The sTeX3 Package *

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

2022-03-09

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 2022-03-09)

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   The sdefinition-environment provides two additional macros, \definame and \definiendum
\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, SᴛᴇX expressions are converted to OMDOC/MMT terms, which use the syntax of OPENMATH.

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

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

1. If the SᴛᴇX package is loaded with the option `mathhub=/path/to/mathhub`, then SᴛᴇX 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 SᴛᴇX-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, SᴛᴇX 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.

3.2.2   The Structure of SᴛᴇX Archives

An SᴛᴇX 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 SᴛᴇX 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 SᴛᴇX 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 SŢEX 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

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

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

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

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

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

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

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

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

creators ($\langle$*string*$\rangle*$) names of the creators.

contributors ($\langle$*string*$\rangle*$) names of contributors.

srccite ($\langle$*string*$\rangle$) a source citation for the content of this module.

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

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

Example 1

Input:

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

Output:

```
Hello World
```

.

<div style="margin-left:2em">\stexpatchmodule</div>

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

For example:

Example 2

Input:

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

Output:

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

.

3.3.2 Declaring New Symbols and Notations

Inside an smodule environment, we can declare new STEX symbols.

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

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

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

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

Example 3
Input:

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

Output:

```
Given a foo, we can...
```

.

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

Example 4
Input:

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

Output:

```
this is a symbol taking two arguments.
```

.

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→ MMT/OMDOC as `OMA`-terms with head `<OMS name="...?binarysymbol"/>`.
⤳T⤳ Semantic macros with no arguments correspond to `OMS` directly.

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`.

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 OMDOC/MMT constants.
> ⟶M⟶ Correspondingly, the name "type" should be taken with a grain of salt, since
> ⟿T⟿ OMDOC/MMT– being foundation-independent – does not a priori implement a fixed typing system.

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

Example 13

Input:

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

Output:

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

.

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

Example 14

Input:

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

Output:

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

.

3.3.5   Precedences and Automated Bracketing

Having done `\addition`, the obvious next thing to implement is `\multiplication`. This is 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.

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

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

In the example above:

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

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

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

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

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

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

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

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

3.3.6 Variables

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

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

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

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

\vardef

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

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

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

TODO: bind=forall/exists

3.3.7 Variable Sequences

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

\varseq

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

This is best shown by example:

> **Example 20**
>
> Input:
>
> ```
> 1  \vardef{varn}[name=n,type=\Nat]{\comp{n}}
> 2 \varseq{seqa}[name=a,type=\Nat]{1}{\varn}{\comp{a}_{#1}}
> 3
> 4 The $i$th index of $\seqa!$ is $\seqa{i}$.
> ```
>
> Output:
>
> The ith index of $a_1, \ldots, a_n$ is a_i.

.

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

TODO: more notations for invoking sequences.

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

Example 21

Input:

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

Output:

$a_1 + \ldots + a_n$

.

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

Example 22

Input:

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

Output:

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

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

Example 23

Input:

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

Output:

$a_1^1, \ldots, a_n^1, a_1^2, \ldots, a_1^m, \ldots, a_n^m$ 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 $\mathtt{lcm}(a,b)$ in english, but is called *kleinstes gemeinsames Vielfaches* in german and consequently denoted as $\mathtt{kgV}(a,b)$ there.

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

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

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

3.4.2 Simple Inheritance and Namespaces

`\importmodule[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

file $\langle$*top-directory*$\rangle$`/some/path/Foo[.`$\langle$*lang*$\rangle$`].tex`, or in $\langle$*top-directory*$\rangle$`/some/path[.`$\langle$*lang*$\rangle$`].tex` (which are checked in that order).

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

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

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

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

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

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

3.4.3   The `mathstructure` Environment

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

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

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

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

This phenomenon is important and common enough to warrant special support, in particular because it requires being able to *instantiate* such structures (or, 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` envi-
ronment 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.

> \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:

```
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:

32

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: $a \cdot c \circ c$

.

TODO: explain donotclone

3.4.5 The `interpretmodule` Environment

TODO: explain

Example 30

Input:

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

Output:

.

3.5 Primitive Symbols (The sTeX Metatheory)

TODO: metatheory documentation

Chapter 4

Using sTeX Symbols

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

4.1 \symref and its variants

`\symref`
`\symname`

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

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

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

Example 31

Input:

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

Output:

> A natural number is...

.

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

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 SᴛᴇX resolves a string `symbolname` to an actual symbol.
>
> If `\symbolname` is a semantic macro, then SᴛᴇX 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...`. SᴛᴇX attempts to handle this case thusly:
>
> If `string` does *not* correspond to a semantic macro `\string`, then SᴛᴇX 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 `addition`s are in scope.
>
> However, this also means that if we have symbols `foo` and e.g. `miraculous-foo`, then SᴛᴇX 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 SᴛᴇX 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/Mᴍᴛ 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 Mᴍᴛ 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, sTeX can not check that the assertions are the "correct" ones – but if the assertions (both in `monoid` as well as those for addition and zero) are properly marked up, 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 SIEX highlights variables, notation components, \symrefs and \definiendums, respectively.

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

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

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

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

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

Chapter 7

Additional Packages

TODO: tikzinput documentation

7.1 Modular Document Structuring

TODO: document-structure documentation

7.2 Slides and Course Notes

TODO: notesslides documentation

7.3 Homework, Problems and Exams

TODO: problem documentation
     TODO: hwexam documentation

Part II
Documentation

Chapter 8

STEX-Basics

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

8.1   Macros and Environments

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

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

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

8.1.1   HTML Annotations

`\if@latexml`   LATEX2e conditional for LATEXML

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

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

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

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

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

Annotates the HTML generated by ⟨*content*⟩ with

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

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

8.1.2 Babel Languages

| `\c_stex_languages_prop` | |
| `\c_stex_language_abbrevs_prop` | |

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

8.1.3 Auxiliary Methods

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

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

$\quad$ `\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 (repsectively) a file path to the file stack, to keep track of the current file. Are called in hooks `file/before` and `file/after`, respectively.

9.1.2 MathHub Archives

`\mathhub`
`\c_stex_mathhub_seq`
`\c_stex_mathhub_str`   We determine the path to the local MathHub folder via one of three 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.

In all three 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

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

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

$\overline{\text{\srefsym}}$ `\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*⟩.

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

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

Chapter 11

sTEX-Modules

This sub package contains code related to Modules

11.1   Macros and Environments

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

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

name The *name* of the module,

ns the *namespace* in field **ns**,

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

lang the module's *language*,

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

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

meta the metatheory of the module.

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

\c_stex_module_<URI>_constants

The names of all constants declared in the module

\c_stex_module_<URI>_constants

The full URIs of all modules imported in this module

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

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

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

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

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

`\stex_add_to_current_module:n`
`\STEXexport`

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

`\stex_add_constant_to_current_module:n`

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

`\stex_add_import_to_current_module:n`

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

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

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

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

Establishes a context for `\comp`. Stores the URI in a variable so that `\comp` knows which symbol governs the current notation.

`\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 `spfstep`s in the body uses a third argument for a proof sketch.

sproofcomment

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

17.2.5   Proof End Markers

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

\sproofend

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

\sProofEndSymbol

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

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

17.2.6   Configuration of the Presentation

Finally, we provide configuration hooks in Figure 1 for the keywords in proofs. These are mainly intended for package authors building on `statements`, e.g. for multi-language

EdN:3

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.

71

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 OM-Doc. 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="margin-left:2em">

ignore
showignores

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`.

\prematurestop

\afterprematurestop

For prematurely stopping the formatting of a document, sTeX provides the `\prematurestop` macro. It can be used everywhere in a document and ignores all input after that – backing out of the 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

\STRlabel
\STRcopy

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.

\STRsemantics

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]

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

\setSGvar
\useSGvar
\ifSGvar

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

</div>

EdN:5

---

[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

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

\blue
\red
...
\black

20.3 Limitations

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

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

Chapter 21

NotesSlides – Slides and Course Notes

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

21.1 Introduction

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

21.2 The User Interface

The `notesslides` class takes the notion of a slide frame from Till Tantau's excellent `beamer` class and adapts its notion of frames for use in the 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
notes
- The options `slides` and `notes` switch between slides mode and notes mode (see Section 21.2.2).

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

21.2.3   Header and Footer Lines of the Slides

The default logo provided by the `notesslides` package is the sℸEX logo it can be customized 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 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 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 sℸEXnotes. In this case we can use `\frameimage[⟨opt⟩]{⟨path⟩}`, where ⟨opt⟩ are the options of `\includegraphics` from the `graphicx` package [CR99] and ⟨path⟩ is the file path (extension can be left off like in `\includegraphics`). We have added the `label` key that allows to give a frame label that can be referenced like a regular `beamer` frame.[7]

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}`

Margin notes (left column):
\inputref*

nparagraph

nfragment
ndefinition
nexample
nsproof
nassertion

\setslidelogo

\setsource

\setlicensing

\frameimage

EdN:7

\mhframeimage

---
[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
notes
hints
gnotes
pts
min
boxed
test

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

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

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

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

The `solution` environment can be to specify a solution to a problem. If the
`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-
ment with the keys `id` for an identifier that can be reference `for` to specify which problem
this is a solution for, and `height` that allows to specify the amount of space to be left in
test situations (i.e. if the `test` option is set in the `\usepackage` statement).

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

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

may arise in grading.

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]
  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

The \inputassignment macro can be used to input an assignment from another file. It takes an optional KeyVal argument and a second argument which is a path to the file containing the problem (the macro assumes that there is only one `assignment` environment

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)

2022-03-09

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.	To be used for grading, do not write here											grade
	0.1	0.2	0.3	1.1	2.1	2.2	2.3	3.1	3.2	3.3	Sum	
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  \LoadClass[border=1px,varwidth]{standalone}
8  \setlength\textwidth{15cm}
9
10 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
11 \ProcessOptions
12
13 \RequirePackage{stex}
14 ⟨/cls⟩
```

24.2 Preliminaries

```
15 ⟨*package⟩
16
17 %%%%%%%%%%%%  basics.dtx  %%%%%%%%%%%%
18
19 \RequirePackage{expl3,l3keys2e,ltxcmds}
20 \ProvidesExplPackage{stex}{2022/03/03}{3.1.0}{sTeX package}
21
22 %\RequirePackage{morewrites}
23 %\RequirePackage{amsmath}
24
```

Package options:
```
25 \keys_define:nn { stex } {
```

```
26    debug      .clist_set:N  = \c_stex_debug_clist ,
27    lang       .clist_set:N  = \c_stex_languages_clist ,
28    mathhub    .tl_set_x:N   = \mathhub ,
29    sms        .bool_set:N   = \c_stex_persist_mode_bool ,
30    image      .bool_set:N   = \c_tikzinput_image_bool,
31    unknown    .code:n       = {}
32  }
33  \ProcessKeysOptions { stex }
```

\stex The STEXlogo:
\sTeX

```
34  \protected\def\stex{
35    \texorpdfstring{\raisebox{-.5ex}S\kern-.5ex\TeX}{sTeX}\xspace%
36  }
37  \let\sTeX\stex
```

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

24.3   Messages and logging

```
38  ⟨@@=stex_log⟩
```

Warnings and error messages

```
39  \msg_new:nnn{stex}{error/unknownlanguage}{
40    Unknown~language:~#1
41  }
42  \msg_new:nnn{stex}{warning/nomathhub}{
43    MATHHUB~system~variable~not~found~and~no~
44    \detokenize{\mathhub}-value~set!
45  }
46  \msg_new:nnn{stex}{error/deactivated-macro}{
47    The~\detokenize{#1}~command~is~only~allowed~in~#2!
48  }
```

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

```
49  \cs_new_protected:Nn \stex_debug:nn {
50    \clist_if_in:NnTF \c_stex_debug_clist { all } {
51      \msg_set:nnn{stex}{debug / #1}{
52        \\Debug~#1:~#2\\
53      }
54      \msg_none:nn{stex}{debug / #1}
55    }{
56      \clist_if_in:NnT \c_stex_debug_clist { #1 } {
57        \msg_set:nnn{stex}{debug / #1}{
58          \\Debug~#1:~#2\\
59        }
60        \msg_none:nn{stex}{debug / #1}
61      }
62    }
63  }
```

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

Redirecting messages:

```
64  \clist_if_in:NnTF \c_stex_debug_clist {all} {
65    \msg_redirect_module:nnn{ stex }{ none }{ term }
```

```
66 }{
67   \clist_map_inline:Nn \c_stex_debug_clist {
68     \msg_redirect_name:nnn{ stex }{ debug / ##1 }{ term }
69   }
70 }
71
72 \stex_debug:nn{log}{debug~mode~on}
```

24.4   HTML Annotations

```
73 ⟨@@=stex_annotate⟩
74 \RequirePackage{rustex}
```

We add the namespace abbreviation `ns:stex="http://kwarc.info/ns/sTeX"` to
R_{US}TeX:

```
75 \rustex_add_Namespace:nn{stex}{http://kwarc.info/ns/sTeX}
```

Conditionals for LaTeXML:

\if@latexml

```
76 \ifcsname if@latexml\endcsname\else
77     \expandafter\newif\csname if@latexml\endcsname\@latexmlfalse
78 \fi
```

(*End definition for* `\if@latexml`*. This function is documented on page 46.*)

\latexml_if_p:
\latexml_if:TF

```
79 \prg_new_conditional:Nnn \latexml_if: {p, T, F, TF} {
80     \if@latexml
81       \prg_return_true:
82     \else:
83       \prg_return_false:
84     \fi:
85 }
```

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

`\l__stex_annotate_arg_tl`   Used by annotation macros to ensure that the HTML output to annotate is not empty.
`\c__stex_annotate_emptyarg_tl`

```
86 \tl_new:N \l__stex_annotate_arg_tl
87 \tl_const:Nx \c__stex_annotate_emptyarg_tl {
88   \rustex_if:TF {
89     \rustex_direct_HTML:n { \c_ampersand_str lrm; }
90   }{~}
91 }
```

(*End definition for* `\l__stex_annotate_arg_tl` *and* `\c__stex_annotate_emptyarg_tl`*.*)

`\__stex_annotate_checkempty:n`

```
92 \cs_new_protected:Nn \__stex_annotate_checkempty:n {
93   \tl_set:Nn \l__stex_annotate_arg_tl { #1 }
94   \tl_if_empty:NT \l__stex_annotate_arg_tl {
95     \tl_set_eq:NN \l__stex_annotate_arg_tl \c__stex_annotate_emptyarg_tl
96   }
97 }
```

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

`\stex_if_do_html_p:`
`\stex_if_do_html:TF`

Whether to (locally) produce HTML output

```
98 \bool_new:N \_stex_html_do_output_bool
99 \bool_set_true:N \_stex_html_do_output_bool
100
101 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
102   \bool_if:nTF \_stex_html_do_output_bool
103     \prg_return_true: \prg_return_false:
104 }
```

(*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

```
105 \cs_new_protected:Nn \stex_suppress_html:n {
106   \exp_args:Nne \use:nn {
107     \bool_set_false:N \_stex_html_do_output_bool
108     #1
109   }{
110     \stex_if_do_html:T {
111       \bool_set_true:N \_stex_html_do_output_bool
112     }
113   }
114 }
```

(*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.

```
115 \rustex_if:TF{
116   \cs_new_protected:Nn \stex_annotate:nnn {
117     \_stex_annotate_checkempty:n { #3 }
118     \rustex_annotate_HTML:nn {
119       property="stex:#1" ~
120       resource="#2"
121     } {
122       \mode_if_vertical:TF{
123         \tl_use:N \l__stex_annotate_arg_tl\par
124       }{
125         \tl_use:N \l__stex_annotate_arg_tl
126       }
127     }
128   }
129   \cs_new_protected:Nn \stex_annotate_invisible:n {
130     \_stex_annotate_checkempty:n { #1 }
131     \rustex_annotate_HTML:nn {
132       stex:visible="false" ~
133       style:display="none"
134     } {
135       \mode_if_vertical:TF{
136         \tl_use:N \l__stex_annotate_arg_tl\par
137       }{
138         \tl_use:N \l__stex_annotate_arg_tl
139       }
```

```
140        }
141      }
142      \cs_new_protected:Nn \stex_annotate_invisible:nnn {
143        \__stex_annotate_checkempty:n { #3 }
144        \rustex_annotate_HTML:nn {
145          property="stex:#1" ~
146          resource="#2" ~
147          stex:visible="false" ~
148          style:display="none"
149        } {
150          \mode_if_vertical:TF{
151            \tl_use:N \l__stex_annotate_arg_tl\par
152          }{
153            \tl_use:N \l__stex_annotate_arg_tl
154          }
155        }
156      }
157      \NewDocumentEnvironment{stex_annotate_env} { m m } {
158        \par
159        \rustex_annotate_HTML_begin:n {
160          property="stex:#1" ~
161          resource="#2"
162        }
163      }{
164        \par\rustex_annotate_HTML_end:
165      }
166    }{
167      \latexml_if:TF {
168        \cs_new_protected:Nn \stex_annotate:nnn {
169          \__stex_annotate_checkempty:n { #3 }
170          \mode_if_math:TF {
171            \cs:w latexml@annotate@math\cs_end:{#1}{#2}{
172              \tl_use:N \l__stex_annotate_arg_tl
173            }
174          }{
175            \cs:w latexml@annotate@text\cs_end:{#1}{#2}{
176              \tl_use:N \l__stex_annotate_arg_tl
177            }
178          }
179        }
180        \cs_new_protected:Nn \stex_annotate_invisible:n {
181          \__stex_annotate_checkempty:n { #1 }
182          \mode_if_math:TF {
183            \cs:w latexml@invisible@math\cs_end:{
184              \tl_use:N \l__stex_annotate_arg_tl
185            }
186          } {
187            \cs:w latexml@invisible@text\cs_end:{
188              \tl_use:N \l__stex_annotate_arg_tl
189            }
190          }
191        }
192        \cs_new_protected:Nn \stex_annotate_invisible:nnn {
193          \__stex_annotate_checkempty:n { #3 }
```

```
194        \cs:w latexml@annotate@invisible\cs_end:{#1}{#2}{
195          \tl_use:N \l__stex_annotate_arg_tl
196        }
197      }
198      \NewDocumentEnvironment{stex_annotate_env} { m m } {
199        \par\begin{latexml@annotateenv}{#1}{#2}
200      }{
201        \par\end{latexml@annotateenv}
202      }
203    }{
204      \cs_new_protected:Nn \stex_annotate:nnn {#3}
205      \cs_new_protected:Nn \stex_annotate_invisible:n {}
206      \cs_new_protected:Nn \stex_annotate_invisible:nnn {}
207      \NewDocumentEnvironment{stex_annotate_env} { m m } {}{}
208    }
209 }
```

(*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

```
210 ⟨@@=stex_language⟩
```

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

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

```
211 \prop_const_from_keyval:Nn \c_stex_languages_prop {
212    en = english ,
213    de = ngerman ,
214    ar = arabic ,
215    bg = bulgarian ,
216    ru = russian ,
217    fi = finnish ,
218    ro = romanian ,
219    tr = turkish ,
220    fr = french
221 }
222
223 \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop {
224    english   = en ,
225    ngerman   = de ,
226    arabic    = ar ,
227    bulgarian = bg ,
228    russian   = ru ,
229    finnish   = fi ,
230    romanian  = ro ,
231    turkish   = tr ,
232    french    = fr
233 }
234 % todo: chinese simplified (zhs)
235 %        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:

```
236  \clist_if_empty:NF \c_stex_languages_clist {
237    \clist_clear:N \l_tmpa_clist
238    \clist_map_inline:Nn \c_stex_languages_clist {
239      \prop_get:NnNTF \c_stex_languages_prop { # 1 } \l_tmpa_str {
240        \clist_put_right:No \l_tmpa_clist \l_tmpa_str
241      } {
242        \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
243      }
244    }
245    \stex_debug:nn{lang} {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
246    \RequirePackage[\clist_use:Nn \l_tmpa_clist,]{babel}
247  }
```

24.6 Auxiliary Methods

\stex_deactivate_macro:Nn

```
248  \cs_new_protected:Nn \stex_deactivate_macro:Nn {
249    \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
250    \def#1{
251      \msg_error:nnnn{stex}{error/deactivated-macro}{\detokenize{#1}}{#2}
252    }
253  }
```

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

\stex_reactivate_macro:N

```
254  \cs_new_protected:Nn \stex_reactivate_macro:N {
255    \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname
256  }
```

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

\ignorespacesandpars

```
257  \protected\def\ignorespacesandpars{
258    \begingroup\catcode13=10\relax
259    \@ifnextchar\par{
260      \endgroup\expandafter\ignorespacesandpars\@gobble
261    }{
262      \endgroup
263    }
264  }
265  ⟨/package⟩
```

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

Chapter 25

STEX -MathHub Implementation

```
266 ⟨*package⟩
267
268 %%%%%%%%%%%%   mathhub.dtx   %%%%%%%%%%%%
269
270 ⟨@@=stex_path⟩
```

Warnings and error messages

```
271 \msg_new:nnn{stex}{error/norepository}{
272   No~archive~#1~found~in~#2
273 }
274 \msg_new:nnn{stex}{error/notinarchive}{
275   Not~currently~in~an~archive,~but~\detokenize{#1}~
276   needs~one!
277 }
278 \msg_new:nnn{stex}{error/nofile}{
279   \detokenize{#1}~could~not~find~file~#2
280 }
281 \msg_new:nnn{stex}{error/twofiles}{
282   \detokenize{#1}~found~two~candidates~for~#2
283 }
```

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

```
284 \cs_new_protected:Nn \stex_path_from_string:Nn {
285   \str_set:Nx \l_tmpa_str { #2 }
286   \str_if_empty:NTF \l_tmpa_str {
287     \seq_clear:N #1
288   }{
289     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
290     \sys_if_platform_windows:T{
291       \seq_clear:N \l_tmpa_tl
```

```
292       \seq_map_inline:Nn #1 {
293           \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
294           \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
295       }
296       \seq_set_eq:NN #1 \l_tmpa_tl
297   }
298   \stex_path_canonicalize:N #1
299   }
300 }
301
```

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

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

```
302 \cs_new_protected:Nn \stex_path_to_string:NN {
303   \exp_args:NNe \str_set:Nn #2 { \seq_use:Nn #1 / }
304 }
305
306 \cs_new:Nn \stex_path_to_string:N {
307   \seq_use:Nn #1 /
308 }
```

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

`\c__stex_path_dot_str`   `.` and `..`, respectively.
`\c__stex_path_up_str`

```
309 \str_const:Nn \c__stex_path_dot_str {.}
310 \str_const:Nn \c__stex_path_up_str {..}
```

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

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

```
311 \cs_new_protected:Nn \stex_path_canonicalize:N {
312   \seq_if_empty:NF #1 {
313     \seq_clear:N \l_tmpa_seq
314     \seq_get_left:NN #1 \l_tmpa_tl
315     \str_if_empty:NT \l_tmpa_tl {
316       \seq_put_right:Nn \l_tmpa_seq {}
317     }
318     \seq_map_inline:Nn #1 {
319       \str_set:Nn \l_tmpa_tl { ##1 }
320       \str_if_eq:NNF \l_tmpa_tl \c__stex_path_dot_str {
321         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
322           \seq_if_empty:NTF \l_tmpa_seq {
323             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
324               \c__stex_path_up_str
325             }
326           }{
327             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
328             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
329               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
330                 \c__stex_path_up_str
331               }
332             }{
```

```
333                        \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
334                      }
335                    }
336                }{
337                  \str_if_empty:NF \l_tmpa_tl {
338                    \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
339                  }
340                }
341              }
342          }
343        \seq_gset_eq:NN #1 \l_tmpa_seq
344      }
345  }
```

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

`\stex_path_if_absolute_p:N`
`\stex_path_if_absolute:N`*TF*

```
346  \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
347    \seq_if_empty:NTF #1 {
348      \prg_return_false:
349    }{
350      \seq_get_left:NN #1 \l_tmpa_tl
351      \sys_if_platform_windows:TF{
352        \str_if_in:NnTF \l_tmpa_tl {:}{
353          \prg_return_true:
354        }{
355          \prg_return_false:
356        }
357      }{
358        \str_if_empty:NTF \l_tmpa_tl {
359          \prg_return_true:
360        }{
361          \prg_return_false:
362        }
363      }
364    }
365  }
```

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

25.2   PWD and kpsewhich

`\stex_kpsewhich:n`

```
366  \str_new:N\l_stex_kpsewhich_return_str
367  \cs_new_protected:Nn \stex_kpsewhich:n {
368    \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
369    \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
370    \tl_trim_spaces:N \l_stex_kpsewhich_return_str
371  }
```

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

We determine the PWD

`\c_stex_pwd_seq`
`\c_stex_pwd_str`

```
372 \sys_if_platform_windows:TF{
373   \begingroup\escapechar=-1\catcode'\\=12
374   \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
375   \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
376   \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_
377 }{
378   \stex_kpsewhich:n{-var-value~PWD}
379 }
380
381 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
382 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
383 \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

```
384 ⟨@@=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 ST_EX-purposes.

`\g__stex_files_stack`   keeps track of file changes

```
385 \seq_gclear_new:N\g__stex_files_stack
```

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

`\c_stex_mainfile_seq`
`\c_stex_mainfile_str`

```
386 \str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
387 \stex_path_from_string:Nn \c_stex_mainfile_seq
388   \c_stex_mainfile_str
```

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

`\g_stex_currentfile_seq`

```
389 \seq_gclear_new:N\g_stex_currentfile_seq
```

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

`\stex_filestack_push:n`

```
390 \cs_new_protected:Nn \stex_filestack_push:n {
391   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
392   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
393     \stex_path_from_string:Nn\g_stex_currentfile_seq{
394       \c_stex_pwd_str/#1
395     }
396   }
397   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
398   \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
399 }
```

105

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

`\stex_filestack_pop:`

```
400 \cs_new_protected:Nn \stex_filestack_pop: {
401   \seq_if_empty:NF\g__stex_files_stack{
402     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
403   }
404   \seq_if_empty:NTF\g__stex_files_stack{
405     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
406   }{
407     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
408     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
409   }
410 }
```

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

Hooks for the current file:

```
411 \AddToHook{file/before}{
412   \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
413 }
414 \AddToHook{file/after}{
415   \stex_filestack_pop:
416 }
```

25.4   MathHub Repositories

```
417 ⟨@@=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.

```
418 \str_if_empty:NTF\mathhub{
419   \sys_if_platform_windows:TF{
420     \begingroup\escapechar=-1\catcode`\\=12
421     \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str MATHHUB\c_percent_str}
422     \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
423     \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_ste
424   }{
425     \stex_kpsewhich:n{-var-value~MATHHUB}
426   }
427   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
428
429   \str_if_empty:NTF\c_stex_mathhub_str{
430     \msg_warning:nn{stex}{warning/nomathhub}
431   }{
432     \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
433     \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
434   }
435 }{
436   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
437   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
438     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
439       \c_stex_pwd_str/\mathhub
440     }
```

```
441    }
442    \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
443    \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
444 }
```

(*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

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

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

`\l__stex_mathhub_manifest_file_seq`

```
461 \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`:

```
462 \cs_new_protected:Nn \__stex_mathhub_find_manifest:N {
463    \seq_set_eq:NN\l_tmpa_seq #1
464    \bool_set_true:N\l_tmpa_bool
465    \bool_while_do:Nn \l_tmpa_bool {
466       \seq_if_empty:NTF \l_tmpa_seq {
467          \bool_set_false:N\l_tmpa_bool
468       }{
469          \file_if_exist:nTF{
470             \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
471          }{
472             \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
473             \bool_set_false:N\l_tmpa_bool
474          }{
475             \file_if_exist:nTF{
476                \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
477             }{
478                \seq_put_right:Nn\l_tmpa_seq{META-INF}
479                \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
```

```
480              \bool_set_false:N\l_tmpa_bool
481            }{
482              \file_if_exist:nTF{
483                \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
484              }{
485                \seq_put_right:Nn\l_tmpa_seq{meta-inf}
486                \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
487                \bool_set_false:N\l_tmpa_bool
488              }{
489                \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
490              }
491            }
492          }
493        }
494      }
495      \seq_set_eq:NN\l__stex_mathhub_manifest_file_seq\l_tmpa_seq
496  }
```

(End definition for __stex_mathhub_find_manifest:N.*)*

<div style="text-align:right">\c__stex_mathhub_manifest_ior</div>

File variable used for MANIFEST-files

```
497  \ior_new:N \c__stex_mathhub_manifest_ior
```

(End definition for \c__stex_mathhub_manifest_ior.*)*

<div style="text-align:right">__stex_mathhub_parse_manifest:n</div>

Stores the entries in manifest file in the corresponding property list:

```
498  \cs_new_protected:Nn \__stex_mathhub_parse_manifest:n {
499    \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
500    \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
501    \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
502      \str_set:Nn \l_tmpa_str {##1}
503      \exp_args:NNoo \seq_set_split:Nnn
504          \l_tmpb_seq \c_colon_str \l_tmpa_str
505      \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
506        \exp_args:NNe \str_set:Nn \l_tmpb_tl {
507          \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
508        }
509        \exp_args:No \str_case:nnTF \l_tmpa_tl {
510          {id} {
511            \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
512              { id } \l_tmpb_tl
513          }
514          {narration-base} {
515            \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
516              { narr } \l_tmpb_tl
517          }
518          {url-base} {
519            \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
520              { docurl } \l_tmpb_tl
521          }
522          {source-base} {
523            \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
524              { ns } \l_tmpb_tl
525          }
```

```
526        {ns} {
527          \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
528            { ns } \l_tmpb_tl
529        }
530        {dependencies} {
531          \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
532            { deps } \l_tmpb_tl
533        }
534      }{}{}
535    }{}
536  }
537  \ior_close:N \c__stex_mathhub_manifest_ior
538 }
```

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

```
539 \cs_new_protected:Nn \stex_set_current_repository:n {
540   \stex_require_repository:n { #1 }
541   \prop_set_eq:Nc \l_stex_current_repository_prop {
542     c_stex_mathhub_#1_manifest_prop
543   }
544 }
```

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

\stex_require_repository:n

```
545 \cs_new_protected:Nn \stex_require_repository:n {
546   \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
547     \stex_debug:nn{mathhub}{Opening~archive:~#1}
548     \__stex_mathhub_do_manifest:n { #1 }
549   }
550 }
```

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

\l_stex_current_repository_prop   Current MathHub repository

```
551 %\prop_new:N \l_stex_current_repository_prop
552
553 \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
554 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
555   \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
556 } {
557   \__stex_mathhub_parse_manifest:n { main }
558   \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
559     \l_tmpa_str
560   \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str _manifest_prop }
561     \c_stex_mathhub_main_manifest_prop
562   \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
563   \stex_debug:nn{mathhub}{Current~repository:~
564     \prop_item:Nn \l_stex_current_repository_prop {id}
565   }
566 }
```

(*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.

```
567 \cs_new_protected:Nn \stex_in_repository:nn {
568   \str_set:Nx \l_tmpa_str { #1 }
569   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
570   \str_if_empty:NTF \l_tmpa_str {
571     \prop_if_exist:NTF \l_stex_current_repository_prop {
572       \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
573       \exp_args:Ne \l_tmpa_cs{
574         \prop_item:Nn \l_stex_current_repository_prop { id }
575       }
576     }{
577       \l_tmpa_cs{}
578     }
579   }{
580     \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}
581     \stex_require_repository:n \l_tmpa_str
582     \str_set:Nx \l_tmpa_str { #1 }
583     \exp_args:Nne \use:nn {
584       \stex_set_current_repository:n \l_tmpa_str
585       \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
586     }{
587       \stex_debug:nn{mathhub}{switching~back~to:~
588         \prop_if_exist:NTF \l_stex_current_repository_prop {
589           \prop_item:Nn \l_stex_current_repository_prop { id }:~
590           \meaning\l_stex_current_repository_prop
591         }{
592           no~repository
593         }
594       }
595       \prop_if_exist:NTF \l_stex_current_repository_prop {
596         \stex_set_current_repository:n {
597           \prop_item:Nn \l_stex_current_repository_prop { id }
598         }
599       }{
600         \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
601       }
602     }
603   }
604 }
```

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

25.5   Using Content in Archives

\mhpath

```
605 \def \mhpath #1 #2 {
606   \exp_args:Ne \tl_if_empty:nTF{#1}{
607     \c_stex_mathhub_str /
608       \prop_item:Nn \l_stex_current_repository_prop { id }
609       / source / #2
610   }{
611     \c_stex_mathhub_str / #1 / source / #2
```

`\inputref`
`\mhinput`

```
612       }
613     }
614   \newif \ifinputref \inputreffalse
615
616   \cs_new_protected:Nn \__stex_mathhub_mhinput:nn {
617     \stex_in_repository:nn {#1} {
618       \ifinputref
619         \input{ \c_stex_mathhub_str / ##1 / source / #2 }
620       \else
621         \inputreftrue
622         \input{ \c_stex_mathhub_str / ##1 / source / #2 }
623         \inputreffalse
624       \fi
625     }
626   }
627   \NewDocumentCommand \mhinput { O{} m}{
628     \stex_mhinput:nn{ #1 }{ #2 }
629   }
630
631   \cs_new_protected:Nn \__stex_mathhub_inputref:nn {
632     \stex_in_repository:nn {#1} {
633       \bool_lazy_any:nTF {
634         {\rustex_if_p:}
635         {\latexml_if_p:}
636       } {
637         \str_clear:N \l_tmpa_str
638         \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
639           \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
640         }
641         \stex_annotate_invisible:nnn{inputref}{
642           \l_tmpa_str / #2
643         }{}
644       }{
645         \begingroup
646           \inputreftrue
647           \tl_if_empty:nTF{ ##1 }{
648             \input{#2}
649           }{
650             \input{ \c_stex_mathhub_str / ##1 / source / #2 }
651           }
652         \endgroup
653       }
654     }
655   }
656   \NewDocumentCommand \inputref { O{} m}{
657     \__stex_mathhub_inputref:nn{ #1 }{ #2 }
658   }
```

```
659 \cs_new_protected:Nn \__stex_mathhub_mhbibresource:nn {
660   \stex_in_repository:nn {#1} {
661     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
662   }
663 }
664 \newcommand\addmhbibresource[2][]{
665   \__stex_mathhub_mhbibresource:nn{ #1 }{ #2 }
666 }
```

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

\libinput

```
667 \cs_new_protected:Npn \libinput #1 {
668   \prop_if_exist:NF \l_stex_current_repository_prop {
669     \msg_error:nnn{stex}{error/notinarchive}\libinput
670   }
671   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
672     \msg_error:nnn{stex}{error/notinarchive}\libinput
673   }
674   \seq_clear:N \l__stex_mathhub_libinput_files_seq
675   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
676   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
677
678   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
679     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #1.tex}
680     \IfFileExists{ \l_tmpa_str }{
681       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
682     }{}
683     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
684     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
685   }
686
687   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}
688   \IfFileExists{ \l_tmpa_str }{
689     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
690   }{}
691
692   \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
693     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
694   }{
695     \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
696       \input{ ##1 }
697     }
698   }
699 }
```

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

\libusepackage

```
700 \NewDocumentCommand \libusepackage {O{} m} {
701   \prop_if_exist:NF \l_stex_current_repository_prop {
702     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
703   }
```

112

```
704    \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
705      \msg_error:nnn{stex}{error/notinarchive}\libusepackage
706    }
707    \seq_clear:N \l__stex_mathhub_libinput_files_seq
708    \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
709    \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
710
711    \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
712      \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #2}
713      \IfFileExists{ \l_tmpa_str.sty }{
714        \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
715      }{}
716      \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
717      \seq_put_right:No \l_tmpa_seq \l_tmpa_str
718    }
719
720    \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2}
721    \IfFileExists{ \l_tmpa_str.sty }{
722      \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
723    }{}
724
725    \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
726      \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
727    }{
728      \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
729        \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
730          \usepackage[#1]{ ##1 }
731        }
732      }{
733        \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
734      }
735    }
736  }
```

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

`\mhgraphics`
`\cmhgraphics`

```
737
738  \AddToHook{begindocument}{
739  \ltx@ifpackageloaded{graphicx}{
740      \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
741      \newcommand\mhgraphics[2][]{%
742        \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
743        \includegraphics[#1]{\mhpath\Gin@mhrepos{#2}}}
744      \newcommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
745  }{}
```

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

`\lstinputmhlisting`
`\clstinputmhlisting`

```
746  \ltx@ifpackageloaded{listings}{
747      \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
748      \newcommand\lstinputmhlisting[2][]{%
749        \def\lst@mhrepos{}\setkeys{lst}{#1}%
750        \lstinputlisting[#1]{\mhpath\lst@mhrepos{#2}}}
```

```
751      \newcommand\clstinputmhlisting[2][]{\begin{center}\lstinputmhlisting[#1]{#2}\end{center}
752    }{}
753  }
754
755  ⟨/package⟩
```

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

Chapter 26

sTEX
-References Implementation

```
756 ⟨*package⟩
757
758 %%%%%%%%%%%%   references.dtx   %%%%%%%%%%%%
759
760 ⟨@@=stex_refs⟩
```

Warnings and error messages

```
761
```

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

```
762 %\iow_new:N \c__stex_refs_refs_iow
763 \AddToHook{begindocument}{
764 %  \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
765 }
766 \AddToHook{enddocument}{
767 %  \iow_close:N \c__stex_refs_refs_iow
768 }
```

\STEXreftitle

```
769 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
770
771 \NewDocumentCommand \STEXreftitle { m } {
772   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
773 }
```

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

26.1   Document URIs and URLs

\l_stex_current_docns_str

```
774 \str_new:N \l_stex_current_docns_str
```

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

```
775 \cs_new_protected:Nn \stex_get_document_uri: {
776   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
777   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
778   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
779   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
780   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
781
782   \str_clear:N \l_tmpa_str
783   \prop_if_exist:NT \l_stex_current_repository_prop {
784     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
785       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
786     }
787   }
788
789   \str_if_empty:NTF \l_tmpa_str {
790     \str_set:Nx \l_stex_current_docns_str {
791       file:/\stex_path_to_string:N \l_tmpa_seq
792     }
793   }{
794     \bool_set_true:N \l_tmpa_bool
795     \bool_while_do:Nn \l_tmpa_bool {
796       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
797       \exp_args:No \str_case:nnTF { \l_tmpb_str } {
798         {source} { \bool_set_false:N \l_tmpa_bool }
799       }{}{
800         \seq_if_empty:NT \l_tmpa_seq {
801           \bool_set_false:N \l_tmpa_bool
802         }
803       }
804     }
805
806     \seq_if_empty:NTF \l_tmpa_seq {
807       \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
808     }{
809       \str_set:Nx \l_stex_current_docns_str {
810         \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
811       }
812     }
813   }
814 }
```

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

```
815 \str_new:N \l_stex_current_docurl_str
```

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

```
816 \cs_new_protected:Nn \stex_get_document_url: {
817   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
818   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
819   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
```

```
820    \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
821    \seq_put_right:No \l_tmpa_seq \l_tmpb_str
822
823    \str_clear:N \l_tmpa_str
824    \prop_if_exist:NT \l_stex_current_repository_prop {
825      \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
826        \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
827          \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
828        }
829      }
830    }
831
832    \str_if_empty:NTF \l_tmpa_str {
833      \str_set:Nx \l_stex_current_docurl_str {
834        file:/\stex_path_to_string:N \l_tmpa_seq
835      }
836    }{
837      \bool_set_true:N \l_tmpa_bool
838      \bool_while_do:Nn \l_tmpa_bool {
839        \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
840        \exp_args:No \str_case:nnTF { \l_tmpb_str } {
841          {source} { \bool_set_false:N \l_tmpa_bool }
842        }{}{
843          \seq_if_empty:NT \l_tmpa_seq {
844            \bool_set_false:N \l_tmpa_bool
845          }
846        }
847      }
848
849      \seq_if_empty:NTF \l_tmpa_seq {
850        \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
851      }{
852        \str_set:Nx \l_stex_current_docurl_str {
853          \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
854        }
855      }
856    }
857  }
```

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

26.2 Setting Reference Targets

```
858  \str_const:Nn \c__stex_refs_url_str{URL}
859  \str_const:Nn \c__stex_refs_ref_str{REF}
860  \str_new:N \l__stex_refs_curr_label_str
861  % @currentlabel -> number
862  % @currentlabelname -> title
863  % @currentHref -> name.number <- id of some kind
864  % \theH# -> \arabic{section}
865  % \the#  -> number
866  % \hyper@makecurrent{#}
867  \int_new:N \l__stex_refs_unnamed_counter_int
```

```
868 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
869   \stex_get_document_uri:
870   \str_clear:N \l__stex_refs_curr_label_str
871   \str_set:Nx \l_tmpa_str { #1 }
872   \str_if_empty:NT \l_tmpa_str {
873     \int_incr:N \l__stex_refs_unnamed_counter_int
874     \str_set:Nx \l_tmpa_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
875   }
876   \str_set:Nx \l__stex_refs_curr_label_str {
877     \l_stex_current_docns_str?\l_tmpa_str
878   }
879   \seq_if_exist:cF{g__stex_refs_labels_\l_tmpa_str _seq}{
880     \seq_new:c {g__stex_refs_labels_\l_tmpa_str _seq}
881   }
882   \seq_if_in:coF{g__stex_refs_labels_\l_tmpa_str _seq}\l__stex_refs_curr_label_str {
883     \seq_gput_right:co{g__stex_refs_labels_\l_tmpa_str _seq}\l__stex_refs_curr_label_str
884   }
885   \stex_if_smsmode:TF {
886     \stex_get_document_url:
887     \str_gset_eq:cN {sref_url_\l__stex_refs_curr_label_str _str}\l_stex_current_docurl_str
888     \str_gset_eq:cN {sref_\l__stex_refs_curr_label_str _type}\c__stex_refs_url_str
889   }{
890     %\iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=~\expandafter\unexpanded\expandafter{
891     \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
892     \immediate\write\@auxout{\stexauxadddocref{\l_stex_current_docns_str}{\l_tmpa_str}}
893     \str_gset:cx {sref_\l__stex_refs_curr_label_str _type}\c__stex_refs_ref_str
894   }
895 }
```

(*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.

