The sTeX3 Package *

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

2022-02-11

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 2022-02-11)

i

Contents

Part I
Manual

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:

EdN:1

- **The sTeX-Package** available here[1]. Note, that the CTAN repository for LaTeX packages may contain outdated versions of the sTeX package, so make sure, that your `TEXMF` system variable is configured such that the packages available in the linked repository are prioritized over potential default packages that come with your TeX distribution.

EdN:2

- **The Mmt System** available here[2]. We recommend following the setup routine documented here.

  Following the setup routine (Step 3) will entail designating a `MathHub`-directory on your local file system, where the Mmt system will look for sTeX/Mmt content archives.

- 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 chapter 4).

- **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.

---

[1] EdNote: For now, we require the `latex3-branch`
[2] EdNote: For now, we require the `sTeX-branch`, requiring manually compiling the MMT sources

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

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.

The document we will consider is the following:

```
\documentclass{article}
\usepackage{stex}
\usepackage{xcolor}
\def\compemph#1{\textcolor{blue}{#1}}

\begin{document}
  \usemodule[smglom/calculus]{series}
  \usemodule[smglom/arithmetics]{realarith}

  The \symref{series}{series} $\infinitesum{n}{1}{
    \realdivide[frac]{1}{
      \realpower{2}{n}
    }
  }$ \symref{converges}{converges} towards $1$.

\end{document}
```

Compiling this document with `pdflatex` should yield the output

> The **series** $\sum_{n=1}^{\infty} \frac{1}{2^n}$ **converges** towards 1.

Note that the $\sum$ and ∞-symbols are highlighted in blue, and the words "series" and "converges" in bold. This signifies that these words and symbols reference sTₑX *symbols* formally declared somewhere; associating their *presentation* in the document with their (formal) definition - i.e. their semantics. The precise way in which they are highlighted (if at all) can of course be customized (see [3]).

EdN:3

`\usemodule` The command `\usemodule[some/archive]{modulename}` finds some module in the appropriate archive – in the first case (`\usemodule[smglom/calculus]{series}`), sTₑX looks for the archive `smglom/calculus` in our local MathHub-directory (see chapter 4), and in its source-folder for a file `series.tex`. Since no such file exists, and by default the document is assumed to be in *english*, it picks the file `series.en.tex`, and indeed, in here we find a statement `\begin{module}{series}`.

sTₑX now reads this file and makes all semantic macros therein available to use, along with all its dependencies. This enables the usage of `\infinitesum` later on.

Analogously, `\usemodule[smglom/arithmetics]{realarith}` opens the file `realarith.en.tex` in the `.../smglom/arithmetics/source`-folder and makes its contents available, e.g. `\realdivide` and `\realpower`.

---

[3]EᴅNᴏᴛᴇ: somewhere later

\symref
\symname

The command `\symref{symbolname}{text}` marks the `text` in the second argument as representing the `symbolname` in the first argument – which is why the word "series" is set in boldface. In the pdf, this is all that happens. In the `xhtml` (which we will investigate shortly) however, we will note that the word "series" is now annotated with the full URI of the symbol denoting the *mathematical concept of a series*. In other words, the word is associated with an unambiguous semantics.

Notably, in both cases above (*series* and *converges*) the text that *references* the symbol and the name of the symbol are identical. Since this occurs quite often, the shorthand `\symname{converges}` would have worked as well, where `\symname{foo-bar}` behaves exactly like `\symref{foo-bar}{foo bar}` - i.e. the text is simply the name of the symbol with "`-`" replaced by a space.

\importmodule

If you investigated the contents of the imported modules (`realarith` and `series`) more closely, you'll note that none of them contain a symbol "converges". Yet, we can use `\symref` to refer to "converges". That is because the symbol `converges` is found in `smglom/calculus/source/sequenceConvergence.en.tex`, and `series.en.tex` contains the line `\importmodule{sequenceConvergence}`. The `\importmodule`-statement makes the module referenced available to all documents that include the current module. As such, a "current module" has to exist for `\importmodule` to work, which is why the command is only allowed within a `module`-environment.

TODO explain `xhtml` conversion, MMT compilation (requires an archive...?).

Chapter 3

Using Semantic Macros

TODO

Chapter 4

sTEX Archives

4.1 The Local MathHub-Directory

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

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

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

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

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

4.2 The Structure of sTEX Archives

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

Each such archive needs two subdirectories:

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

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

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

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

4.3   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: 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 Mᴍᴛ can pick up on them to resolve dependencies, e.g. for `lmh install`.

