part@prefix{\arabic{chapter}.}
6926        }
6927        {chapter}{
6928          \int_set:Nn \l_document_structure_section_level_int {1}
6929          \def\thesection{\arabic{chapter}.\arabic{section}}
6930          \def\part@prefix{\arabic{chapter}.}
6931        }
6932      }{
6933        \int_set:Nn \l_document_structure_section_level_int {2}
6934        \def\part@prefix{}
6935      }
6936    }
6937
6938    \bool_if:NF \c__notesslides_notes_bool { % only in slides
```

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

The new counters are used in the `omgroup` environment that choses the LaTeX sectioning macros according to `\section@level`.

sfragment

```
6939    \renewenvironment{sfragment}[2][]{
6940      \__document_structure_omgroup_args:n { #1 }
6941      \int_incr:N \l_document_structure_section_level_int
6942      \bool_if:NT \c__notesslides_sectocframes_bool {
6943        \stepcounter{slide}
6944        \begin{frame}[noframenumbering]
6945        \vfill\Large\centering
6946        \red{
6947          \ifcase\l_document_structure_section_level_int\or
6948            \stepcounter{part}
6949            \def\__notesslideslabel{\omdoc@part@kw~\Roman{part}}
6950            \def\currentsectionlevel{\omdoc@part@kw}
6951          \or
6952            \stepcounter{chapter}
6953            \def\__notesslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
6954            \def\currentsectionlevel{\omdoc@chapter@kw}
6955          \or
6956            \stepcounter{section}
6957            \def\__notesslideslabel{\part@prefix\arabic{section}}
6958            \def\currentsectionlevel{\omdoc@section@kw}
6959          \or
6960            \stepcounter{subsection}
6961            \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
6962            \def\currentsectionlevel{\omdoc@subsection@kw}
6963          \or
6964            \stepcounter{subsubsection}
6965            \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
6966            \def\currentsectionlevel{\omdoc@subsubsection@kw}
6967          \or
6968            \stepcounter{paragraph}
6969            \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
```

254

```
6970        \def\currentsectionlevel{\omdoc@paragraph@kw}
6971      \else
6972        \def\__notesslideslabel{}
6973        \def\currentsectionlevel{\omdoc@paragraph@kw}
6974      \fi% end ifcase
6975      \__notesslideslabel%\sref@label@id\__notesslideslabel
6976      \quad #2%
6977    }%
6978    \vfill%
6979    \end{frame}%
6980  }
6981  \str_if_empty:NF \l__document_structure_omgroup_id_str {
6982    \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
6983  }
6984  }{}
6985 }
```

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

```
6986 \def\inserttheorembodyfont{\normalfont}
6987 %\bool_if:NF \c__notesslides_notes_bool {
6988 %  \defbeamertemplate{theorem begin}{miko}
6989 %  {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
6990 %    \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
6991 %    \inserttheorempunctuation\inserttheorembodyfont\xspace}
6992 %  \defbeamertemplate{theorem end}{miko}{}
```

and we set it as the default one.

```
6993 %  \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.

```
6994 %  \expandafter\def\csname Parent2\endcsname{}
6995 %}
6996
6997 \AddToHook{begindocument}{ % this does not work for some reasone
6998   \setbeamertemplate{theorems}[ams style]
6999 }
7000 \bool_if:NT \c__notesslides_notes_bool {
7001   \renewenvironment{columns}[1][]{%
7002     \par\noindent%
7003     \begin{minipage}%
7004     \slidewidth\centering\leavevmode%
7005   }{%
7006     \end{minipage}\par\noindent%
7007   }%
7008   \newsavebox\columnbox%
7009   \renewenvironment<>{column}[2][]{%
7010     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
7011   }{%
7012     \end{minipage}\end{lrbox}\usebox\columnbox%
7013   }%
7014 }
```

```
7015  \bool_if:NTF \c__notesslides_noproblems_bool {
7016    \newenvironment{problems}{}{}
7017  }{
7018    \excludecomment{problems}
7019  }
```

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.

```
7020  \gdef\printexcursions{}
7021  \newcommand\excursionref[2]{% label, text
7022    \bool_if:NT \c__notesslides_notes_bool {
7023      \begin{sparagraph}[title=Excursion]
7024        #2 \sref[fallback=the appendix]{#1}.
7025      \end{sparagraph}
7026    }
7027  }
7028  \newcommand\activate@excursion[2][]{
7029    \gappto\printexcursions{\inputref[#1]{#2}}
7030  }
7031  \newcommand\excursion[4][]{% repos, label, path, text
7032    \bool_if:NT \c__notesslides_notes_bool {
7033      \activate@excursion[#1]{#3}\excursionref{#2}{#4}
7034    }
7035  }
```