```
896 \cs_new_protected:Npn \stexauxadddocref #1 #2 {
897   \str_set:Nn \l_tmpa_str {#1?#2}
898   \str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
899   \seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
900     \seq_new:c {g__stex_refs_labels_#2_seq}
901   }
902   \seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
903     \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
904   }
905 }
```

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

```
906 \AtEndDocument{
907   \def\stexauxadddocref#1 #2 {}{}
908 }
```

```
909 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
910   \stex_if_smsmode:TF {
911     \str_if_exist:cF{sref_sym_#1_type}{
912       \stex_get_document_url:
913       \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str
```

118

```
914      \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
915    }
916  }{
917    \str_if_empty:NF \l__stex_refs_curr_label_str {
918      \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
919      \immediate\write\@auxout{
920        \exp_not:N\expandafter\def\exp_not:N\csname \exp_not:N\detokenize{sref_sym_#1_label_
921          \l__stex_refs_curr_label_str
922        }
923      }
924    }
925  }
926 }
```

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

26.3   Using References

```
927 \str_new:N \l__stex_refs_indocument_str
```

\sref   Optional arguments:

```
928
929 \keys_define:nn { stex / sref } {
930    linktext      .tl_set:N  = \l__stex_refs_linktext_tl ,
931    fallback      .tl_set:N  = \l__stex_refs_fallback_tl ,
932    pre           .tl_set:N  = \l__stex_refs_pre_tl ,
933    post          .tl_set:N  = \l__stex_refs_post_tl ,
934 }
935 \cs_new_protected:Nn \__stex_refs_args:n {
936    \tl_clear:N \l__stex_refs_linktext_tl
937    \tl_clear:N \l__stex_refs_fallback_tl
938    \tl_clear:N \l__stex_refs_pre_tl
939    \tl_clear:N \l__stex_refs_post_tl
940    \str_clear:N \l__stex_refs_repo_str
941    \keys_set:nn { stex / sref } { #1 }
942 }
```

The actual macro:

```
943 \NewDocumentCommand \sref { O{} m}{
944    \__stex_refs_args:n { #1 }
945    \str_if_empty:NTF \l__stex_refs_indocument_str {
946      \str_set:Nx \l_tmpa_str { #2 }
947      \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
948      \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
949        \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
950          \seq_get_left:cNF {g__stex_refs_labels_\l_tmpa_str _seq} \l_tmpa_str {
951            \str_clear:N \l_tmpa_str
952          }
953        }{
954          \str_clear:N \l_tmpa_str
955        }
956      }{
957        \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
958        \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
```

```
959        \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
960        \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
961          \str_set_eq:NN \l_tmpc_str \l_tmpa_str
962          \str_clear:N \l_tmpa_str
963          \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str _seq} {
964            \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
965              \str_range:nnn { ##1 }{ -\l_tmpa_int}{ -1 }
966            }{
967              \seq_map_break:n {
968                \str_set:Nn \l_tmpa_str { ##1 }
969              }
970            }
971          }
972        }{
973          \str_clear:N \l_tmpa_str
974        }
975      }
976      \str_if_empty:NTF \l_tmpa_str {
977        \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_lin
978      }{
979        \str_if_eq:cNTF {sref_\l_tmpa_str _type} \c__stex_refs_ref_str {
980          \tl_if_empty:NTF \l__stex_refs_linktext_tl {
981            \cs_if_exist:cTF{autoref}{
982              \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
983            }{
984              \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
985            }
986          }{
987            \ltx@ifpackageloaded{hyperref}{
988              \hyperref[sref_\l_tmpa_str]\l__stex_refs_linktext_tl
989            }{
990              \l__stex_refs_linktext_tl
991            }
992          }
993        }{
994          \ltx@ifpackageloaded{hyperref}{
995            \href{\use:c{sref_url_\l_tmpa_str _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_t
996          }{
997            \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs
998          }
999        }
1000     }
1001   }{
1002     % TODO
1003   }
1004 }
```

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

\srefsym

```
1005 \NewDocumentCommand \srefsym { O{} m}{
1006   \stex_get_symbol:n { #2 }
1007   \__stex_refs_sym_aux:nn{#1}{\l_stex_get_symbol_uri_str}
1008 }
```

120

```
1009
1010 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1011   \str_if_exist:cTF {sref_sym_#2 _label_str }{
1012     \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1013   }{
1014     \__stex_refs_args:n { #1 }
1015     \str_if_empty:NTF \l__stex_refs_indocument_str {
1016       \tl_if_exist:cTF{sref_sym_#2 _type}{
1017         % doc uri in \l_tmpb_str
1018         \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1019         \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1020           % reference
1021           \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1022             \cs_if_exist:cTF{autoref}{
1023               \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1024             }{
1025               \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1026             }
1027           }{
1028             \ltx@ifpackageloaded{hyperref}{
1029               \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1030             }{
1031               \l__stex_refs_linktext_tl
1032             }
1033           }
1034         }{
1035           % URL
1036           \ltx@ifpackageloaded{hyperref}{
1037             \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl \
1038           }{
1039             \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_re
1040           }
1041         }
1042       }{
1043         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_l
1044       }
1045     }{
1046       % TODO
1047     }
1048   }
1049 }
```

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

```
1050 \cs_new_protected:Npn \srefsymuri #1 #2 {
1051   \__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1052 }
```

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

```
1053 ⟨/package⟩
```

Chapter 27

sTeX
-Modules Implementation

```
1054 ⟨*package⟩
1055
1056 %%%%%%%%%%%%    modules.dtx    %%%%%%%%%%%%
1057
1058 ⟨@@=stex_modules⟩
```

Warnings and error messages
```
1059 \msg_new:nnn{stex}{error/unknownmodule}{
1060   No~module~#1~found
1061 }
1062 \msg_new:nnn{stex}{error/syntax}{
1063   Syntax~error:~#1
1064 }
1065 \msg_new:nnn{stex}{error/siglanguage}{
1066   Module~#1~declares~signature~#2,~but~does~not~
1067   declare~its~language
1068 }
1069 \msg_new:nnn{stex}{warning/deprecated}{
1070   #1~is~deprecated;~please~use~#2~instead!
1071 }
1072
1073 \msg_new:nnn{stex}{error/conflictingmodules}{
1074   Conflicting~imports~for~module~#1
1075 }
```

\l_stex_current_module_str   The current module:
```
1076 \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
```
1077 \seq_new:N \l_stex_all_modules_seq
```

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

`\stex_if_in_module_p:`
`\stex_if_in_module:`*TF*

```
1078 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1079   \str_if_empty:NTF \l_stex_current_module_str
1080     \prg_return_false: \prg_return_true:
1081 }
```

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

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

```
1082 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1083   \prop_if_exist:cTF { c_stex_module_#1_prop }
1084     \prg_return_true: \prg_return_false:
1085 }
```

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

`\stex_add_to_current_module:n`
`\STEXexport`

Only allowed within modules:

```
1086 \cs_new_protected:Nn \stex_add_to_current_module:n {
1087   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1088 }
1089 \cs_new_protected:Npn \STEXexport {
1090   \begingroup
1091   \newlinechar=-1\relax
1092   \endlinechar=-1\relax
1093   %\catcode`\ = 9\relax
1094   \expandafter\endgroup\__stex_modules_export:n
1095 }
1096 \cs_new_protected:Nn \__stex_modules_export:n {
1097   \ignorespaces #1
1098   \stex_add_to_current_module:n { \ignorespaces #1 }
1099   \stex_smsmode_do:
1100 }
1101 \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*.)

`\stex_add_constant_to_current_module:n`

```
1102 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1103   \str_set:Nx \l_tmpa_str { #1 }
1104   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
1105 }
```

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

`\stex_add_import_to_current_module:n`

```
1106 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1107   \str_set:Nx \l_tmpa_str { #1 }
1108   \exp_args:Nno
1109   \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1110     \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1111   }
1112 }
```

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

`\stex_collect_imports:n`

```
1113 \cs_new_protected:Nn \stex_collect_imports:n {
1114   \seq_clear:N \l_stex_collect_imports_seq
1115   \__stex_modules_collect_imports:n {#1}
1116 }
1117 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1118   \seq_map_inline:cn {c_stex_module_#1_imports} {
1119     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1120       \__stex_modules_collect_imports:n { ##1 }
1121     }
1122   }
1123   \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1124     \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1125   }
1126 }
```

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

`\stex_do_up_to_module:n`

```
1127 \int_new:N \l__stex_modules_group_depth_int
1128 \tl_new:N \l__stex_modules_aftergroup_tl
1129 \cs_new_protected:Nn \stex_do_up_to_module:n {
1130   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1131     #1
1132   }{
1133     #1
1134     \expandafter \tl_gset:Nn \expandafter \l__stex_modules_aftergroup_tl \expandafter { \l__
1135     \aftergroup\__stex_modules_aftergroup_do:
1136   }
1137 }
1138 \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1139   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1140     \l__stex_modules_aftergroup_tl
1141     \tl_clear:N \l__stex_modules_aftergroup_tl
1142   }{
1143     \l__stex_modules_aftergroup_tl
1144     \aftergroup\__stex_modules_aftergroup_do:
1145   }
1146 }
1147 \cs_new_protected:Nn \_stex_reset_up_to_module: {
1148
1149   \tl_gset_eq:NN \l__stex_modules_aftergroup_tl \l__stex_modules_aftergroup_outer_tl
1150 }
```

(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`).

```
1151
```

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

124

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

```
1152 \str_new:N \l_stex_modules_ns_str
1153 \str_new:N \l_stex_modules_subpath_str
1154 \cs_new_protected:Nn \__stex_modules_compute_namespace:nN {
1155   \str_set:Nx \l_tmpa_str { #1 }
1156   \seq_set_eq:NN \l_tmpa_seq #2
1157   % split off file extension
1158   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1159   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1160   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1161   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1162
1163   \bool_set_true:N \l_tmpa_bool
1164   \bool_while_do:Nn \l_tmpa_bool {
1165     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1166     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1167       {source} { \bool_set_false:N \l_tmpa_bool }
1168     }{}{
1169       \seq_if_empty:NT \l_tmpa_seq {
1170         \bool_set_false:N \l_tmpa_bool
1171       }
1172     }
1173   }
1174
1175   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
1176   \str_if_empty:NTF \l_stex_modules_subpath_str {
1177     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
1178   }{
1179     \str_set:Nx \l_stex_modules_ns_str {
1180       \l_tmpa_str/\l_stex_modules_subpath_str
1181     }
1182   }
1183 }
1184
1185 \cs_new_protected:Nn \stex_modules_current_namespace: {
1186   \str_clear:N \l_stex_modules_subpath_str
1187   \prop_if_exist:NTF \l_stex_current_repository_prop {
1188     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1189     \__stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1190   }{
1191     % split off file extension
1192     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1193     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1194     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1195     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1196     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1197     \str_set:Nx \l_stex_modules_ns_str {
1198       file:/\stex_path_to_string:N \l_tmpa_seq
1199     }
1200   }
1201 }
```

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

27.1 The `smodule` environment

`smodule` arguments:

```
1202 \keys_define:nn { stex / module } {
1203   title         .tl_set:N    = \smoduletitle ,
1204   type          .str_set_x:N = \smoduletype ,
1205   id            .str_set_x:N = \smoduleid ,
1206   deprecate     .str_set_x:N = \l_stex_module_deprecate_str ,
1207   ns            .str_set_x:N = \l_stex_module_ns_str ,
1208   lang          .str_set_x:N = \l_stex_module_lang_str ,
1209   sig           .str_set_x:N = \l_stex_module_sig_str ,
1210   creators      .str_set_x:N = \l_stex_module_creators_str ,
1211   contributors  .str_set_x:N = \l_stex_module_contributors_str ,
1212   meta          .str_set_x:N = \l_stex_module_meta_str ,
1213   srccite       .str_set_x:N = \l_stex_module_srccite_str
1214 }
1215
1216 \cs_new_protected:Nn \__stex_modules_args:n {
1217   \str_clear:N \smoduletitle
1218   \str_clear:N \smoduletype
1219   \str_clear:N \smoduleid
1220   \str_clear:N \l_stex_module_ns_str
1221   \str_clear:N \l_stex_module_deprecate_str
1222   \str_clear:N \l_stex_module_lang_str
1223   \str_clear:N \l_stex_module_sig_str
1224   \str_clear:N \l_stex_module_creators_str
1225   \str_clear:N \l_stex_module_contributors_str
1226   \str_clear:N \l_stex_module_meta_str
1227   \str_clear:N \l_stex_module_srccite_str
1228   \keys_set:nn { stex / module } { #1 }
1229 }
1230
1231 % module parameters here? In the body?
1232
```

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

```
1233 \cs_new_protected:Nn \stex_module_setup:nn {
1234   \tl_gset_eq:NN \l__stex_modules_aftergroup_outer_tl \l__stex_modules_aftergroup_tl
1235   \tl_clear:N \l__stex_modules_aftergroup_tl
1236   \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1237   \str_set:Nx \l_stex_module_name_str { #2 }
1238   \__stex_modules_args:n { #1 }
```

First, we set up the name and namespace of the module.

Are we in a nested module?

```
1239   \stex_if_in_module:TF {
1240     % Nested module
1241     \prop_get:cnN {c_stex_module_\l_stex_current_module_str _prop}
1242       { ns } \l_stex_module_ns_str
1243     \str_set:Nx \l_stex_module_name_str {
1244       \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
1245         { name } / \l_stex_module_name_str
1246     }
1247   }{
```

126

```
1248    % not nested:
1249    \str_if_empty:NT \l_stex_module_ns_str {
1250      \stex_modules_current_namespace:
1251      \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1252      \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1253        / {\l_stex_module_ns_str}
1254      \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1255      \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1256        \str_set:Nx \l_stex_module_ns_str {
1257          \stex_path_to_string:N \l_tmpa_seq
1258        }
1259      }
1260    }
1261  }
```

Next, we determine the language of the module:

```
1262  \str_if_empty:NT \l_stex_module_lang_str {
1263    \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1264    \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1265    \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1266    \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1267    \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1268      \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1269        inferred~from~file~name}
1270      \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1271    }
1272  }
1273
1274  \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
1275    \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1276      \l_tmpa_str {
1277        \ltx@ifpackageloaded{babel}{
1278          \exp_args:Nx \selectlanguage { \l_tmpa_str }
1279        }{}
1280      } {
1281        \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
1282      }
1283  }}
```

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

```
1284  \str_if_empty:NTF \l_stex_module_sig_str {
1285    \exp_args:Nnx \prop_gset_from_keyval:cn {
1286      c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1287    } {
1288      name      = \l_stex_module_name_str ,
1289      ns        = \l_stex_module_ns_str ,
1290      file      = \exp_not:o { \g_stex_currentfile_seq } ,
1291      lang      = \l_stex_module_lang_str ,
1292      sig       = \l_stex_module_sig_str ,
1293      deprecate = \l_stex_module_deprecate_str ,
1294      meta      = \l_stex_module_meta_str
1295    }
1296    \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
```

```
1297    \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1298    \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1299    \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
```

We load the metatheory:

```
1300      \str_if_empty:NT \l_stex_module_meta_str {
1301        \str_set:Nx \l_stex_module_meta_str {
1302          \c_stex_metatheory_ns_str ? Metatheory
1303        }
1304      }
1305      \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1306        \bool_set_true:N \l_stex_in_meta_bool
1307        \exp_args:Nx \stex_add_to_current_module:n {
1308          \bool_set_true:N \l_stex_in_meta_bool
1309          \stex_activate_module:n {\l_stex_module_meta_str}
1310          \bool_set_false:N \l_stex_in_meta_bool
1311        }
1312        \stex_activate_module:n {\l_stex_module_meta_str}
1313        \bool_set_false:N \l_stex_in_meta_bool
1314      }
1315    }{
1316      \str_if_empty:NT \l_stex_module_lang_str {
1317        \msg_error:nnxx{stex}{error/siglanguage}{
1318          \l_stex_module_ns_str?\l_stex_module_name_str
1319        }{\l_stex_module_sig_str}
1320      }
1321
1322      \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1323      \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1324      \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1325      \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1326      \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1327      \str_set:Nx \l_tmpa_str {
1328        \stex_path_to_string:N \l_tmpa_seq /
1329        \l_tmpa_str . \l_stex_module_sig_str .tex
1330      }
1331      \IfFileExists \l_tmpa_str {
1332        \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1333          \str_clear:N \l_stex_current_module_str
1334          \seq_clear:N \l_stex_all_modules_seq
1335          \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1336        }
1337      }{
1338        \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1339      }
1340      \stex_if_smsmode:F {
1341        \stex_activate_module:n {
1342          \l_stex_module_ns_str ? \l_stex_module_name_str
1343        }
1344      }
1345      \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1346    }
1347    \str_if_empty:NF \l_stex_module_deprecate_str {
1348      \msg_warning:nnxx{stex}{warning/deprecated}{
```

```
1349        Module~\l_stex_current_module_str
1350      }{
1351        \l_stex_module_deprecate_str
1352      }
1353    }
1354    \seq_put_right:Nx \l_stex_all_modules_seq {
1355      \l_stex_module_ns_str ? \l_stex_module_name_str
1356    }
1357  }
```

(*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}`

```
1358  \cs_new_protected:Nn \__stex_modules_begin_module: {
1359    \stex_reactivate_macro:N \STEXexport
1360    \stex_reactivate_macro:N \importmodule
1361    \stex_reactivate_macro:N \symdecl
1362    \stex_reactivate_macro:N \notation
1363    \stex_reactivate_macro:N \symdef
1364
1365    \stex_debug:nn{modules}{
1366      New~module:\\
1367      Namespace:~\l_stex_module_ns_str\\
1368      Name:~\l_stex_module_name_str\\
1369      Language:~\l_stex_module_lang_str\\
1370      Signature:~\l_stex_module_sig_str\\
1371      Metatheory:~\l_stex_module_meta_str\\
1372      File:~\stex_path_to_string:N \g_stex_currentfile_seq
1373    }
1374
1375    \stex_if_smsmode:F{
1376      \begin{stex_annotate_env} {theory} {
1377        \l_stex_module_ns_str ? \l_stex_module_name_str
1378      }
1379
1380      \stex_annotate_invisible:nnn{header}{} {
1381        \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1382        \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1383        \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1384          \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1385        }
1386        \str_if_empty:NF \smoduletype {
1387          \stex_annotate:nnn{type}{\smoduletype}{}
1388        }
1389      }
1390    }
1391    % TODO: Inherit metatheory for nested modules?
1392  }
1393  \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}`

```
1394 \cs_new_protected:Nn \__stex_modules_end_module: {
1395   \stex_debug:nn{modules}{Closing~module~\prop_item:cn {c_stex_module_\l_stex_current_module
1396 }
```

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

The core environment

```
1397 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1398 \NewDocumentEnvironment { smodule } { O{} m } {
1399   \stex_module_setup:nn{#1}{#2}
1400   \par
1401   \stex_if_smsmode:F{
1402     \tl_clear:N \l_tmpa_tl
1403     \clist_map_inline:Nn \smoduletype {
1404       \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1405         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1406       }
1407     }
1408     \tl_if_empty:NTF \l_tmpa_tl {
1409       \__stex_modules_smodule_start:
1410     }{
1411       \l_tmpa_tl
1412     }
1413   }
1414   \__stex_modules_begin_module:
1415   \str_if_empty:NF \smoduleid {
1416     \stex_ref_new_doc_target:n \smoduleid
1417   }
1418   \stex_smsmode_do:
1419 } {
1420   \__stex_modules_end_module:
1421   \stex_if_smsmode:F {
1422     \end{stex_annotate_env}
1423     \clist_set:No \l_tmpa_clist \smoduletype
1424     \tl_clear:N \l_tmpa_tl
1425     \clist_map_inline:Nn \l_tmpa_clist {
1426       \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1427         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1428       }
1429     }
1430     \tl_if_empty:NTF \l_tmpa_tl {
1431       \__stex_modules_smodule_end:
1432     }{
1433       \l_tmpa_tl
1434     }
1435   }
1436 }
```

`\stexpatchmodule`

```
1437 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1438 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1439
1440 \newcommand\stexpatchmodule[3][] {
```

130

```
1441      \str_set:Nx \l_tmpa_str{ #1 }
1442      \str_if_empty:NTF \l_tmpa_str {
1443         \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1444         \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1445      }{
1446         \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1447         \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1448      }
1449 }
```

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

27.2   Invoking modules

```
1450 \NewDocumentCommand \STEXModule { m } {
1451   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1452   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1453   \tl_set:Nn \l_tmpa_tl {
1454      \msg_error:nnx{stex}{error/unknownmodule}{#1}
1455   }
1456   \seq_map_inline:Nn \l_stex_all_modules_seq {
1457      \str_set:Nn \l_tmpb_str { ##1 }
1458      \str_if_eq:eeT { \l_tmpa_str } {
1459         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1460      } {
1461         \seq_map_break:n {
1462            \tl_set:Nn \l_tmpa_tl {
1463               \stex_invoke_module:n { ##1 }
1464            }
1465         }
1466      }
1467   }
1468   \l_tmpa_tl
1469 }
1470
1471 \cs_new_protected:Nn \stex_invoke_module:n {
1472   \stex_debug:nn{modules}{Invoking~module~#1}
1473   \peek_charcode_remove:NTF ! {
1474      \__stex_modules_invoke_uri:nN { #1 }
1475   } {
1476      \peek_charcode_remove:NTF ? {
1477         \__stex_modules_invoke_symbol:nn { #1 }
1478      } {
1479         \msg_error:nnx{stex}{error/syntax}{
1480            ?~or~!~expected~after~
1481            \c_backslash_str STEXModule{#1}
1482         }
1483      }
1484   }
1485 }
1486
1487 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
```

```
1488     \str_set:Nn #2 { #1 }
1489 }
1490
1491 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1492     \stex_invoke_symbol:n{#1?#2}
1493 }
```

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

`\stex_activate_module:n`

```
1494 \bool_new:N \l_stex_in_meta_bool
1495 \bool_set_false:N \l_stex_in_meta_bool
1496 \cs_new_protected:Nn \stex_activate_module:n {
1497     \stex_debug:nn{modules}{Activating~module~#1}
1498     \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
1499         \msg_error:nnn{stex}{error/conflictingmodules}{ #1 }
1500     }
1501     \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1502         \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1503         \use:c{ c_stex_module_#1_code }
1504     }
1505 }
```

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

```
1506 ⟨/package⟩
```

Chapter 28

sTeX-Module Inheritance Implementation

```
1507 ⟨∗package⟩
1508
1509 %%%%%%%%%%%%   inheritance.dtx   %%%%%%%%%%%%
1510
```

28.1 SMS Mode

```
1511 ⟨@@=stex_smsmode⟩
```

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

```
1512 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1513 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1514 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1515
1516 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1517   \makeatletter
1518   \makeatother
1519   \ExplSyntaxOn
1520   \ExplSyntaxOff
1521   \rustexBREAK
1522 }
1523
1524 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1525   \symdef
1526   \importmodule
1527   \notation
1528   \symdecl
1529   \STEXexport
1530   \inlineass
1531   \inlinedef
1532   \inlineex
1533   \endinput
1534   \setnotation
```

133

```
1535        \copynotation
1536    }
1537
1538    \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1539        \tl_to_str:n {
1540            smodule,
1541            copymodule,
1542            interpretmodule,
1543            sdefinition,
1544            sexample,
1545            sassertion,
1546            sparagraph
1547        }
1548    }
```

(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`

```
1549    \bool_new:N \g__stex_smsmode_bool
1550    \bool_set_false:N \g__stex_smsmode_bool
1551    \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1552        \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1553    }
```

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

`\__stex_smsmode_in_smsmode:nn`

```
1554    \cs_new_protected:Nn \__stex_smsmode_in_smsmode:nn {
1555        \vbox_set:Nn \l_tmpa_box {
1556            \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1557            \bool_gset_true:N \g__stex_smsmode_bool
1558            #2
1559            \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1560        }
1561        \box_clear:N \l_tmpa_box
1562    }
```

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

`\stex_file_in_smsmode:nn`

```
1563    \quark_new:N \q__stex_smsmode_break
1564
1565    \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1566        \stex_filestack_push:n{#1}
1567        \__stex_smsmode_in_smsmode:nn{#1} {
1568            #2
1569            \everyeof{\q__stex_smsmode_break\noexpand}
1570            \expandafter\expandafter\expandafter
1571            \stex_smsmode_do:
1572            \csname @ @ input\endcsname "#1"\relax
1573        }
1574        \stex_filestack_pop:
1575    }
```

(*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:

```
1576 \cs_new_protected:Npn \stex_smsmode_do: {
1577   \stex_if_smsmode:T {
1578     \__stex_smsmode_do:w
1579   }
1580 }
1581 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1582   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 }}{
1583     \expandafter\if\expandafter\relax\noexpand#1
1584       \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1585     \else\expandafter\__stex_smsmode_do:w\fi
1586   }{
1587     \__stex_smsmode_do:w %#1
1588   }
1589 }
1590 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1591   \cs_if_eq:NNF #1 \q__stex_smsmode_break {
1592     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1593       #1\__stex_smsmode_do:w
1594     }{
1595       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1596         #1
1597       }{
1598         \cs_if_eq:NNTF \begin #1 {
1599           \__stex_smsmode_check_begin:n
1600         }{
1601           \cs_if_eq:NNTF \end #1 {
1602             \__stex_smsmode_check_end:n
1603           }{
1604             \__stex_smsmode_do:w
1605           }
1606         }
1607       }
1608     }
1609   }
1610 }
1611
1612 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1613   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1614     \begin{#1}
1615   }{
1616     \__stex_smsmode_do:w
1617   }
1618 }
1619 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
1620   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1621     \end{#1}\__stex_smsmode_do:w
1622   }{
1623     \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1624   }
1625 }
```

135

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

28.2 Inheritance

```
1626 ⟨@@=stex_importmodule⟩
```

`\stex_import_module_uri:nn`

```
1627 \cs_new_protected:Nn \stex_import_module_uri:nn {
1628   \str_set:Nx \l_stex_import_archive_str { #1 }
1629   \str_set:Nn \l_stex_import_path_str { #2 }
1630
1631   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1632   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1633   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1634
1635   \stex_modules_current_namespace:
1636   \bool_lazy_all:nTF {
1637     {\str_if_empty_p:N \l_stex_import_archive_str}
1638     {\str_if_empty_p:N \l_stex_import_path_str}
1639     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1640   }{
1641     \str_set_eq:NN \l_stex_import_path_str \l_stex_modules_subpath_str
1642     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1643   }{
1644     \str_if_empty:NT \l_stex_import_archive_str {
1645       \prop_if_exist:NT \l_stex_current_repository_prop {
1646         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1647       }
1648     }
1649     \str_if_empty:NTF \l_stex_import_archive_str {
1650       \str_if_empty:NF \l_stex_import_path_str {
1651         \str_set:Nx \l_stex_import_ns_str {
1652           \l_stex_module_ns_str / \l_stex_import_path_str
1653         }
1654       }
1655     }{
1656       \stex_require_repository:n \l_stex_import_archive_str
1657       \prop_get:cnN { c_stex_mathhub_\l_stex_import_archive_str _manifest_prop } { ns }
1658         \l_stex_import_ns_str
1659       \str_if_empty:NF \l_stex_import_path_str {
1660         \str_set:Nx \l_stex_import_ns_str {
1661           \l_stex_import_ns_str / \l_stex_import_path_str
1662         }
1663       }
1664     }
1665   }
1666 }
```

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

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

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

```
1667 \str_new:N \l_stex_import_name_str
1668 \str_new:N \l_stex_import_archive_str
1669 \str_new:N \l_stex_import_path_str
```

136

```
1670 \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*⟩}

```
1671 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1672   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1673
1674     % archive
1675     \str_set:Nx \l_tmpa_str { #2 }
1676     \str_if_empty:NTF \l_tmpa_str {
1677       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1678     } {
1679       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1680       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1681       \seq_put_right:Nn \l_tmpa_seq { source }
1682     }
1683
1684     % path
1685     \str_set:Nx \l_tmpb_str { #3 }
1686     \str_if_empty:NTF \l_tmpb_str {
1687       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1688
1689       \ltx@ifpackageloaded{babel} {
1690         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1691             { \languagename } \l_tmpb_str {
1692               \msg_error:nnx{stex}{error/unknownlanguage}{\languagename}
1693             }
1694       } {
1695         \str_clear:N \l_tmpb_str
1696       }
1697
1698       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1699       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1700         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1701       }{
1702         \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1703         \IfFileExists{ \l_tmpa_str.tex }{
1704           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1705         }{
1706           % try english as default
1707           \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1708           \IfFileExists{ \l_tmpa_str.en.tex }{
1709             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1710           }{
1711             \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1712           }
1713         }
1714       }
1715
1716     } {
1717       \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1718       \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1719
```

```
1720        \ltx@ifpackageloaded{babel} {
1721          \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1722              { \languagename } \l_tmpb_str {
1723                  \msg_error:nnx{stex}{error/unknownlanguage}{\languagename}
1724              }
1725        } {
1726          \str_clear:N \l_tmpb_str
1727        }
1728
1729        \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1730
1731        \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1732        \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1733          \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1734        }{
1735          \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1736          \IfFileExists{ \l_tmpa_str/#4.tex }{
1737            \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1738          }{
1739            % try english as default
1740            \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1741            \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1742              \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1743            }{
1744              \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1745              \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1746                \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1747              }{
1748                \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1749                \IfFileExists{ \l_tmpa_str.tex }{
1750                  \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1751                }{
1752                  % try english as default
1753                  \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1754                  \IfFileExists{ \l_tmpa_str.en.tex }{
1755                    \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1756                  }{
1757                    \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1758                  }
1759                }
1760              }
1761            }
1762          }
1763        }
1764      }
1765
1766      \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
1767        \seq_clear:N \l_stex_all_modules_seq
1768        \str_clear:N \l_stex_current_module_str
1769        \str_set:Nx \l_tmpb_str { #2 }
1770        \str_if_empty:NF \l_tmpb_str {
1771          \stex_set_current_repository:n { #2 }
1772        }
1773        \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
```

```
1774        }
1775
1776        \stex_if_module_exists:nF { #1 ? #4 } {
1777          \msg_error:nnx{stex}{error/unknownmodule}{
1778            #1?#4~(in~file~\g__stex_importmodule_file_str)
1779          }
1780        }
1781      }
1782      \stex_activate_module:n { #1 ? #4 }
1783 }
```

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

\importmodule
```
1784 \NewDocumentCommand \importmodule { O{} m } {
1785      \stex_import_module_uri:nn { #1 } { #2 }
1786      \stex_debug:nn{modules}{Importing~module:~
1787        \l_stex_import_ns_str ? \l_stex_import_name_str
1788      }
1789      \stex_if_smsmode:F {
1790        \stex_import_require_module:nnnn
1791        { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1792        { \l_stex_import_path_str } { \l_stex_import_name_str }
1793        \stex_annotate_invisible:nnn
1794          {import} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1795      }
1796      \exp_args:Nx \stex_add_to_current_module:n {
1797        \stex_import_require_module:nnnn
1798        { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1799        { \l_stex_import_path_str } { \l_stex_import_name_str }
1800      }
1801      \exp_args:Nx \stex_add_import_to_current_module:n {
1802        \l_stex_import_ns_str ? \l_stex_import_name_str
1803      }
1804      \stex_smsmode_do:
1805      \ignorespacesandpars
1806 }
1807 \stex_deactivate_macro:Nn \importmodule {module~environments}
```

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

\usemodule
```
1808 \NewDocumentCommand \usemodule { O{} m } {
1809      \stex_if_smsmode:F {
1810        \stex_import_module_uri:nn { #1 } { #2 }
1811        \stex_import_require_module:nnnn
1812        { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1813        { \l_stex_import_path_str } { \l_stex_import_name_str }
1814        \stex_annotate_invisible:nnn
1815          {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1816      }
1817      \stex_smsmode_do:
1818      \ignorespacesandpars
1819 }
```

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

1820 ⟨/package⟩

Chapter 29

SтеX
-Symbols Implementation

```
1821 ⟨∗package⟩
1822
1823 %%%%%%%%%%%%%   symbols.dtx   %%%%%%%%%%%%%
1824
```

Warnings and error messages
```
1825 \msg_new:nnn{stex}{error/wrongargs}{
1826   args~value~in~symbol~declaration~for~#1~
1827   needs~to~be~i,~a,~b~or~B,~but~#2~given
1828 }
1829 \msg_new:nnn{stex}{error/unknownsymbol}{
1830   No~symbol~#1~found!
1831 }
1832 \msg_new:nnn{stex}{error/seqlength}{
1833   Expected~#1~arguments;~got~#2!
1834 }
```

29.1   Symbol Declarations

```
1835 ⟨@@=stex_symdecl⟩
```

\stex_all_symbols:n   Map over all available symbols
```
1836 \cs_new_protected:Nn \stex_all_symbols:n {
1837   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
1838   \seq_map_inline:Nn \l_stex_all_modules_seq {
1839     \seq_map_inline:cn{c_stex_module_##1_constants}{
1840       \__stex_symdecl_all_symbols_cs{##1?####1}
1841     }
1842   }
1843 }
```

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

\STEXsymbol

```
1844 \NewDocumentCommand \STEXsymbol { m } {
1845   \stex_get_symbol:n { #1 }
```

```
1846    \exp_args:No
1847    \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1848  }
```

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

symdecl arguments:

```
1849  \keys_define:nn { stex / symdecl } {
1850    name        .str_set_x:N  = \l_stex_symdecl_name_str ,
1851    local       .bool_set:N   = \l_stex_symdecl_local_bool ,
1852    args        .str_set_x:N  = \l_stex_symdecl_args_str ,
1853    type        .tl_set:N     = \l_stex_symdecl_type_tl ,
1854    deprecate   .str_set_x:N  = \l_stex_symdecl_deprecate_str ,
1855    align       .str_set:N    = \l_stex_symdecl_align_str , % TODO(?)
1856    gfc         .str_set:N    = \l_stex_symdecl_gfc_str , % TODO(?)
1857    specializes .str_set:N    = \l_stex_symdecl_specializes_str , % TODO(?)
1858    def         .tl_set:N     = \l_stex_symdecl_definiens_tl ,
1859    assoc       .choices:nn   =
1860        {bin,binl,binr,pre,conj,pwconj}
1861        {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}
1862  }
1863
1864  \bool_new:N \l_stex_symdecl_make_macro_bool
1865
1866  \cs_new_protected:Nn \__stex_symdecl_args:n {
1867    \str_clear:N \l_stex_symdecl_name_str
1868    \str_clear:N \l_stex_symdecl_args_str
1869    \str_clear:N \l_stex_symdecl_deprecate_str
1870    \str_clear:N \l_stex_symdecl_assoctype_str
1871    \bool_set_false:N \l_stex_symdecl_local_bool
1872    \tl_clear:N \l_stex_symdecl_type_tl
1873    \tl_clear:N \l_stex_symdecl_definiens_tl
1874
1875    \keys_set:nn { stex / symdecl } { #1 }
1876  }
```

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

```
1877
1878  \NewDocumentCommand \symdecl { s m O{}} {
1879    \__stex_symdecl_args:n { #3 }
1880    \IfBooleanTF #1 {
1881      \bool_set_false:N \l_stex_symdecl_make_macro_bool
1882    } {
1883      \bool_set_true:N \l_stex_symdecl_make_macro_bool
1884    }
1885    \stex_symdecl_do:n { #2 }
1886    \stex_smsmode_do:
1887  }
1888
1889  \cs_new_protected:Nn \stex_symdecl_do:nn {
1890    \__stex_symdecl_args:n{#1}
1891    \bool_set_false:N \l_stex_symdecl_make_macro_bool
1892    \stex_symdecl_do:n{#2}
1893  }
```

```
1895 \stex_deactivate_macro:Nn \symdecl {module~environments}
```

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

\stex_symdecl_do:n

```
1896 \cs_new_protected:Nn \stex_symdecl_do:n {
1897   \stex_if_in_module:F {
1898     % TODO throw error? some default namespace?
1899   }
1900
1901   \str_if_empty:NT \l_stex_symdecl_name_str {
1902     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1903   }
1904
1905   \prop_if_exist:cT { l_stex_symdecl_
1906       \l_stex_current_module_str ?
1907       \l_stex_symdecl_name_str
1908     _prop
1909   }{
1910     % TODO throw error (beware of circular dependencies)
1911   }
1912
1913   \prop_clear:N \l_tmpa_prop
1914   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
1915   \seq_clear:N \l_tmpa_seq
1916   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1917   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1918
1919   \str_if_empty:NT \l_stex_symdecl_deprecate_str {
1920     \str_if_empty:NF \l_stex_module_deprecate_str {
1921       \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
1922     }
1923   }
1924   \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
1925
1926   \exp_args:No \stex_add_constant_to_current_module:n {
1927     \l_stex_symdecl_name_str
1928   }
1929
1930   % arity/args
1931   \int_zero:N \l_tmpb_int
1932
1933   \bool_set_true:N \l_tmpa_bool
1934   \str_map_inline:Nn \l_stex_symdecl_args_str {
1935     \token_case_meaning:NnF ##1 {
1936       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1937       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1938       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1939       {\tl_to_str:n a} {
1940         \bool_set_false:N \l_tmpa_bool
1941         \int_incr:N \l_tmpb_int
1942       }
1943       {\tl_to_str:n B} {
```

```
1944        \bool_set_false:N \l_tmpa_bool
1945        \int_incr:N \l_tmpb_int
1946      }
1947    }{
1948      \msg_error:nnxx{stex}{error/wrongargs}{
1949        \l_stex_current_module_str ?
1950        \l_stex_symdecl_name_str
1951      }{##1}
1952    }
1953  }
1954  \bool_if:NTF \l_tmpa_bool {
1955    % possibly numeric
1956    \str_if_empty:NTF \l_stex_symdecl_args_str {
1957      \prop_put:Nnn \l_tmpa_prop { args } {}
1958      \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1959    }{
1960      \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1961      \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1962      \str_clear:N \l_tmpa_str
1963      \int_step_inline:nn \l_tmpa_int {
1964        \str_put_right:Nn \l_tmpa_str i
1965      }
1966      \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1967    }
1968  } {
1969    \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1970    \prop_put:Nnx \l_tmpa_prop { arity }
1971      { \str_count:N \l_stex_symdecl_args_str }
1972  }
1973  \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
1974
1975  \tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
1976    \prop_put:Nnx \l_tmpa_prop { defined }{ false }
1977  }{
1978    \prop_put:Nnx \l_tmpa_prop { defined }{ true }
1979  }
1980
1981  % semantic macro
1982
1983  \bool_if:NT \l_stex_symdecl_make_macro_bool {
1984    \exp_args:Nx \stex_do_up_to_module:n {
1985      \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1986        \l_stex_current_module_str ? \l_stex_symdecl_name_str
1987      }}
1988    }
1989
1990    \bool_if:NF \l_stex_symdecl_local_bool {
1991      \exp_args:Nx \stex_add_to_current_module:n {
1992        \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1993          \l_stex_current_module_str ? \l_stex_symdecl_name_str
1994        } }
1995      }
1996    }
1997  }
```

144

```
1998
1999    \stex_debug:nn{symbols}{New~symbol:~
2000      \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2001      Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2002      Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
2003      Definiens:~\exp_not:o {\l_stex_symdecl_definiens_tl}
2004    }
2005
2006    % circular dependencies require this:
2007
2008    \prop_if_exist:cF {
2009      l_stex_symdecl_
2010      \l_stex_current_module_str ? \l_stex_symdecl_name_str
2011      _prop
2012    } {
2013      \exp_args:Nx \stex_do_up_to_module:n {
2014        \prop_set_from_keyval:cn {
2015          l_stex_symdecl_
2016          \l_stex_current_module_str ? \l_stex_symdecl_name_str
2017          _prop
2018        } {\prop_to_keyval:N \l_tmpa_prop}
2019        \seq_clear:c {
2020          l_stex_symdecl_
2021          \l_stex_current_module_str ? \l_stex_symdecl_name_str
2022          _notations
2023        }
2024      }
2025    }
2026
2027
2028
2029    \bool_if:NF \l_stex_symdecl_local_bool {
2030      \exp_args:Nx
2031      \stex_add_to_current_module:n {
2032        \seq_clear:c {
2033          l_stex_symdecl_
2034          \l_stex_current_module_str ? \l_stex_symdecl_name_str
2035          _notations
2036        }
2037        \prop_set_from_keyval:cn {
2038          l_stex_symdecl_
2039          \l_stex_current_module_str ? \l_stex_symdecl_name_str
2040          _prop
2041        } {
2042          name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2043          module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2044          type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2045          args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2046          arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2047          assocs    = \prop_item:Nn \l_tmpa_prop { assocs }
2048        }
2049      }
2050    }
2051
```

145

```
2052    \stex_if_smsmode:F {
2053 %      \exp_args:Nx \stex_do_up_to_module:n {
2054 %        \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2055 %        \l_stex_current_module_str ? \l_stex_symdecl_name_str
2056 %      }
2057 %    }
2058    \stex_if_do_html:T {
2059      \stex_annotate_invisible:nnn {symdecl} {
2060        \l_stex_current_module_str ? \l_stex_symdecl_name_str
2061      } {
2062        \tl_if_empty:NF \l_stex_symdecl_type_tl {\stex_annotate_invisible:nnn{type}{}{$\l_st
2063        \stex_annotate_invisible:nnn{args}{}{
2064          \prop_item:Nn \l_tmpa_prop { args }
2065        }
2066        \stex_annotate_invisible:nnn{macroname}{#1}{}
2067        \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2068          \stex_annotate_invisible:nnn{definiens}{}
2069            {$\l_stex_symdecl_definiens_tl$}
2070        }
2071        \str_if_empty:NF \l_stex_symdecl_assoctype_str {
2072          \stex_annotate_invisible:nnn{assoctype}{\l_stex_symdecl_assoctype_str}{}
2073        }
2074      }
2075    }
2076  }
2077 }
```

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

`\stex_get_symbol:n`

```
2078 \str_new:N \l_stex_get_symbol_uri_str
2079
2080 \cs_new_protected:Nn \stex_get_symbol:n {
2081   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2082     \tl_set:Nn \l_tmpa_tl { #1 }
2083     \__stex_symdecl_get_symbol_from_cs:
2084   }{
2085     % argument is a string
2086     % is it a command name?
2087     \cs_if_exist:cTF { #1 }{
2088       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2089       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2090       \str_if_empty:NTF \l_tmpa_str {
2091         \exp_args:Nx \cs_if_eq:NNTF {
2092           \tl_head:N \l_tmpa_tl
2093         } \stex_invoke_symbol:n {
2094           \__stex_symdecl_get_symbol_from_cs:
2095         }{
2096           \__stex_symdecl_get_symbol_from_string:n { #1 }
2097         }
2098       } {
2099         \__stex_symdecl_get_symbol_from_string:n { #1 }
2100       }
2101     }{
```

146

```
2102        % argument is not a command name
2103        \__stex_symdecl_get_symbol_from_string:n { #1 }
2104        % \l_stex_all_symbols_seq
2105      }
2106    }
2107    \str_if_eq:eeF {
2108      \prop_item:cn {
2109        l_stex_symdecl_\l_stex_get_symbol_uri_str _prop
2110      }{ deprecate }
2111    }{}{
2112      \msg_warning:nnxx{stex}{warning/deprecated}{
2113        Symbol~\l_stex_get_symbol_uri_str
2114      }{
2115        \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{ deprecate }
2116      }
2117    }
2118  }
2119
2120  \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2121    \tl_set:Nn \l_tmpa_tl {
2122      \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2123    }
2124    \str_set:Nn \l_tmpa_str { #1 }
2125    \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2126
2127    \stex_all_symbols:n {
2128      \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2129        \seq_map_break:n{\seq_map_break:n{
2130          \tl_set:Nn \l_tmpa_tl {
2131            \str_set:Nn \l_stex_get_symbol_uri_str { ##1 }
2132          }
2133        }}
2134      }
2135    }
2136
2137    \l_tmpa_tl
2138  }
2139
2140  \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
2141    \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2142      { \tl_tail:N \l_tmpa_tl }
2143    \tl_if_single:NTF \l_tmpa_tl {
2144      \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2145        \exp_after:wN \str_set:Nn \exp_after:wN
2146          \l_stex_get_symbol_uri_str \l_tmpa_tl
2147      }{
2148        % TODO
2149        % tail is not a single group
2150      }
2151    }{
2152      % TODO
2153      % tail is not a single group
2154    }
2155  }
```

147

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

29.2   Notations

```
2156 ⟨@@=stex_notation⟩
```

notation arguments:
```
2157 \keys_define:nn { stex / notation } {
2158   lang    .tl_set_x:N  = \l__stex_notation_lang_str ,
2159   variant .tl_set_x:N  = \l__stex_notation_variant_str ,
2160   prec    .str_set_x:N = \l__stex_notation_prec_str ,
2161   op      .tl_set:N    = \l__stex_notation_op_tl ,
2162   primary .bool_set:N  = \l__stex_notation_primary_bool ,
2163   primary .default:n   = {true} ,
2164   unknown .code:n      = \str_set:Nx
2165       \l__stex_notation_variant_str \l_keys_key_str
2166 }
2167
2168 \cs_new_protected:Nn \_stex_notation_args:n {
2169   \str_clear:N \l__stex_notation_lang_str
2170   \str_clear:N \l__stex_notation_variant_str
2171   \str_clear:N \l__stex_notation_prec_str
2172   \tl_clear:N \l__stex_notation_op_tl
2173   \bool_set_false:N \l__stex_notation_primary_bool
2174
2175   \keys_set:nn { stex / notation } { #1 }
2176 }
```

\notation

```
2177 \NewDocumentCommand \notation { s m O{}} {
2178   \_stex_notation_args:n { #3 }
2179   \tl_clear:N \l_stex_symdecl_definiens_tl
2180   \stex_get_symbol:n { #2 }
2181   \tl_set:Nn \l_stex_notation_after_do_tl {
2182     \__stex_notation_final:
2183     \IfBooleanTF#1{
2184       \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2185     }{}
2186     \stex_smsmode_do:\ignorespacesandpars
2187   }
2188   \stex_notation_do:nnnnn
2189     { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } }
2190     { \prop_item:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } }
2191     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2192     { \l__stex_notation_prec_str}
2193 }
2194 \stex_deactivate_macro:Nn \notation {module~environments}
```

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

\stex_notation_do:nnnnn

```
2195 \seq_new:N \l__stex_notation_precedences_seq
2196 \tl_new:N \l__stex_notation_opprec_tl
2197 \int_new:N \l__stex_notation_currarg_int
```

148

```
2198  \tl_new:N \stex_symbol_after_invokation_tl
2199
2200  \cs_new_protected:Nn \stex_notation_do:nnnnn {
2201    \let\l_stex_current_symbol_str\relax
2202    \seq_clear:N \l__stex_notation_precedences_seq
2203    \tl_clear:N \l__stex_notation_opprec_tl
2204    \str_set:Nx \l__stex_notation_args_str { #1 }
2205    \str_set:Nx \l__stex_notation_arity_str { #2 }
2206    \str_set:Nx \l__stex_notation_suffix_str { #3 }
2207    \str_set:Nx \l__stex_notation_prec_str { #4 }
2208
2209    % precedences
2210    \str_if_empty:NTF \l__stex_notation_prec_str {
2211      \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2212        \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2213      }{
2214        \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2215      }
2216    } {
2217      \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2218        \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2219        \int_step_inline:nn { \l__stex_notation_arity_str } {
2220          \exp_args:NNo
2221          \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2222        }
2223      }{
2224        \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2225        \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2226          \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2227          \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2228            \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2229              \l_tmpa_seq {\tl_to_str:n{x} } { \l_tmpa_str }
2230            \seq_map_inline:Nn \l_tmpa_seq {
2231              \seq_put_right:Nn \l_tmpb_seq { ##1 }
2232            }
2233          }
2234        }{
2235          \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2236            \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2237          }{
2238            \tl_set:No \l__stex_notation_opprec_tl { 0 }
2239          }
2240        }
2241      }
2242    }
2243
2244    \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2245    \int_step_inline:nn { \l__stex_notation_arity_str } {
2246      \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
2247        \exp_args:NNo
2248        \seq_put_right:No \l__stex_notation_precedences_seq {
2249          \l__stex_notation_opprec_tl
2250        }
2251      }
```

149

```
2252      }
2253      \tl_clear:N \l_stex_notation_dummyargs_tl
2254
2255      \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2256        \exp_args:NNe
2257        \cs_set:Npn \l_stex_notation_macrocode_cs {
2258          \_stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2259            { \l__stex_notation_suffix_str }
2260            { \l__stex_notation_opprec_tl }
2261            { \exp_not:n { #5 } }
2262        }
2263        \l_stex_notation_after_do_tl
2264      }{
2265        \str_if_in:NnTF \l__stex_notation_args_str b {
2266          \exp_args:Nne \use:nn
2267          {
2268          \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2269          \cs_set:Npn \l__stex_notation_arity_str } { {
2270            \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2271              { \l__stex_notation_suffix_str }
2272              { \l__stex_notation_opprec_tl }
2273              { \exp_not:n { #5 } }
2274        }}
2275      }{
2276        \str_if_in:NnTF \l__stex_notation_args_str B {
2277          \exp_args:Nne \use:nn
2278          {
2279          \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2280          \cs_set:Npn \l__stex_notation_arity_str } { {
2281            \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2282              { \l__stex_notation_suffix_str }
2283              { \l__stex_notation_opprec_tl }
2284              { \exp_not:n { #5 } }
2285          } }
2286        }{
2287          \exp_args:Nne \use:nn
2288          {
2289          \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2290          \cs_set:Npn \l__stex_notation_arity_str } { {
2291            \_stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2292              { \l__stex_notation_suffix_str }
2293              { \l__stex_notation_opprec_tl }
2294              { \exp_not:n { #5 } }
2295          } }
2296        }
2297      }
2298
2299      \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2300      \int_zero:N \l__stex_notation_currarg_int
2301      \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2302      \__stex_notation_arguments:
2303    }
2304  }
```

(*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

```
2305 \cs_new_protected:Nn \__stex_notation_arguments: {
2306   \int_incr:N \l__stex_notation_currarg_int
2307   \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2308     \l_stex_notation_after_do_tl
2309   }{
2310     \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2311     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2312     \str_if_eq:VnTF \l_tmpa_str a {
2313       \__stex_notation_argument_assoc:n
2314     }{
2315       \str_if_eq:VnTF \l_tmpa_str B {
2316         \__stex_notation_argument_assoc:n
2317       }{
2318         \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2319         \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2320           { \_stex_term_math_arg:nnn
2321             { \int_use:N \l__stex_notation_currarg_int }
2322             { \l_tmpa_str }
2323             { ####\int_use:N \l__stex_notation_currarg_int }
2324           }
2325         }
2326         \__stex_notation_arguments:
2327       }
2328     }
2329   }
2330 }
```

(*End definition for* __stex_notation_arguments:.)

__stex_notation_argument_assoc:n

```
2331 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
2332
2333   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2334     {\l__stex_notation_arity_str}{
2335     #1
2336   }
2337   \int_zero:N \l_tmpa_int
2338   \tl_clear:N \l_tmpa_tl
2339   \str_map_inline:Nn \l__stex_notation_args_str {
2340     \int_incr:N \l_tmpa_int
2341     \tl_put_right:Nx \l_tmpa_tl {
2342       \str_if_eq:nnTF {##1}{a}{ {} }{
2343         \str_if_eq:nnTF {##1}{B}{ {} }{
2344           {\_stex_term_arg:nn{\int_use:N \l_tmpa_int}{############### \int_use:N \l_tmpa_in
2345         }
2346       }
2347     }
2348   }
2349   \exp_after:wN\exp_after:wN\exp_after:wN \def
2350   \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2351   \exp_after:wN\exp_after:wN\exp_after:wN ##
2352   \exp_after:wN\exp_after:wN\exp_after:wN 1
2353   \exp_after:wN\exp_after:wN\exp_after:wN ##
```

151

```
2354    \exp_after:wN\exp_after:wN\exp_after:wN 2
2355    \exp_after:wN\exp_after:wN\exp_after:wN {
2356      \exp_after:wN \exp_after:wN \exp_after:wN
2357      \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2358        \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2359      }
2360    }
2361
2362    \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2363    \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
2364      \_stex_term_math_assoc_arg:nnnn
2365        { \int_use:N \l__stex_notation_currarg_int }
2366        { \l_tmpa_str }
2367        { ####\int_use:N \l__stex_notation_currarg_int }
2368        { \l_tmpa_cs {####1} {####2} }
2369    } }
2370    \__stex_notation_arguments:
2371 }
```

(End definition for __stex_notation_argument_assoc:n.*)*

__stex_notation_final: Called after processing all notation arguments

```
2372 \cs_new_protected:Nn \__stex_notation_final: {
2373 %  \exp_args:Nne \use:nn
2374 %  {
2375 %  \cs_generate_from_arg_count:cNnn {
2376 %      stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2377 %      \l__stex_notation_suffix_str
2378 %      _cs
2379 %    }
2380 %    \cs_set:Npn \l__stex_notation_arity_str } { {
2381 %      \exp_after:wN \exp_after:wN \exp_after:wN
2382 %      \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2383 %      { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sym
2384 %  } } }
2385
2386 %  \tl_if_empty:NF \l__stex_notation_op_tl {
2387 %    \cs_set:cpx {
2388 %      stex_op_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2389 %      \l__stex_notation_suffix_str
2390 %      _cs
2391 %    } { \exp_not:N \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } } }
2392 %  }
2393
2394    \exp_args:Nx \stex_do_up_to_module:n {
2395      \cs_generate_from_arg_count:cNnn {
2396        stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2397        \l__stex_notation_suffix_str
2398        _cs
2399      } \cs_set:Npn {\l__stex_notation_arity_str} {
2400          \exp_after:wN \exp_after:wN \exp_after:wN
2401          \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2402          { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2403      }
```

152

```
\tl_if_empty:NF \l__stex_notation_op_tl {
  \cs_set:cpn {
    stex_op_notation_\l_stex_get_symbol_uri_str \c_hash_str
    \l__stex_notation_suffix_str
    _cs
  } { \exp_not:N \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
}
}

\exp_args:Ne
\stex_add_to_current_module:n {
  \cs_generate_from_arg_count:cNnn {
    stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
    \l__stex_notation_suffix_str
    _cs
  } \cs_set:Npn {\l__stex_notation_arity_str} {
      \exp_after:wN \exp_after:wN \exp_after:wN
      \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
      { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
  }
  \tl_if_empty:NF \l__stex_notation_op_tl {
    \cs_set:cpn {
      stex_op_notation_\l_stex_get_symbol_uri_str \c_hash_str
      \l__stex_notation_suffix_str
      _cs
    } { \exp_not:N \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
  }
}

\stex_debug:nn{symbols}{
  Notation~\l__stex_notation_suffix_str
  ~for~\l_stex_get_symbol_uri_str^^J
  Operator~precedence:~\l__stex_notation_opprec_tl^^J
  Argument~precedences:~
    \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
  Notation: \cs_meaning:c {
    stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
    \l__stex_notation_suffix_str
    _cs
  }
}

\exp_args:Nx
\stex_do_up_to_module:n {
  \seq_put_right:cx {
    l_stex_symdecl_ \l_stex_get_symbol_uri_str
    _notations
  } {
    \l__stex_notation_suffix_str
  }
}
\exp_args:Ne
\stex_add_to_current_module:n {
  \seq_put_right:cn {
```

```
2458        l_stex_symdecl_\l_stex_get_symbol_uri_str
2459          _notations
2460      } { \l__stex_notation_suffix_str }
2461    }
2462
2463    \stex_if_smsmode:F {
2464
2465      % HTML annotations
2466      \stex_if_do_html:T {
2467        \stex_annotate_invisible:nnn { notation }
2468        { \l_stex_get_symbol_uri_str } {
2469          \stex_annotate_invisible:nnn { notationfragment }
2470            { \l__stex_notation_suffix_str }{}
2471          \stex_annotate_invisible:nnn { precedence }
2472            { \l__stex_notation_prec_str }{}
2473
2474          \int_zero:N \l_tmpa_int
2475          \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2476          \tl_clear:N \l_tmpa_tl
2477          \int_step_inline:nn { \l__stex_notation_arity_str }{
2478            \int_incr:N \l_tmpa_int
2479            \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2480            \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_r
2481            \str_if_eq:VnTF \l_tmpb_str a {
2482              \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2483                \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2484                \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2485              } }
2486            }{
2487              \str_if_eq:VnTF \l_tmpb_str B {
2488                \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2489                  \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2490                  \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2491                } }
2492              }{
2493                \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2494                  \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2495                } }
2496              }
2497            }
2498          }
2499          \stex_annotate_invisible:nnn { notationcomp }{}{
2500            \str_set:Nx \l_stex_current_symbol_str {\l_stex_get_symbol_uri_str }
2501            $ \exp_args:Nno \use:nn { \use:c {
2502              stex_notation_ \l_stex_current_symbol_str
2503              \c_hash_str \l__stex_notation_suffix_str _cs
2504            } } { \l_tmpa_tl } $
2505          }
2506        }
2507      }
2508    }
2509 }
```

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

```
2510 \keys_define:nn { stex / setnotation } {
2511   lang    .tl_set_x:N  = \l__stex_notation_lang_str ,
2512   variant .tl_set_x:N  = \l__stex_notation_variant_str ,
2513   unknown .code:n       = \str_set:Nx
2514     \l__stex_notation_variant_str \l_keys_key_str
2515 }
2516
2517 \cs_new_protected:Nn \_stex_setnotation_args:n {
2518   \str_clear:N \l__stex_notation_lang_str
2519   \str_clear:N \l__stex_notation_variant_str
2520   \keys_set:nn { stex / setnotation } { #1 }
2521 }
2522
2523 \cs_new_protected:Nn \stex_setnotation:n {
2524   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1 _notations }
2525     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2526       \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2527         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2528       \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2529         { \c_hash_str }
2530       \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl_#1 _notations }
2531         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2532       \exp_args:Nx \stex_add_to_current_module:n {
2533         \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2534           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2535         \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl_#1 _notations }
2536           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2537         \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2538           { \c_hash_str }
2539       }
2540       \stex_debug:nn {notations}{
2541         Setting~default~notation~
2542         {\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str}~for~
2543         #1 \\
2544         \expandafter\meaning\csname
2545         l_stex_symdecl_#1 _notations\endcsname
2546       }
2547     }{
2548       % todo throw error
2549     }
2550 }
2551
2552 \NewDocumentCommand \setnotation {m m} {
2553   \stex_get_symbol:n { #1 }
2554   \_stex_setnotation_args:n { #2 }
2555   \stex_setnotation:n{\l_stex_get_symbol_uri_str}
2556   \stex_smsmode_do:\ignorespacesandpars
2557 }
2558
2559 \cs_new_protected:Nn \stex_copy_notations:nn {
2560   \stex_debug:nn {notations}{
2561     Copying~notations~from~#2~to~#1\\
2562     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
```

```
2563        }
2564      \tl_clear:N \l_tmpa_tl
2565      \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } } {
2566        \tl_put_right:Nn \l_tmpa_tl { {## ##1} }
2567      }
2568      \seq_map_inline:cn {l_stex_symdecl_#2_notations}{
2569        \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2570        \edef \l_tmpa_tl {
2571          \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2572          \exp_after:wN\exp_after:wN\exp_after:wN {
2573            \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2574          }
2575        }
2576        \exp_args:Nx
2577        \stex_do_up_to_module:n {
2578          \seq_put_right:cn{l_stex_symdecl_#1_notations}{##1}
2579          \cs_generate_from_arg_count:cNnn {
2580            stex_notation_ #1 \c_hash_str ##1 _cs
2581          } \cs_set:Npn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }{
2582            \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl}
2583          }
2584        }
2585      }
2586    }
2587
2588    \NewDocumentCommand \copynotation {m m} {
2589      \stex_get_symbol:n { #1 }
2590      \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2591      \stex_get_symbol:n { #2 }
2592      \exp_args:Noo
2593      \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2594      \exp_args:Nx \stex_add_import_to_current_module:n{
2595        \stex_copy_notations:nn {\l_tmpa_str} {\l_stex_get_symbol_uri_str}
2596      }
2597      \stex_smsmode_do:\ignorespacesandpars
2598    }
2599
```

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

\symdef

```
2600    \keys_define:nn { stex / symdef } {
2601      name     .str_set_x:N = \l_stex_symdecl_name_str ,
2602      local    .bool_set:N  = \l_stex_symdecl_local_bool ,
2603      args     .str_set_x:N = \l_stex_symdecl_args_str ,
2604      type     .tl_set:N    = \l_stex_symdecl_type_tl ,
2605      def      .tl_set:N    = \l_stex_symdecl_definiens_tl ,
2606      op       .tl_set:N    = \l__stex_notation_op_tl ,
2607      lang     .str_set_x:N = \l__stex_notation_lang_str ,
2608      variant  .str_set_x:N = \l__stex_notation_variant_str ,
2609      prec     .str_set_x:N = \l__stex_notation_prec_str ,
2610      assoc    .choices:nn  =
2611        {bin,binl,binr,pre,conj,pwconj}
2612        {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
```

```
2613    unknown  .code:n      = \str_set:Nx
2614        \l__stex_notation_variant_str \l_keys_key_str
2615  }
2616
2617  \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2618    \str_clear:N \l_stex_symdecl_name_str
2619    \str_clear:N \l_stex_symdecl_args_str
2620    \str_clear:N \l_stex_symdecl_assoctype_str
2621    \bool_set_false:N \l_stex_symdecl_local_bool
2622    \tl_clear:N \l_stex_symdecl_type_tl
2623    \tl_clear:N \l_stex_symdecl_definiens_tl
2624    \str_clear:N \l__stex_notation_lang_str
2625    \str_clear:N \l__stex_notation_variant_str
2626    \str_clear:N \l__stex_notation_prec_str
2627    \tl_clear:N \l__stex_notation_op_tl
2628
2629    \keys_set:nn { stex / symdef } { #1 }
2630  }
2631
2632  \NewDocumentCommand \symdef { m O{} } {
2633    \__stex_notation_symdef_args:n { #2 }
2634    \bool_set_true:N \l_stex_symdecl_make_macro_bool
2635    \stex_symdecl_do:n { #1 }
2636    \tl_set:Nn \l_stex_notation_after_do_tl {
2637      \__stex_notation_final:
2638      \stex_smsmode_do:\ignorespacesandpars
2639    }
2640    \str_set:Nx \l_stex_get_symbol_uri_str {
2641      \l_stex_current_module_str ? \l_stex_symdecl_name_str
2642    }
2643    \exp_args:Nx \stex_notation_do:nnnnn
2644      { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } } }
2645      { \prop_item:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } } }
2646      { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2647      { \l__stex_notation_prec_str}
2648  }
2649  \stex_deactivate_macro:Nn \symdef {module~environments}
```

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

29.3   Variables

```
2650  ⟨@@=stex_variables⟩
2651
2652  \keys_define:nn { stex / vardef } {
2653    name     .str_set_x:N  = \l__stex_variables_name_str ,
2654    args     .str_set_x:N  = \l__stex_variables_args_str ,
2655    type     .tl_set:N     = \l__stex_variables_type_tl ,
2656    def      .tl_set:N     = \l__stex_variables_def_tl ,
2657    op       .tl_set:N     = \l__stex_variables_op_tl ,
2658    prec     .str_set_x:N  = \l__stex_variables_prec_str ,
2659    assoc    .choices:nn    =
2660        {bin,binl,binr,pre,conj,pwconj}
2661        {\str_set:Nx \l_stex_variables_assoctype_str {\l_keys_choice_tl}},
```

```
2662   bind     .choices:nn   =
2663        {forall,exists}
2664        {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2665 }
2666
2667 \cs_new_protected:Nn \__stex_variables_args:n {
2668   \str_clear:N \l__stex_variables_name_str
2669   \str_clear:N \l__stex_variables_args_str
2670   \str_clear:N \l__stex_variables_prec_str
2671   \str_clear:N \l__stex_variables_assoctype_str
2672   \str_clear:N \l__stex_variables_bind_str
2673   \tl_clear:N \l__stex_variables_type_tl
2674   \tl_clear:N \l__stex_variables_def_tl
2675   \tl_clear:N \l__stex_variables_op_tl
2676
2677   \keys_set:nn { stex / vardef } { #1 }
2678 }
2679
2680 \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{}} {
2681   \__stex_variables_args:n {#2}
2682   \str_if_empty:NT \l__stex_variables_name_str {
2683     \str_set:Nx \l__stex_variables_name_str { #1 }
2684   }
2685   \prop_clear:N \l_tmpa_prop
2686   \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
2687
2688   \int_zero:N \l_tmpb_int
2689   \bool_set_true:N \l_tmpa_bool
2690   \str_map_inline:Nn \l__stex_variables_args_str {
2691     \token_case_meaning:NnF ##1 {
2692       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2693       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2694       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2695       {\tl_to_str:n a} {
2696         \bool_set_false:N \l_tmpa_bool
2697         \int_incr:N \l_tmpb_int
2698       }
2699       {\tl_to_str:n B} {
2700         \bool_set_false:N \l_tmpa_bool
2701         \int_incr:N \l_tmpb_int
2702       }
2703     }{
2704       \msg_error:nnxx{stex}{error/wrongargs}{
2705         variable~\l__stex_variables_name_str
2706       }{##1}
2707     }
2708   }
2709   \bool_if:NTF \l_tmpa_bool {
2710     % possibly numeric
2711     \str_if_empty:NTF \l__stex_variables_args_str {
2712       \prop_put:Nnn \l_tmpa_prop { args } {}
2713       \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2714     }{
2715       \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
```

```
2716        \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2717        \str_clear:N \l_tmpa_str
2718        \int_step_inline:nn \l_tmpa_int {
2719          \str_put_right:Nn \l_tmpa_str i
2720        }
2721        \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
2722        \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2723      }
2724    } {
2725      \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2726      \prop_put:Nnx \l_tmpa_prop { arity }
2727        { \str_count:N \l__stex_variables_args_str }
2728    }
2729    \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2730    \tl_set:cx { #1 }{ \stex_invoke_variable:n { \l__stex_variables_name_str } }
2731
2732    \prop_set_eq:cN { l_stex_variable_\l__stex_variables_name_str _prop} \l_tmpa_prop
2733
2734    \tl_if_empty:NF \l__stex_variables_op_tl {
2735      \cs_set:cpx {
2736        stex_var_op_notation_ \l__stex_variables_name_str _cs
2737      } { \exp_not:N\comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } } }
2738    }
2739
2740    \tl_set:Nn \l_stex_notation_after_do_tl {
2741      \exp_args:Nne \use:nn {
2742        \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str _cs }
2743          \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
2744      } {{
2745        \exp_after:wN \exp_after:wN \exp_after:wN
2746        \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2747        { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_symb
2748      }}
2749      \stex_if_do_html:T {
2750        \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
2751          \stex_annotate_invisible:nnn { precedence }
2752            { \l__stex_variables_prec_str }{}
2753          \tl_if_empty:NF \l__stex_variables_type_tl {\stex_annotate_invisible:nnn{type}{}{$\l
2754          \stex_annotate_invisible:nnn{args}{}{ \l__stex_variables_args_str }
2755          \stex_annotate_invisible:nnn{macroname}{#1}{}
2756          \tl_if_empty:NF \l__stex_variables_def_tl {
2757            \stex_annotate_invisible:nnn{definiens}{}
2758              {$\l__stex_variables_def_tl$}
2759          }
2760          \str_if_empty:NF \l__stex_variables_assoctype_str {
2761            \stex_annotate_invisible:nnn{assoctype}{\l__stex_variables_assoctype_str}{}
2762          }
2763          \int_zero:N \l_tmpa_int
2764          \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
2765          \tl_clear:N \l_tmpa_tl
2766          \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
2767            \int_incr:N \l_tmpa_int
2768            \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
2769            \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables
```

159

```
2770            \str_if_eq:VnTF \l_tmpb_str a {
2771              \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2772                \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2773                \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2774              } }
2775            }{
2776              \str_if_eq:VnTF \l_tmpb_str B {
2777                \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2778                  \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2779                  \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2780                } }
2781              }{
2782                \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2783                  \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2784                } }
2785              }
2786            }
2787          }
2788          \stex_annotate_invisible:nnn { notationcomp }{}{
2789            \str_set:Nx \l_stex_current_symbol_str {var://\l__stex_variables_name_str }
2790            $ \exp_args:Nno \use:nn { \use:c {
2791              stex_var_notation_\l__stex_variables_name_str _cs
2792            } } { \l_tmpa_tl } $
2793          }
2794        }
2795      }\ignorespacesandpars
2796    }
2797
2798    \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
2799 }
2800
2801 \cs_new:Nn \_stex_reset:N {
2802    \tl_if_exist:NTF #1 {
2803      \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
2804    }{
2805      \let \exp_not:N #1 \exp_not:N \undefined
2806    }
2807 }
2808
2809 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
2810    \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
2811    \exp_args:Nnx \use:nn {
2812      % TODO
2813      \stex_annotate_invisible:nnn {vardecls}{\clist_use:Nn\l__stex_variables_names,}{
2814        #2
2815      }
2816    }{
2817      \_stex_reset:N \varnot
2818      \_stex_reset:N \vartype
2819      \_stex_reset:N \vardefi
2820    }
2821 }
2822
2823 \NewDocumentCommand \vardef { s } {
```

```
2824    \IfBooleanTF#1 {
2825      \__stex_variables_do_complex:nn
2826    }{
2827      \__stex_variables_do_simple:nnn
2828    }
2829  }

2830
2831  \NewDocumentCommand \svar { O{} m }{
2832    \tl_if_empty:nTF {#1}{
2833      \str_set:Nn \l_tmpa_str { #2 }
2834    }{
2835      \str_set:Nn \l_tmpa_str { #1 }
2836    }
2837    \_stex_term_omv:nn {
2838      var://\l_tmpa_str
2839    }{
2840      \exp_args:Nnx \use:nn {
2841        \def\comp{\_varcomp}
2842        \str_set:Nx \l_stex_current_symbol_str { var://\l_tmpa_str }
2843        \comp{ #2 }
2844      }{
2845        \_stex_reset:N \comp
2846        \_stex_reset:N \l_stex_current_symbol_str
2847      }
2848    }
2849  }