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

\excursiongroup

```
7036  \keys_define:nn{notesslides / excursiongroup }{
7037    id        .str_set_x:N  = \l__notesslides_excursion_id_str,
7038    intro     .tl_set:N     = \l__notesslides_excursion_intro_tl,
7039    mhrepos   .str_set_x:N  = \l__notesslides_excursion_mhrepos_str
7040  }
7041  \cs_new_protected:Nn \__notesslides_excursion_args:n {
7042    \tl_clear:N \l__notesslides_excursion_intro_tl
7043    \str_clear:N \l__notesslides_excursion_id_str
7044    \str_clear:N \l__notesslides_excursion_mhrepos_str
7045    \keys_set:nn {notesslides / excursiongroup }{ #1 }
7046  }
7047  \newcommand\excursiongroup[1][]{
7048    \__notesslides_excursion_args:n{ #1 }
7049    \ifdefempty\printexcursions{}% only if there are excursions
7050    {\begin{note}
7051      \begin{sfragment}[#1]{Excursions}%
7052        \ifdefempty\l__notesslides_excursion_intro_tl{}{
7053          \inputref[\l__notesslides_excursion_mhrepos_str]{
7054            \l__notesslides_excursion_intro_tl
7055          }
7056        }
7057        \printexcursions%
```

```
7058        \end{sfragment}
7059      \end{note}}
7060 }
7061 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
7062 ⟨/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.

```
7063 ⟨∗package⟩
7064 ⟨@@=problems⟩
7065 \ProvidesExplPackage{problem}{2022/02/26}{3.0.1}{Semantic Markup for Problems}
7066 \RequirePackage{l3keys2e,stex}
7067
7068 \keys_define:nn { problem / pkg }{
7069   notes     .default:n    = { true },
7070   notes     .bool_set:N   = \c__problems_notes_bool,
7071   gnotes    .default:n    = { true },
7072   gnotes    .bool_set:N   = \c__problems_gnotes_bool,
7073   hints     .default:n    = { true },
7074   hints     .bool_set:N   = \c__problems_hints_bool,
7075   solutions .default:n    = { true },
7076   solutions .bool_set:N   = \c__problems_solutions_bool,
7077   pts       .default:n    = { true },
7078   pts       .bool_set:N   = \c__problems_pts_bool,
7079   min       .default:n    = { true },
7080   min       .bool_set:N   = \c__problems_min_bool,
7081   boxed     .default:n    = { true },
7082   boxed     .bool_set:N   = \c__problems_boxed_bool,
7083   unknown   .code:n       = {}
7084 }
7085 \newif\ifsolutions
7086
7087 \ProcessKeysOptions{ problem / pkg }
7088 \bool_if:NTF \c__problems_solutions_bool {
7089   \solutionstrue
7090 }{
7091   \solutionsfalse
7092 }
```

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

```
7093  \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.

```
7094  \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.

```
7095  \def\prob@problem@kw{Problem}
7096  \def\prob@solution@kw{Solution}
7097  \def\prob@hint@kw{Hint}
7098  \def\prob@note@kw{Note}
7099  \def\prob@gnote@kw{Grading}
7100  \def\prob@pt@kw{pt}
7101  \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
7102  \AddToHook{begindocument}{
7103    \ltx@ifpackageloaded{babel}{
7104       \makeatletter
7105       \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7106       \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7107         \input{problem-ngerman.ldf}
7108       }
7109       \clist_if_in:NnT \l_tmpa_clist {finnish}{
7110         \input{problem-finnish.ldf}
7111       }
7112       \clist_if_in:NnT \l_tmpa_clist {french}{
7113         \input{problem-french.ldf}
7114       }
7115       \clist_if_in:NnT \l_tmpa_clist {russian}{
7116         \input{problem-russian.ldf}
7117       }
7118       \makeatother
7119    }{}
7120  }
```

39.2   Problems and Solutions

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

```
7121  \keys_define:nn{ problem / problem }{
7122    id      .str_set_x:N  = \l__problems_prob_id_str,
7123    pts     .tl_set:N     = \l__problems_prob_pts_tl,
7124    min     .tl_set:N     = \l__problems_prob_min_tl,
7125    title   .tl_set:N     = \l__problems_prob_title_tl,
7126    type    .tl_set:N     = \l__problems_prob_type_tl,
7127    imports .tl_set:N     = \l__problems_prob_imports_tl,
7128    name    .str_set_x:N  = \l__problems_prob_name_str,
7129    refnum  .int_set:N    = \l__problems_prob_refnum_int
```

```
7130  }
7131  \cs_new_protected:Nn \__problems_prob_args:n {
7132    \str_clear:N \l__problems_prob_id_str
7133    \str_clear:N \l__problems_prob_name_str
7134    \tl_clear:N \l__problems_prob_pts_tl
7135    \tl_clear:N \l__problems_prob_min_tl
7136    \tl_clear:N \l__problems_prob_title_tl
7137    \tl_clear:N \l__problems_prob_type_tl
7138    \tl_clear:N \l__problems_prob_imports_tl
7139    \int_zero_new:N \l__problems_prob_refnum_int
7140    \keys_set:nn { problem / problem }{ #1 }
7141    \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
7142      \let\l__problems_prob_refnum_int\undefined
7143    }
7144  }
```

Then we set up a counter for problems.

\numberproblemsin