2850

2851

2852
2853  \keys_define:nn { stex / varseq } {
2854    name     .str_set_x:N  = \l__stex_variables_name_str ,
2855    args     .int_set:N    = \l__stex_variables_args_int ,
2856    type     .tl_set:N     = \l__stex_variables_type_tl  ,
2857    mid      .tl_set:N     = \l__stex_variables_mid_tl    ,
2858    bind     .choices:nn   =
2859        {forall,exists}
2860        {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2861  }

2862
2863  \cs_new_protected:Nn \__stex_variables_seq_args:n {
2864    \str_clear:N \l__stex_variables_name_str
2865    \int_set:Nn \l__stex_variables_args_int 1
2866    \tl_clear:N \l__stex_variables_type_tl
2867    \str_clear:N \l__stex_variables_bind_str

2868
2869    \keys_set:nn { stex / varseq } { #1 }
2870  }

2871
2872  \NewDocumentCommand \varseq {m O{} m m m}{
2873    \__stex_variables_seq_args:n { #2 }
2874    \str_if_empty:NT \l__stex_variables_name_str {
2875      \str_set:Nx \l__stex_variables_name_str { #1 }
2876    }
2877    \prop_clear:N \l_tmpa_prop
```

161

```latex
2878    \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
2879
2880    \seq_set_from_clist:Nn \l_tmpa_seq {#3}
2881    \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
2882      \msg_error:nnxx{stex}{error/seqlength}
2883        {\int_use:N \l__stex_variables_args_int}
2884        {\seq_count:N \l_tmpa_seq}
2885    }
2886    \seq_set_from_clist:Nn \l_tmpb_seq {#4}
2887    \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
2888      \msg_error:nnxx{stex}{error/seqlength}
2889        {\int_use:N \l__stex_variables_args_int}
2890        {\seq_count:N \l_tmpb_seq}
2891    }
2892    \prop_put:Nnn \l_tmpa_prop {starts} {#3}
2893    \prop_put:Nnn \l_tmpa_prop {ends} {#4}
2894
2895    \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
2896      \cs_set:Npn  {\int_use:N \l__stex_variables_args_int} { #5 }
2897
2898    \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
2899    \int_step_inline:nn \l__stex_variables_args_int {
2900      \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
2901    }
2902    \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
2903    \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
2904    \tl_if_empty:NF \l__stex_variables_mid_tl {
2905      \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
2906      \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
2907    }
2908    \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
2909    \int_step_inline:nn \l__stex_variables_args_int {
2910      \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
2911    }
2912    \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
2913    \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
2914
2915
2916    \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl
2917
2918    \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}
2919
2920    \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
2921
2922    \int_step_inline:nn \l__stex_variables_args_int {
2923      \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
2924        \_stex_term_math_arg:nnn{##1}{0}{\exp_not:n{####}##1}
2925      }}
2926    }
2927
2928    \tl_set:Nx \l_tmpa_tl {
2929      \_stex_term_math_oma:nnnn { varseq://\l__stex_variables_name_str}{}{0}{
2930        \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
2931      }
```

```
2932    }

2933

2934    \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \stex_symbol_after_invokation_tl} }

2935

2936    \exp_args:Nno \use:nn {
2937    \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
2938      \cs_set:Npn {\int_use:N \l__stex_variables_args_int}}{\l_tmpa_tl}

2939

2940    \stex_debug:nn{sequences}{New~Sequence:~
2941      \expandafter\meaning\csname stex_varseq_\l__stex_variables_name_str _cs\endcsname\\~\\
2942      \prop_to_keyval:N \l_tmpa_prop
2943    }

2944

2945    \prop_set_eq:cN {stex_varseq_\l__stex_variables_name_str _prop}\l_tmpa_prop
2946    \ignorespacesandpars
2947 }

2948

2949 ⟨/package⟩
```

163

Chapter 30

STEX
-Terms Implementation

```
2950  ⟨∗package⟩
2951
2952  %%%%%%%%%%%%    terms.dtx    %%%%%%%%%%%%
2953
2954  ⟨@@=stex_terms⟩
```

Warnings and error messages

```
2955  \msg_new:nnn{stex}{error/nonotation}{
2956    Symbol~#1~invoked,~but~has~no~notation#2!
2957  }
2958  \msg_new:nnn{stex}{error/notationarg}{
2959    Error~in~parsing~notation~#1
2960  }
2961  \msg_new:nnn{stex}{error/noop}{
2962    Symbol~#1~has~no~operator~notation~for~notation~#2
2963  }
2964  \msg_new:nnn{stex}{error/notallowed}{
2965    Symbol~invokation~#1~not~allowed~in~notation~component~of~#2
2966  }
2967
```

30.1   Symbol Invocations

`\stex_invoke_symbol:n`   Invokes a semantic macro

```
2968
2969
2970  \bool_new:N \l_stex_allow_semantic_bool
2971  \bool_set_true:N \l_stex_allow_semantic_bool
2972
2973  \cs_new_protected:Nn \stex_invoke_symbol:n {
2974    \bool_if:NTF \l_stex_allow_semantic_bool {
2975      \str_if_eq:eeF {
2976        \prop_item:cn {
2977          l_stex_symdecl_#1_prop
2978        }{ deprecate }
```

```
2979    }{}{
2980      \msg_warning:nnxx{stex}{warning/deprecated}{
2981        Symbol~#1
2982      }{
2983        \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
2984      }
2985    }
2986    \if_mode_math:
2987      \exp_after:wN \__stex_terms_invoke_math:n
2988    \else:
2989      \exp_after:wN \__stex_terms_invoke_text:n
2990    \fi: { #1 }
2991  }{
2992    \msg_error:nnxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
2993  }
2994 }

2995
2996 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
2997   \peek_charcode_remove:NTF ! {
2998     \__stex_terms_invoke_op_custom:nn {#1}
2999   }{
3000     \__stex_terms_invoke_custom:nn {#1}
3001   }
3002 }

3003
3004 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
3005   \peek_charcode_remove:NTF ! {
3006     % operator
3007     \peek_charcode_remove:NTF * {
3008       % custom op
3009       \__stex_terms_invoke_op_custom:nn {#1}
3010     }{
3011       % op notation
3012       \peek_charcode:NTF [ {
3013         \__stex_terms_invoke_op_notation:nw {#1}
3014       }{
3015         \__stex_terms_invoke_op_notation:nw {#1}[]
3016       }
3017     }
3018   }{
3019     \peek_charcode_remove:NTF * {
3020       \__stex_terms_invoke_custom:nn {#1}
3021       % custom
3022     }{
3023       % normal
3024       \peek_charcode:NTF [ {
3025         \__stex_terms_invoke_notation:nw {#1}
3026       }{
3027         \__stex_terms_invoke_notation:nw {#1}[]
3028       }
3029     }
3030   }
3031 }

3032
```

```
3033
3034 \cs_new_protected:Nn \__stex_terms_invoke_op_custom:nn {
3035   \exp_args:Nnx \use:nn {
3036     \def\comp{\_comp}
3037     \str_set:Nn \l_stex_current_symbol_str { #1 }
3038     \bool_set_false:N \l_stex_allow_semantic_bool
3039     \_stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
3040       \comp{ #2 }
3041     }
3042   }{
3043     \_stex_reset:N \comp
3044     \_stex_reset:N \l_stex_current_symbol_str
3045     \bool_set_true:N \l_stex_allow_semantic_bool
3046   }
3047 }
3048
3049 \keys_define:nn { stex / terms } {
3050   lang    .tl_set_x:N = \l_stex_notation_lang_str ,
3051   variant .tl_set_x:N = \l_stex_notation_variant_str ,
3052   unknown .code:n     = \str_set:Nx
3053       \l_stex_notation_variant_str \l_keys_key_str
3054 }
3055
3056 \cs_new_protected:Nn \__stex_terms_args:n {
3057   \str_clear:N \l_stex_notation_lang_str
3058   \str_clear:N \l_stex_notation_variant_str
3059
3060   \keys_set:nn { stex / terms } { #1 }
3061 }
3062
3063 \cs_new_protected:Nn \stex_find_notation:nn {
3064   \__stex_terms_args:n { #2 }
3065   \seq_if_empty:cTF {
3066     l_stex_symdecl_ #1 _notations
3067   } {
3068     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
3069   } {
3070     \bool_lazy_all:nTF {
3071       {\str_if_empty_p:N \l_stex_notation_variant_str}
3072       {\str_if_empty_p:N \l_stex_notation_lang_str}
3073     }{
3074       \seq_get_left:cN {l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
3075     }{
3076       \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
3077         \l_stex_notation_variant_str \c_hash_str \l_stex_notation_lang_str
3078       }{
3079         \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3080       }{
3081         \msg_error:nnxx{stex}{error/nonotation}{#1}{
3082           ~\l_stex_notation_variant_str \c_hash_str \l_stex_notation_lang_str
3083         }
3084       }
3085     }
3086   }
```

166

```
3087 }
3088
3089 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3090   \exp_args:Nnx \use:nn {
3091     \def\comp{\_comp}
3092     \str_set:Nn \l_stex_current_symbol_str { #1 }
3093     \stex_find_notation:nn { #1 }{ #2 }
3094     \bool_set_false:N \l_stex_allow_semantic_bool
3095     \cs_if_exist:cTF {
3096       stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3097     }{
3098       \_stex_term_oms:nnn {
3099         #1 \c_hash_str \l_stex_notation_variant_str
3100       }{ #1 }{
3101         \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3102       }
3103     }{
3104       \int_compare:nNnTF {\prop_item:cn {l_stex_symdecl_#1_prop}{arity}} = 0{
3105         \cs_if_exist:cTF {
3106           stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3107         }{
3108           \tl_set:Nx \stex_symbol_after_invokation_tl {
3109             \_stex_reset:N \comp
3110             \_stex_reset:N \stex_symbol_after_invokation_tl
3111             \_stex_reset:N \l_stex_current_symbol_str
3112             \bool_set_true:N \l_stex_allow_semantic_bool
3113           }
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           \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3118         }{
3119           \msg_error:nnxx{stex}{error/nonotation}{#1}{
3120             ~\l_stex_notation_variant_str
3121           }
3122         }
3123       }{
3124         \msg_error:nnxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3125       }
3126     }
3127   }{
3128     \_stex_reset:N \comp
3129     \_stex_reset:N \l_stex_current_symbol_str
3130     \bool_set_true:N \l_stex_allow_semantic_bool
3131   }
3132 }
3133
3134 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {
3135   \stex_find_notation:nn { #1 }{ #2 }
3136   \cs_if_exist:cTF {
3137     stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3138   }{
3139     \tl_set:Nx \stex_symbol_after_invokation_tl {
3140       \_stex_reset:N \comp
```

```
3141        \_stex_reset:N \stex_symbol_after_invokation_tl
3142        \_stex_reset:N \l_stex_current_symbol_str
3143        \bool_set_true:N \l_stex_allow_semantic_bool
3144      }
3145      \def\comp{\_comp}
3146      \str_set:Nn \l_stex_current_symbol_str { #1 }
3147      \bool_set_false:N \l_stex_allow_semantic_bool
3148      \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3149    }{
3150      \msg_error:nnxx{stex}{error/nonotation}{#1}{
3151        ~\l_stex_notation_variant_str
3152      }
3153    }
3154 }
3155
3156 \prop_new:N \l__stex_terms_custom_args_prop
3157
3158 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3159    \exp_args:Nnx \use:nn {
3160      \bool_set_false:N \l_stex_allow_semantic_bool
3161      \def\comp{\_comp}
3162      \str_set:Nn \l_stex_current_symbol_str { #1 }
3163      \prop_clear:N \l__stex_terms_custom_args_prop
3164      \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3165      \prop_get:cnN {
3166        l_stex_symdecl_#1 _prop
3167      }{ args } \l_tmpa_str
3168      \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3169      \tl_set:Nn \arg { \__stex_terms_arg: }
3170      \str_if_empty:NTF \l_tmpa_str {
3171        \_stex_term_oms:nnn {#1}{#1}{#2}
3172      }{
3173        \str_if_in:NnTF \l_tmpa_str b {
3174          \_stex_term_ombind:nnn {#1}{#1}{#2}
3175        }{
3176          \str_if_in:NnTF \l_tmpa_str B {
3177            \_stex_term_ombind:nnn {#1}{#1}{#2}
3178          }{
3179            \_stex_term_oma:nnn {#1}{#1}{#2}
3180          }
3181        }
3182      }
3183      % TODO check that all arguments exist
3184    }{
3185      \_stex_reset:N \l_stex_current_symbol_str
3186      \_stex_reset:N \arg
3187      \_stex_reset:N \comp
3188      \_stex_reset:N \l__stex_terms_custom_args_prop
3189      \bool_set_true:N \l_stex_allow_semantic_bool
3190    }
3191 }
3192
3193 \NewDocumentCommand \__stex_terms_arg: { s O{} m}{
3194    \tl_if_empty:nTF {#2}{
```

```
3195    \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
3196    \bool_set_true:N \l_tmpa_bool
3197    \bool_do_while:Nn \l_tmpa_bool {
3198      \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int
3199        \int_incr:N \l_tmpa_int
3200      }{
3201        \bool_set_false:N \l_tmpa_bool
3202      }
3203    }
3204  }{
3205    \int_set:Nn \l_tmpa_int { #2 }
3206    \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
3207      % TODO throw error
3208    }
3209  }
3210  \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
3211  \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
3212    % TODO throw error
3213  }
3214  \bool_set_true:N \l_stex_allow_semantic_bool
3215  \IfBooleanTF#1{
3216    \stex_annotate_invisible:n {
3217      \exp_args:No \_stex_term_arg:nn {\l_stex_current_symbol_str}{#3}
3218    }
3219  }{
3220    \exp_args:No \_stex_term_arg:nn {\l_stex_current_symbol_str}{#3}
3221  }
3222  \bool_set_false:N \l_stex_allow_semantic_bool
3223 }
3224
3225
3226 \cs_new_protected:Nn \_stex_term_arg:nn {
3227   \bool_set_true:N \l_stex_allow_semantic_bool
3228   \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3229   \bool_set_false:N \l_stex_allow_semantic_bool
3230 }
3231
3232 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
3233   \exp_args:Nnx \use:nn
3234     { \int_set:Nn \l__stex_terms_downprec { #2 }
3235         \_stex_term_arg:nn { #1 }{ #3 }
3236     }
3237     { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3238 }
```

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

```
3239 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
3240   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3241   \tl_set:Nn \l_tmpb_tl {\_stex_term_math_arg:nnn{#1}{#2}}
3242   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 }}{
3243     \expandafter\if\expandafter\relax\noexpand#3
3244       \expandafter\__stex_terms_math_assoc_arg_maybe_sequence:N\expandafter#3
```

169

```
3245        \else\expandafter\__stex_terms_math_assoc_arg_simple:n\expandafter#3\fi
3246      }{
3247        \__stex_terms_math_assoc_arg_simple:n{#3}
3248      }
3249  }
3250
3251  \cs_new_protected:Nn \__stex_terms_math_assoc_arg_maybe_sequence:N {
3252    \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
3253    \str_if_empty:NTF \l_tmpa_str {
3254      \exp_args:Nx \cs_if_eq:NNTF {
3255        \tl_head:N #1
3256      } \stex_invoke_sequence:n {
3257        \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
3258        \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
3259        \tl_set:Nx \l_tmpa_tl {\prop_item:cn {stex_varseq_\l_tmpa_str _prop}{notation}}
3260        \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
3261        \tl_set:Nx \l_tmpa_tl {{\exp_not:N \exp_not:n{
3262          \exp_not:n{\exp_args:Nnx \use:nn} {
3263            \exp_not:n {
3264              \def\comp{\_varcomp}
3265              \str_set:Nn \l_stex_current_symbol_str
3266            } {varseq://\l_tmpa_str}
3267            \exp_not:n{ ##1 }
3268          }{
3269            \exp_not:n {
3270              \_stex_reset:N \comp
3271              \_stex_reset:N \l_stex_current_symbol_str
3272            }
3273          }
3274        }}}
3275        \exp_args:Nno \use:nn {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
3276        \seq_reverse:N \l_tmpa_seq
3277        \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
3278        \seq_map_inline:Nn \l_tmpa_seq {
3279          \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3280            \exp_args:Nno
3281            \l_tmpa_cs { ##1 } \l_tmpa_tl
3282          }
3283        }
3284        \tl_set:Nx \l_tmpa_tl {
3285          \_stex_term_omv:nn {varseq://\l_tmpa_str}{
3286            \exp_args:No \exp_not:n \l_tmpa_tl
3287          }
3288        }
3289        \exp_args:No\l_tmpb_tl\l_tmpa_tl
3290      }{
3291        \__stex_terms_math_assoc_arg_simple:n { #1 }
3292      }
3293    } {
3294      \__stex_terms_math_assoc_arg_simple:n { #1 }
3295    }
3296
3297  }
3298
```

```
\cs_new_protected:Nn \__stex_terms_math_assoc_arg_simple:n {
  \clist_set:Nn \l_tmpa_clist{ #1 }
  \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
    \tl_set:Nn \l_tmpa_tl { #1 }
  }{
    \clist_reverse:N \l_tmpa_clist
    \clist_pop:NN \l_tmpa_clist \l_tmpa_tl

    \clist_map_inline:Nn \l_tmpa_clist {
      \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
        \exp_args:Nno
        \l_tmpa_cs { ##1 } \l_tmpa_tl
      }
    }
  }
  \exp_args:No\l_tmpb_tl\l_tmpa_tl
}
```

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

30.2 Terms

Precedences:

\infprec
\neginfprec
\l__stex_terms_downprec

```
\tl_const:Nx \infprec {\int_use:N \c_max_int}
\tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
\int_new:N \l__stex_terms_downprec
\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:

\l__stex_terms_left_bracket_str
\l__stex_terms_right_bracket_str

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

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

__stex_terms_maybe_brackets:nn

Compares precedences and insert brackets accordingly

```
\cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
  \bool_if:NTF \l__stex_terms_brackets_done_bool {
    \bool_set_false:N \l__stex_terms_brackets_done_bool
    #2
  } {
    \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
      \bool_if:NTF \l_stex_inparray_bool { #2 }{
        \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
        \dobrackets { #2 }
      }
    }{ #2 }
  }
}
```

(*End definition for* __stex_terms_maybe_brackets:nn.)

\dobrackets

```
3335 \bool_new:N \l__stex_terms_brackets_done_bool
3336 %\RequirePackage{scalerel}
3337 \cs_new_protected:Npn \dobrackets #1 {
3338   %\ThisStyle{\if D\m@switch
3339   %    \exp_args:Nnx \use:nn
3340   %    { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
3341   %    { \exp_not:N\right\l__stex_terms_right_bracket_str }
3342   % \else
3343      \exp_args:Nnx \use:nn
3344      {
3345        \bool_set_true:N \l__stex_terms_brackets_done_bool
3346        \int_set:Nn \l__stex_terms_downprec \infprec
3347        \l__stex_terms_left_bracket_str
3348        #1
3349      }
3350      {
3351        \bool_set_false:N \l__stex_terms_brackets_done_bool
3352        \l__stex_terms_right_bracket_str
3353        \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3354      }
3355   %\fi}
3356 }
```

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

\withbrackets

```
3357 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
3358   \exp_args:Nnx \use:nn
3359   {
3360     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
3361     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
3362     #3
3363   }
3364   {
3365     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
3366       {\l__stex_terms_left_bracket_str}
3367     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
3368       {\l__stex_terms_right_bracket_str}
3369   }
3370 }
```

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

\STEXinvisible

```
3371 \cs_new_protected:Npn \STEXinvisible #1 {
3372   \stex_annotate_invisible:n { #1 }
3373 }
```

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

OMDoc terms:

_stex_term_math_oms:nnnn

```
3374 \cs_new_protected:Nn \_stex_term_oms:nnn {
3375   \stex_annotate:nnn{ OMID }{ #2 }{
3376     \stex_highlight_term:nn { #1 } { #3 }
3377   }
3378 }
3379
3380 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
3381   \__stex_terms_maybe_brackets:nn { #3 }{
3382     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3383   }
3384 }
```

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

_stex_term_math_omv:nn

```
3385 \cs_new_protected:Nn \_stex_term_omv:nn {
3386   \stex_annotate:nnn{ OMV }{ #1 }{
3387     \stex_highlight_term:nn { #1 } { #2 }
3388   }
3389 }
```

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

_stex_term_math_oma:nnnn

```
3390 \cs_new_protected:Nn \_stex_term_oma:nnn {
3391   \stex_annotate:nnn{ OMA }{ #2 }{
3392     \stex_highlight_term:nn { #1 } { #3 }
3393   }
3394 }
3395
3396 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
3397   \__stex_terms_maybe_brackets:nn { #3 }{
3398     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3399   }
3400 }
```

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

_stex_term_math_omb:nnnn

```
3401 \cs_new_protected:Nn \_stex_term_ombind:nnn {
3402   \stex_annotate:nnn{ OMBIND }{ #2 }{
3403     \stex_highlight_term:nn { #1 } { #3 }
3404   }
3405 }
3406
3407 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
3408   \__stex_terms_maybe_brackets:nn { #3 }{
3409     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3410   }
3411 }
```

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

173

```
3412 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3413
3414 \keys_define:nn { stex / symname } {
3415   pre      .tl_set_x:N   = \l__stex_terms_pre_tl ,
3416   post     .tl_set_x:N   = \l__stex_terms_post_tl ,
3417   root     .tl_set_x:N   = \l__stex_terms_root_tl
3418 }
3419
3420 \cs_new_protected:Nn \stex_symname_args:n {
3421   \tl_clear:N \l__stex_terms_post_tl
3422   \tl_clear:N \l__stex_terms_pre_tl
3423   \tl_clear:N \l__stex_terms_root_str
3424   \keys_set:nn { stex / symname } { #1 }
3425 }
3426
3427 \NewDocumentCommand \symref { m m }{
3428   \let\compemph_uri_prev:\compemph@uri
3429   \let\compemph@uri\symrefemph@uri
3430   \STEXsymbol{#1}!{ #2 }
3431   \let\compemph@uri\compemph_uri_prev:
3432 }
3433
3434 \NewDocumentCommand \synonym { O{} m m}{
3435   \stex_symname_args:n { #1 }
3436   \let\compemph_uri_prev:\compemph@uri
3437   \let\compemph@uri\symrefemph@uri
3438   % TODO
3439   \STEXsymbol{#2}!{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
3440   \let\compemph@uri\compemph_uri_prev:
3441 }
3442
3443 \NewDocumentCommand \symname { O{} m }{
3444   \stex_symname_args:n { #1 }
3445   \stex_get_symbol:n { #2 }
3446   \str_set:Nx \l_tmpa_str {
3447     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3448   }
3449   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3450
3451   \let\compemph_uri_prev:\compemph@uri
3452   \let\compemph@uri\symrefemph@uri
3453   \exp_args:NNx \use:nn
3454   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!{
3455     \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
3456   } }
3457   \let\compemph@uri\compemph_uri_prev:
3458 }
3459
3460 \NewDocumentCommand \Symname { O{} m }{
3461   \stex_symname_args:n { #1 }
3462   \stex_get_symbol:n { #2 }
3463   \str_set:Nx \l_tmpa_str {
3464     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
```

174

```
3465        }
3466      \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3467      \let\compemph_uri_prev:\compemph@uri
3468      \let\compemph@uri\symrefemph@uri
3469      \exp_args:NNx \use:nn
3470      \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!{
3471        \exp_after:wN \stex_capitalize:n \l_tmpa_str
3472          \l__stex_terms_post_tl
3473      } }
3474      \let\compemph@uri\compemph_uri_prev:
3475    }
```

(*End definition for* `\symref` *and* `\symname`. *These functions are documented on page* *62.*)

30.3 Notation Components

```
3476    ⟨@@=stex_notationcomps⟩
```

```
3477    \cs_new_protected:Nn \stex_highlight_term:nn {
3478      #2
3479    }
3480
3481    \cs_new_protected:Nn \stex_unhighlight_term:n {
3482    %  \latexml_if:TF {
3483    %    #1
3484    %  } {
3485    %    \rustex_if:TF {
3486    %      #1
3487    %    } {
3488        #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
3489    %    }
3490    %  }
3491    }
```

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

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

```
3492    \cs_new_protected:Npn \_comp #1 {
3493      \str_if_empty:NF \l_stex_current_symbol_str {
3494        \rustex_if:TF {
3495          \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
3496        }{
3497          \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3498        }
3499      }
3500    }
3501
3502    \cs_new_protected:Npn \_varcomp #1 {
3503      \str_if_empty:NF \l_stex_current_symbol_str {
3504        \rustex_if:TF {
3505          \stex_annotate:nnn { varcomp }{ \l_stex_current_symbol_str }{ #1 }
3506        }{
3507          \exp_args:Nnx \varemph@uri { #1 } { \l_stex_current_symbol_str }
```

```
3508          }
3509        }
3510     }
3511
3512     \def\comp{\_comp}
3513
3514     \cs_new_protected:Npn \compemph@uri #1 #2 {
3515        \compemph{ #1 }
3516     }
3517
3518
3519     \cs_new_protected:Npn \compemph #1 {
3520        #1
3521     }
3522
3523     \cs_new_protected:Npn \defemph@uri #1 #2 {
3524        \defemph{#1}
3525     }
3526
3527     \cs_new_protected:Npn \defemph #1 {
3528        \textbf{#1}
3529     }
3530
3531     \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3532        \symrefemph{#1}
3533     }
3534
3535     \cs_new_protected:Npn \symrefemph #1 {
3536        \textbf{#1}
3537     }
3538
3539     \cs_new_protected:Npn \varemph@uri #1 #2 {
3540        \varemph{#1}
3541     }
3542
3543     \cs_new_protected:Npn \varemph #1 {
3544        #1
3545     }
```

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

\ellipses

```
3546     \NewDocumentCommand \ellipses {} { \ldots }
```

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

\parray
\prmatrix
\parrayline
\parraylineh
\parraycell

```
3547     \bool_new:N \l_stex_inparray_bool
3548     \bool_set_false:N \l_stex_inparray_bool
3549     \NewDocumentCommand \parray { m m } {
3550        \begingroup
3551        \bool_set_true:N \l_stex_inparray_bool
3552        \begin{array}{#1}
3553          #2
3554        \end{array}
```

```
3555    \endgroup
3556 }
3557
3558 \NewDocumentCommand \prmatrix { m } {
3559    \begingroup
3560    \bool_set_true:N \l_stex_inparray_bool
3561    \begin{matrix}
3562       #1
3563    \end{matrix}
3564    \endgroup
3565 }
3566
3567 \def \maybephline {
3568    \bool_if:NT \l_stex_inparray_bool {\hline}
3569 }
3570
3571 \def \parrayline #1 #2 {
3572    #1 #2 \bool_if:NT \l_stex_inparray_bool {\\}
3573 }
3574
3575 \def \pmrow #1 { \parrayline{}{ #1 } }
3576
3577 \def \parraylineh #1 #2 {
3578    #1 #2 \bool_if:NT \l_stex_inparray_bool {\\\hline}
3579 }
3580
3581 \def \parraycell #1 {
3582    #1 \bool_if:NT \l_stex_inparray_bool {&}
3583 }
```

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

30.4   Variables

```
3584 ⟨@@=stex_variables⟩
```

\stex_invoke_variable:n   Invokes a variable

```
3585 \cs_new_protected:Nn \stex_invoke_variable:n {
3586    \if_mode_math:
3587       \exp_after:wN \__stex_variables_invoke_math:n
3588    \else:
3589       \exp_after:wN \__stex_variables_invoke_text:n
3590    \fi: {#1}
3591 }
3592
3593 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
3594    %TODO
3595 }
3596
3597
3598 \cs_new_protected:Nn \__stex_variables_invoke_math:n {
3599    \peek_charcode_remove:NTF ! {
3600       \peek_charcode_remove:NTF ! {
3601          \peek_charcode:NTF [ {
```

```
3602          \__stex_variables_invoke_op_custom:nw
3603        }{
3604          % TODO throw error
3605        }
3606      }{
3607        \__stex_variables_invoke_op:n { #1 }
3608      }
3609    }{
3610      \peek_charcode_remove:NTF * {
3611        \__stex_variables_invoke_text:n { #1 }
3612      }{
3613        \__stex_variables_invoke_math_ii:n { #1 }
3614      }
3615    }
3616  }
3617
3618  \cs_new_protected:Nn \__stex_variables_invoke_op:n {
3619    \cs_if_exist:cTF {
3620      stex_var_op_notation_ #1 _cs
3621    }{
3622      \exp_args:Nnx \use:nn {
3623        \def\comp{\_varcomp}
3624        \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3625        \_stex_term_omv:nn { var://#1 }{
3626          \use:c{stex_var_op_notation_ #1 _cs }
3627        }
3628      }{
3629        \_stex_reset:N \comp
3630        \_stex_reset:N \l_stex_current_symbol_str
3631      }
3632    }{
3633      \int_compare:nNnTF {\prop_item:cn {l_stex_variable_#1_prop}{arity}} = 0{
3634        \__stex_variables_invoke_math_ii:n {#1}
3635      }{
3636        \msg_error:nnxx{stex}{error/noop}{variable~#1}{}
3637      }
3638    }
3639  }
3640
3641  \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n  #1 {
3642    \cs_if_exist:cTF {
3643      stex_var_notation_#1cs
3644    }{
3645      \tl_set:Nx \stex_symbol_after_invokation_tl {
3646        \_stex_reset:N \comp
3647        \_stex_reset:N \stex_symbol_after_invokation_tl
3648        \_stex_reset:N \l_stex_current_symbol_str
3649        \bool_set_true:N \l_stex_allow_semantic_bool
3650      }
3651      \def\comp{\_varcomp}
3652      \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3653      \bool_set_false:N \l_stex_allow_semantic_bool
3654      \use:c{stex_var_notation_#1_cs}
3655    }{
```

```
3656        \msg_error:nnxx{stex}{error/nonotation}{variable~#1}{s}
3657    }
3658 }
```

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

30.5   Sequences

```
3659 ⟨@@=stex_sequences⟩
3660
3661 \cs_new_protected:Nn \stex_invoke_sequence:n {
3662    \peek_charcode_remove:NTF ! {
3663      \_stex_term_omv:nn {varseq://#1}{
3664        \exp_args:Nnx \use:nn {
3665          \def\comp{\_varcomp}
3666          \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3667          \prop_item:cn{stex_varseq_#1_prop}{notation}
3668        }{
3669          \_stex_reset:N \comp
3670          \_stex_reset:N \l_stex_current_symbol_str
3671        }
3672      }
3673    }{
3674      \bool_set_false:N \l_stex_allow_semantic_bool
3675      \def\comp{\_varcomp}
3676      \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3677      \tl_set:Nx \stex_symbol_after_invokation_tl {
3678        \_stex_reset:N \comp
3679        \_stex_reset:N \stex_symbol_after_invokation_tl
3680        \_stex_reset:N \l_stex_current_symbol_str
3681        \bool_set_true:N \l_stex_allow_semantic_bool
3682      }
3683      \use:c { stex_varseq_#1_cs }
3684    }
3685 }
3686 ⟨/package⟩
```

Chapter 31

sTeX -Structural Features Implementation

```
3687 ⟨∗package⟩
3688
3689 %%%%%%%%%%%%   features.dtx   %%%%%%%%%%%%
3690
```

Warnings and error messages

```
3691 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3692   Symbol~#1~can~not~be~assigned~in~copymodule~#2
3693 }
3694 \msg_new:nnn{stex}{error/interpretmodule/nodefiniens}{
3695   Symbol~#1~not~assigned~in~interpretmodule~#2
3696 }
3697
3698 \msg_new:nnn{stex}{error/unknownstructure}{
3699   No~structure~#1~found!
3700 }
3701
3702 \msg_new:nnn{stex}{error/unknownfield}{
3703   No~field~#1~in~instance~#2~found!\\#3
3704 }
3705
3706 \msg_new:nnn{stex}{error/keyval}{
3707   Invalid~key=value~pair:#1
3708 }
3709 \msg_new:nnn{stex}{error/instantiate/missing}{
3710   Assignments~missing~in~instantiate:~#1
3711 }
3712 \msg_new:nnn{stex}{error/incompatible}{
3713   Incompatible~signature:~#1~(#2)~and~#3~(#4)
3714 }
3715
```

31.1 Imports with modification

```
3716 ⟨@@=stex_copymodule⟩
3717 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
3718   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3719     \tl_set:Nn \l_tmpa_tl { #1 }
3720     \__stex_copymodule_get_symbol_from_cs:
3721   }{
3722     % argument is a string
3723     % is it a command name?
3724     \cs_if_exist:cTF { #1 }{
3725       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3726       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3727       \str_if_empty:NTF \l_tmpa_str {
3728         \exp_args:Nx \cs_if_eq:NNTF {
3729           \tl_head:N \l_tmpa_tl
3730         } \stex_invoke_symbol:n {
3731           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
3732         }{
3733           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3734         }
3735       } {
3736         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3737       }
3738     }{
3739       % argument is not a command name
3740       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3741       % \l_stex_all_symbols_seq
3742     }
3743   }
3744 }
3745
3746 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
3747   \str_set:Nn \l_tmpa_str { #1 }
3748   \bool_set_false:N \l_tmpa_bool
3749   \bool_if:NF \l_tmpa_bool {
3750     \tl_set:Nn \l_tmpa_tl {
3751       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
3752     }
3753   \str_set:Nn \l_tmpa_str { #1 }
3754   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3755   \seq_map_inline:Nn #2 {
3756     \str_set:Nn \l_tmpb_str { ##1 }
3757     \str_if_eq:eeT { \l_tmpa_str } {
3758       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3759     } {
3760       \seq_map_break:n {
3761         \tl_set:Nn \l_tmpa_tl {
3762           \str_set:Nn \l_stex_get_symbol_uri_str {
3763             ##1
3764           }
3765         }
3766       }
3767     }
```

```
3768        }
3769      \l_tmpa_tl
3770    }
3771  }
3772
3773  \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
3774    \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3775      { \tl_tail:N \l_tmpa_tl }
3776    \tl_if_single:NTF \l_tmpa_tl {
3777      \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3778        \exp_after:wN \str_set:Nn \exp_after:wN
3779          \l_stex_get_symbol_uri_str \l_tmpa_tl
3780        \__stex_copymodule_get_symbol_check:n { #1 }
3781      }{
3782        % TODO
3783        % tail is not a single group
3784      }
3785    }{
3786      % TODO
3787      % tail is not a single group
3788    }
3789  }
3790
3791  \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
3792    \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
3793      \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3794        :~\seq_use:Nn #1 {,~}
3795      }
3796    }
3797  }
3798
3799  \cs_new_protected:Nn \stex_copymodule_start:nnnn {
3800    \stex_import_module_uri:nn { #1 } { #2 }
3801    \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3802    \stex_import_require_module:nnnn
3803      { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3804      { \l_stex_import_path_str } { \l_stex_import_name_str }
3805    \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3806    \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
3807    \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
3808    \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
3809      \seq_map_inline:cn {c_stex_module_##1_constants}{
3810        \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
3811          ##1 ? ####1
3812        }
3813      }
3814    }
3815    \seq_clear:N \l_tmpa_seq
3816    \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3817      name     = \l_stex_current_copymodule_name_str ,
3818      module   = \l_stex_current_module_str ,
3819      from     = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3820      includes = \l_tmpa_seq ,
3821      fields   = \l_tmpa_seq
```

```
3822    }
3823    \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3824      as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3825      \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_se
3826    \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,
3827    \stex_if_smsmode:F {
3828      \begin{stex_annotate_env} {#4} {
3829        \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3830      }
3831      \stex_annotate_invisible:nnn{from}{\l_stex_import_ns_str ?\l_stex_import_name_str}{}
3832    }
3833    \bool_set_eq:NN \l__stex_copymodule_oldhtml_bool \_stex_html_do_output_bool
3834    \bool_set_false:N \_stex_html_do_output_bool
3835  }
3836  \cs_new_protected:Nn \stex_copymodule_end:n {
3837    \def \l_tmpa_cs ##1 ##2 {#1}
3838    \bool_set_eq:NN \_stex_html_do_output_bool \l__stex_copymodule_oldhtml_bool
3839    \tl_clear:N \l_tmpa_tl
3840    \tl_clear:N \l_tmpb_tl
3841    \prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3842    \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
3843      \seq_map_inline:cn {c_stex_module_##1_constants}{
3844        \tl_clear:N \l_tmpc_tl
3845        \l_tmpa_cs{##1}{####1}
3846        \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
3847          \tl_put_right:Nx \l_tmpa_tl {
3848            \prop_set_from_keyval:cn {
3849              l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule
3850            }{
3851              \exp_after:wN \prop_to_keyval:N \csname
3852                l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodu
3853              \endcsname
3854            }
3855            \seq_clear:c {
3856              l_stex_symdecl_
3857              \l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_name
3858              _notations
3859            }
3860          }
3861          \tl_put_right:Nx \l_tmpc_tl {
3862            \stex_copy_notations:nn {\l_stex_current_module_str ? \use:c{l__stex_copymodule_co
3863            \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?####1
3864          }
3865          \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \use:c{l__stex_copymodul
3866          \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
3867            \tl_put_right:Nx \l_tmpc_tl {
3868              \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1
3869            }
3870            \tl_put_right:Nx \l_tmpa_tl {
3871              \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
3872                \stex_invoke_symbol:n {
3873                  \l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_
3874                }
3875              }
```

```
3876              }
3877            }
3878          }{
3879            \tl_put_right:Nx \l_tmpc_tl {
3880              \stex_copy_notations:nn {\l_stex_current_module_str ? \l_stex_current_copymodule_n
3881            }
3882            \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
3883            \prop_put:Nnx \l_tmpa_prop { name }{ \l_stex_current_copymodule_name_str / ####1 }
3884            \prop_put:Nnx \l_tmpa_prop { module }{ \l_stex_current_module_str }
3885            \tl_put_right:Nx \l_tmpa_tl {
3886              \prop_set_from_keyval:cn {
3887                l_stex_symdecl_\l_stex_current_module_str ? \l_stex_current_copymodule_name_str
3888              }{
3889                \prop_to_keyval:N \l_tmpa_prop
3890              }
3891              \seq_clear:c {
3892                l_stex_symdecl_
3893                \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3894                _notations
3895              }
3896            }
3897            \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \l_stex_current_copymodu
3898            \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
3899              \tl_put_right:Nx \l_tmpc_tl {
3900                \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1
3901              }
3902              \tl_put_right:Nx \l_tmpa_tl {
3903                \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
3904                  \stex_invoke_symbol:n {
3905                    \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3906                  }
3907                }
3908              }
3909            }
3910          }
3911          \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
3912            \tl_put_right:Nx \l_tmpc_tl {
3913              \stex_annotate_invisible:nnn{definiens}{}{$\use:c{l__stex_copymodule_copymodule_##
3914            }
3915          }
3916          \tl_put_right:Nx \l_tmpb_tl {
3917            \stex_annotate:nnn{assignment} {##1?####1} { \l_tmpc_tl }
3918          }
3919        }
3920      }
3921      \prop_put:Nno \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3922      \tl_put_left:Nx \l_tmpa_tl {
3923        \prop_set_from_keyval:cn {
3924          l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _pro
3925        }{
3926          \prop_to_keyval:N \l_stex_current_copymodule_prop
3927        }
3928      }
3929      \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl
```