```
7145  \newcounter{problem}
7146  \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.

```
7147  \newcommand\prob@label[1]{#1}
```

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

\prob@number   We consolidate the problem number into a reusable internal macro

```
7148  \newcommand\prob@number{
7149    \int_if_exist:NTF \l__problems_inclprob_refnum_int {
7150      \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
7151    }{
7152      \int_if_exist:NTF \l__problems_prob_refnum_int {
7153        \prob@label{\int_use:N \l__problems_prob_refnum_int }
7154      }{
7155          \prob@label\theproblem
7156      }
7157    }
7158  }
```

(*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.

```
7159  \newcommand\prob@title[3]{%
7160    \tl_if_exist:NTF \l__problems_inclprob_title_tl {
7161      #2 \l__problems_inclprob_title_tl #3
7162    }{
7163      \tl_if_exist:NTF \l__problems_prob_title_tl {
7164        #2 \l__problems_prob_title_tl #3
7165      }{
7166        #1
```

```
7167        }
7168      }
7169 }
```

(*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.

```
7170 \def\prob@heading{
7171   {\prob@problem@kw}\ \prob@number\prob@title{~}{~(}{)\strut}
7172   %\sref@label@id{\prob@problem@kw~\prob@number}{}
7173 }
```

(*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`

```
7174 \newenvironment{sproblem}[1][]{
7175   \__problems_prob_args:n{#1}%\sref@target%
7176   \@in@omtexttrue% we are in a statement (for inline definitions)
7177   \stepcounter{problem}\record@problem
7178   \def\current@section@level{\prob@problem@kw}
7179
7180   \str_if_empty:NT \l__problems_prob_name_str {
7181     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
7182     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
7183     \seq_get_left:NN \l_tmpa_seq \l__problems_prob_name_str
7184   }
7185   \exp_args:Nno\stex_module_setup:nn{type=problem}\l__problems_prob_name_str
7186
7187   \stex_csl_to_imports:No \importmodule \l__problems_prob_imports_tl
7188
7189
7190   \tl_if_exist:NTF \l__problems_inclprob_type_tl {
7191     \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
7192   }{
7193     \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
7194   }
7195   \str_if_exist:NTF \l__problems_inclprob_id_str {
7196     \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
7197   }{
7198     \str_set_eq:NN \sproblemid \l__problems_prob_id_str
7199   }
7200
7201
7202   \clist_set:No \l_tmpa_clist \sproblemtype
7203   \tl_clear:N \l_tmpa_tl
7204   \clist_map_inline:Nn \l_tmpa_clist {
7205     \tl_if_exist:cT {__problems_sproblem_##1_start:}{
7206       \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_start:}}
```

```
7207        }
7208      }
7209      \tl_if_empty:NTF \l_tmpa_tl {
7210        \__problems_sproblem_start:
7211      }{
7212        \l_tmpa_tl
7213      }
7214      \stex_ref_new_doc_target:n \sproblemid
7215    }{
7216      \__stex_modules_end_module:
7217      \clist_set:No \l_tmpa_clist \sproblemtype
7218      \tl_clear:N \l_tmpa_tl
7219      \clist_map_inline:Nn \l_tmpa_clist {
7220        \tl_if_exist:cT {__problems_sproblem_##1_end:}{
7221          \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_end:}}}
7222      }
7223    }
7224    \tl_if_empty:NTF \l_tmpa_tl {
7225      \__problems_sproblem_end:
7226    }{
7227      \l_tmpa_tl
7228    }
7229
7230
7231    \smallskip
7232  }
7233
7234
7235  \cs_new_protected:Nn \__problems_sproblem_start: {
7236    \par\noindent\textbf\prob@heading\show@pts\show@min\\\ignorespacesandpars
7237  }
7238  \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
7239
7240  \newcommand\stexpatchproblem[3][] {
7241    \str_set:Nx \l_tmpa_str{ #1 }
7242    \str_if_empty:NTF \l_tmpa_str {
7243      \tl_set:Nn \__problems_sproblem_start: { #2 }
7244      \tl_set:Nn \__problems_sproblem_end: { #3 }
7245    }{
7246      \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
7247      \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
7248    }
7249  }
7250
7251
7252  \bool_if:NT \c__problems_boxed_bool {
7253    \surroundwithmdframed{problem}
7254  }
```

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

```
7255  \def\record@problem{
7256    \protected@write\@auxout{}
7257    {
7258      \string\@problem{\prob@number}
```

```
7259        {
7260          \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7261            \l__problems_inclprob_pts_tl
7262          }{
7263            \l__problems_prob_pts_tl
7264          }
7265        }%
7266        {
7267          \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7268            \l__problems_inclprob_min_tl
7269          }{
7270            \l__problems_prob_min_tl
7271          }
7272        }
7273      }
7274    }
```

(*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).

```
7275  \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.

```
7276  \keys_define:nn { problem / solution }{
7277    id            .str_set_x:N  = \l__problems_solution_id_str ,
7278    for           .tl_set:N     = \l__problems_solution_for_tl ,
7279    height        .dim_set:N    = \l__problems_solution_height_dim ,
7280    creators      .clist_set:N  = \l__problems_solution_creators_clist ,
7281    contributors  .clist_set:N  = \l__problems_solution_contributors_clist ,
7282    srccite       .tl_set:N     = \l__problems_solution_srccite_tl
7283  }
7284  \cs_new_protected:Nn \__problems_solution_args:n {
7285    \str_clear:N \l__problems_solution_id_str
7286    \tl_clear:N \l__problems_solution_for_tl
7287    \tl_clear:N \l__problems_solution_srccite_tl
7288    \clist_clear:N \l__problems_solution_creators_clist
7289    \clist_clear:N \l__problems_solution_contributors_clist
7290    \dim_zero:N \l__problems_solution_height_dim
7291    \keys_set:nn { problem / solution }{ #1 }
7292  }
```

the next step is to define a helper macro that does what is needed to start a solution.

```
7293  \newcommand\@startsolution[1][]{
7294    \__problems_solution_args:n { #1 }
7295    \@in@omtexttrue% we are in a statement.
7296    \bool_if:NF \c__problems_boxed_bool { \hrule }
7297    \smallskip\noindent
7298    {\textbf\prob@solution@kw :\enspace}
7299    \begin{small}
7300    \def\current@section@level{\prob@solution@kw}
7301    \ignorespacesandpars
7302  }
```

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.

```
7303 \newcommand\startsolutions{
7304   \specialcomment{solution}{\@startsolution}{
7305     \bool_if:NF \c__problems_boxed_bool {
7306       \hrule\medskip
7307     }
7308     \end{small}%
7309   }
7310   \bool_if:NT \c__problems_boxed_bool {
7311     \surroundwithmdframed{solution}
7312   }
7313 }
```

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

```
7314 \newcommand\stopsolutions{\excludecomment{solution}}
```

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

so it only remains to start/stop solutions depending on what option was specified.

```
7315 \ifsolutions
7316   \startsolutions
7317 \else
7318   \stopsolutions
7319 \fi
```

```
7320 \bool_if:NTF \c__problems_notes_bool {
7321   \newenvironment{exnote}[1][]{
7322     \par\smallskip\hrule\smallskip
7323     \noindent\textbf{\prob@note@kw : }\small
7324   }{
7325     \smallskip\hrule
7326   }
7327 }{
7328   \excludecomment{exnote}
7329 }
```

```
7330 \bool_if:NTF \c__problems_notes_bool {
7331   \newenvironment{hint}[1][]{
7332     \par\smallskip\hrule\smallskip
7333     \noindent\textbf{\prob@hint@kw :~ }\small
7334   }{
7335     \smallskip\hrule
7336   }
7337   \newenvironment{exhint}[1][]{
7338     \par\smallskip\hrule\smallskip
7339     \noindent\textbf{\prob@hint@kw :~ }\small
7340   }{
7341     \smallskip\hrule
```

```
7342        }
7343    }{
7344      \excludecomment{hint}
7345      \excludecomment{exhint}
7346    }
```

gnote

```
7347    \bool_if:NTF \c__problems_notes_bool {
7348      \newenvironment{gnote}[1][]{
7349        \par\smallskip\hrule\smallskip
7350        \noindent\textbf{\prob@gnote@kw : }\small
7351      }{
7352        \smallskip\hrule
7353      }
7354    }{
7355      \excludecomment{gnote}
7356    }
```

39.3 Multiple Choice Blocks

mcb [16]

```
7357    \newenvironment{mcb}{
7358      \begin{enumerate}
7359    }{
7360      \end{enumerate}
7361    }
```

we define the keys for the mcc macro

```
7362    \cs_new_protected:Nn \__problems_do_yes_param:Nn {
7363      \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
7364        \bool_set_true:N #1
7365      }{
7366        \bool_set_false:N #1
7367      }
7368    }
7369    \keys_define:nn { problem / mcc }{
7370      id        .str_set_x:N  = \l__problems_mcc_id_str ,
7371      feedback  .tl_set:N     = \l__problems_mcc_feedback_tl ,
7372      T         .default:n    = { true } ,
7373      T         .bool_set:N   = \l__problems_mcc_t_bool ,
7374      F         .default:n    = { true } ,
7375      F         .bool_set:N   = \l__problems_mcc_f_bool ,
7376      Ttext     .code:n       = {
7377        \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
7378      } ,
7379      Ftext     .code:n       = {
7380        \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
7381      }
7382    }
7383    \cs_new_protected:Nn \l__problems_mcc_args:n {
7384      \str_clear:N \l__problems_mcc_id_str
```

---

[16]EDNOTE: MK: maybe import something better here from a dedicated MC package