184

```
3930    \stex_debug:nn{copymodule}{result:\meaning \l_tmpa_tl}
3931    \exp_args:Nx \stex_do_up_to_module:n {
3932        \exp_args:No \exp_not:n \l_tmpa_tl
3933    }
3934    \l_tmpb_tl
3935    \stex_if_smsmode:F {
3936        \end{stex_annotate_env}
3937    }
3938 }
3939
3940 \NewDocumentEnvironment {copymodule} { O{} m m}{
3941    \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ structure }
3942    \stex_deactivate_macro:Nn \symdecl {module~environments}
3943    \stex_deactivate_macro:Nn \symdef {module~environments}
3944    \stex_deactivate_macro:Nn \notation {module~environments}
3945    \stex_reactivate_macro:N \assign
3946    \stex_reactivate_macro:N \renamedecl
3947    \stex_reactivate_macro:N \donotcopy
3948    \stex_smsmode_do:
3949 }{
3950    \stex_copymodule_end:n {}
3951 }
3952
3953 \NewDocumentEnvironment {interpretmodule} { O{} m m}{
3954    \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ realization }
3955    \stex_deactivate_macro:Nn \symdecl {module~environments}
3956    \stex_deactivate_macro:Nn \symdef {module~environments}
3957    \stex_deactivate_macro:Nn \notation {module~environments}
3958    \stex_reactivate_macro:N \assign
3959    \stex_reactivate_macro:N \renamedecl
3960    \stex_reactivate_macro:N \donotcopy
3961    \stex_smsmode_do:
3962 }{
3963    \stex_copymodule_end:n {
3964        \tl_if_exist:cF {
3965            l__stex_copymodule_copymodule_##1?##2_def_tl
3966        }{
3967            \str_if_eq:eeF {
3968                \prop_item:cn{
3969                    l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
3970            }{ true }{
3971                \msg_error:nnxx{stex}{error/interpretmodule/nodefiniens}{
3972                    ##1?##2
3973                }{\l_stex_current_copymodule_name_str}
3974            }
3975        }
3976    }
3977 }
3978
3979 \NewDocumentCommand \donotcopy { O{} m}{
3980    \stex_import_module_uri:nn { #1 } { #2 }
3981    \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3982    \seq_map_inline:Nn \l_stex_collect_imports_seq {
3983        \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
```

185

```
3984     \seq_map_inline:cn {c_stex_module_##1_constants}{
3985       \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? ####1 }
3986       \bool_lazy_any:nT {
3987         { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_name_str}}
3988         { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_macroname_str}}
3989         { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_def_tl}}
3990       }{
3991         % TODO throw error
3992       }
3993     }
3994   }
3995
3996   \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
3997   \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_ns_str ?\l_stex_import_name_str }
3998   \prop_put:Nno \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
3999 }
4000
4001 \NewDocumentCommand \assign { m m }{
4002   \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
4003   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
4004   \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _def_tl}{#2}
4005 }
4006
4007 \keys_define:nn { stex / renamedecl } {
4008   name          .str_set_x:N  = \l_stex_renamedecl_name_str
4009 }
4010 \cs_new_protected:Nn \__stex_copymodule_renamedecl_args:n {
4011   \str_clear:N \l_stex_renamedecl_name_str
4012   \keys_set:nn { stex / renamedecl } { #1 }
4013 }
4014
4015 \NewDocumentCommand \renamedecl { O{} m m}{
4016   \__stex_copymodule_renamedecl_args:n { #1 }
4017   \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
4018   \stex_debug:nn{renamedecl}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
4019   \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
4020   \str_if_empty:NTF \l_stex_renamedecl_name_str {
4021     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4022       \l_stex_get_symbol_uri_str
4023     } }
4024   } {
4025     \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex
4026     \stex_debug:nn{renamedecl}{@~\l_stex_current_module_str ? \l_stex_renamedecl_name_str}
4027     \prop_set_eq:cc {l_stex_symdecl_
4028       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4029       _prop
4030     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
4031     \seq_set_eq:cc {l_stex_symdecl_
4032       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4033       _notations
4034     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
4035     \prop_put:cnx {l_stex_symdecl_
4036       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4037       _prop
```

```
4038        }{ name }{ \l_stex_renamedecl_name_str }
4039        \prop_put:cnx {l_stex_symdecl_
4040          \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4041          _prop
4042        }{ module }{ \l_stex_current_module_str }
4043        \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
4044          \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4045        }
4046        \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4047          \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4048        } }
4049      }
4050    }
4051
4052    \stex_deactivate_macro:Nn \assign {copymodules}
4053    \stex_deactivate_macro:Nn \renamedecl {copymodules}
4054    \stex_deactivate_macro:Nn \donotcopy {copymodules}
4055
4056
4057    \seq_new:N \l_stex_implicit_morphisms_seq
4058    \NewDocumentCommand \implicitmorphism { O{} m m}{
4059      \stex_import_module_uri:nn { #1 } { #2 }
4060      \stex_debug:nn{implicits}{
4061        Implicit~morphism:~
4062        \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4063      }
4064      \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
4065        \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4066      }{
4067        \msg_error:nnn{stex}{error/conflictingmodules}{
4068          \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4069        }
4070      }
4071
4072      % TODO
4073
4074
4075
4076      \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
4077        \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4078      }
4079    }
4080
```

31.2   The feature environment

structural@feature

```
4081   ⟨@@=stex_features⟩
4082
4083   \NewDocumentEnvironment{structural_feature_module}{ m m m }{
4084     \stex_if_in_module:F {
4085       \msg_set:nnn{stex}{error/nomodule}{
4086         Structural~Feature~has~to~occur~in~a~module:\\
```

```
4087        Feature~#2~of~type~#1\\
4088        In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
4089      }
4090      \msg_error:nn{stex}{error/nomodule}
4091    }
4092
4093    \stex_module_setup:nn{meta=NONE}{#2 - #1}
4094
4095    \stex_if_smsmode:F {
4096      \begin{stex_annotate_env}{ feature:#1 }{}
4097        \stex_annotate_invisible:nnn{header}{}{ #3 }
4098    }
4099 }{
4100    \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
4101    \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
4102    \stex_debug:nn{features}{
4103      Feature: \l_stex_last_feature_str
4104    }
4105    \stex_if_smsmode:F {
4106      \end{stex_annotate_env}
4107    }
4108 }
```

31.3   Structure

```
4109 ⟨@@=stex_structures⟩
4110 \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
4111    \prop_if_exist:cF {c_stex_module_\l_stex_current_module_str _structures}{
4112      \prop_new:c {c_stex_module_\l_stex_current_module_str _structures}
4113    }
4114    \prop_gput:cxx{c_stex_module_\l_stex_current_module_str _structures}
4115      {#1}{#2}
4116 }
4117
4118 \keys_define:nn { stex / features / structure } {
4119    name          .str_set_x:N  = \l__stex_structures_name_str ,
4120 }
4121
4122 \cs_new_protected:Nn \__stex_structures_structure_args:n {
4123    \str_clear:N \l__stex_structures_name_str
4124    \keys_set:nn { stex / features / structure } { #1 }
4125 }
4126
4127 \NewDocumentEnvironment{mathstructure}{m O{}}{
4128    \__stex_structures_structure_args:n { #2 }
4129    \str_if_empty:NT \l__stex_structures_name_str {
4130      \str_set:Nx \l__stex_structures_name_str { #1 }
4131    }
4132    \exp_args:Nx \stex_symdecl_do:nn {
4133        name = \l__stex_structures_name_str ,
4134        type = \metacollection ,
4135        def  = {\STEXsymbol{module-type}{
```

188

```
4136        \_stex_term_math_oms:nnnn {
4137          \prop_get:cn {c_stex_module_\l_stex_current_module_str _prop}
4138            { ns } ?
4139            \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4140              { name } / \l__stex_structures_name_str - structure
4141          }{}{0}{}
4142        }}
4143      }{ #1 }
4144    \exp_args:Nnnx
4145    \begin{structural_feature_module}{ structure }
4146      { \l__stex_structures_name_str }{}
4147    \stex_smsmode_do:
4148  }{
4149    \end{structural_feature_module}
4150    \_stex_reset_up_to_module:
4151    \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
4152    \seq_clear:N \l_tmpa_seq
4153    \seq_map_inline:Nn \l_stex_collect_imports_seq {
4154      \seq_map_inline:cn{c_stex_module_##1_constants}{
4155        \seq_put_right:Nn \l_tmpa_seq { ##1 ? ####1 }
4156      }
4157    }
4158    \exp_args:Nnno
4159    \prop_gput:cnn {c_stex_module_ \l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
4160    \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
4161    \stex_add_structure_to_current_module:nn
4162      \l__stex_structures_name_str
4163      \l_stex_last_feature_str
4164    \exp_args:Nx
4165    \stex_add_to_current_module:n {
4166      \tl_set:cn { #1 }{
4167        \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structure
4168      }
4169    }
4170    \exp_args:Nx
4171    \stex_do_up_to_module:n {
4172      \tl_set:cn { #1 }{
4173        \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structure
4174      }
4175    }
4176  }
4177  \seq_put_right:Nx \g_stex_smsmode_allowedenvs_seq { \tl_to_str:n {mathstructure}}
4178
4179  \cs_new:Nn \stex_invoke_structure:nn {
4180    \stex_invoke_symbol:n { #1?#2 }
4181  }
4182
4183  \cs_new_protected:Nn \stex_get_structure:n {
4184    \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4185      \tl_set:Nn \l_tmpa_tl { #1 }
4186      \__stex_structures_get_from_cs:
4187    }{
4188      \cs_if_exist:cTF { #1 }{
4189        \cs_set_eq:Nc \l_tmpa_cs { #1 }
```

```
4190        \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
4191        \str_if_empty:NTF \l_tmpa_str {
4192          \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs} \stex_invoke_structure:nn {
4193            \__stex_structures_get_from_cs:
4194          }{
4195            \__stex_structures_get_from_string:n { #1 }
4196          }
4197        }{
4198          \__stex_structures_get_from_string:n { #1 }
4199        }
4200      }{
4201        \__stex_structures_get_from_string:n { #1 }
4202      }
4203    }
4204  }
4205
4206  \cs_new_protected:Nn \__stex_structures_get_from_cs: {
4207    \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4208      { \tl_tail:N \l_tmpa_tl }
4209    \str_set:Nx \l_tmpa_str {
4210      \exp_after:wN \use_i:nn \l_tmpa_tl
4211    }
4212    \str_set:Nx \l_tmpb_str {
4213      \exp_after:wN \use_ii:nn \l_tmpa_tl
4214    }
4215    \str_set:Nx \l_stex_get_structure_str {
4216      \l_tmpa_str ? \l_tmpb_str
4217    }
4218    \str_set:Nx \l_stex_get_structure_module_str {
4219      \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
4220    }
4221  }
4222
4223  \cs_new_protected:Nn \__stex_structures_get_from_string:n {
4224    \tl_set:Nn \l_tmpa_tl {
4225      \msg_error:nnn{stex}{error/unknownstructure}{#1}
4226    }
4227    \str_set:Nn \l_tmpa_str { #1 }
4228    \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4229
4230    \seq_map_inline:Nn \l_stex_all_modules_seq {
4231      \prop_if_exist:cT {c_stex_module_##1_structures} {
4232        \prop_map_inline:cn {c_stex_module_##1_structures} {
4233          \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
4234            \prop_map_break:n{\seq_map_break:n{
4235              \tl_set:Nn \l_tmpa_tl {
4236                \str_set:Nn \l_stex_get_structure_str {##1?####1}
4237                \str_set:Nn \l_stex_get_structure_module_str {####2}
4238              }
4239            }}
4240          }
4241        }
4242      }
4243    }
```

```
4244    \l_tmpa_tl
4245  }
```

\instantiate

```
4246
4247  \keys_define:nn { stex / instantiate } {
4248    name          .str_set_x:N  = \l__stex_structures_name_str
4249  }
4250  \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
4251    \str_clear:N \l__stex_structures_name_str
4252    \keys_set:nn { stex / instantiate } { #1 }
4253  }
4254
4255  \NewDocumentCommand \instantiate {m O{} m m m}{
4256    \begingroup
4257      \stex_get_structure:n {#4}
4258      \__stex_structures_instantiate_args:n { #2 }
4259      \str_if_empty:NT \l__stex_structures_name_str {
4260        \str_set:Nn \l__stex_structures_name_str { #1 }
4261      }
4262      \seq_clear:N \l__stex_structures_fields_seq
4263      \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4264      \seq_map_inline:Nn \l_stex_collect_imports_seq {
4265        \seq_map_inline:cn {c_stex_module_##1_constants}{
4266          \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4267        }
4268      }
4269      \seq_set_split:Nnn \l_tmpa_seq , {#3}
4270      \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4271      \prop_clear:N \l_tmpa_prop
4272      \seq_map_inline:Nn \l_tmpa_seq {
4273        \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4274        \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4275          \msg_error:nnn{stex}{error/keyval}{##1}
4276        }
4277        \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_structur
4278        \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4279        \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_uri
4280        \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}
4281        \exp_args:Nxx \str_if_eq:nnF
4282          {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4283          {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
4284          \msg_error:nnxxxx{stex}{error/incompatible}
4285            {\l__stex_structures_dom_str}
4286            {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4287            {\l_stex_get_symbol_uri_str}
4288            {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
4289        }
4290        \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str
4291      }
4292      \seq_if_empty:NF \l__stex_structures_fields_seq {
4293        \msg_error:nnx{stex}{error/instantiate/missing}{\seq_use:Nn\l__stex_structures_fields_
4294      }
4295      \exp_args:Nx
```

```
4296       \stex_add_to_current_module:n {
4297         \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
4298           domain = \l_stex_get_structure_module_str ,
4299           \prop_to_keyval:N \l_tmpa_prop
4300         }
4301         \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
4302       }
4303       \exp_args:Nx
4304       \stex_do_up_to_module:n {
4305         \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
4306           domain = \l_stex_get_structure_module_str ,
4307           \prop_to_keyval:N \l_tmpa_prop
4308         }
4309         \tl_set:cn{ #1 }{\stex_invoke_instance:n{\l_stex_current_module_str?\l__stex_structure
4310       }
4311       \stex_debug:nn{instantiate}{
4312         Instance~\l_stex_current_module_str?\l__stex_structures_name_str \\
4313         \prop_to_keyval:N \l_tmpa_prop
4314       }
4315       \exp_args:Nxx \stex_symdecl_do:nn {
4316         type={\STEXsymbol{module-type}{
4317           \_stex_term_math_oms:nnnn {
4318             \l_stex_get_structure_module_str
4319           }{}{0}{}
4320         }}
4321       }{\l__stex_structures_name_str}
4322       \exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}
4323    \endgroup
4324    \stex_smsmode_do:\ignorespacesandpars
4325 }
4326 \tl_put_right:Nx \g_stex_smsmode_allowedmacros_escape_tl {\instantiate}
4327
4328 \cs_new_protected:Nn \stex_symbol_or_var:n {
4329    \cs_if_exist:cTF{#1}{
4330      \cs_set_eq:Nc \l_tmpa_tl { #1 }
4331      \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4332      \str_if_empty:NTF \l_tmpa_str {
4333        \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
4334          \stex_invoke_variable:n {
4335            \bool_set_true:N \l_stex_symbol_or_var_bool
4336            \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
4337            \str_set:Nx \l_stex_get_symbol_uri_str {
4338              \exp_after:wN \use:n \l_tmpa_tl
4339            }
4340          }{
4341            \bool_set_false:N \l_stex_symbol_or_var_bool
4342            \stex_get_symbol:n{#1}
4343          }
4344      }{
4345        \__stex_structures_symbolorvar_from_string:n{ #1 }
4346      }
4347    }{
4348      \__stex_structures_symbolorvar_from_string:n{ #1 }
4349    }
```

```
4350 }
4351
4352 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
4353   \prop_if_exist:cTF {l_stex_variable_#1 _prop}{
4354     \bool_set_true:N \l_stex_symbol_or_var_bool
4355     \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
4356   }{
4357     \bool_set_false:N \l_stex_symbol_or_var_bool
4358     \stex_get_symbol:n{#1}
4359   }
4360 }
4361
4362 \keys_define:nn { stex / varinstantiate } {
4363   name         .str_set_x:N  = \l__stex_structures_name_str,
4364   bind         .choices:nn   =
4365       {forall,exists}
4366       {\str_set:Nx \l__stex_structures_bind_str {\l_keys_choice_tl}}
4367
4368 }
4369 \cs_new_protected:Nn \__stex_structures_varinstantiate_args:n {
4370   \str_clear:N \l__stex_structures_name_str
4371   \str_clear:N \l__stex_structures_bind_str
4372   \keys_set:nn { stex / varinstantiate } { #1 }
4373 }
4374
4375 \NewDocumentCommand \varinstantiate {m O{} m m m}{
4376   \begingroup
4377     \stex_get_structure:n {#4}
4378     \__stex_structures_varinstantiate_args:n { #2 }
4379     \str_if_empty:NT \l__stex_structures_name_str {
4380       \str_set:Nn \l__stex_structures_name_str { #1 }
4381     }
4382     \seq_clear:N \l__stex_structures_fields_seq
4383     \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4384     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4385       \seq_map_inline:cn {c_stex_module_##1_constants}{
4386         \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4387       }
4388     }
4389     \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4390     \prop_clear:N \l_tmpa_prop
4391     \tl_if_empty:nF {#3} {
4392       \seq_set_split:Nnn \l_tmpa_seq , {#3}
4393       \seq_map_inline:Nn \l_tmpa_seq {
4394         \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4395         \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4396           \msg_error:nnn{stex}{error/keyval}{##1}
4397         }
4398         \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
4399         \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4400         \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
4401         \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
4402         \bool_if:NTF \l_stex_symbol_or_var_bool {
4403           \exp_args:Nxx \str_if_eq:nnF
```

193

```
4404            {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4405            {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}{
4406         \msg_error:nnxxxx{stex}{error/incompatible}
4407            {\l__stex_structures_dom_str}
4408            {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4409            {\l_stex_get_symbol_uri_str}
4410            {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}
4411       }
4412       \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n {
4413     }{
4414       \exp_args:Nxx \str_if_eq:nnF
4415         {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4416         {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
4417         \msg_error:nnxxxx{stex}{error/incompatible}
4418            {\l__stex_structures_dom_str}
4419            {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4420            {\l_stex_get_symbol_uri_str}
4421            {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
4422       }
4423       \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n {\l
4424     }
4425   }
4426 }
4427 \tl_gclear:N \g__stex_structures_aftergroup_tl
4428 \seq_map_inline:Nn \l__stex_structures_fields_seq {
4429   \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdecl_
4430   \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4431     \stex_find_notation:nn{##1}{}
4432     \cs_gset_eq:cc{g__stex_structures_tmpa_\l_tmpa_str _cs}
4433        {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4434     \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
4435        \cs_gset_eq:cc {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
4436          {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4437     }
4438   }
4439
4440   \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4441     \prop_set_from_keyval:cn { l_stex_variable_ \l_tmpa_str _prop}{
4442       name   = \l_tmpa_str ,
4443       args   = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4444       arity  = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4445       assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4446     }
4447     \cs_set_eq:cc {stex_var_notation_\l_tmpa_str _cs}
4448        {g__stex_structures_tmpa_\l_tmpa_str _cs}
4449     \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str _cs}
4450        {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
4451   }
4452   \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\stex_invok
4453 }
4454 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4455   \prop_set_from_keyval:cn {l_stex_varinstance_\l__stex_structures_name_str _prop }{
4456     domain = \l_stex_get_structure_module_str ,
4457     \prop_to_keyval:N \l_tmpa_prop
```

194

```
4458        }
4459        \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
4460        \tl_set:cn {l_stex_varinstance_\l__stex_structures_name_str _op_tl}{
4461          \exp_args:Nnx \exp_not:N \use:nn {
4462            \str_set:Nn \exp_not:N \l_stex_current_symbol_str {var://\l__stex_structures_name_
4463            \_stex_term_omv:nn {var://\l__stex_structures_name_str}{
4464              \exp_not:n{
4465                \_varcomp{#5}
4466              }
4467            }
4468          }{
4469            \exp_not:n{\_stex_reset:N \l_stex_current_symbol_str}
4470          }
4471        }
4472      }
4473      \aftergroup\g__stex_structures_aftergroup_tl
4474    \endgroup
4475    \stex_smsmode_do:\ignorespacesandpars
4476 }
4477
4478 \cs_new_protected:Nn \stex_invoke_instance:n {
4479    \peek_charcode_remove:NTF ! {
4480      \stex_invoke_symbol:n{#1}
4481    }{
4482      \_stex_invoke_instance:nn {#1}
4483    }
4484 }
4485
4486
4487 \cs_new_protected:Nn \stex_invoke_varinstance:n {
4488    \peek_charcode_remove:NTF ! {
4489      \exp_args:Nnx \use:nn {
4490        \def\comp{\_varcomp}
4491        \use:c{l_stex_varinstance_#1_op_tl}
4492      }{
4493        \_stex_reset:N \comp
4494      }
4495    }{
4496      \_stex_invoke_varinstance:nn {#1}
4497    }
4498 }
4499
4500 \cs_new_protected:Nn \_stex_invoke_instance:nn {
4501    \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
4502      \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
4503    }{
4504      \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
4505      \msg_error:nnnnn{stex}{error/unknownfield}{#2}{#1}{
4506        \prop_to_keyval:N \l_tmpa_prop
4507      }
4508    }
4509 }
4510
4511 \cs_new_protected:Nn \_stex_invoke_varinstance:nn {
```

195

```
4512    \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
4513      \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
4514      \l_tmpa_tl
4515    }{
4516      \msg_error:nnnnn{stex}{error/unknownfield}{#2}{#1}{}
4517    }
4518  }
```

(*End definition for* `\instantiate`*. This function is documented on page* *31*.)

\stex_invoke_structure:nnn

```
4519  % #1: URI of the instance
4520  % #2: URI of the instantiated module
4521  \cs_new_protected:Nn \stex_invoke_structure:nnn {
4522    \tl_if_empty:nTF{ #3 }{
4523      \prop_set_eq:Nc \l__stex_structures_structure_prop {
4524        c_stex_feature_ #2 _prop
4525      }
4526    }
4527    \tl_clear:N \l_tmpa_tl
4528    \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
4528    \seq_map_inline:Nn \l_tmpa_seq {
4529      \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
4530      \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
4531      \cs_if_exist:cT {
4532        stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
4533      }{
4534        \tl_if_empty:NF \l_tmpa_tl {
4535          \tl_put_right:Nn \l_tmpa_tl {,}
4536        }
4537        \tl_put_right:Nx \l_tmpa_tl {
4538          \stex_invoke_symbol:n {#1/\l_tmpa_str}!
4539        }
4540      }
4541    }
4542    \exp_args:No \mathstruct \l_tmpa_tl
4543  }{
4544    \stex_invoke_symbol:n{#1/#3}
4545  }
4546 }
```

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

```
4547  ⟨/package⟩
```

196

Chapter 32

sTeX
-Statements Implementation

```
4548 ⟨∗package⟩
4549
4550 %%%%%%%%%%%%    features.dtx    %%%%%%%%%%%%
4551
4552 ⟨@@=stex_statements⟩
```

Warnings and error messages

```
4553
```

```
4554 \def\titleemph#1{\textbf{#1}}
```

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

32.1   Definitions

definiendum

```
4555 \keys_define:nn {stex / definiendum }{
4556   pre      .tl_set:N     = \l__stex_statements_definiendum_pre_tl,
4557   post     .tl_set:N     = \l__stex_statements_definiendum_post_tl,
4558   root     .str_set_x:N  = \l__stex_statements_definiendum_root_str,
4559   gfa      .str_set_x:N  = \l__stex_statements_definiendum_gfa_str
4560 }
4561 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
4562   \str_clear:N \l__stex_statements_definiendum_root_str
4563   \tl_clear:N \l__stex_statements_definiendum_post_tl
4564   \str_clear:N \l__stex_statements_definiendum_gfa_str
4565   \keys_set:nn { stex / definiendum }{ #1 }
4566 }
4567 \NewDocumentCommand \definiendum { O{} m m} {
4568   \__stex_statements_definiendum_args:n { #1 }
4569   \stex_get_symbol:n { #2 }
4570   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4571   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
4572     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

197

```
4573        \tl_set:Nn \l_tmpa_tl { #3 }
4574      } {
4575        \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
4576        \tl_set:Nn \l_tmpa_tl {
4577          \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
4578        }
4579      }
4580    } {
4581      \tl_set:Nn \l_tmpa_tl { #3 }
4582    }
4583
4584    % TODO root
4585    \rustex_if:TF {
4586      \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
4587    } {
4588      \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
4589    }
4590 }
4591 \stex_deactivate_macro:Nn \definiendum {definition~environments}
```

(*End definition for* definiendum. *This function is documented on page 40.*)

definame

```
4592
4593 \NewDocumentCommand \definame { O{} m } {
4594    \__stex_statements_definiendum_args:n { #1 }
4595    % TODO: root
4596    \stex_get_symbol:n { #2 }
4597    \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4598    \str_set:Nx \l_tmpa_str {
4599      \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4600    }
4601    \str_replace_all:Nnn \l_tmpa_str {-} {~}
4602    \rustex_if:TF {
4603      \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4604        \l_tmpa_str\l__stex_statements_definiendum_post_tl
4605      }
4606    } {
4607      \exp_args:Nnx \defemph@uri {
4608        \l_tmpa_str\l__stex_statements_definiendum_post_tl
4609      } { \l_stex_get_symbol_uri_str }
4610    }
4611 }
4612 \stex_deactivate_macro:Nn \definame {definition~environments}
4613
4614 \NewDocumentCommand \Definame { O{} m } {
4615    \__stex_statements_definiendum_args:n { #1 }
4616    \stex_get_symbol:n { #2 }
4617    \str_set:Nx \l_tmpa_str {
4618      \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4619    }
4620    \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4621    \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4622    \rustex_if:TF {
```

```
4623      \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4624        \l_tmpa_str\l__stex_statements_definiendum_post_tl
4625      }
4626    } {
4627      \exp_args:Nnx \defemph@uri {
4628        \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
4629      } { \l_stex_get_symbol_uri_str }
4630    }
4631  }
4632  \stex_deactivate_macro:Nn \Definame {definition~environments}
4633
4634  \NewDocumentCommand \premise { m }{
4635    \stex_annotate:nnn{ premise }{}{ #1 }
4636  }
4637  \NewDocumentCommand \conclusion { m }{
4638    \stex_annotate:nnn{ conclusion }{}{ #1 }
4639  }
4640  \NewDocumentCommand \definiens { O{} m }{
4641    \str_clear:N \l_stex_get_symbol_uri_str
4642    \tl_if_empty:nF {#1} {
4643      \stex_get_symbol:n { #1 }
4644    }
4645    \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
4646  }
4647
4648  \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
4649  \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
4650  \stex_deactivate_macro:Nn \definiens {definition~environments}
4651
```

(*End definition for* definame. *This function is documented on page 40.*)

sdefinition

```
4652
4653  \keys_define:nn {stex / sdefinition }{
4654    type     .str_set_x:N  = \sdefinitiontype,
4655    id       .str_set_x:N  = \sdefinitionid,
4656    name     .str_set_x:N  = \sdefinitionname,
4657    for      .clist_set:N   = \l__stex_statements_sdefinition_for_clist ,
4658    title    .tl_set:N      = \sdefinitiontitle
4659  }
4660  \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
4661    \str_clear:N \sdefinitiontype
4662    \str_clear:N \sdefinitionid
4663    \str_clear:N \sdefinitionname
4664    \clist_clear:N \l__stex_statements_sdefinition_for_clist
4665    \tl_clear:N \sdefinitiontitle
4666    \keys_set:nn { stex / sdefinition }{ #1 }
4667  }
4668
4669  \NewDocumentEnvironment{sdefinition}{O{}}{
4670    \__stex_statements_sdefinition_args:n{ #1 }
4671    \stex_reactivate_macro:N \definiendum
4672    \stex_reactivate_macro:N \definame
```

199

```
4673    \stex_reactivate_macro:N \Definame
4674    \stex_reactivate_macro:N \premise
4675    \stex_reactivate_macro:N \definiens
4676    \stex_if_smsmode:F{
4677      \seq_clear:N \l_tmpa_seq
4678      \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4679        \tl_if_empty:nF{ ##1 }{
4680          \stex_get_symbol:n { ##1 }
4681          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4682            \l_stex_get_symbol_uri_str
4683          }
4684        }
4685      }
4686      \exp_args:Nnnx
4687      \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {,}}
4688      \str_if_empty:NF \sdefinitiontype {
4689        \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{}
4690      }
4691      \clist_set:No \l_tmpa_clist \sdefinitiontype
4692      \tl_clear:N \l_tmpa_tl
4693      \clist_map_inline:Nn \l_tmpa_clist {
4694        \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
4695          \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
4696        }
4697      }
4698      \tl_if_empty:NTF \l_tmpa_tl {
4699        \__stex_statements_sdefinition_start:
4700      }{
4701        \l_tmpa_tl
4702      }
4703    }
4704    \stex_ref_new_doc_target:n \sdefinitionid
4705    \stex_smsmode_do:
4706  }{
4707    \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
4708    \stex_if_smsmode:F {
4709      \clist_set:No \l_tmpa_clist \sdefinitiontype
4710      \tl_clear:N \l_tmpa_tl
4711      \clist_map_inline:Nn \l_tmpa_clist {
4712        \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
4713          \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
4714        }
4715      }
4716      \tl_if_empty:NTF \l_tmpa_tl {
4717        \__stex_statements_sdefinition_end:
4718      }{
4719        \l_tmpa_tl
4720      }
4721      \end{stex_annotate_env}
4722    }
4723  }
```

\stexpatchdefinition

```
4724 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
```

```
4725    \par\noindent\titleemph{Definition\tl_if_empty:NF \sdefinitiontitle {
4726      ~(\sdefinitiontitle)
4727    }~}
4728 }
4729 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\par\medskip}
4730
4731 \newcommand\stexpatchdefinition[3][] {
4732    \str_set:Nx \l_tmpa_str{ #1 }
4733    \str_if_empty:NTF \l_tmpa_str {
4734      \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
4735      \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
4736    }{
4737      \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2
4738      \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
4739    }
4740 }
```

(End definition for \stexpatchdefinition. This function is documented on page 42.)

\inlinedef   inline:

```
4741 \keys_define:nn {stex / inlinedef }{
4742    type     .str_set_x:N  = \sdefinitiontype,
4743    id       .str_set_x:N  = \sdefinitionid,
4744    for      .clist_set:N   = \l__stex_statements_sdefinition_for_clist ,
4745    name     .str_set_x:N  = \sdefinitionname
4746 }
4747 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
4748    \str_clear:N \sdefinitiontype
4749    \str_clear:N \sdefinitionid
4750    \str_clear:N \sdefinitionname
4751    \clist_clear:N \l__stex_statements_sdefinition_for_clist
4752    \keys_set:nn { stex / inlinedef }{ #1 }
4753 }
4754 \NewDocumentCommand \inlinedef { O{} m } {
4755    \begingroup
4756    \__stex_statements_inlinedef_args:n{ #1 }
4757    \stex_reactivate_macro:N \definiendum
4758    \stex_reactivate_macro:N \definame
4759    \stex_reactivate_macro:N \Definame
4760    \stex_reactivate_macro:N \premise
4761    \stex_reactivate_macro:N \definiens
4762    \stex_ref_new_doc_target:n \sdefinitionid
4763    \stex_if_smsmode:TF{
4764      \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
4765    }{
4766      \seq_clear:N \l_tmpa_seq
4767      \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4768        \tl_if_empty:nF{ ##1 }{
4769          \stex_get_symbol:n { ##1 }
4770          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4771            \l_stex_get_symbol_uri_str
4772          }
4773        }
4774      }
```

```
4775        \exp_args:Nnx
4776        \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
4777          \str_if_empty:NF \sdefinitiontype {
4778            \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{}
4779          }
4780          #2
4781          \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
4782        }
4783      }
4784      \endgroup
4785      \stex_smsmode_do:
4786    }
```

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

32.2   Assertions

```
4787
4788  \keys_define:nn {stex / sassertion }{
4789    type     .str_set_x:N  = \sassertiontype,
4790    id       .str_set_x:N  = \sassertionid,
4791    title    .tl_set:N     = \sassertiontitle ,
4792    for      .clist_set:N  = \l__stex_statements_sassertion_for_clist ,
4793    name     .str_set_x:N  = \sassertionname
4794  }
4795  \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
4796    \str_clear:N \sassertiontype
4797    \str_clear:N \sassertionid
4798    \str_clear:N \sassertionname
4799    \clist_clear:N \l__stex_statements_sassertion_for_clist
4800    \tl_clear:N \sassertiontitle
4801    \keys_set:nn { stex / sassertion }{ #1 }
4802  }
4803
4804  %\tl_new:N \g__stex_statements_aftergroup_tl
4805
4806  \NewDocumentEnvironment{sassertion}{O{}}{
4807    \__stex_statements_sassertion_args:n{ #1 }
4808    \stex_reactivate_macro:N \premise
4809    \stex_reactivate_macro:N \conclusion
4810    \stex_if_smsmode:F {
4811      \seq_clear:N \l_tmpa_seq
4812      \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4813        \tl_if_empty:nF{ ##1 }{
4814          \stex_get_symbol:n { ##1 }
4815          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4816            \l_stex_get_symbol_uri_str
4817          }
4818        }
4819      }
4820      \exp_args:Nnnx
4821      \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}
```

```
4822        \str_if_empty:NF \sassertiontype {
4823          \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
4824        }
4825        \clist_set:No \l_tmpa_clist \sassertiontype
4826        \tl_clear:N \l_tmpa_tl
4827        \clist_map_inline:Nn \l_tmpa_clist {
4828          \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
4829            \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
4830          }
4831        }
4832        \tl_if_empty:NTF \l_tmpa_tl {
4833          \__stex_statements_sassertion_start:
4834        }{
4835          \l_tmpa_tl
4836        }
4837      }
4838      \str_if_empty:NTF \sassertionid {
4839        \str_if_empty:NF \sassertionname {
4840          \stex_ref_new_doc_target:n {}
4841        }
4842      } {
4843        \stex_ref_new_doc_target:n \sassertionid
4844      }
4845      \stex_smsmode_do:
4846    }{
4847      \str_if_empty:NF \sassertionname {
4848        \stex_symdecl_do:nn{}{\sassertionname}
4849        \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4850      }
4851      \stex_if_smsmode:F {
4852        \clist_set:No \l_tmpa_clist \sassertiontype
4853        \tl_clear:N \l_tmpa_tl
4854        \clist_map_inline:Nn \l_tmpa_clist {
4855          \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
4856            \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
4857          }
4858        }
4859        \tl_if_empty:NTF \l_tmpa_tl {
4860          \__stex_statements_sassertion_end:
4861        }{
4862          \l_tmpa_tl
4863        }
4864        \end{stex_annotate_env}
4865      }
4866    }
```

```
4867
4868 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
4869   \par\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {
4870     (\sassertiontitle)
4871   }~}
4872 }
4873 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
```

```
4874
4875 \newcommand\stexpatchassertion[3][] {
4876     \str_set:Nx \l_tmpa_str{ #1 }
4877     \str_if_empty:NTF \l_tmpa_str {
4878         \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
4879         \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
4880     }{
4881         \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
4882         \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3
4883     }
4884 }
```

(*End definition for* \stexpatchassertion. *This function is documented on page 42.*)

\inlineass   inline:

```
4885 \keys_define:nn {stex / inlineass }{
4886   type     .str_set_x:N  = \sassertiontype,
4887   id       .str_set_x:N  = \sassertionid,
4888   for      .clist_set:N  = \l__stex_statements_sassertion_for_clist ,
4889   name     .str_set_x:N  = \sassertionname
4890 }
4891 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
4892   \str_clear:N \sassertiontype
4893   \str_clear:N \sassertionid
4894   \str_clear:N \sassertionname
4895   \clist_clear:N \l__stex_statements_sassertion_for_clist
4896   \keys_set:nn { stex / inlineass }{ #1 }
4897 }
4898 \NewDocumentCommand \inlineass { O{} m } {
4899   \begingroup
4900   \stex_reactivate_macro:N \premise
4901   \stex_reactivate_macro:N \conclusion
4902   \__stex_statements_inlineass_args:n{ #1 }
4903   \str_if_empty:NTF \sassertionid {
4904     \str_if_empty:NF \sassertionname {
4905       \stex_ref_new_doc_target:n {}
4906     }
4907   } {
4908     \stex_ref_new_doc_target:n \sassertionid
4909   }
4910
4911   \stex_if_smsmode:TF{
4912     \str_if_empty:NF \sassertionname {
4913       \stex_symdecl_do:nn{}{\sassertionname}
4914       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4915     }
4916   }{
4917     \seq_clear:N \l_tmpa_seq
4918     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4919       \tl_if_empty:nF{ ##1 }{
4920         \stex_get_symbol:n { ##1 }
4921         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4922           \l_stex_get_symbol_uri_str
4923         }
```

```
4924          }
4925        }
4926      \exp_args:Nnx
4927      \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{
4928        \str_if_empty:NF \sassertiontype {
4929          \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
4930        }
4931        #2
4932        \str_if_empty:NF \sassertionname {
4933          \stex_symdecl_do:nn{}{\sassertionname}
4934          \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4935        }
4936      }
4937    }
4938    \endgroup
4939    \stex_smsmode_do:
4940 }
```

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

32.3   Examples

sexample

```
4941
4942 \keys_define:nn {stex / sexample }{
4943   type    .str_set_x:N  = \exampletype,
4944   id      .str_set_x:N  = \sexampleid,
4945   title   .tl_set:N     = \sexampletitle,
4946   name    .str_set_x:N  = \sexamplename ,
4947   for     .clist_set:N  = \l__stex_statements_sexample_for_clist,
4948 }
4949 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
4950   \str_clear:N \sexampletype
4951   \str_clear:N \sexampleid
4952   \str_clear:N \sexamplename
4953   \tl_clear:N \sexampletitle
4954   \clist_clear:N \l__stex_statements_sexample_for_clist
4955   \keys_set:nn { stex / sexample }{ #1 }
4956 }
4957
4958 \NewDocumentEnvironment{sexample}{O{}}{
4959   \__stex_statements_sexample_args:n{ #1 }
4960   \stex_reactivate_macro:N \premise
4961   \stex_reactivate_macro:N \conclusion
4962   \stex_if_smsmode:F {
4963     \seq_clear:N \l_tmpa_seq
4964     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4965       \tl_if_empty:nF{ ##1 }{
4966         \stex_get_symbol:n { ##1 }
4967         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4968           \l_stex_get_symbol_uri_str
4969         }
4970       }
```

```
4971          }
4972          \exp_args:Nnnx
4973          \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
4974          \str_if_empty:NF \sexampletype {
4975            \stex_annotate_invisible:nnn{type}{\sexampletype}{}
4976          }
4977          \clist_set:No \l_tmpa_clist \sexampletype
4978          \tl_clear:N \l_tmpa_tl
4979          \clist_map_inline:Nn \l_tmpa_clist {
4980            \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
4981              \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
4982            }
4983          }
4984          \tl_if_empty:NTF \l_tmpa_tl {
4985            \__stex_statements_sexample_start:
4986          }{
4987            \l_tmpa_tl
4988          }
4989        }
4990        \str_if_empty:NF \sexampleid {
4991          \stex_ref_new_doc_target:n \sexampleid
4992        }
4993        \stex_smsmode_do:
4994      }{
4995        \str_if_empty:NF \sexamplename { \stex_symdecl_do:nn{}{\sexamplename} }
4996        \stex_if_smsmode:F {
4997          \clist_set:No \l_tmpa_clist \sexampletype
4998          \tl_clear:N \l_tmpa_tl
4999          \clist_map_inline:Nn \l_tmpa_clist {
5000            \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
5001              \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
5002            }
5003          }
5004          \tl_if_empty:NTF \l_tmpa_tl {
5005            \__stex_statements_sexample_end:
5006          }{
5007            \l_tmpa_tl
5008          }
5009          \end{stex_annotate_env}
5010        }
5011 }
```

\stexpatchexample

```
5012
5013 \cs_new_protected:Nn \__stex_statements_sexample_start: {
5014   \par\noindent\titleemph{Example~\tl_if_empty:NF \sexampletitle {
5015     (\sexampletitle)
5016   }~}
5017 }
5018 \cs_new_protected:Nn \__stex_statements_sexample_end: {\par\medskip}
5019
5020 \newcommand\stexpatchexample[3][] {
5021     \str_set:Nx \l_tmpa_str{ #1 }
5022     \str_if_empty:NTF \l_tmpa_str {
```

206

```
5023        \tl_set:Nn \__stex_statements_sexample_start: { #2 }
5024        \tl_set:Nn \__stex_statements_sexample_end: { #3 }
5025      }{
5026        \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
5027        \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
5028      }
5029 }
```

(*End definition for* \stexpatchexample. *This function is documented on page 42.*)

\inlineex   inline:

```
5030 \keys_define:nn {stex / inlineex }{
5031   type    .str_set_x:N  = \sexampletype,
5032   id      .str_set_x:N  = \sexampleid,
5033   for     .clist_set:N   = \l__stex_statements_sexample_for_clist ,
5034   name    .str_set_x:N  = \sexamplename
5035 }
5036 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
5037   \str_clear:N \sexampletype
5038   \str_clear:N \sexampleid
5039   \str_clear:N \sexamplename
5040   \clist_clear:N \l__stex_statements_sexample_for_clist
5041   \keys_set:nn { stex / inlineex }{ #1 }
5042 }
5043 \NewDocumentCommand \inlineex { O{} m } {
5044   \begingroup
5045   \stex_reactivate_macro:N \premise
5046   \stex_reactivate_macro:N \conclusion
5047   \__stex_statements_inlineex_args:n{ #1 }
5048   \str_if_empty:NF \sexampleid {
5049     \stex_ref_new_doc_target:n \sexampleid
5050   }
5051   \stex_if_smsmode:TF{
5052     \str_if_empty:NF \sexamplename { \stex_symdecl_do:nn{}{\examplename} }
5053   }{
5054     \seq_clear:N \l_tmpa_seq
5055     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5056       \tl_if_empty:nF{ ##1 }{
5057         \stex_get_symbol:n { ##1 }
5058         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5059           \l_stex_get_symbol_uri_str
5060         }
5061       }
5062     }
5063     \exp_args:Nnx
5064     \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpa_seq {,}}{
5065       \str_if_empty:NF \sexampletype {
5066         \stex_annotate_invisible:nnn{type}{\sexampletype}{}
5067       }
5068       #2
5069       \str_if_empty:NF \sexamplename { \stex_symdecl_do:nn{}{\sexamplename} }
5070     }
5071   }
5072   \endgroup
```

```
5073     \stex_smsmode_do:
5074 }
```

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

32.4   Logical Paragraphs

sparagraph

```
5075 \keys_define:nn { stex / sparagraph} {
5076   id       .str_set_x:N   = \sparagraphid ,
5077   title    .tl_set:N       = \l_stex_sparagraph_title_tl ,
5078   type     .str_set_x:N   = \sparagraphtype ,
5079   for      .clist_set:N   = \l__stex_statements_sparagraph_for_clist ,
5080   from     .tl_set:N       = \sparagraphfrom ,
5081   to       .tl_set:N       = \sparagraphto ,
5082   start    .tl_set:N       = \l_stex_sparagraph_start_tl ,
5083   name     .str_set:N       = \sparagraphname
5084 }
5085
5086 \cs_new_protected:Nn \stex_sparagraph_args:n {
5087   \tl_clear:N \l_stex_sparagraph_title_tl
5088   \tl_clear:N \sparagraphfrom
5089   \tl_clear:N \sparagraphto
5090   \tl_clear:N \l_stex_sparagraph_start_tl
5091   \str_clear:N \sparagraphid
5092   \str_clear:N \sparagraphtype
5093   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5094   \str_clear:N \sparagraphname
5095   \keys_set:nn { stex / sparagraph }{ #1 }
5096 }
5097 \newif\if@in@omtext\@in@omtextfalse
5098
5099 \NewDocumentEnvironment {sparagraph} { O{} } {
5100   \stex_sparagraph_args:n { #1 }
5101   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5102     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
5103   }{
5104     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
5105   }
5106   \@in@omtexttrue
5107   \stex_if_smsmode:F {
5108     \seq_clear:N \l_tmpa_seq
5109     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5110       \tl_if_empty:nF{ ##1 }{
5111         \stex_get_symbol:n { ##1 }
5112         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5113           \l_stex_get_symbol_uri_str
5114         }
5115       }
5116     }
5117     \exp_args:Nnnx
5118     \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
5119     \str_if_empty:NF \sparagraphtype {
```

```
5120        \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}
5121      }
5122      \str_if_empty:NF \sparagraphfrom {
5123        \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5124      }
5125      \str_if_empty:NF \sparagraphto {
5126        \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5127      }
5128      \clist_set:No \l_tmpa_clist \sparagraphtype
5129      \tl_clear:N \l_tmpa_tl
5130      \clist_map_inline:Nn \sparagraphtype {
5131        \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
5132          \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
5133        }
5134      }
5135      \tl_if_empty:NTF \l_tmpa_tl {
5136        \__stex_statements_sparagraph_start:
5137      }{
5138        \l_tmpa_tl
5139      }
5140    }
5141    \clist_set:No \l_tmpa_clist \sparagraphtype
5142    \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}
5143    {
5144      \stex_reactivate_macro:N \definiendum
5145      \stex_reactivate_macro:N \definame
5146      \stex_reactivate_macro:N \Definame
5147      \stex_reactivate_macro:N \premise
5148      \stex_reactivate_macro:N \definiens
5149    }
5150    \str_if_empty:NTF \sparagraphid {
5151      \str_if_empty:NTF \sparagraphname {
5152        \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5153          \stex_ref_new_doc_target:n {}
5154        }
5155      } {
5156        \stex_ref_new_doc_target:n {}
5157      }
5158    } {
5159      \stex_ref_new_doc_target:n \sparagraphid
5160    }
5161    \exp_args:NNx
5162    \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5163      \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5164        \tl_if_empty:nF{ ##1 }{
5165          \stex_get_symbol:n { ##1 }
5166          \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5167        }
5168      }
5169    }
5170    \stex_smsmode_do:
5171    \ignorespacesandpars
5172  }{
5173    \str_if_empty:NF \sparagraphname {
```

```
5174        \stex_symdecl_do:nn{}{\sparagraphname}
5175        \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5176      }
5177      \stex_if_smsmode:F {
5178        \clist_set:No \l_tmpa_clist \sparagraphtype
5179        \tl_clear:N \l_tmpa_tl
5180        \clist_map_inline:Nn \l_tmpa_clist {
5181          \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
5182            \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
5183          }
5184        }
5185        \tl_if_empty:NTF \l_tmpa_tl {
5186          \__stex_statements_sparagraph_end:
5187        }{
5188          \l_tmpa_tl
5189        }
5190        \end{stex_annotate_env}
5191      }
5192    }
```

\stexpatchparagraph

```
5193
5194  \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
5195    \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5196      \tl_if_empty:NF \l_stex_sparagraph_title_tl {
5197        \titleemph{\l_stex_sparagraph_title_tl}:~
5198      }
5199    }{
5200      \titleemph{\l_stex_sparagraph_start_tl}~
5201    }
5202  }
5203  \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
5204
5205  \newcommand\stexpatchparagraph[3][] {
5206      \str_set:Nx \l_tmpa_str{ #1 }
5207      \str_if_empty:NTF \l_tmpa_str {
5208        \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
5209        \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
5210      }{
5211        \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
5212        \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3
5213      }
5214  }
5215
5216  \keys_define:nn { stex / inlinepara} {
5217    id      .str_set_x:N  = \sparagraphid ,
5218    type    .str_set_x:N  = \sparagraphtype ,
5219    for     .clist_set:N  = \l__stex_statements_sparagraph_for_clist ,
5220    from    .tl_set:N     = \sparagraphfrom ,
5221    to      .tl_set:N     = \sparagraphto ,
5222    name    .str_set:N    = \sparagraphname
5223  }
5224  \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
5225    \tl_clear:N \sparagraphfrom
```

```
5226    \tl_clear:N \sparagraphto
5227    \str_clear:N \sparagraphid
5228    \str_clear:N \sparagraphtype
5229    \clist_clear:N \l__stex_statements_sparagraph_for_clist
5230    \str_clear:N \sparagraphname
5231    \keys_set:nn { stex / inlinepara }{ #1 }
5232 }
5233 \NewDocumentCommand \inlinepara { O{} m } {
5234    \begingroup
5235    \__stex_statements_inlinepara_args:n{ #1 }
5236    \clist_set:No \l_tmpa_clist \sparagraphtype
5237    \str_if_empty:NTF \sparagraphid {
5238       \str_if_empty:NTF \sparagraphname {
5239          \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5240             \stex_ref_new_doc_target:n {}
5241          }
5242       } {
5243          \stex_ref_new_doc_target:n {}
5244       }
5245    } {
5246       \stex_ref_new_doc_target:n \sparagraphid
5247    }
5248    \stex_if_smsmode:TF{
5249       \str_if_empty:NF \sparagraphname {
5250          \stex_symdecl_do:nn{}{\sparagraphname}
5251          \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5252       }
5253    }{
5254       \seq_clear:N \l_tmpa_seq
5255       \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5256          \tl_if_empty:nF{ ##1 }{
5257             \stex_get_symbol:n { ##1 }
5258             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5259                \l_stex_get_symbol_uri_str
5260             }
5261          }
5262       }
5263       \exp_args:Nnx
5264       \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}{
5265          \str_if_empty:NF \sparagraphtype {
5266             \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}
5267          }
5268          \str_if_empty:NF \sparagraphfrom {
5269             \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5270          }
5271          \str_if_empty:NF \sparagraphto {
5272             \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5273          }
5274          \str_if_empty:NF \sparagraphname {
5275             \stex_symdecl_do:nn{}{\sparagraphname}
5276             \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5277          }
5278          \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5279             \clist_map_inline:Nn \l_tmpa_seq {
```

```
5280            \stex_ref_new_sym_target:n {##1}
5281          }
5282        }
5283        #2
5284      }
5285    }
5286    \endgroup
5287    \stex_smsmode_do:
5288 }
5289
```

(*End definition for* `\stexpatchparagraph`. *This function is documented on page* *42*.)

```
5290 ⟨/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]

```
5291 ⟨∗package⟩
5292 ⟨@@=stex_sproof⟩
5293
5294 %%%%%%%%%%%%   sproof.dtx   %%%%%%%%%%%%
5295
```

33.2 Proofs

We first define some keys for the proof environment.

```
5296 \keys_define:nn { stex / spf } {
5297   id          .str_set_x:N  = \spfid,
5298   for         .clist_set:N  = \l__stex_sproof_spf_for_clist ,
5299   from        .tl_set:N     = \l__stex_sproof_spf_from_tl ,
5300   proofend    .tl_set:N     = \l__stex_sproof_spf_proofend_tl,
5301   type        .str_set_x:N  = \spftype,
5302   title       .tl_set:N     = \spftitle,
5303   continues   .tl_set:N     = \l__stex_sproof_spf_continues_tl,
5304   functions   .tl_set:N     = \l__stex_sproof_spf_functions_tl,
5305   method      .tl_set:N     = \l__stex_sproof_spf_method_tl
5306 }
5307 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
5308 \str_clear:N \spfid
5309 \tl_clear:N \l__stex_sproof_spf_for_tl
5310 \tl_clear:N \l__stex_sproof_spf_from_tl
5311 \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
5312 \str_clear:N \spftype
5313 \tl_clear:N \spftitle
5314 \tl_clear:N \l__stex_sproof_spf_continues_tl
5315 \tl_clear:N \l__stex_sproof_spf_functions_tl
```

---

[8]EDNOTE: need an implementation for LaTeXML

```
5316 \tl_clear:N \l__stex_sproof_spf_method_tl
5317   \bool_set_false:N \l__stex_sproof_inc_counter_bool
5318 \keys_set:nn { stex / spf }{ #1 }
5319 }
```

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

```
5320 \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.

```
5321 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
5322 \cs_new_protected:Npn \sproofnumber {
5323   \int_set:Nn \l_tmpa_int {1}
5324   \bool_while_do:nn {
5325     \int_compare_p:nNn {
5326       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5327     } > 0
5328   }{
5329     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
5330     \int_incr:N \l_tmpa_int
5331   }
5332 }
5333 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
5334   \int_set:Nn \l_tmpa_int {1}
5335   \bool_while_do:nn {
5336     \int_compare_p:nNn {
5337       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5338     } > 0
5339   }{
5340     \int_incr:N \l_tmpa_int
5341   }
5342   \int_compare:nNnF \l_tmpa_int = 1 {
5343     \int_decr:N \l_tmpa_int
5344   }
5345   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
5346     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1
```

---

[7]This gets the labeling right but only works 8 levels deep

```
5347        }
5348    }
5349
5350    \cs_new_protected:Npn \__stex_sproof_add_counter: {
5351      \int_set:Nn \l_tmpa_int {1}
5352      \bool_while_do:nn {
5353        \int_compare_p:nNn {
5354          \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5355        } > 0
5356      }{
5357        \int_incr:N \l_tmpa_int
5358      }
5359      \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 1 }
5360    }
5361
5362    \cs_new_protected:Npn \__stex_sproof_remove_counter: {
5363      \int_set:Nn \l_tmpa_int {1}
5364      \bool_while_do:nn {
5365        \int_compare_p:nNn {
5366          \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5367        } > 0
5368      }{
5369        \int_incr:N \l_tmpa_int
5370      }
5371      \int_decr:N \l_tmpa_int
5372      \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 0 }
5373    }
```

\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

```
5374    \def\sproof@box{
5375      \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
5376    }
5377    \def\sproofend{
5378      \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
5379        \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
5380      }
5381    }
```

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

spf@*@kw

```
5382    \def\spf@proofsketch@kw{Proof~Sketch}
5383    \def\spf@proof@kw{Proof}
5384    \def\spf@step@kw{Step}
```

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

For the other languages, we set up triggers

```
5385    \AddToHook{begindocument}{
5386      \ltx@ifpackageloaded{babel}{
5387        \makeatletter
5388        \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5389        \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5390          \input{sproof-ngerman.ldf}
```

```
5391        }
5392        \clist_if_in:NnT \l_tmpa_clist {finnish}{
5393          \input{sproof-finnish.ldf}
5394        }
5395        \clist_if_in:NnT \l_tmpa_clist {french}{
5396          \input{sproof-french.ldf}
5397        }
5398        \clist_if_in:NnT \l_tmpa_clist {russian}{
5399          \input{sproof-russian.ldf}
5400        }
5401        \makeatother
5402     }{}
5403 }
```

spfsketch

```
5404 \newcommand\spfsketch[2][]{
5405    \begingroup
5406    \let \premise \stex_proof_premise:
5407    \__stex_sproof_spf_args:n{#1}
5408    \stex_if_smsmode:TF {
5409      \str_if_empty:NF \spfid {
5410        \stex_ref_new_doc_target:n \spfid
5411      }
5412    }{
5413      \seq_clear:N \l_tmpa_seq
5414      \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5415        \tl_if_empty:nF{ ##1 }{
5416          \stex_get_symbol:n { ##1 }
5417          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5418            \l_stex_get_symbol_uri_str
5419          }
5420        }
5421      }
5422      \exp_args:Nnx
5423      \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
5424        \str_if_empty:NF \spftype {
5425          \stex_annotate_invisible:nnn{type}{\spftype}{}
5426        }
5427        \clist_set:No \l_tmpa_clist \spftype
5428        \tl_set:Nn \l_tmpa_tl {
5429          \titleemph{
5430            \tl_if_empty:NTF \spftitle {
5431              \spf@proofsketch@kw
5432            }{
5433              \spftitle
5434            }
5435          }:~
5436        }
5437        \clist_map_inline:Nn \l_tmpa_clist {
5438          \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5439            \tl_clear:N \l_tmpa_tl
5440          }
5441        }
5442        \str_if_empty:NF \spfid {
```

```
5443          \stex_ref_new_doc_target:n \spfid
5444        }
5445      \l_tmpa_tl #2 \sproofend
5446    }
5447  }
5448  \endgroup
5449  \stex_smsmode_do:
5450 }
5451
```

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

spfeq This is very similar to \spfsketch, but uses a computation array[9][10]

```
5452 \newenvironment{spfeq}[2][]{
5453   \__stex_sproof_spf_args:n{#1}
5454   \let \premise \stex_proof_premise:
5455   \stex_if_smsmode:TF {
5456     \str_if_empty:NF \spfid {
5457       \stex_ref_new_doc_target:n \spfid
5458     }
5459 }{
5460     \seq_clear:N \l_tmpa_seq
5461     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5462       \tl_if_empty:nF{ ##1 }{
5463         \stex_get_symbol:n { ##1 }
5464         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5465           \l_stex_get_symbol_uri_str
5466         }
5467       }
5468     }
5469     \exp_args:Nnnx
5470     \begin{stex_annotate_env}{spfeq}{\seq_use:Nn \l_tmpa_seq {,}}
5471     \str_if_empty:NF \spftype {
5472       \stex_annotate_invisible:nnn{type}{\spftype}{}
5473     }
5474
5475     \clist_set:No \l_tmpa_clist \spftype
5476     \tl_clear:N \l_tmpa_tl
5477     \clist_map_inline:Nn \l_tmpa_clist {
5478       \tl_if_exist:cT {__stex_sproof_spfeq_##1_start:}{
5479         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_start:}}
5480       }
5481       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5482         \tl_set:Nn \l_tmpa_tl {\use:n{}}
5483       }
5484     }
5485     \tl_if_empty:NTF \l_tmpa_tl {
5486       \__stex_sproof_spfeq_start:
5487     }{
5488       \l_tmpa_tl
5489     }{~#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

217

```
5490        \str_if_empty:NF \spfid {
5491          \stex_ref_new_doc_target:n \spfid
5492        }
5493        \begin{displaymath}\begin{array}{rcll}
5494      }
5495      \stex_smsmode_do:
5496    }{
5497      \stex_if_smsmode:F {
5498        \end{array}\end{displaymath}
5499        \clist_set:No \l_tmpa_clist \spftype
5500        \tl_clear:N \l_tmpa_tl
5501        \clist_map_inline:Nn \l_tmpa_clist {
5502          \tl_if_exist:cT {__stex_sproof_spfeq_##1_end:}{
5503            \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_end:}}
5504          }
5505        }
5506        \tl_if_empty:NTF \l_tmpa_tl {
5507          \__stex_sproof_spfeq_end:
5508        }{
5509          \l_tmpa_tl
5510        }
5511        \end{stex_annotate_env}
5512      }
5513    }
5514
5515    \cs_new_protected:Nn \__stex_sproof_spfeq_start: {
5516      \titleemph{
5517        \tl_if_empty:NTF \spftitle {
5518          \spf@proof@kw
5519        }{
5520          \spftitle
5521        }
5522      }:
5523    }
5524    \cs_new_protected:Nn \__stex_sproof_spfeq_end: {\sproofend}
5525
5526    \newcommand\stexpatchspfeq[3][] {
5527        \str_set:Nx \l_tmpa_str{ #1 }
5528        \str_if_empty:NTF \l_tmpa_str {
5529          \tl_set:Nn \__stex_sproof_spfeq_start: { #2 }
5530          \tl_set:Nn \__stex_sproof_spfeq_end: { #3 }
5531        }{
5532          \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_start:\endcsname{ #2 }
5533          \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_end:\endcsname{ #3 }
5534        }
5535    }
5536
```

(*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.

```
5537 \newenvironment{sproof}[2][]{
```

```
5538    \let \premise \stex_proof_premise:
5539    \intarray_gzero:N \l__stex_sproof_counter_intarray
5540    \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
5541    \__stex_sproof_spf_args:n{#1}
5542    \stex_if_smsmode:TF {
5543      \str_if_empty:NF \spfid {
5544        \stex_ref_new_doc_target:n \spfid
5545      }
5546    }{
5547      \seq_clear:N \l_tmpa_seq
5548      \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5549        \tl_if_empty:nF{ ##1 }{
5550          \stex_get_symbol:n { ##1 }
5551          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5552            \l_stex_get_symbol_uri_str
5553          }
5554        }
5555      }
5556      \exp_args:Nnnx
5557      \begin{stex_annotate_env}{sproof}{\seq_use:Nn \l_tmpa_seq {,}}
5558      \str_if_empty:NF \spftype {
5559        \stex_annotate_invisible:nnn{type}{\spftype}{}
5560      }
5561
5562      \clist_set:No \l_tmpa_clist \spftype
5563      \tl_clear:N \l_tmpa_tl
5564      \clist_map_inline:Nn \l_tmpa_clist {
5565        \tl_if_exist:cT {__stex_sproof_sproof_##1_start:}{
5566          \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_start:}}
5567        }
5568        \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5569          \tl_set:Nn \l_tmpa_tl {\use:n{}}
5570        }
5571      }
5572      \tl_if_empty:NTF \l_tmpa_tl {
5573        \__stex_sproof_sproof_start:
5574      }{
5575        \l_tmpa_tl
5576      }{~#2}
5577      \str_if_empty:NF \spfid {
5578        \stex_ref_new_doc_target:n \spfid
5579      }
5580      \begin{description}
5581    }
5582    \stex_smsmode_do:
5583  }{
5584    \stex_if_smsmode:F{
5585      \end{description}
5586      \clist_set:No \l_tmpa_clist \spftype
5587      \tl_clear:N \l_tmpa_tl
5588      \clist_map_inline:Nn \l_tmpa_clist {
5589        \tl_if_exist:cT {__stex_sproof_sproof_##1_end:}{
5590          \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_end:}}
5591        }
```

```
5592        }
5593      \tl_if_empty:NTF \l_tmpa_tl {
5594        \__stex_sproof_sproof_end:
5595      }{
5596        \l_tmpa_tl
5597      }
5598      \end{stex_annotate_env}
5599    }
5600  }
5601
5602  \cs_new_protected:Nn \__stex_sproof_sproof_start: {
5603    \par\noindent\titleemph{
5604      \tl_if_empty:NTF \spftype {
5605        \spf@proof@kw
5606      }{
5607        \spftype
5608      }
5609    }:
5610  }
5611  \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
5612
5613  \newcommand\stexpatchproof[3][] {
5614    \str_set:Nx \l_tmpa_str{ #1 }
5615    \str_if_empty:NTF \l_tmpa_str {
5616      \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
5617      \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
5618    }{
5619      \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
5620      \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
5621    }
5622  }
```

```
5623  \newcommand\spfidea[2][]{
5624    \__stex_sproof_spf_args:n{#1}
5625    \titleemph{
5626      \tl_if_empty:NTF \spftype {Proof~Idea}{
5627        \spftype
5628      }:
5629    }~#2
5630    \sproofend
5631  }
```

(*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.

```
5632  \newenvironment{spfstep}[1][]{
5633    \__stex_sproof_spf_args:n{#1}
5634    \stex_if_smsmode:TF {
```

```
5635        \str_if_empty:NF \spfid {
5636          \stex_ref_new_doc_target:n \spfid
5637        }
5638      }{
5639        \@in@omtexttrue
5640        \seq_clear:N \l_tmpa_seq
5641        \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5642          \tl_if_empty:nF{ ##1 }{
5643            \stex_get_symbol:n { ##1 }
5644            \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5645              \l_stex_get_symbol_uri_str
5646            }
5647          }
5648        }
5649        \exp_args:Nnnx
5650        \begin{stex_annotate_env}{spfstep}{\seq_use:Nn \l_tmpa_seq {,}}
5651        \str_if_empty:NF \spftype {
5652          \stex_annotate_invisible:nnn{type}{\spftype}{}
5653        }
5654        \clist_set:No \l_tmpa_clist \spftype
5655        \tl_set:Nn \l_tmpa_tl {
5656          \item[\sproofnumber]
5657          \bool_set_true:N \l__stex_sproof_inc_counter_bool
5658        }
5659        \clist_map_inline:Nn \l_tmpa_clist {
5660          \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5661            \tl_clear:N \l_tmpa_tl
5662          }
5663        }
5664        \l_tmpa_tl
5665        \tl_if_empty:NF \spftitle {
5666          {(\titleemph{\spftitle})\enspace}
5667        }
5668        \str_if_empty:NF \spfid {
5669          \stex_ref_new_doc_target:n \spfid
5670        }
5671      }
5672      \stex_smsmode_do:
5673      \ignorespacesandpars
5674    }{
5675      \bool_if:NT \l__stex_sproof_inc_counter_bool {
5676        \__stex_sproof_inc_counter:
5677      }
5678      \stex_if_smsmode:F {
5679        \end{stex_annotate_env}
5680      }
5681  }
```

sproofcomment

```
5682  \newenvironment{sproofcomment}[1][]{
5683    \__stex_sproof_spf_args:n{#1}
5684    \clist_set:No \l_tmpa_clist \spftype
5685    \tl_set:Nn \l_tmpa_tl {
5686      \item[\sproofnumber]
```

```
5687        \bool_set_true:N \l__stex_sproof_inc_counter_bool
5688      }
5689    \clist_map_inline:Nn \l_tmpa_clist {
5690      \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5691        \tl_clear:N \l_tmpa_tl
5692      }
5693    }
5694    \l_tmpa_tl
5695  }{
5696    \bool_if:NT \l__stex_sproof_inc_counter_bool {
5697      \__stex_sproof_inc_counter:
5698    }
5699  }
```

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.

```
5700  \newenvironment{subproof}[2][]{
5701    \__stex_sproof_spf_args:n{#1}
5702    \stex_if_smsmode:TF{
5703      \str_if_empty:NF \spfid {
5704        \stex_ref_new_doc_target:n \spfid
5705      }
5706    }{
5707      \seq_clear:N \l_tmpa_seq
5708      \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5709        \tl_if_empty:nF{ ##1 }{
5710          \stex_get_symbol:n { ##1 }
5711          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5712            \l_stex_get_symbol_uri_str
5713          }
5714        }
5715      }
5716      \exp_args:Nnnx
5717      \begin{stex_annotate_env}{subproof}{\seq_use:Nn \l_tmpa_seq {,}}
5718      \str_if_empty:NF \spftype {
5719        \stex_annotate_invisible:nnn{type}{\spftype}{}
5720      }
5721
5722      \clist_set:No \l_tmpa_clist \spftype
5723      \tl_set:Nn \l_tmpa_tl {
5724        \item[\sproofnumber]
5725        \bool_set_true:N \l__stex_sproof_inc_counter_bool
5726      }
5727      \clist_map_inline:Nn \l_tmpa_clist {
5728        \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5729          \tl_clear:N \l_tmpa_tl
5730        }
5731      }
5732      \l_tmpa_tl
5733      \tl_if_empty:NF \spftitle {
5734        {(\titleemph{\spftitle})\enspace}
5735      }
```

```
5736        {~#2}
5737        \str_if_empty:NF \spfid {
5738          \stex_ref_new_doc_target:n \spfid
5739        }
5740      }
5741      \__stex_sproof_add_counter:
5742      \stex_smsmode_do:
5743    }{
5744      \__stex_sproof_remove_counter:
5745      \bool_if:NT \l__stex_sproof_inc_counter_bool {
5746        \__stex_sproof_inc_counter:
5747      }
5748      \stex_if_smsmode:F{
5749        \end{stex_annotate_env}
5750      }
5751    }
```

spfcases In the `pfcases` environment, the start text is displayed as the first comment of the proof.

```
5752 \newenvironment{spfcases}[2][]{
5753   \tl_if_empty:nTF{#1}{
5754     \begin{subproof}[method=by-cases]{#2}
5755   }{
5756     \begin{subproof}[#1,method=by-cases]{#2}
5757   }
5758 }{
5759   \end{subproof}
5760 }
```

spfcase In the `pfcase` environment, the start text is displayed specification of the case after the `\item`

```
5761 \newenvironment{spfcase}[2][]{
5762   \__stex_sproof_spf_args:n{#1}
5763   \stex_if_smsmode:TF {
5764     \str_if_empty:NF \spfid {
5765       \stex_ref_new_doc_target:n \spfid
5766     }
5767   }{
5768     \seq_clear:N \l_tmpa_seq
5769     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5770       \tl_if_empty:nF{ ##1 }{
5771         \stex_get_symbol:n { ##1 }
5772         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5773           \l_stex_get_symbol_uri_str
5774         }
5775       }
5776     }
5777     \exp_args:Nnnx
5778     \begin{stex_annotate_env}{spfcase}{\seq_use:Nn \l_tmpa_seq {,}}
5779     \str_if_empty:NF \spftype {
5780       \stex_annotate_invisible:nnn{type}{\spftype}{}
5781     }
5782     \clist_set:No \l_tmpa_clist \spftype
5783     \tl_set:Nn \l_tmpa_tl {
5784       \item[\sproofnumber]
```

```
5785        \bool_set_true:N \l__stex_sproof_inc_counter_bool
5786      }
5787      \clist_map_inline:Nn \l_tmpa_clist {
5788        \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5789          \tl_clear:N \l_tmpa_tl
5790        }
5791      }
5792      \l_tmpa_tl
5793      \tl_if_empty:nF{#2}{
5794        \titleemph{#2}:~
5795      }
5796    }
5797    \__stex_sproof_add_counter:
5798    \stex_smsmode_do:
5799  }{
5800    \__stex_sproof_remove_counter:
5801    \bool_if:NT \l__stex_sproof_inc_counter_bool {
5802      \__stex_sproof_inc_counter:
5803    }
5804    \stex_if_smsmode:F{
5805      \clist_set:No \l_tmpa_clist \spftype
5806      \tl_set:Nn \l_tmpa_tl{\sproofend}
5807      \clist_map_inline:Nn \l_tmpa_clist {
5808        \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5809          \tl_clear:N \l_tmpa_tl
5810        }
5811      }
5812      \l_tmpa_tl
5813      \end{stex_annotate_env}
5814    }
5815  }
```

spfcase   similar to spfcase, takes a third argument.

```
5816  \newcommand\spfcasesketch[3][]{
5817    \begin{spfcase}[#1]{#2}#3\end{spfcase}
5818  }
```

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.

```
5819  \keys_define:nn { stex / just }{
5820    id        .str_set_x:N  = \l__stex_sproof_just_id_str,
5821    method    .tl_set:N     = \l__stex_sproof_just_method_tl,
5822    premises  .tl_set:N     = \l__stex_sproof_just_premises_tl,
5823    args      .tl_set:N     = \l__stex_sproof_just_args_tl
5824  }
```

The next three environments and macros are purely semantic, so we ignore the keyval arguments for now and only display the content.[11]

EdN:11

---
[11]EDNOTE: need to do something about the premise in draft mode.

justification

```
5825  \newenvironment{justification}[1][]{}{}
```

\premise

```
5826  \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.