```
7385     \tl_clear:N \l__problems_mcc_feedback_tl
7386     \bool_set_true:N \l__problems_mcc_t_bool
7387     \bool_set_true:N \l__problems_mcc_f_bool
7388     \bool_set_true:N \l__problems_mcc_Ttext_bool
7389     \bool_set_false:N \l__problems_mcc_Ftext_bool
7390     \keys_set:nn { problem / mcc }{ #1 }
7391 }
```

\mcc

```
7392 \newcommand\mcc[2][]{
7393     \l__problems_mcc_args:n{ #1 }
7394     \item #2
7395     \ifsolutions
7396         \\
7397         \bool_if:NT \l__problems_mcc_t_bool {
7398             % TODO!
7399             % \ifcsstring{mcc@T}{T}{}{\mcc@Ttext}%
7400         }
7401         \bool_if:NT \l__problems_mcc_f_bool {
7402             % TODO!
7403             % \ifcsstring{mcc@F}{F}{}{\mcc@Ftext}%
7404         }
7405         \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
7406             !
7407         }{
7408             \l__problems_mcc_feedback_tl
7409         }
7410     \fi
7411 } %solutions
```

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

39.4   Including Problems

\includeproblem   The \includeproblem command is essentially a glorified \input statement, it sets some internal macros first that overwrite the local points. Importantly, it resets the inclprob keys after the input.

```
7412
7413 \keys_define:nn{ problem / inclproblem }{
7414     id      .str_set_x:N  = \l__problems_inclprob_id_str,
7415     pts     .tl_set:N     = \l__problems_inclprob_pts_tl,
7416     min     .tl_set:N     = \l__problems_inclprob_min_tl,
7417     title   .tl_set:N     = \l__problems_inclprob_title_tl,
7418     refnum  .int_set:N     = \l__problems_inclprob_refnum_int,
7419     type    .tl_set:N     = \l__problems_inclprob_type_tl,
7420     mhrepos .str_set_x:N  = \l__problems_inclprob_mhrepos_str
7421 }
7422 \cs_new_protected:Nn \__problems_inclprob_args:n {
7423     \str_clear:N \l__problems_prob_id_str
7424     \tl_clear:N \l__problems_inclprob_pts_tl
7425     \tl_clear:N \l__problems_inclprob_min_tl
7426     \tl_clear:N \l__problems_inclprob_title_tl
7427     \tl_clear:N \l__problems_inclprob_type_tl
```

```
7428    \int_zero_new:N \l__problems_inclprob_refnum_int
7429    \str_clear:N \l__problems_inclprob_mhrepos_str
7430    \keys_set:nn { problem / inclproblem }{ #1 }
7431    \tl_if_empty:NT \l__problems_inclprob_pts_tl {
7432      \let\l__problems_inclprob_pts_tl\undefined
7433    }
7434    \tl_if_empty:NT \l__problems_inclprob_min_tl {
7435      \let\l__problems_inclprob_min_tl\undefined
7436    }
7437    \tl_if_empty:NT \l__problems_inclprob_title_tl {
7438      \let\l__problems_inclprob_title_tl\undefined
7439    }
7440    \tl_if_empty:NT \l__problems_inclprob_type_tl {
7441      \let\l__problems_inclprob_type_tl\undefined
7442    }
7443    \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
7444      \let\l__problems_inclprob_refnum_int\undefined
7445    }
7446  }
7447
7448  \cs_new_protected:Nn \__problems_inclprob_clear: {
7449    \let\l__problems_inclprob_id_str\undefined
7450    \let\l__problems_inclprob_pts_tl\undefined
7451    \let\l__problems_inclprob_min_tl\undefined
7452    \let\l__problems_inclprob_title_tl\undefined
7453    \let\l__problems_inclprob_type_tl\undefined
7454    \let\l__problems_inclprob_refnum_int\undefined
7455    \let\l__problems_inclprob_mhrepos_str\undefined
7456  }
7457  \__problems_inclprob_clear:
7458
7459  \newcommand\includeproblem[2][]{
7460    \__problems_inclprob_args:n{ #1 }
7461    \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
7462      \input{#2}
7463    }{
7464      \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
7465        \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
7466    }
7467  }
7468    \__problems_inclprob_clear:
7469  }
```

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

39.5   Reporting Metadata

For messages it is OK to have them in English as the whole documentation is, and we can therefore assume authors can deal with it.

```
7470  \AddToHook{enddocument}{
7471    \bool_if:NT \c__problems_pts_bool {
7472      \message{Total:~\arabic{pts}~points}
7473    }
```

```
7474    \bool_if:NT \c__problems_min_bool {
7475      \message{Total:~\arabic{min}~minutes}
7476    }
7477 }
```

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

```
7478 \def\pts#1{
7479    \bool_if:NT \c__problems_pts_bool {
7480      \marginpar{#1~\prob@pt@kw}
7481    }
7482 }
7483 \def\min#1{
7484    \bool_if:NT \c__problems_min_bool {
7485      \marginpar{#1~\prob@min@kw}
7486    }
7487 }
```

\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.