```
5827  \newcommand\justarg[2][]{#2}
5828  ⟨/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

sT_EX -Others Implementation

```
5829 ⟨*package⟩
5830
5831 %%%%%%%%%%%%   others.dtx   %%%%%%%%%%%%
5832
5833 ⟨@@=stex_others⟩
```

Warnings and error messages

```
5834   % None
```

\MSC Math subject classifier

```
5835 \NewDocumentCommand \MSC {m} {
5836   % TODO
5837 }
```

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

Patching tikzinput, if loaded

```
5838 \@ifpackageloaded{tikzinput}{
5839   \RequirePackage{stex-tikzinput}
5840 }{}
5841 ⟨/package⟩
```

Chapter 35

sTeX
-Metatheory Implementation

```
5842 ⟨∗package⟩
5843 ⟨@@=stex_modules⟩
5844
5845 %%%%%%%%%%%%   metatheory.dtx   %%%%%%%%%%%%
5846
5847 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
5848 \begingroup
5849 \stex_module_setup:nn{
5850   ns=\c_stex_metatheory_ns_str,
5851   meta=NONE
5852 }{Metatheory}
5853 \stex_reactivate_macro:N \symdecl
5854 \stex_reactivate_macro:N \notation
5855 \stex_reactivate_macro:N \symdef
5856 \ExplSyntaxOff
5857 \csname stex_suppress_html:n\endcsname{
5858   % is-a (a:A, a \in A, a is an A, etc.)
5859   \symdecl{isa}[args=ai]
5860   \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
5861   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
5862   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
5863
5864   % bind (\forall, \Pi, \lambda etc.)
5865   \symdecl{bind}[args=Bi]
5866   \notation{bind}[forall]{\comp\forall #1.\;#2}{##1 \comp, ##2}
5867   \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
5868   \notation{bind}[depfun]{\comp( #1 \comp()\;\to\;} #2}{##1 \comp, ##2}
5869
5870   % implicit bind
5871   \symdef{implicitbind}[args=Bi]{\comp\prod_{#1}#2}{##1\comp,##2}
5872
5873   % dummy variable
5874   \symdecl{dummyvar}
5875   \notation{dummyvar}[underscore]{\comp\_}
5876   \notation{dummyvar}[dot]{\comp\cdot}
```

```
5877    \notation{dummyvar}[dash]{\comp{{\rm --}}}
5878
5879    %fromto (function space, Hom-set, implication etc.)
5880    \symdecl{fromto}[args=ai]
5881    \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
5882    \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
5883
5884    % mapto (lambda etc.)
5885    %\symdecl{mapto}[args=Bi]
5886    %\notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
5887    %\notation{mapto}[lambda]{\comp\lambda #1 \comp.\; #2}{#1 \comp, #2}
5888    %\notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.\; #2}{#1 \comp, #2}
5889
5890    % function/operator application
5891    \symdecl{apply}[args=ia]
5892    \notation{apply}[prec=0;0x\infprec,parens]{#1 \comp( #2 \comp)}{##1 \comp, ##2}
5893    \notation{apply}[prec=0;0x\infprec,lambda]{#1 \; #2 }{##1 \; ##2}
5894
5895    % ``type'' of all collections (sets,classes,types,kinds)
5896    \symdecl{metacollection}
5897    \notation{metacollection}[U]{\comp{\mathcal{U}}}
5898    \notation{metacollection}[set]{\comp{\textsf{Set}}}
5899
5900    % collection of propositions/booleans/truth values
5901    \symdecl{prop}[name=proposition]
5902    \notation{prop}[prop]{\comp{{\rm prop}}}
5903    \notation{prop}[BOOL]{\comp{{\rm BOOL}}}
5904
5905    % sequences
5906    \symdecl{seqtype}[args=1]
5907    \notation{seqtype}[kleene]{#1^{\comp\ast}}
5908
5909    \symdef{sequence-index}[args=2,li,prec=nobrackets]{{#1}_{#2}}
5910    \notation{sequence-index}[ui,prec=nobrackets]{{#1}^{#2}}
5911
5912    \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{,\ellipses}}{##1\comp,##2}
5913    \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{,\ellipses,}#2}{##1\comp,##2}
5914    \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{,\ellipses,}#2\comp{,\ellipses,}#
5915
5916    % letin (``let'', local definitions, variable substitution)
5917    \symdecl{letin}[args=bii]
5918    \notation{letin}[let]{\comp{{\rm let}}\;#1\comp{=}#2\;\comp{{\rm in}}\;#3}
5919    \notation{letin}[subst]{#3 \comp[ #1 \comp/ #2 \comp]}
5920    \notation{letin}[frac]{#3 \comp[ \frac{#2}{#1} \comp]}
5921
5922    % structures
5923    \symdecl*{module-type}[args=1]
5924    \notation{module-type}{\mathtt{MOD} #1}
5925    \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
5926    \notation{mathstruct}[angle,prec=nobrackets]{\comp\langle #1 \comp\rangle}{##1 \comp, ##2}
5927
5928 }
5929    \ExplSyntaxOn
5930    \stex_add_to_current_module:n{
```

228

```
5931      \let\nappa\apply
5932      \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
5933      \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
5934      \def\livar{\csname sequence-index\endcsname[li]}
5935      \def\uivar{\csname sequence-index\endcsname[ui]}
5936      \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
5937      \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
5938      \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
5939    }
5940  \__stex_modules_end_module:
5941  \endgroup
5942  ⟨/package⟩
```

Chapter 36

Tikzinput Implementation

```
5943  ⟨*package⟩
5944
5945  %%%%%%%%%%%%   tikzinput.dtx   %%%%%%%%%%%%
5946
5947  \ProvidesExplPackage{tikzinput}{2022/02/26}{3.0.1}{tikzinput package}
5948  \RequirePackage{l3keys2e}
5949
5950  \keys_define:nn { tikzinput } {
5951    image    .bool_set:N   = \c_tikzinput_image_bool,
5952    image    .default:n    = false ,
5953    unknown    .code:n       = {}
5954  }
5955
5956  \ProcessKeysOptions { tikzinput }
5957
5958  \bool_if:NTF \c_tikzinput_image_bool {
5959    \RequirePackage{graphicx}
5960
5961    \providecommand\usetikzlibrary[]{}
5962    \newcommand\tikzinput[2][]{\includegraphics[#1]{#2}}
5963  }{
5964    \RequirePackage{tikz}
5965    \RequirePackage{standalone}
5966
5967    \newcommand \tikzinput [2] [] {
5968      \setkeys{Gin}{#1}
5969      \ifx \Gin@ewidth \Gin@exclamation
5970        \ifx \Gin@eheight \Gin@exclamation
5971          \input { #2 }
5972        \else
5973          \resizebox{!}{ \Gin@eheight }{
5974            \input { #2 }
5975          }
5976        \fi
5977      \else
5978        \ifx \Gin@eheight \Gin@exclamation
5979          \resizebox{ \Gin@ewidth }{!}{
5980            \input { #2 }
```

```
5981            }
5982         \else
5983            \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
5984               \input { #2 }
5985            }
5986         \fi
5987      \fi
5988   }
5989 }
5990
5991 \newcommand \ctikzinput [2] [] {
5992   \begin{center}
5993      \tikzinput [#1] {#2}
5994   \end{center}
5995 }
5996
5997 \@ifpackageloaded{stex}{
5998   \RequirePackage{stex-tikzinput}
5999 }{}
6000
6001 ⟨/package⟩
6002 ⟨∗stex⟩
6003 \ProvidesExplPackage{stex-tikzinput}{2022/02/26}{3.0.1}{stex-tikzinput}
6004 \RequirePackage{stex}
6005 \RequirePackage{tikzinput}
6006
6007 \newcommand\mhtikzinput[2][]{%
6008   \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
6009   \stex_in_repository:nn\Gin@mhrepos{
6010      \tikzinput[#1]{\mhpath{##1}{#2}}
6011   }
6012 }
6013 \newcommand\cmhtikzinput[2][]{\begin{center}\mhtikzinput[#1]{#2}\end{center}}
6014 ⟨/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

37.1 The document-structure Class

The functionality is spread over the `document-structure` class and package. The class provides the `document` environment and the `document-structure` element corresponds to it, whereas the package provides the concrete functionality.

```
6015  ⟨∗cls⟩
6016  ⟨@@=document_structure⟩
6017  \ProvidesExplClass{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure Class}
6018  \RequirePackage{l3keys2e}
```

37.2 Class Options

To initialize the `document-structure` class, we declare and process the necessary options using the `kvoptions` package for key/value options handling. For `omdoc.cls` this is quite simple. We have options `report` and `book`, which set the `\omdoc@cls@class` macro and pass on the macro to `omdoc.sty` for further processing.

`\omdoc@cls@class`

```
6019  \keys_define:nn{ document-structure / pkg }{
6020    class        .str_set_x:N  = \c_document_structure_class_str,
6021    minimal      .bool_set:N   = \c_document_structure_minimal_bool,
6022    report       .code:n       = {
6023      \ClassWarning{document-structure}{the option 'report' is deprecated, use 'class=report',
6024      \str_set:Nn \c_document_structure_class_str {report}
6025    },
6026    book         .code:n       = {
6027      \ClassWarning{document-structure}{the option 'book' is deprecated, use 'class=book', ins
6028      \str_set:Nn \c_document_structure_class_str {book}
6029    },
6030    bookpart     .code:n       = {
6031      \ClassWarning{document-structure}{the option 'bookpart' is deprecated, use 'class=book,t
6032      \str_set:Nn \c_document_structure_class_str {book}
6033      \str_set:Nn \c_document_structure_topsect_str {chapter}
6034    },
```

```
6035     docopt      .str_set_x:N  = \c_document_structure_docopt_str,
6036     unknown     .code:n       = {
6037       \PassOptionsToPackage{ \CurrentOption }{ document-structure }
6038     }
6039 }
6040 \ProcessKeysOptions{ document-structure / pkg }
6041 \str_if_empty:NT \c_document_structure_class_str {
6042     \str_set:Nn \c_document_structure_class_str {article}
6043 }
6044 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
6045     {\c_document_structure_class_str}
6046
```

37.3   Beefing up the `document` environment

Now, – unless the option `minimal` is defined – we include the `stex` package

```
6047 \RequirePackage{document-structure}
6048 \bool_if:NF \c_document_structure_minimal_bool {
```

And define the environments we need. The top-level one is the `document` environment, which we redefined so that we can provide keyval arguments.

`document` For the moment we do not use them on the LATEX level, but the document identifier is
EdN:12 picked up by LATEXML.[12]

```
6049 \keys_define:nn { document-structure / document }{
6050     id .str_set_x:N = \c_document_structure_document_id_str
6051 }
6052 \let\__document_structure_orig_document=\document
6053 \renewcommand{\document}[1][]{
6054     \keys_set:nn{ document-structure / document }{ #1 }
6055     \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
6056     \__document_structure_orig_document
6057 }
```

Finally, we end the test for the `minimal` option.

```
6058 }
6059 ⟨/cls⟩
```

37.4   Implementation: document-structure Package

```
6060 ⟨*package⟩
6061 \ProvidesExplPackage{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure}
6062 \RequirePackage{l3keys2e}
```

37.5   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).

---

[12]EDNOTE: faking documentkeys for now. @HANG, please implement

```
6063
6064 \keys_define:nn{ document-structure / pkg }{
6065   class       .str_set_x:N  = \c_document_structure_class_str,
6066   topsect     .str_set_x:N  = \c_document_structure_topsect_str,
6067 %  showignores .bool_set:N   = \c_document_structure_showignores_bool,
6068 }
6069 \ProcessKeysOptions{ document-structure / pkg }
6070 \str_if_empty:NT \c_document_structure_class_str {
6071   \str_set:Nn \c_document_structure_class_str {article}
6072 }
6073 \str_if_empty:NT \c_document_structure_topsect_str {
6074   \str_set:Nn \c_document_structure_topsect_str {section}
6075 }
```

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

```
6076 \RequirePackage{xspace}
6077 \RequirePackage{comment}
6078 \AddToHook{begindocument}{
6079 \ltx@ifpackageloaded{babel}{
6080    \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6081    \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6082       \makeatletter\input{document-structure-ngerman.ldf}\makeatother
6083    }
6084   }{}
6085 }
```

`\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.

```
6086 \int_new:N \l_document_structure_section_level_int
6087 \str_case:VnF \c_document_structure_topsect_str {
6088   {part}{
6089     \int_set:Nn \l_document_structure_section_level_int {0}
6090   }
6091   {chapter}{
6092     \int_set:Nn \l_document_structure_section_level_int {1}
6093   }
6094 }{
6095   \str_case:VnF \c_document_structure_class_str {
6096     {book}{
6097       \int_set:Nn \l_document_structure_section_level_int {0}
6098     }
6099     {report}{
6100       \int_set:Nn \l_document_structure_section_level_int {0}
6101     }
6102   }{
6103     \int_set:Nn \l_document_structure_section_level_int {2}
6104   }
6105 }
```

234

37.6 Document Structure

The structure of the document is given by the `omgroup` environment just like in OMDoc. The hierarchy is adjusted automatically according to the LATEX class in effect.

\currentsectionlevel

For the `\currentsectionlevel` and `\Currentsectionlevel` macros we use an internal macro `\current@section@level` that only contains the keyword (no markup). We initialize it with "document" as a default. In the generated OMDoc, we only generate a text element of class `omdoc_currentsectionlevel`, wich will be instantiated by CSS later.[13]

EdN:13

```
6106 \def\current@section@level{document}%
6107 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
6108 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

\skipomgroup

```
6109 \cs_new_protected:Npn \skipomgroup {
6110   \ifcase\l_document_structure_section_level_int
6111   \or\stepcounter{part}
6112   \or\stepcounter{chapter}
6113   \or\stepcounter{section}
6114   \or\stepcounter{subsection}
6115   \or\stepcounter{subsubsection}
6116   \or\stepcounter{paragraph}
6117   \or\stepcounter{subparagraph}
6118   \fi
6119 }
```

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

blindfragment

```
6120 \newcommand\at@begin@blindomgroup[1]{}
6121 \newenvironment{blindfragment}
6122 {
6123   \int_incr:N\l_document_structure_section_level_int
6124   \at@begin@blindomgroup\l_document_structure_section_level_int
6125 }{}
```

\omgroup@nonum

convenience macro: `\omgroup@nonum{⟨level⟩}{⟨title⟩}` makes an unnumbered sectioning with title ⟨*title*⟩ at level ⟨*level*⟩.

```
6126 \newcommand\omgroup@nonum[2]{
6127   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
6128   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
6129 }
```

(*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.

```
6130 \newcommand\omgroup@num[2]{
```

---

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

```
6131    \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
6132      \@nameuse{#1}{#2}
6133    }{
6134      \cs_if_exist:NTF\rdfmeta@sectioning{
6135        \@nameuse{rdfmeta@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
6136      }{
6137        \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
6138      }
6139    }
6140  %\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\omgroup@id
6141  }
```

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

sfragment

```
6142  \keys_define:nn { document-structure / omgroup }{
6143    id            .str_set_x:N = \l__document_structure_omgroup_id_str,
6144    date          .str_set_x:N = \l__document_structure_omgroup_date_str,
6145    creators      .clist_set:N = \l__document_structure_omgroup_creators_clist,
6146    contributors  .clist_set:N = \l__document_structure_omgroup_contributors_clist,
6147    srccite       .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
6148    type          .tl_set:N    = \l__document_structure_omgroup_type_tl,
6149    short         .tl_set:N    = \l__document_structure_omgroup_short_tl,
6150    display       .tl_set:N    = \l__document_structure_omgroup_display_tl,
6151    intro         .tl_set:N    = \l__document_structure_omgroup_intro_tl,
6152    loadmodules   .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
6153  }
6154  \cs_new_protected:Nn \__document_structure_omgroup_args:n {
6155    \str_clear:N \l__document_structure_omgroup_id_str
6156    \str_clear:N \l__document_structure_omgroup_date_str
6157    \clist_clear:N \l__document_structure_omgroup_creators_clist
6158    \clist_clear:N \l__document_structure_omgroup_contributors_clist
6159    \tl_clear:N \l__document_structure_omgroup_srccite_tl
6160    \tl_clear:N \l__document_structure_omgroup_type_tl
6161    \tl_clear:N \l__document_structure_omgroup_short_tl
6162    \tl_clear:N \l__document_structure_omgroup_display_tl
6163    \tl_clear:N \l__document_structure_omgroup_intro_tl
6164    \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
6165    \keys_set:nn { document-structure / omgroup } { #1 }
6166  }
```

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.

```
6167  \newif\if@mainmatter\@mainmattertrue
6168  \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.

```
6169  \keys_define:nn { document-structure / sectioning }{
6170    name    .str_set_x:N  = \l__document_structure_sect_name_str    ,
6171    ref     .str_set_x:N  = \l__document_structure_sect_ref_str     ,
6172    clear   .bool_set:N   = \l__document_structure_sect_clear_bool  ,
6173    clear   .default:n    = {true}                                 ,
6174    num     .bool_set:N   = \l__document_structure_sect_num_bool    ,
```

236

```
6175    num      .default:n   = {true}
6176  }
6177  \cs_new_protected:Nn \__document_structure_sect_args:n {
6178    \str_clear:N \l__document_structure_sect_name_str
6179    \str_clear:N \l__document_structure_sect_ref_str
6180    \bool_set_false:N \l__document_structure_sect_clear_bool
6181    \bool_set_false:N \l__document_structure_sect_num_bool
6182    \keys_set:nn { document-structure / sectioning } { #1 }
6183  }
6184  \newcommand\omdoc@sectioning[3][]{
6185    \__document_structure_sect_args:n {#1 }
6186    \let\omdoc@sect@name\l__document_structure_sect_name_str
6187    \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
6188    \if@mainmatter% numbering not overridden by frontmatter, etc.
6189      \bool_if:NTF \l__document_structure_sect_num_bool {
6190        \omgroup@num{#2}{#3}
6191      }{
6192        \omgroup@nonum{#2}{#3}
6193      }
6194      \def\current@section@level{\omdoc@sect@name}
6195    \else
6196      \omgroup@nonum{#2}{#3}
6197    \fi
6198  }% 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.

```
6199  \newcommand\omgroup@redefine@addtocontents[1]{%
6200  %\edef\__document_structureimport{#1}%
6201  %\@for\@I:=\__document_structureimport\do{%
6202  %\edef\@path{\csname module@\@I  @path\endcsname}%
6203  %\@ifundefined{tf@toc}\relax%
6204  %    {\protected@write\tf@toc{}{\string\@requiremodules{\@path}}}}
6205  %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
6206  %\def\addcontentsline##1##2##3{%
6207  %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}
6208  %\else% hyperref.sty not loaded
6209  %\def\addcontentsline##1##2##3{%
6210  %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}{
6211  %\fi
6212  }% 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.

```
6213  \newenvironment{sfragment}[2][]% keys, title
6214  {
6215    \__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.

```
6216    \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
6217      \omgroup@redefine@addtocontents{
6218        %\@ifundefined{module@id}\used@modules%
```

237

```
6219        %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}
6220     }
6221   }
```

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

```
6222    \int_incr:N\l_document_structure_section_level_int
6223    \ifcase\l_document_structure_section_level_int
6224      \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
6225      \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
6226      \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
6227      \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
6228      \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
6229      \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#
6230      \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{paragr
6231    \fi
6232    \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
6233    \str_if_empty:NF \l__document_structure_omgroup_id_str {
6234      \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
6235    }
6236 }% for customization
6237 {}
```

and finally, we localize the sections

```
6238 \newcommand\omdoc@part@kw{Part}
6239 \newcommand\omdoc@chapter@kw{Chapter}
6240 \newcommand\omdoc@section@kw{Section}
6241 \newcommand\omdoc@subsection@kw{Subsection}
6242 \newcommand\omdoc@subsubsection@kw{Subsubsection}
6243 \newcommand\omdoc@paragraph@kw{paragraph}
6244 \newcommand\omdoc@subparagraph@kw{subparagraph}
```

37.7   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

```
6245 \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).

```
6246 \cs_if_exist:NTF\frontmatter{
6247   \let\__document_structure_orig_frontmatter\frontmatter
6248   \let\frontmatter\relax
6249 }{
6250   \tl_set:Nn\__document_structure_orig_frontmatter{
6251     \clearpage
6252     \@mainmatterfalse
6253     \pagenumbering{roman}
```

```
6254       }
6255   }
6256   \cs_if_exist:NTF\backmatter{
6257       \let\__document_structure_orig_backmatter\backmatter
6258       \let\backmatter\relax
6259   }{
6260       \tl_set:Nn\__document_structure_orig_backmatter{
6261           \clearpage
6262           \@mainmatterfalse
6263           \pagenumbering{roman}
6264       }
6265   }
```

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.

```
6266   \newenvironment{frontmatter}{
6267       \__document_structure_orig_frontmatter
6268   }{
6269       \cs_if_exist:NTF\mainmatter{
6270           \mainmatter
6271       }{
6272           \clearpage
6273           \@mainmattertrue
6274           \pagenumbering{arabic}
6275       }
6276   }
```

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

```
6277   \newenvironment{backmatter}{
6278       \__document_structure_orig_backmatter
6279   }{
6280       \cs_if_exist:NTF\mainmatter{
6281           \mainmatter
6282       }{
6283           \clearpage
6284           \@mainmattertrue
6285           \pagenumbering{arabic}
6286       }
6287   }
```

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

```
6288   \@mainmattertrue\pagenumbering{arabic}
```

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

```
6289   \def \c__document_structure_document_str{document}
6290   \newcommand\afterprematurestop{}
6291   \def\prematurestop@endomgroup{
6292       \unless\ifx\@currenvir\c__document_structure_document_str
6293           \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafte
6294           \expandafter\prematurestop@endomgroup
```

```
6295     \fi
6296 }
6297 \providecommand\prematurestop{
6298     \message{Stopping~sTeX~processing~prematurely}
6299     \prematurestop@endomgroup
6300     \afterprematurestop
6301     \end{document}
6302 }
```

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

37.8   Global Variables

\setSGvar   set a global variable

```
6303 \RequirePackage{etoolbox}
6304 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}
```

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

\useSGvar   use a global variable

```
6305 \newrobustcmd\useSGvar[1]{%
6306     \@ifundefined{sTeX@Gvar@#1}
6307     {\PackageError{document-structure}
6308         {The sTeX Global variable #1 is undefined}
6309         {set it with \protect\setSGvar}}
6310     \@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.

```
6311 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
6312     \@ifundefined{sTeX@Gvar@#1}
6313     {\PackageError{document-structure}
6314         {The sTeX Global variable #1 is undefined}
6315         {set it with \protect\setSGvar}}
6316     {\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.

```
6317 ⟨*cls⟩
6318 ⟨@@=notesslides⟩
6319 \ProvidesExplClass{notesslides}{2022/02/28}{3.1.0}{notesslides Class}
6320 \RequirePackage{l3keys2e}
6321
6322 \keys_define:nn{notesslides / cls}{
6323   class    .code:n   = {
6324     \PassOptionsToClass{\CurrentOption}{document-structure}
6325     \str_if_eq:nnT{#1}{book}{
6326       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
6327     }
6328     \str_if_eq:nnT{#1}{report}{
6329       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
6330     }
6331   },
6332   notes    .bool_set:N  = \c__notesslides_notes_bool ,
6333   slides   .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
6334   unknown .code:n       = {
6335     \PassOptionsToClass{\CurrentOption}{document-structure}
6336     \PassOptionsToClass{\CurrentOption}{beamer}
6337     \PassOptionsToPackage{\CurrentOption}{notesslides}
6338   }
6339 }
6340 \ProcessKeysOptions{ notesslides / cls }
6341 \bool_if:NTF \c__notesslides_notes_bool {
6342   \PassOptionsToPackage{notes=true}{notesslides}
6343 }{
6344   \PassOptionsToPackage{notes=false}{notesslides}
6345 }
6346 ⟨/cls⟩
```

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

```
6347 ⟨∗package⟩
6348 \ProvidesExplPackage{notesslides}{2022/02/28}{3.1.0}{notesslides Package}
6349 \RequirePackage{l3keys2e}
6350
6351 \keys_define:nn{notesslides / pkg}{
6352   topsect        .str_set_x:N  = \c__notesslides_topsect_str,
6353   defaulttopsect .str_set_x:N  = \c__notesslides_defaulttopsec_str,
6354   notes          .bool_set:N   = \c__notesslides_notes_bool ,
6355   slides         .code:n       = { \bool_set_false:N \c__notesslides_notes_bool },
6356   sectocframes   .bool_set:N   = \c__notesslides_sectocframes_bool ,
6357   frameimages    .bool_set:N   = \c__notesslides_frameimages_bool ,
6358   fiboxed        .bool_set:N   = \c__notesslides_fiboxed_bool ,
6359   noproblems     .bool_set:N   = \c__notesslides_noproblems_bool,
6360   unknown        .code:n       = {
6361     \PassOptionsToClass{\CurrentOption}{stex}
6362     \PassOptionsToClass{\CurrentOption}{tikzinput}
6363   }
6364 }
6365 \ProcessKeysOptions{ notesslides / pkg }
6366 \newif\ifnotes
6367 \bool_if:NTF \c__notesslides_notes_bool {
6368   \notestrue
6369 }{
6370   \notesfalse
6371 }
6372
```

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

```
6373 \str_if_empty:NTF \c__notesslides_topsect_str {
6374   \str_set_eq:NN \__notesslidestopsect \c__notesslides_defaulttopsec_str
6375 }{
6376   \str_set_eq:NN \__notesslidestopsect \c__notesslides_topsect_str
6377 }
6378 ⟨/package⟩
```

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

```
6379 ⟨∗cls⟩
6380 \bool_if:NTF \c__notesslides_notes_bool {
6381   \LoadClass{document-structure}
6382 }{
6383   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
6384   \newcounter{Item}
6385   \newcounter{paragraph}
6386   \newcounter{subparagraph}
6387   \newcounter{Hfootnote}
6388   \RequirePackage{document-structure}
6389 }
```

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

```
6390 \RequirePackage{notesslides}
6391 ⟨/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.

```
6392 ⟨∗package⟩
6393 \bool_if:NT \c__notesslides_notes_bool {
6394     \RequirePackage{a4wide}
6395     \RequirePackage{marginnote}
6396     \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
6397     \RequirePackage{mdframed}
6398     \RequirePackage[noxcolor,noamsthm]{beamerarticle}
6399     \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
6400 }
6401 \RequirePackage{stex-tikzinput}
6402 \RequirePackage{etoolbox}
6403 \RequirePackage{amssymb}
6404 \RequirePackage{amsmath}
6405 \RequirePackage{comment}
6406 \RequirePackage{textcomp}
6407 \RequirePackage{url}
6408 \RequirePackage{graphicx}
6409 \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`.[14]

EdN:14

```
6410 \bool_if:NT \c__notesslides_notes_bool {
6411     \renewcommand\usetheme[2][]{\usepackage[#1]{beamernotestheme#2}}
6412 }
6413
6414
6415 \NewDocumentCommand \libusetheme {O{} m} {
6416     \bool_if:NTF \c__notesslides_notes_bool {
6417         \libusepackage[#1]{beamernotestheme#2}
6418     }{
6419     \libusepackage[#1]{beamertheme#2}
6420     }
6421 }
```

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

```
6422 \newcounter{slide}
6423 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
6424 \newlength{\slideheight}\setlength{\slideheight}{9cm}
```

---

[14]EDNOTE: MK: This is not ideal, but I am not sure that I want to be able to provide the full theme functionality there.

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.

```
6425 \bool_if:NTF \c__notesslides_notes_bool {
6426     \renewenvironment{note}{\ignorespaces}{}
6427 }{
6428     \excludecomment{note}
6429 }
```

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.

```
6430 \bool_if:NT \c__notesslides_notes_bool {
6431     \newlength{\slideframewidth}
6432     \setlength{\slideframewidth}{1.5pt}
```

frame We first define the keys.

```
6433     \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
6434         \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
6435             \bool_set_true:N #1
6436         }{
6437             \bool_set_false:N #1
6438         }
6439     }
6440     \keys_define:nn{notesslides / frame}{
6441         label                .str_set_x:N  = \l__notesslides_frame_label_str,
6442         allowframebreaks     .code:n       = {
6443             \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
6444         },
6445         allowdisplaybreaks   .code:n       = {
6446             \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
6447         },
6448         fragile              .code:n       = {
6449             \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
6450         },
6451         shrink               .code:n       = {
6452             \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
6453         },
6454         squeeze              .code:n       = {
6455             \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
6456         },
6457         t                    .code:n       = {
6458             \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
6459         },
6460     }
6461     \cs_new_protected:Nn \__notesslides_frame_args:n {
6462         \str_clear:N \l__notesslides_frame_label_str
6463         \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
6464         \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
6465         \bool_set_true:N \l__notesslides_frame_fragile_bool
6466         \bool_set_true:N \l__notesslides_frame_shrink_bool
6467         \bool_set_true:N \l__notesslides_frame_squeeze_bool
6468         \bool_set_true:N \l__notesslides_frame_t_bool
```

```
6469        \keys_set:nn { notesslides / frame }{ #1 }
6470    }
```

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

```
6471    \renewenvironment{frame}[1][]{
6472      \__notesslides_frame_args:n{#1}
6473      \sffamily
6474      \stepcounter{slide}
6475      \def\@currentlabel{\theslide}
6476      \str_if_empty:NF \l__notesslides_frame_label_str {
6477        \label{\l__notesslides_frame_label_str}
6478      }
```

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

```
6479      \def\itemize@level{outer}
6480      \def\itemize@outer{outer}
6481      \def\itemize@inner{inner}
6482      \renewcommand\newpage{\addtocounter{framenumber}{1}}
6483      \newcommand\metakeys@show@keys[2]{\marginnote{{\scriptsize ##2}}}
6484      \renewenvironment{itemize}{
6485        \ifx\itemize@level\itemize@outer
6486          \def\itemize@label{$\rhd$}
6487        \fi
6488        \ifx\itemize@level\itemize@inner
6489          \def\itemize@label{$\scriptstyle\rhd$}
6490        \fi
6491        \begin{list}
6492        {\itemize@label}
6493        {\setlength{\labelsep}{.3em}
6494         \setlength{\labelwidth}{.5em}
6495         \setlength{\leftmargin}{1.5em}
6496        }
6497        \edef\itemize@level{\itemize@inner}
6498      }{
6499        \end{list}
6500      }
```

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

```
6501      \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth
6502    }{
6503      \medskip\miko@slidelabel\end{mdframed}
6504    }
```

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

\frametitle

```
6505      \renewcommand{\frametitle}[1]{{\Large\bf\sf\color{blue}{#1}}\medskip}
6506 }
```

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

EdN:15   \pause   [15]

```
6507 \bool_if:NT \c__notesslides_notes_bool {
6508   \newcommand\pause{}
6509 }
```

---
[15]EdNote: MK: fake it in notes mode for now

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

nparagraph

```
6510 \bool_if:NTF \c__notesslides_notes_bool {
6511   \newenvironment{nparagraph}[1][]{\begin{sparagraph}[#1]}{\end{sparagraph}}
6512 }{
6513   \excludecomment{nparagraph}
6514 }
```

nfragment

```
6515 \bool_if:NTF \c__notesslides_notes_bool {
6516   \newenvironment{nfragment}[2][]{\begin{sfragment}[#1]{#2}}{\end{sfragment}}
6517 }{
6518   \excludecomment{nfragment}
6519 }
```

ndefinition

```
6520 \bool_if:NTF \c__notesslides_notes_bool {
6521   \newenvironment{ndefinition}[1][]{\begin{sdefinition}[#1]}{\end{sdefinition}}
6522 }{
6523   \excludecomment{ndefinition}
6524 }
```

nassertion

```
6525 \bool_if:NTF \c__notesslides_notes_bool {
6526   \newenvironment{nassertion}[1][]{\begin{sassertion}[#1]}{\end{sassertion}}
6527 }{
6528   \excludecomment{nassertion}
6529 }
```

nsproof

```
6530 \bool_if:NTF \c__notesslides_notes_bool {
6531   \newenvironment{nproof}[2][]{\begin{sproof}[#1]{#2}}{\end{sproof}}
6532 }{
6533   \excludecomment{nproof}
6534 }
```

nexample

```
6535 \bool_if:NTF \c__notesslides_notes_bool {
6536   \newenvironment{nexample}[1][]{\begin{sexample}[#1]}{\end{sexample}}
6537 }{
6538   \excludecomment{nexample}
6539 }
```

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

```
6540 \def\inputref@preskip{\smallskip}
6541 \def\inputref@postskip{\medskip}
```

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

246

```
6542  \let\orig@inputref\inputref
6543  \def\inputref{\@ifstar\ninputref\orig@inputref}
6544  \newcommand\ninputref[2][]{
6545    \bool_if:NT \c__notesslides_notes_bool {
6546      \orig@inputref[#1]{#2}
6547    }
6548  }
```

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

38.3   Header and Footer Lines

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

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

```
6549  \newlength{\slidelogoheight}
6550
6551  \bool_if:NTF \c__notesslides_notes_bool {
6552    \setlength{\slidelogoheight}{.4cm}
6553  }{
6554    \setlength{\slidelogoheight}{1cm}
6555  }
6556  \newsavebox{\slidelogo}
6557  \sbox{\slidelogo}{\sTeX}
6558  \newrobustcmd{\setslidelogo}[1]{
6559    \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
6560  }
```

(*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.

```
6561  \def\source{Michael Kohlhase}% customize locally
6562  \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.

```
6563  \def\copyrightnotice{\footnotesize\copyright :\hspace{.3ex}{\source}}
6564  \newsavebox{\cclogo}
6565  \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{stex-cc_somerights}}
6566  \newif\ifcchref\cchreffalse
6567  \AtBeginDocument{
6568    \@ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
6569  }
6570  \def\licensing{
6571    \ifcchref
```

```
6572        \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
6573     \else
6574       {\usebox{\cclogo}}
6575     \fi
6576 }
6577 \newrobustcmd{\setlicensing}[2][]{
6578    \def\@url{#1}
6579    \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
6580    \ifx\@url\@empty
6581      \def\licensing{{\usebox{\cclogo}}}
6582    \else
6583      \def\licensing{
6584        \ifcchref
6585        \href{#1}{\usebox{\cclogo}}
6586        \else
6587        {\usebox{\cclogo}}
6588        \fi
6589      }
6590    \fi
6591 }
```

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

EdN:16   `\slidelabel`   Now, we set up the slide label for the `article` mode.[16]

```
6592 \newrobustcmd\miko@slidelabel{
6593    \vbox to \slidelogoheight{
6594      \vss\hbox to \slidewidth
6595      {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
6596    }
6597 }
```

(*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.

```
6598 \def\Gin@mhrepos{}
6599 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
6600 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
6601 \newrobustcmd\frameimage[2][]{
6602    \stepcounter{slide}
6603    \bool_if:NT \c__notesslides_frameimages_bool {
6604      \def\Gin@ewidth{}\setkeys{Gin}{#1}
6605      \bool_if:NF \c__notesslides_notes_bool { \vfill }
6606      \begin{center}
6607        \bool_if:NTF \c__notesslides_fiboxed_bool {
6608          \fbox{
6609            \ifx\Gin@ewidth\@empty
6610              \ifx\Gin@mhrepos\@empty
6611                \mhgraphics[width=\slidewidth,#1]{#2}
6612              \else
```

---

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

```
6613              \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
6614            \fi
6615          \else% Gin@ewidth empty
6616            \ifx\Gin@mhrepos\@empty
6617              \mhgraphics[#1]{#2}
6618            \else
6619              \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
6620            \fi
6621          \fi% Gin@ewidth empty
6622        }
6623      }{
6624        \ifx\Gin@ewidth\@empty
6625          \ifx\Gin@mhrepos\@empty
6626            \mhgraphics[width=\slidewidth,#1]{#2}
6627          \else
6628            \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
6629          \fi
6630          \ifx\Gin@mhrepos\@empty
6631            \mhgraphics[#1]{#2}
6632          \else
6633            \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
6634          \fi
6635        \fi% Gin@ewidth empty
6636      }
6637      \end{center}
6638      \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
6639      \bool_if:NF \c__notesslides_notes_bool { \vfill }
6640    }
6641 } % 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.

```
6642 \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

```
6643 \AddToHook{begindocument}{
6644   \definecolor{green}{rgb}{0,.5,0}
6645   \definecolor{purple}{cmyk}{.3,1,0,.17}
6646 }
```

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.

```
6647 % \def\STpresent#1{\textcolor{blue}{#1}}
6648 \def\defemph#1{{\textcolor{magenta}{#1}}}
6649 \def\symrefemph#1{{\textcolor{cyan}{#1}}}
6650 \def\compemph#1{{\textcolor{blue}{#1}}}
6651 \def\titleemph#1{{\textcolor{blue}{#1}}}
6652 \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.

```
6653 \pgfdeclareimage[width=.8em]{miko@small@dbend}{stex-dangerous-bend}
6654 \def\smalltextwarning{
6655   \pgfuseimage{miko@small@dbend}
6656   \xspace
6657 }
6658 \pgfdeclareimage[width=1.2em]{miko@dbend}{stex-dangerous-bend}
6659 \newrobustcmd\textwarning{
6660   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
6661   \xspace
6662 }
6663 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{stex-dangerous-bend}
6664 \newrobustcmd\bigtextwarning{
6665   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
6666   \xspace
6667 }
```

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

```
6668 \newrobustcmd\putgraphicsat[3]{
6669   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
6670 }
6671 \newrobustcmd\putat[2]{
6672   \begin{picture}(0,0)\put(#1){#2}\end{picture}
6673 }
```

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`.

```
6674 \bool_if:NT \c__notesslides_sectocframes_bool {
6675   \str_if_eq:VnTF \__notesslidestopsect{part}{
6676     \newcounter{chapter}\counterwithin*{section}{chapter}
6677   }{
6678     \str_if_eq:VnT\__notesslidestopsect{chapter}{
6679       \newcounter{chapter}\counterwithin*{section}{chapter}
6680     }
6681   }
6682 }
```

\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

```
6683 \def\part@prefix{}
6684 \@ifpackageloaded{document-structure}{}{
6685   \str_case:VnF \__notesslidestopsect {
6686     {part}{
6687       \int_set:Nn \l_document_structure_section_level_int {0}
6688       \def\thesection{\arabic{chapter}.\arabic{section}}
```

```
6689        \def\part@prefix{\arabic{chapter}.}
6690      }
6691      {chapter}{
6692        \int_set:Nn \l_document_structure_section_level_int {1}
6693        \def\thesection{\arabic{chapter}.\arabic{section}}
6694        \def\part@prefix{\arabic{chapter}.}
6695      }
6696    }{
6697      \int_set:Nn \l_document_structure_section_level_int {2}
6698      \def\part@prefix{}
6699    }
6700  }
6701
6702  \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.

```
6703    \renewenvironment{sfragment}[2][]{
6704      \__document_structure_omgroup_args:n { #1 }
6705      \int_incr:N \l_document_structure_section_level_int
6706      \bool_if:NT \c__notesslides_sectocframes_bool {
6707        \stepcounter{slide}
6708        \begin{frame}[noframenumbering]
6709        \vfill\Large\centering
6710        \red{
6711          \ifcase\l_document_structure_section_level_int\or
6712            \stepcounter{part}
6713            \def\__notesslideslabel{\omdoc@part@kw~\Roman{part}}
6714            \def\currentsectionlevel{\omdoc@part@kw}
6715          \or
6716            \stepcounter{chapter}
6717            \def\__notesslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
6718            \def\currentsectionlevel{\omdoc@chapter@kw}
6719          \or
6720            \stepcounter{section}
6721            \def\__notesslideslabel{\part@prefix\arabic{section}}
6722            \def\currentsectionlevel{\omdoc@section@kw}
6723          \or
6724            \stepcounter{subsection}
6725            \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
6726            \def\currentsectionlevel{\omdoc@subsection@kw}
6727          \or
6728            \stepcounter{subsubsection}
6729            \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
6730            \def\currentsectionlevel{\omdoc@subsubsection@kw}
6731          \or
6732            \stepcounter{paragraph}
6733            \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
6734            \def\currentsectionlevel{\omdoc@paragraph@kw}
6735          \else
6736            \def\__notesslideslabel{}
```

251

```
6737          \def\currentsectionlevel{\omdoc@paragraph@kw}
6738        \fi% end ifcase
6739        \__notesslideslabel%\sref@label@id\__notesslideslabel
6740        \quad #2%
6741      }%
6742      \vfill%
6743      \end{frame}%
6744    }
6745    \str_if_empty:NF \l__document_structure_omgroup_id_str {
6746      \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
6747    }
6748  }{}
6749 }
```

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

```
6750 \def\inserttheorembodyfont{\normalfont}
6751 %\bool_if:NF \c__notesslides_notes_bool {
6752 %  \defbeamertemplate{theorem begin}{miko}
6753 %  {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
6754 %    \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
6755 %    \inserttheorempunctuation\inserttheorembodyfont\xspace}
6756 %  \defbeamertemplate{theorem end}{miko}{}
```

and we set it as the default one.

```
6757 %  \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.