```
7488 \newcounter{pts}
7489 \def\show@pts{
7490    \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7491      \bool_if:NT \c__problems_pts_bool {
7492        \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
7493        \addtocounter{pts}{\l__problems_inclprob_pts_tl}
7494      }
7495    }{
7496      \tl_if_exist:NT \l__problems_prob_pts_tl {
7497        \bool_if:NT \c__problems_pts_bool {
7498          \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
7499          \addtocounter{pts}{\l__problems_prob_pts_tl}
7500        }
7501      }
7502    }
7503 }
```

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

and now the same for the minutes

\show@min

```
7504 \newcounter{min}
7505 \def\show@min{
7506    \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7507      \bool_if:NT \c__problems_min_bool {
7508        \marginpar{\l__problems_inclprob_pts_tl\ min}
7509        \addtocounter{min}{\l__problems_inclprob_min_tl}
7510      }
7511    }{
7512      \tl_if_exist:NT \l__problems_prob_min_tl {
7513        \bool_if:NT \c__problems_min_bool {
7514          \tl_if_empty:NT\l__problems_prob_min_tl{
7515            \tl_set:Nn \l__problems_prob_min_tl {0}
7516          }
```

```
7517          \marginpar{\l__problems_prob_min_tl\ min}
7518          \addtocounter{min}{\l__problems_prob_min_tl}
7519        }
7520      }
7521    }
7522 }
7523 ⟨/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.

```
7524 ⟨*package⟩
7525 \ProvidesExplPackage{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
7526 \RequirePackage{l3keys2e}
7527
7528 \newif\iftest\testfalse
7529 \DeclareOption{test}{\testtrue}
7530 \newif\ifmultiple\multiplefalse
7531 \DeclareOption{multiple}{\multipletrue}
7532 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
7533 \ProcessOptions
```

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

```
7534 \RequirePackage{keyval}[1997/11/10]
7535 \RequirePackage{problem}
```

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

```
7536 \newcommand\hwexam@assignment@kw{Assignment}
7537 \newcommand\hwexam@given@kw{Given}
7538 \newcommand\hwexam@due@kw{Due}
7539 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
7540 blank~for~extra~space}
7541 \def\hwexam@minutes@kw{minutes}
7542 \newcommand\correction@probs@kw{prob.}
7543 \newcommand\correction@pts@kw{total}
7544 \newcommand\correction@reached@kw{reached}
7545 \newcommand\correction@sum@kw{Sum}
7546 \newcommand\correction@grade@kw{grade}
7547 \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

```
7548 \AddToHook{begindocument}{
7549 \ltx@ifpackageloaded{babel}{
7550 \makeatletter
7551 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7552 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7553   \input{hwexam-ngerman.ldf}
7554 }
7555 \clist_if_in:NnT \l_tmpa_clist {finnish}{
7556   \input{hwexam-finnish.ldf}
7557 }
7558 \clist_if_in:NnT \l_tmpa_clist {french}{
7559   \input{hwexam-french.ldf}
7560 }
7561 \clist_if_in:NnT \l_tmpa_clist {russian}{
7562   \input{hwexam-russian.ldf}
7563 }
7564 \makeatother
7565 }{}
7566 }
7567
```

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.

```
7568 \newcounter{assignment}
7569 \numberproblemsin{assignment}
7570 \renewcommand\prob@label[1]{\assignment@number.#1}
```

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

```
7571 \keys_define:nn { hwexam / assignment } {
7572 id   .str_set_x:N = \l_@@_assign_id_str,
7573 number  .int_set:N  = \l_@@_assign_number_int,
7574 title  .tl_set:N  = \l_@@_assign_title_tl,
7575 type   .tl_set:N  = \l_@@_assign_type_tl,
7576 given .tl_set:N  = \l_@@_assign_given_tl,
7577 due .tl_set:N  = \l_@@_assign_due_tl,
7578 loadmodules .code:n  = {
7579 \bool_set_true:N \l_@@_assign_loadmodules_bool
7580 }
7581 }
7582 \cs_new_protected:Nn \_@@_assignment_args:n {
7583 \str_clear:N \l_@@_assign_id_str
7584 \int_set:Nn \l_@@_assign_number_int {-1}
7585 \tl_clear:N \l_@@_assign_title_tl
7586 \tl_clear:N \l_@@_assign_type_tl
7587 \tl_clear:N \l_@@_assign_given_tl
7588 \tl_clear:N \l_@@_assign_due_tl
7589 \bool_set_false:N \l_@@_assign_loadmodules_bool
```

```
7590  \keys_set:nn { hwexam / assignment }{ #1 }
7591  }
```

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.

```
7592  \newcommand\given@due[2]{
7593  \bool_lazy_all:nF {
7594  {\tl_if_empty_p:V \l_@@_inclassign_given_tl}
7595  {\tl_if_empty_p:V \l_@@_assign_given_tl}
7596  {\tl_if_empty_p:V \l_@@_inclassign_due_tl}
7597  {\tl_if_empty_p:V \l_@@_assign_due_tl}
7598  }{ #1 }
7599
7600  \tl_if_empty:NTF \l_@@_inclassign_given_tl {
7601  \tl_if_empty:NF \l_@@_assign_given_tl {
7602  \hwexam@given@kw\xspace\l_@@_assign_given_tl
7603  }
7604  }{
7605  \hwexam@given@kw\xspace\l_@@_inclassign_given_tl
7606  }
7607
7608  \bool_lazy_or:nnF {
7609  \bool_lazy_and_p:nn {
7610  \tl_if_empty_p:V \l_@@_inclassign_due_tl
7611  }{
7612  \tl_if_empty_p:V \l_@@_assign_due_tl
7613  }
7614  }{
7615  \bool_lazy_and_p:nn {
7616  \tl_if_empty_p:V \l_@@_inclassign_due_tl
7617  }{
7618  \tl_if_empty_p:V \l_@@_assign_due_tl
7619  }
7620  }{ ,~ }
7621
7622  \tl_if_empty:NTF \l_@@_inclassign_due_tl {
7623  \tl_if_empty:NF \l_@@_assign_due_tl {
7624  \hwexam@due@kw\xspace \l_@@_assign_due_tl
7625  }
7626  }{
7627  \hwexam@due@kw\xspace \l_@@_inclassign_due_tl
7628  }
7629
7630  \bool_lazy_all:nF {
7631  { \tl_if_empty_p:V \l_@@_inclassign_given_tl }
7632  { \tl_if_empty_p:V \l_@@_assign_given_tl }
7633  { \tl_if_empty_p:V \l_@@_inclassign_due_tl }
7634  { \tl_if_empty_p:V \l_@@_assign_due_tl }
7635  }{ #2 }
7636  }
```

\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.

```
7637 \newcommand\assignment@title[3]{
7638 \tl_if_empty:NTF \l_@@_inclassign_title_tl {
7639 \tl_if_empty:NTF \l_@@_assign_title_tl {
7640 #1
7641 }{
7642 #2\l_@@_assign_title_tl#3
7643 }
7644 }{
7645 #2\l_@@_inclassign_title_tl#3
7646 }
7647 }
```

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

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

```
7648 \newcommand\assignment@number{
7649 \int_compare:nNnTF \l_@@_inclassign_number_int = {-1} {
7650 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
7651 \arabic{assignment}
7652 } {
7653 \int_use:N \l_@@_assign_number_int
7654 }
7655 }{
7656 \int_use:N \l_@@_inclassign_number_int
7657 }
7658 }
```

(*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.

```
7659 \newenvironment{assignment}[1][]{
7660 \_@@_assignment_args:n { #1 }
7661 %\sref@target
7662 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
7663 \global\stepcounter{assignment}
7664 }{
7665 \global\setcounter{assignment}{\int_use:N\l_@@_assign_number_int}
7666 }
7667 \setcounter{problem}{0}
7668 \def\current@section@level{\document@hwexamtype}
7669 %\sref@label@id{\document@hwexamtype \thesection}
7670 \begin{@assignment}
7671 }{
7672 \end{@assignment}
7673 }
```

273

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

```
7674  \def\ass@title{
7675  \protect\document@hwexamtype~\arabic{assignment}
7676  \assignment@title{}{\;(}{)\;} -- \given@due{}{}
7677  }
7678  \ifmultiple
7679  \newenvironment{@assignment}{
7680  \bool_if:NTF \l_@@_assign_loadmodules_bool {
7681  \begin{sfragment}[loadmodules]{\ass@title}
7682  }{
7683  \begin{sfragment}{\ass@title}
7684  }
7685  }{
7686  \end{sfragment}
7687  }
```

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

```
7688  \else
7689  \newenvironment{@assignment}{
7690  \begin{center}\bf
7691  \Large\@title\strut\\
7692  \document@hwexamtype~\arabic{assignment}\assignment@title{\;}{:\;}{\\}
7693  \large\given@due{--\;}{\;--}
7694  \end{center}
7695  }{}
7696  \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.

```
7697  \keys_define:nn { hwexam / inclassignment } {
7698  %id   .str_set_x:N = \l_@@_assign_id_str,
7699  number   .int_set:N = \l_@@_inclassign_number_int,
7700  title  .tl_set:N  = \l_@@_inclassign_title_tl,
7701  type  .tl_set:N  = \l_@@_inclassign_type_tl,
7702  given .tl_set:N  = \l_@@_inclassign_given_tl,
7703  due .tl_set:N  = \l_@@_inclassign_due_tl,
7704  mhrepos   .str_set_x:N = \l_@@_inclassign_mhrepos_str
7705  }
7706  \cs_new_protected:Nn \_@@_inclassignment_args:n {
7707  \int_set:Nn \l_@@_inclassign_number_int {-1}
7708  \tl_clear:N \l_@@_inclassign_title_tl
7709  \tl_clear:N \l_@@_inclassign_type_tl
7710  \tl_clear:N \l_@@_inclassign_given_tl
7711  \tl_clear:N \l_@@_inclassign_due_tl
7712  \str_clear:N \l_@@_inclassign_mhrepos_str
7713  \keys_set:nn { hwexam / inclassignment }{ #1 }
7714  }
7715  \_@@_inclassignment_args:n {}
7716
7717  \newcommand\inputassignment[2][]{
```

```
7718  \_@@_inclassignment_args:n { #1 }
7719  \str_if_empty:NTF \l_@@_inclassign_mhrepos_str {
7720  \input{#2}
7721  }{
7722  \stex_in_repository:nn{\l_@@_inclassign_mhrepos_str}{
7723  \input{\mhpath{\l_@@_inclassign_mhrepos_str}{#2}}
7724  }
7725  }
7726  \_@@_inclassignment_args:n {}
7727  }
7728  \newcommand\includeassignment[2][]{
7729  \newpage
7730  \inputassignment[#1]{#2}
7731  }
```