```
6758 %  \expandafter\def\csname Parent2\endcsname{}
6759 %}
6760
6761 \AddToHook{begindocument}{ % this does not work for some reasone
6762   \setbeamertemplate{theorems}[ams style]
6763 }
6764 \bool_if:NT \c__notesslides_notes_bool {
6765   \renewenvironment{columns}[1][]{%
6766     \par\noindent%
6767     \begin{minipage}%
6768     \slidewidth\centering\leavevmode%
6769   }{%
6770     \end{minipage}\par\noindent%
6771   }%
6772   \newsavebox\columnbox%
6773   \renewenvironment<>{column}[2][]{%
6774     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
6775   }{%
6776     \end{minipage}\end{lrbox}\usebox\columnbox%
6777   }%
6778 }
6779 \bool_if:NTF \c__notesslides_noproblems_bool {
6780   \newenvironment{problems}{}{}
6781 }{
6782   \excludecomment{problems}
6783 }
```

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.

```
6784 \gdef\printexcursions{}
6785 \newcommand\excursionref[2]{% label, text
6786   \bool_if:NT \c__notesslides_notes_bool {
6787     \begin{sparagraph}[title=Excursion]
6788       #2 \sref[fallback=the appendix]{#1}.
6789     \end{sparagraph}
6790   }
6791 }
6792 \newcommand\activate@excursion[2][]{
6793   \gappto\printexcursions{\inputref[#1]{#2}}
6794 }
6795 \newcommand\excursion[4][]{% repos, label, path, text
6796   \bool_if:NT \c__notesslides_notes_bool {
6797     \activate@excursion[#1]{#3}\excursionref{#2}{#4}
6798   }
6799 }
```

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

\excursiongroup

```
6800 \keys_define:nn{notesslides / excursiongroup }{
6801   id        .str_set_x:N = \l__notesslides_excursion_id_str,
6802   intro     .tl_set:N    = \l__notesslides_excursion_intro_tl,
6803   mhrepos   .str_set_x:N = \l__notesslides_excursion_mhrepos_str
6804 }
6805 \cs_new_protected:Nn \__notesslides_excursion_args:n {
6806   \tl_clear:N \l__notesslides_excursion_intro_tl
6807   \str_clear:N \l__notesslides_excursion_id_str
6808   \str_clear:N \l__notesslides_excursion_mhrepos_str
6809   \keys_set:nn {notesslides / excursiongroup }{ #1 }
6810 }
6811 \newcommand\excursiongroup[1][]{
6812   \__notesslides_excursion_args:n{ #1 }
6813   \ifdefempty\printexcursions{}% only if there are excursions
6814   {\begin{note}
6815     \begin{sfragment}[#1]{Excursions}%
6816       \ifdefempty\l__notesslides_excursion_intro_tl{}{
6817         \inputref[\l__notesslides_excursion_mhrepos_str]{
6818           \l__notesslides_excursion_intro_tl
6819         }
6820       }
6821       \printexcursions%
6822     \end{sfragment}
6823   \end{note}}
6824 }
6825 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
6826 ⟨/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.

```
6827  ⟨*package⟩
6828  ⟨@@=problems⟩
6829  \ProvidesExplPackage{problem}{2022/02/26}{3.0.1}{Semantic Markup for Problems}
6830  \RequirePackage{l3keys2e,stex}
6831
6832  \keys_define:nn { problem / pkg }{
6833    notes     .default:n   = { true },
6834    notes     .bool_set:N  = \c__problems_notes_bool,
6835    gnotes    .default:n   = { true },
6836    gnotes    .bool_set:N  = \c__problems_gnotes_bool,
6837    hints     .default:n   = { true },
6838    hints     .bool_set:N  = \c__problems_hints_bool,
6839    solutions .default:n   = { true },
6840    solutions .bool_set:N  = \c__problems_solutions_bool,
6841    pts       .default:n   = { true },
6842    pts       .bool_set:N  = \c__problems_pts_bool,
6843    min       .default:n   = { true },
6844    min       .bool_set:N  = \c__problems_min_bool,
6845    boxed     .default:n   = { true },
6846    boxed     .bool_set:N  = \c__problems_boxed_bool,
6847    unknown   .code:n      = {}
6848  }
6849  \newif\ifsolutions
6850
6851  \ProcessKeysOptions{ problem / pkg }
6852  \bool_if:NTF \c__problems_solutions_bool {
6853    \solutionstrue
6854  }{
6855    \solutionsfalse
6856  }
```

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

```
6857  \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.

```
6858  \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.

```
6859  \def\prob@problem@kw{Problem}
6860  \def\prob@solution@kw{Solution}
6861  \def\prob@hint@kw{Hint}
6862  \def\prob@note@kw{Note}
6863  \def\prob@gnote@kw{Grading}
6864  \def\prob@pt@kw{pt}
6865  \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
6866  \AddToHook{begindocument}{
6867    \ltx@ifpackageloaded{babel}{
6868        \makeatletter
6869        \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6870        \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6871          \input{problem-ngerman.ldf}
6872        }
6873        \clist_if_in:NnT \l_tmpa_clist {finnish}{
6874          \input{problem-finnish.ldf}
6875        }
6876        \clist_if_in:NnT \l_tmpa_clist {french}{
6877          \input{problem-french.ldf}
6878        }
6879        \clist_if_in:NnT \l_tmpa_clist {russian}{
6880          \input{problem-russian.ldf}
6881        }
6882        \makeatother
6883    }{}
6884  }
```

39.2   Problems and Solutions

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

```
6885  \keys_define:nn{ problem / problem }{
6886    id       .str_set_x:N  = \l__problems_prob_id_str,
6887    pts      .tl_set:N      = \l__problems_prob_pts_tl,
6888    min      .tl_set:N      = \l__problems_prob_min_tl,
6889    title    .tl_set:N      = \l__problems_prob_title_tl,
6890    type     .tl_set:N      = \l__problems_prob_type_tl,
6891    refnum   .int_set:N     = \l__problems_prob_refnum_int
6892  }
6893  \cs_new_protected:Nn \__problems_prob_args:n {
```

```
6894    \str_clear:N \l__problems_prob_id_str
6895    \tl_clear:N \l__problems_prob_pts_tl
6896    \tl_clear:N \l__problems_prob_min_tl
6897    \tl_clear:N \l__problems_prob_title_tl
6898    \tl_clear:N \l__problems_prob_type_tl
6899    \int_zero_new:N \l__problems_prob_refnum_int
6900    \keys_set:nn { problem / problem }{ #1 }
6901    \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
6902      \let\l__problems_prob_refnum_int\undefined
6903    }
6904  }
```

Then we set up a counter for problems.

\numberproblemsin

```
6905  \newcounter{problem}
6906  \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.

```
6907  \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

```
6908  \newcommand\prob@number{
6909    \int_if_exist:NTF \l__problems_inclprob_refnum_int {
6910      \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
6911    }{
6912      \int_if_exist:NTF \l__problems_prob_refnum_int {
6913        \prob@label{\int_use:N \l__problems_prob_refnum_int }
6914      }{
6915        \prob@label\theproblem
6916      }
6917    }
6918  }
```

(*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.

```
6919  \newcommand\prob@title[3]{%
6920    \tl_if_exist:NTF \l__problems_inclprob_title_tl {
6921      #2 \l__problems_inclprob_title_tl #3
6922    }{
6923      \tl_if_exist:NTF \l__problems_prob_title_tl {
6924        #2 \l__problems_prob_title_tl #3
6925      }{
6926        #1
6927      }
6928    }
6929  }
```

(*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.

```
6930  \def\prob@heading{
6931    {\prob@problem@kw}\ \prob@number\prob@title{~}{~(}{)\strut}
6932    %\sref@label@id{\prob@problem@kw~\prob@number}{}
6933  }
```

(*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`

```
6934  \newenvironment{sproblem}[1][]{
6935    \__problems_prob_args:n{#1}%\sref@target%
6936    \@in@omtexttrue% we are in a statement (for inline definitions)
6937    \stepcounter{problem}\record@problem
6938    \def\current@section@level{\prob@problem@kw}
6939    \tl_if_exist:NTF \l__problems_inclprob_type_tl {
6940      \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
6941    }{
6942      \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
6943    }
6944    \str_if_exist:NTF \l__problems_inclprob_id_str {
6945      \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
6946    }{
6947      \str_set_eq:NN \sproblemid \l__problems_prob_id_str
6948    }
6949
6950
6951    \clist_set:No \l_tmpa_clist \sproblemtype
6952    \tl_clear:N \l_tmpa_tl
6953    \clist_map_inline:Nn \l_tmpa_clist {
6954      \tl_if_exist:cT {__problems_sproblem_##1_start:}{
6955        \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_start:}}
6956      }
6957    }
6958    \tl_if_empty:NTF \l_tmpa_tl {
6959      \__problems_sproblem_start:
6960    }{
6961      \l_tmpa_tl
6962    }
6963    \stex_ref_new_doc_target:n \sproblemid
6964  }{
6965    \clist_set:No \l_tmpa_clist \sproblemtype
6966    \tl_clear:N \l_tmpa_tl
6967    \clist_map_inline:Nn \l_tmpa_clist {
6968      \tl_if_exist:cT {__problems_sproblem_##1_end:}{
6969        \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_end:}}
6970      }
```

```
6971        }
6972      \tl_if_empty:NTF \l_tmpa_tl {
6973        \__problems_sproblem_end:
6974      }{
6975        \l_tmpa_tl
6976      }
6977
6978
6979      \smallskip
6980    }
6981
6982
6983    \cs_new_protected:Nn \__problems_sproblem_start: {
6984      \par\noindent\textbf\prob@heading\show@pts\show@min\\\ignorespacesandpars
6985    }
6986    \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
6987
6988    \newcommand\stexpatchproblem[3][] {
6989        \str_set:Nx \l_tmpa_str{ #1 }
6990        \str_if_empty:NTF \l_tmpa_str {
6991          \tl_set:Nn \__problems_sproblem_start: { #2 }
6992          \tl_set:Nn \__problems_sproblem_end: { #3 }
6993        }{
6994          \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
6995          \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
6996        }
6997    }
6998
6999
7000    \bool_if:NT \c__problems_boxed_bool {
7001      \surroundwithmdframed{problem}
7002    }
```

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

```
7003    \def\record@problem{
7004      \protected@write\@auxout{}
7005      {
7006        \string\@problem{\prob@number}
7007        {
7008          \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7009            \l__problems_inclprob_pts_tl
7010          }{
7011            \l__problems_prob_pts_tl
7012          }
7013        }%
7014        {
7015          \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7016            \l__problems_inclprob_min_tl
7017          }{
7018            \l__problems_prob_min_tl
7019          }
7020        }
7021      }
7022    }
```

258

(*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).

```
7023 \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.

```
7024 \keys_define:nn { problem / solution }{
7025   id            .str_set_x:N  = \l__problems_solution_id_str ,
7026   for           .tl_set:N     = \l__problems_solution_for_tl ,
7027   height        .dim_set:N    = \l__problems_solution_height_dim ,
7028   creators      .clist_set:N  = \l__problems_solution_creators_clist ,
7029   contributors  .clist_set:N  = \l__problems_solution_contributors_clist ,
7030   srccite       .tl_set:N     = \l__problems_solution_srccite_tl
7031 }
7032 \cs_new_protected:Nn \__problems_solution_args:n {
7033   \str_clear:N \l__problems_solution_id_str
7034   \tl_clear:N \l__problems_solution_for_tl
7035   \tl_clear:N \l__problems_solution_srccite_tl
7036   \clist_clear:N \l__problems_solution_creators_clist
7037   \clist_clear:N \l__problems_solution_contributors_clist
7038   \dim_zero:N \l__problems_solution_height_dim
7039   \keys_set:nn { problem / solution }{ #1 }
7040 }
```

the next step is to define a helper macro that does what is needed to start a solution.

```
7041 \newcommand\@startsolution[1][]{
7042   \__problems_solution_args:n { #1 }
7043   \@in@omtexttrue% we are in a statement.
7044   \bool_if:NF \c__problems_boxed_bool { \hrule }
7045   \smallskip\noindent
7046   {\textbf\prob@solution@kw :\enspace}
7047   \begin{small}
7048   \def\current@section@level{\prob@solution@kw}
7049   \ignorespacesandpars
7050 }
```

\startsolutions for the `\startsolutions` macro we use the `\specialcomment` macro from the `comment` package. Note that we use the `\@startsolution` macro in the start codes, that parses the optional argument.

```
7051 \newcommand\startsolutions{
7052   \specialcomment{solution}{\@startsolution}{
7053     \bool_if:NF \c__problems_boxed_bool {
7054       \hrule\medskip
7055     }
7056     \end{small}%
7057   }
7058   \bool_if:NT \c__problems_boxed_bool {
7059     \surroundwithmdframed{solution}
7060   }
7061 }
```

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

\stopsolutions

```
7062 \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.

```
7063 \ifsolutions
7064   \startsolutions
7065 \else
7066   \stopsolutions
7067 \fi
```

exnote

```
7068 \bool_if:NTF \c__problems_notes_bool {
7069   \newenvironment{exnote}[1][]{
7070     \par\smallskip\hrule\smallskip
7071     \noindent\textbf{\prob@note@kw : }\small
7072   }{
7073     \smallskip\hrule
7074   }
7075 }{
7076   \excludecomment{exnote}
7077 }
```

hint

```
7078 \bool_if:NTF \c__problems_notes_bool {
7079   \newenvironment{hint}[1][]{
7080     \par\smallskip\hrule\smallskip
7081     \noindent\textbf{\prob@hint@kw :~ }\small
7082   }{
7083     \smallskip\hrule
7084   }
7085   \newenvironment{exhint}[1][]{
7086     \par\smallskip\hrule\smallskip
7087     \noindent\textbf{\prob@hint@kw :~ }\small
7088   }{
7089     \smallskip\hrule
7090   }
7091 }{
7092   \excludecomment{hint}
7093   \excludecomment{exhint}
7094 }
```

gnote

```
7095 \bool_if:NTF \c__problems_notes_bool {
7096   \newenvironment{gnote}[1][]{
7097     \par\smallskip\hrule\smallskip
7098     \noindent\textbf{\prob@gnote@kw : }\small
7099   }{
7100     \smallskip\hrule
7101   }
7102 }{
7103   \excludecomment{gnote}
7104 }
```

39.3 Multiple Choice Blocks

mcb [17]

```
7105  \newenvironment{mcb}{
7106    \begin{enumerate}
7107  }{
7108    \end{enumerate}
7109  }
```

we define the keys for the mcc macro

```
7110  \cs_new_protected:Nn \__problems_do_yes_param:Nn {
7111    \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
7112      \bool_set_true:N #1
7113    }{
7114      \bool_set_false:N #1
7115    }
7116  }
7117  \keys_define:nn { problem / mcc }{
7118    id        .str_set_x:N  = \l__problems_mcc_id_str ,
7119    feedback  .tl_set:N     = \l__problems_mcc_feedback_tl ,
7120    T         .default:n    = { true } ,
7121    T         .bool_set:N   = \l__problems_mcc_t_bool ,
7122    F         .default:n    = { true } ,
7123    F         .bool_set:N   = \l__problems_mcc_f_bool ,
7124    Ttext     .code:n       = {
7125      \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
7126    } ,
7127    Ftext     .code:n       = {
7128      \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
7129    }
7130  }
7131  \cs_new_protected:Nn \l__problems_mcc_args:n {
7132    \str_clear:N \l__problems_mcc_id_str
7133    \tl_clear:N \l__problems_mcc_feedback_tl
7134    \bool_set_true:N \l__problems_mcc_t_bool
7135    \bool_set_true:N \l__problems_mcc_f_bool
7136    \bool_set_true:N \l__problems_mcc_Ttext_bool
7137    \bool_set_false:N \l__problems_mcc_Ftext_bool
7138    \keys_set:nn { problem / mcc }{ #1 }
7139  }
```

\mcc

```
7140  \newcommand\mcc[2][]{
7141    \l__problems_mcc_args:n{ #1 }
7142    \item #2
7143    \ifsolutions
7144      \\
7145      \bool_if:NT \l__problems_mcc_t_bool {
7146        % TODO!
7147        % \ifcsstring{mcc@T}{T}{}{\mcc@Ttext}%
7148      }
7149      \bool_if:NT \l__problems_mcc_f_bool {
```

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

```
7150        % TODO!
7151        % \ifcsstring{mcc@F}{F}{}{\mcc@Ftext}%
7152      }
7153      \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
7154        !
7155      }{
7156        \l__problems_mcc_feedback_tl
7157      }
7158    \fi
7159  } %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.

```
7160
7161  \keys_define:nn{ problem / inclproblem }{
7162    id       .str_set_x:N  = \l__problems_inclprob_id_str,
7163    pts      .tl_set:N      = \l__problems_inclprob_pts_tl,
7164    min      .tl_set:N      = \l__problems_inclprob_min_tl,
7165    title    .tl_set:N      = \l__problems_inclprob_title_tl,
7166    refnum   .int_set:N     = \l__problems_inclprob_refnum_int,
7167    type     .tl_set:N      = \l__problems_inclprob_type_tl,
7168    mhrepos  .str_set_x:N  = \l__problems_inclprob_mhrepos_str
7169  }
7170  \cs_new_protected:Nn \__problems_inclprob_args:n {
7171    \str_clear:N \l__problems_prob_id_str
7172    \tl_clear:N \l__problems_inclprob_pts_tl
7173    \tl_clear:N \l__problems_inclprob_min_tl
7174    \tl_clear:N \l__problems_inclprob_title_tl
7175    \tl_clear:N \l__problems_inclprob_type_tl
7176    \int_zero_new:N \l__problems_inclprob_refnum_int
7177    \str_clear:N \l__problems_inclprob_mhrepos_str
7178    \keys_set:nn { problem / inclproblem }{ #1 }
7179    \tl_if_empty:NT \l__problems_inclprob_pts_tl {
7180      \let\l__problems_inclprob_pts_tl\undefined
7181    }
7182    \tl_if_empty:NT \l__problems_inclprob_min_tl {
7183      \let\l__problems_inclprob_min_tl\undefined
7184    }
7185    \tl_if_empty:NT \l__problems_inclprob_title_tl {
7186      \let\l__problems_inclprob_title_tl\undefined
7187    }
7188    \tl_if_empty:NT \l__problems_inclprob_type_tl {
7189      \let\l__problems_inclprob_type_tl\undefined
7190    }
7191    \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
7192      \let\l__problems_inclprob_refnum_int\undefined
7193    }
7194  }
```

```
7195
7196 \cs_new_protected:Nn \__problems_inclprob_clear: {
7197   \let\l__problems_inclprob_id_str\undefined
7198   \let\l__problems_inclprob_pts_tl\undefined
7199   \let\l__problems_inclprob_min_tl\undefined
7200   \let\l__problems_inclprob_title_tl\undefined
7201   \let\l__problems_inclprob_type_tl\undefined
7202   \let\l__problems_inclprob_refnum_int\undefined
7203   \let\l__problems_inclprob_mhrepos_str\undefined
7204 }
7205 \__problems_inclprob_clear:
7206
7207 \newcommand\includeproblem[2][]{
7208   \__problems_inclprob_args:n{ #1 }
7209   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
7210     \input{#2}
7211   }{
7212     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
7213       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
7214     }
7215   }
7216   \__problems_inclprob_clear:
7217 }
```

(*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.

```
7218 \AddToHook{enddocument}{
7219   \bool_if:NT \c__problems_pts_bool {
7220     \message{Total:~\arabic{pts}~points}
7221   }
7222   \bool_if:NT \c__problems_min_bool {
7223     \message{Total:~\arabic{min}~minutes}
7224   }
7225 }
```

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

```
7226 \def\pts#1{
7227   \bool_if:NT \c__problems_pts_bool {
7228     \marginpar{#1~\prob@pt@kw}
7229   }
7230 }
7231 \def\min#1{
7232   \bool_if:NT \c__problems_min_bool {
7233     \marginpar{#1~\prob@min@kw}
7234   }
7235 }
```

\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.

```
7236 \newcounter{pts}
7237 \def\show@pts{
7238   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7239     \bool_if:NT \c__problems_pts_bool {
7240       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
7241       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
7242     }
7243   }{
7244     \tl_if_exist:NT \l__problems_prob_pts_tl {
7245       \bool_if:NT \c__problems_pts_bool {
7246         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
7247         \addtocounter{pts}{\l__problems_prob_pts_tl}
7248       }
7249     }
7250   }
7251 }
```

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

and now the same for the minutes

\show@min

```
7252 \newcounter{min}
7253 \def\show@min{
7254   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7255     \bool_if:NT \c__problems_min_bool {
7256       \marginpar{\l__problems_inclprob_pts_tl\ min}
7257       \addtocounter{min}{\l__problems_inclprob_min_tl}
7258     }
7259   }{
7260     \tl_if_exist:NT \l__problems_prob_min_tl {
7261       \bool_if:NT \c__problems_min_bool {
7262         \marginpar{\l__problems_prob_min_tl\ min}
7263         \addtocounter{min}{\l__problems_prob_min_tl}
7264       }
7265     }
7266   }
7267 }
7268 ⟨/package⟩
```

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

Chapter 40

Implementation: The hwexam Class

The functionality is spread over the `hwexam` class and package. The class provides the `document` environment and pre-loads some convenience packages, whereas the package provides the concrete functionality.

40.1 Class Options

To initialize the `hwexam` class, we declare and process the necessary options by passing them to the respective packages and classes they come from.

```
7269  ⟨@@=hwexam⟩
7270  ⟨∗cls⟩
7271  \ProvidesExplClass{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
7272  \RequirePackage{l3keys2e}
7273  \DeclareOption*{
7274    \PassOptionsToClass{\CurrentOption}{document-structure}
7275    \PassOptionsToPackage{\CurrentOption}{stex}
7276    \PassOptionsToPackage{\CurrentOption}{hwexam}
7277    \PassOptionsToPackage{\CurrentOption}{tikzinput}
7278  }
7279  \ProcessOptions
```

We load `omdoc.cls`, and the desired packages. For the LaTeXML bindings, we make sure the right packages are loaded.

```
7280  \LoadClass{document-structure}
7281  \RequirePackage{stex}
7282  \RequirePackage{hwexam}
7283  \RequirePackage{tikzinput}
7284  \RequirePackage{graphicx}
7285  \RequirePackage{a4wide}
7286  \RequirePackage{amssymb}
7287  \RequirePackage{amstext}
7288  \RequirePackage{amsmath}
```

Finally, we register another keyword for the `document` environment. We give a default assignment type to prevent errors

```
7289  \newcommand\assig@default@type{\hwexam@assignment@kw}
7290  \def\document@hwexamtype{\assig@default@type}
7291  ⟨@@=document_structure⟩
7292  \keys_define:nn { document-structure / document }{
7293  id .str_set_x:N = \c_document_structure_document_id_str,
7294  hwexamtype .tl_set:N = \document@hwexamtype
7295  }
7296  ⟨@@=hwexam⟩
7297  ⟨/cls⟩
```

Chapter 41

Implementation: The hwexam Package

41.1 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.

```
7298 ⟨*package⟩
7299 \ProvidesExplPackage{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
7300 \RequirePackage{l3keys2e}
7301
7302 \newif\iftest\testfalse
7303 \DeclareOption{test}{\testtrue}
7304 \newif\ifmultiple\multiplefalse
7305 \DeclareOption{multiple}{\multipletrue}
7306 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
7307 \ProcessOptions
```

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

```
7308 \RequirePackage{keyval}[1997/11/10]
7309 \RequirePackage{problem}
```

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

```
7310 \newcommand\hwexam@assignment@kw{Assignment}
7311 \newcommand\hwexam@given@kw{Given}
7312 \newcommand\hwexam@due@kw{Due}
7313 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
7314 blank~for~extra~space}
7315 \def\hwexam@minutes@kw{minutes}
7316 \newcommand\correction@probs@kw{prob.}
7317 \newcommand\correction@pts@kw{total}
7318 \newcommand\correction@reached@kw{reached}
7319 \newcommand\correction@sum@kw{Sum}
7320 \newcommand\correction@grade@kw{grade}
7321 \newcommand\correction@forgrading@kw{To~be~used~for~grading,~do~not~write~here}
```

For the other languages, we set up triggers

```
7322 \AddToHook{begindocument}{
7323 \ltx@ifpackageloaded{babel}{
7324 \makeatletter
7325 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7326 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7327   \input{hwexam-ngerman.ldf}
7328 }
7329 \clist_if_in:NnT \l_tmpa_clist {finnish}{
7330   \input{hwexam-finnish.ldf}
7331 }
7332 \clist_if_in:NnT \l_tmpa_clist {french}{
7333   \input{hwexam-french.ldf}
7334 }
7335 \clist_if_in:NnT \l_tmpa_clist {russian}{
7336   \input{hwexam-russian.ldf}
7337 }
7338 \makeatother
7339 }{}
7340 }
7341
```

41.2   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.

```
7342 \newcounter{assignment}
7343 \numberproblemsin{assignment}
7344 \renewcommand\prob@label[1]{\assignment@number.#1}
```

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

```
7345 \keys_define:nn { hwexam / assignment } {
7346 id   .str_set_x:N = \l__hwexam_assign_id_str,
7347 number  .int_set:N  = \l__hwexam_assign_number_int,
7348 title  .tl_set:N  = \l__hwexam_assign_title_tl,
7349 type  .tl_set:N  = \l__hwexam_assign_type_tl,
7350 given .tl_set:N  = \l__hwexam_assign_given_tl,
7351 due .tl_set:N  = \l__hwexam_assign_due_tl,
7352 loadmodules .code:n  = {
7353 \bool_set_true:N \l__hwexam_assign_loadmodules_bool
7354 }
7355 }
7356 \cs_new_protected:Nn \__hwexam_assignment_args:n {
7357 \str_clear:N \l__hwexam_assign_id_str
7358 \int_set:Nn \l__hwexam_assign_number_int {-1}
7359 \tl_clear:N \l__hwexam_assign_title_tl
7360 \tl_clear:N \l__hwexam_assign_type_tl
7361 \tl_clear:N \l__hwexam_assign_given_tl
7362 \tl_clear:N \l__hwexam_assign_due_tl
7363 \bool_set_false:N \l__hwexam_assign_loadmodules_bool
```

```
7364    \keys_set:nn { hwexam / assignment }{ #1 }
7365  }
```

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.

```
7366  \newcommand\given@due[2]{
7367  \bool_lazy_all:nF {
7368  {\tl_if_empty_p:V \l__hwexam_inclassign_given_tl}
7369  {\tl_if_empty_p:V \l__hwexam_assign_given_tl}
7370  {\tl_if_empty_p:V \l__hwexam_inclassign_due_tl}
7371  {\tl_if_empty_p:V \l__hwexam_assign_due_tl}
7372  }{ #1 }
7373
7374  \tl_if_empty:NTF \l__hwexam_inclassign_given_tl {
7375  \tl_if_empty:NF \l__hwexam_assign_given_tl {
7376  \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
7377  }
7378  }{
7379  \hwexam@given@kw\xspace\l__hwexam_inclassign_given_tl
7380  }
7381
7382  \bool_lazy_or:nnF {
7383  \bool_lazy_and_p:nn {
7384  \tl_if_empty_p:V \l__hwexam_inclassign_due_tl
7385  }{
7386  \tl_if_empty_p:V \l__hwexam_assign_due_tl
7387  }
7388  }{
7389  \bool_lazy_and_p:nn {
7390  \tl_if_empty_p:V \l__hwexam_inclassign_due_tl
7391  }{
7392  \tl_if_empty_p:V \l__hwexam_assign_due_tl
7393  }
7394  }{ ,~ }
7395
7396  \tl_if_empty:NTF \l__hwexam_inclassign_due_tl {
7397  \tl_if_empty:NF \l__hwexam_assign_due_tl {
7398  \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
7399  }
7400  }{
7401  \hwexam@due@kw\xspace \l__hwexam_inclassign_due_tl
7402  }
7403
7404  \bool_lazy_all:nF {
7405  { \tl_if_empty_p:V \l__hwexam_inclassign_given_tl }
7406  { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
7407  { \tl_if_empty_p:V \l__hwexam_inclassign_due_tl }
7408  { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
7409  }{ #2 }
7410  }
```

\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.

```
7411 \newcommand\assignment@title[3]{
7412 \tl_if_empty:NTF \l__hwexam_inclassign_title_tl {
7413 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
7414 #1
7415 }{
7416 #2\l__hwexam_assign_title_tl#3
7417 }
7418 }{
7419 #2\l__hwexam_inclassign_title_tl#3
7420 }
7421 }
```

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

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

```
7422 \newcommand\assignment@number{
7423 \int_compare:nNnTF \l__hwexam_inclassign_number_int = {-1} {
7424 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
7425 \arabic{assignment}
7426 } {
7427 \int_use:N \l__hwexam_assign_number_int
7428 }
7429 }{
7430 \int_use:N \l__hwexam_inclassign_number_int
7431 }
7432 }
```

(*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.

```
7433 \newenvironment{assignment}[1][]{
7434 \__hwexam_assignment_args:n { #1 }
7435 %\sref@target
7436 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
7437 \global\stepcounter{assignment}
7438 }{
7439 \global\setcounter{assignment}{\int_use:N\l__hwexam_assign_number_int}
7440 }
7441 \setcounter{problem}{0}
7442 \def\current@section@level{\document@hwexamtype}
7443 %\sref@label@id{\document@hwexamtype \thesection}
7444 \begin{@assignment}
7445 }{
7446 \end{@assignment}
7447 }
```

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

```
7448  \def\ass@title{
7449  \protect\document@hwexamtype~\arabic{assignment}
7450  \assignment@title{}{\;(}{)\;} -- \given@due{}{}
7451  }
7452  \ifmultiple
7453  \newenvironment{@assignment}{
7454  \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
7455  \begin{sfragment}[loadmodules]{\ass@title}
7456  }{
7457  \begin{sfragment}{\ass@title}
7458  }
7459  }{
7460  \end{sfragment}
7461  }
```

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

```
7462  \else
7463  \newenvironment{@assignment}{
7464  \begin{center}\bf
7465  \Large\@title\strut\\
7466  \document@hwexamtype~\arabic{assignment}\assignment@title{\;}{:\;}{\\}
7467  \large\given@due{--\;}{\;--}
7468  \end{center}
7469  }{}
7470  \fi% multiple
```

41.3   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.

```
7471  \keys_define:nn { hwexam / inclassignment } {
7472  %id   .str_set_x:N = \l__hwexam_assign_id_str,
7473  number  .int_set:N = \l__hwexam_inclassign_number_int,
7474  title  .tl_set:N  = \l__hwexam_inclassign_title_tl,
7475  type   .tl_set:N  = \l__hwexam_inclassign_type_tl,
7476  given .tl_set:N  = \l__hwexam_inclassign_given_tl,
7477  due .tl_set:N  = \l__hwexam_inclassign_due_tl,
7478  mhrepos   .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
7479  }
7480  \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
7481  \int_set:Nn \l__hwexam_inclassign_number_int {-1}
7482  \tl_clear:N \l__hwexam_inclassign_title_tl
7483  \tl_clear:N \l__hwexam_inclassign_type_tl
7484  \tl_clear:N \l__hwexam_inclassign_given_tl
7485  \tl_clear:N \l__hwexam_inclassign_due_tl
7486  \str_clear:N \l__hwexam_inclassign_mhrepos_str
7487  \keys_set:nn { hwexam / inclassignment }{ #1 }
7488  }
7489  \__hwexam_inclassignment_args:n {}
7490
7491  \newcommand\inputassignment[2][]{
```

```
7492 \__hwexam_inclassignment_args:n { #1 }
7493 \str_if_empty:NTF \l__hwexam_inclassign_mhrepos_str {
7494 \input{#2}
7495 }{
7496 \stex_in_repository:nn{\l__hwexam_inclassign_mhrepos_str}{
7497 \input{\mhpath{\l__hwexam_inclassign_mhrepos_str}{#2}}}
7498 }
7499 }
7500 \__hwexam_inclassignment_args:n {}
7501 }
7502 \newcommand\includeassignment[2][]{
7503 \newpage
7504 \inputassignment[#1]{#2}
7505 }
```

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

41.4   Typesetting Exams

\quizheading

```
7506 \ExplSyntaxOff
7507 \newcommand\quizheading[1]{%
7508 \def\@tas{#1}%
7509 \large\noindent NAME: \hspace{8cm}   MAILBOX:\\[2ex]%
7510 \ifx\@tas\@empty\else%
7511 \noindent TA:~\@for\@I:=\@tas\do{{\Large$\Box$}\@I\hspace*{1em}}\\[2ex]%
7512 \fi%
7513 }
7514 \ExplSyntaxOn
```

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

\testheading

```
7515
7516 \def\hwexamheader{\input{hwexam-default.header}}
7517
7518 \def\hwexamminutes{
7519 \tl_if_empty:NTF \testheading@duration {
7520 {\testheading@min}~\hwexam@minutes@kw
7521 }{
7522 \testheading@duration
7523 }
7524 }
7525
7526 \keys_define:nn { hwexam / testheading } {
7527 min   .tl_set:N  = \testheading@min,
7528 duration .tl_set:N  = \testheading@duration,
7529 reqpts .tl_set:N  = \testheading@reqpts,
7530 tools .tl_set:N  = \testheading@tools
7531 }
7532 \cs_new_protected:Nn \__hwexam_testheading_args:n {
7533 \tl_clear:N \testheading@min
7534 \tl_clear:N \testheading@duration
```

```
7535   \tl_clear:N \testheading@reqpts
7536   \tl_clear:N \testheading@tools
7537   \keys_set:nn { hwexam / testheading }{ #1 }
7538   }
7539   \newenvironment{testheading}[1][]{
7540   \__hwexam_testheading_args:n{ #1 }
7541   \newcount\check@time\check@time=\testheading@min
7542   \advance\check@time by -\theassignment@totalmin
7543   \newif\if@bonuspoints
7544   \tl_if_empty:NTF \testheading@reqpts {
7545   \@bonuspointsfalse
7546   }{
7547   \newcount\bonus@pts
7548   \bonus@pts=\theassignment@totalpts
7549   \advance\bonus@pts by -\testheading@reqpts
7550   \edef\bonus@pts{\the\bonus@pts}
7551   \@bonuspointstrue
7552   }
7553   \edef\check@time{\the\check@time}
7554
7555   \makeatletter\hwexamheader\makeatother
7556   }{
7557   \newpage
7558   }
```

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

\testspace

```
7559   \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}
```

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

\testnewpage

```
7560   \newcommand\testnewpage{\iftest\newpage\fi}
```

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

\testemptypage

```
7561   \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.

```
7562   ⟨@@=problems⟩
7563   \renewcommand\@problem[3]{
7564   \stepcounter{assignment@probs}
7565   \def\__problemspts{#2}
7566   \ifx\__problemspts\@empty\else
7567   \addtocounter{assignment@totalpts}{#2}
7568   \fi
7569   \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\f
7570   \xdef\correction@probs{\correction@probs & #1}%
7571   \xdef\correction@pts{\correction@pts & #2}
7572   \xdef\correction@reached{\correction@reached &}
```

```
7573  }
7574  ⟨@@=hwexam⟩
```

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

`\correction@table` This macro generates the correction table

```
7575  \newcounter{assignment@probs}
7576  \newcounter{assignment@totalpts}
7577  \newcounter{assignment@totalmin}
7578  \def\correction@probs{\correction@probs@kw}
7579  \def\correction@pts{\correction@pts@kw}
7580  \def\correction@reached{\correction@reached@kw}
7581  \stepcounter{assignment@probs}
7582  \newcommand\correction@table{
7583  \resizebox{\textwidth}{!}{%
7584  \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
7585  &\multicolumn{\theassignment@probs}{c||}%|
7586  {\footnotesize\correction@forgrading@kw} &\\\hline
7587  \correction@probs & \correction@sum@kw & \correction@grade@kw\\\hline
7588  \correction@pts &\theassignment@totalpts & \\\hline
7589  \correction@reached & & \\[.7cm]\hline
7590  \end{tabular}}}
7591  ⟨/package⟩
```

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

41.5   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}
```