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

40.4   Typesetting Exams

```
7732  \ExplSyntaxOff
7733  \newcommand\quizheading[1]{%
7734  \def\@tas{#1}%
7735  \large\noindent NAME: \hspace{8cm}  MAILBOX:\\[2ex]%
7736  \ifx\@tas\@empty\else%
7737  \noindent TA:~\@for\@I:=\@tas\do{{\Large$\Box$}\@I\hspace*{1em}}\\[2ex]%
7738  \fi%
7739  }
7740  \ExplSyntaxOn
```

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

\testheading

```
7741
7742  \def\hwexamheader{\input{hwexam-default.header}}
7743
7744  \def\hwexamminutes{
7745  \tl_if_empty:NTF \testheading@duration {
7746  {\testheading@min}~\hwexam@minutes@kw
7747  }{
7748  \testheading@duration
7749  }
7750  }
7751
7752  \keys_define:nn { hwexam / testheading } {
7753  min    .tl_set:N  = \testheading@min,
7754  duration .tl_set:N  = \testheading@duration,
7755  reqpts .tl_set:N  = \testheading@reqpts,
7756  tools  .tl_set:N  = \testheading@tools
7757  }
7758  \cs_new_protected:Nn \_@@_testheading_args:n {
7759  \tl_clear:N \testheading@min
7760  \tl_clear:N \testheading@duration
```

```
7761  \tl_clear:N \testheading@reqpts
7762  \tl_clear:N \testheading@tools
7763  \keys_set:nn { hwexam / testheading }{ #1 }
7764  }
7765  \newenvironment{testheading}[1][]{
7766  \_@@_testheading_args:n{ #1 }
7767  \newcount\check@time\check@time=\testheading@min
7768  \advance\check@time by -\theassignment@totalmin
7769  \newif\if@bonuspoints
7770  \tl_if_empty:NTF \testheading@reqpts {
7771  \@bonuspointsfalse
7772  }{
7773  \newcount\bonus@pts
7774  \bonus@pts=\theassignment@totalpts
7775  \advance\bonus@pts by -\testheading@reqpts
7776  \edef\bonus@pts{\the\bonus@pts}
7777  \@bonuspointstrue
7778  }
7779  \edef\check@time{\the\check@time}
7780
7781  \makeatletter\hwexamheader\makeatother
7782  }{
7783  \newpage
7784  }
```

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

\testspace

```
7785  \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}
```

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

\testnewpage

```
7786  \newcommand\testnewpage{\iftest\newpage\fi}
```

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

\testemptypage

```
7787  \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.

```
7788  ⟨@@=problems⟩
7789  \renewcommand\@problem[3]{
7790  \stepcounter{assignment@probs}
7791  \def\__problemspts{#2}
7792  \ifx\__problemspts\@empty\else
7793  \addtocounter{assignment@totalpts}{#2}
7794  \fi
7795  \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\f
7796  \xdef\correction@probs{\correction@probs & #1}%
7797  \xdef\correction@pts{\correction@pts & #2}
7798  \xdef\correction@reached{\correction@reached &}
```

7799 `}`

7800 ⟨@@=hwexam⟩

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

\correction@table This macro generates the correction table

7801 `\newcounter{assignment@probs}`
7802 `\newcounter{assignment@totalpts}`
7803 `\newcounter{assignment@totalmin}`
7804 `\def\correction@probs{\correction@probs@kw}`
7805 `\def\correction@pts{\correction@pts@kw}`
7806 `\def\correction@reached{\correction@reached@kw}`
7807 `\stepcounter{assignment@probs}`
7808 `\newcommand\correction@table{`
7809 `\resizebox{\textwidth}{!}{%`
7810 `\begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%`
7811 `&\multicolumn{\theassignment@probs}{c||}%|`
7812 `{\footnotesize\correction@forgrading@kw} &\\\hline`
7813 `\correction@probs & \correction@sum@kw & \correction@grade@kw\\\hline`
7814 `\correction@pts &\theassignment@totalpts & \\\hline`
7815 `\correction@reached & & \\[.7cm]\hline`
7816 `\end{tabular}}}`

7817 ⟨/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}
```