Chapter 5

Creating New Modules and Symbols

5.1 Advanced Structuring Mechanisms

Given modules:

> **Example 1**
>
> ```
> \begin{module}{magma}
> \symdef{universe}{\comp{\mathcal U}}
> \symdef[args=2,op=\circ]{operation}{#1 \comp\circ #2}
> \end{module}
> \begin{module}{monoid}
> \importmodule{magma}
> \symdef{unit}{\comp e}
> \end{module}
> \begin{module}{group}
> \importmodule{monoid}
> \symdef[args=1]{inverse}{{#1}^{\comp{-1}}}
> \end{module}
> ```
>
> | Module 5.1.1[magma] |
> | Module 5.1.2[monoid] |
> | Module 5.1.3[group] |

.

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

Example 2

```
 \begin{module}{ring}
\begin{copymodule}{group}{addition}
\renamedecl[name=universe]{universe}{runiverse}
\renamedecl[name=plus]{operation}{rplus}
\renamedecl[name=zero]{unit}{rzero}
\renamedecl[name=uminus]{inverse}{ruminus}
\end{copymodule}
\notation[plus,op=+,prec=60]{rplus}{#1 \comp+ #2}
\notation[zero]{rzero}{\comp0}
\notation[uminus,op=-]{ruminus}{\comp- #1}
\begin{copymodule}{monoid}{multiplication}
\assign{universe}{\runiverse}
\renamedecl[name=times]{operation}{rtimes}
\renamedecl[name=one]{unit}{rone}
\end{copymodule}
\notation[cdot,op=\cdot,prec=50]{rtimes}{#1 \comp\cdot #2}
\notation[one]{rone}{\comp1}

Test: $\rtimes a{\rplus c{\rtimes de}}$
\end{module}
```

Module 5.1.4[ring]
    Test: $a \cdot (c + d \cdot e)$

.

TODO: explain donotclone

Example 3

```
 \begin{module}{int}
\symdef{Integers}{\comp{\mathbb Z}}
\symdef[args=2,op=+]{plus}{#1 \comp+ #2}
\symdef{zero}{\comp0}
\symdef[args=1,op=-]{uminus}{\comp-#1}

\begin{interpretmodule}{group}{intisgroup}
\assign{universe}{\Integers}
\assign{operation}{\plus!}
\assign{unit}{\zero}
\assign{inverse}{\uminus!}
\end{interpretmodule}
\end{module}
```

Module 5.1.5[int]

.

5.2   Primitive Symbols (The sTeX Metatheory)

Chapter 6

sTeX Statements (Definitions, Theorems, Examples, ...)

Chapter 7

Additional Packages

7.1   Modular Document Structuring

7.2   Slides and Course Notes

7.3   Homework, Problems and Exams

Chapter 8

Stuff

8.1 Modules

Both print this ST_EX logo.

8.1.1 Semantic Macros and Notations

Semantic macros invoke a formally declared symbol.

To declare a symbol (in a module), we use `\symdecl`, which takes as argument the name of the corresponding semantic macro, e.g. `\symdecl{foo}` introduces the macro `\foo`. Additionally, `\symdecl` takes several options, the most important one being its arity. `foo` as declared above yields a *constant* symbol. To introduce an *operator* which takes arguments, we have to specify which arguments it takes.

For example, to introduce binary multiplication, we can do `\symdecl[args=2]{mult}`. We can then supply the semantic macro with arbitrarily many notations, such as `\notation{mult}{#1 #2}`.

> **Example 4**
>
> ```
> \symdecl[args=2]{mult}
> \notation{mult}{#1 #2}
> $\mult{a}{b}$
> ```
>
> ---
> $a\,b$
> ---

.

Since usually, a freshly introduced symbol also comes with a notation from the start, the `\symdef` command combines `\symdecl` and `\notation`. So instead of the above, we could have also written

$$\symdef[args=2]{mult}{#1 #2}$$

Adding more notations like `\notation[cdot]{mult}{#1 \comp{\cdot} #2}` or `\notation[times]{mult}{#1 \comp{\times} #2}` allows us to write `$\mult[cdot]{a}{b}$` and `$\mult[times]{a}{b}$`:

Example 5

```
\notation[cdot]{mult}{#1 \comp{\cdot} #2}
\notation[times]{mult}{#1 \comp{\times} #2}
$\mult[cdot]{a}{b}$ and $\mult[times]{a}{b}$
```

$a \cdot b$ and $a \times b$

.

Not using an explicit option with a semantic macro yields the first declared notation, unless changed[4].

Outside of math mode, or by using the starred variant `\foo*`, allows to provide a custom notation, where notational (or textual) components can be given explicitly in square brackets.

Example 6

```
$\mult*{a}[\comp{\ast}]{b}$ is the
\mult[\comp{product of} ]{$a$}[ \comp{and} ]{$b$}
```

$a * b$ is the product of a and b

.

In custom mode, prefixing an argument with a star will not print that argument, but still export it to OMDoc:

Example 7

```
\mult[\comp{Multiplying}]*{$\mult{a}{b}$}[ again by ]{$b$} yields...
```

Multiplying again by b yields...

˙The syntax `*[⟨int⟩]` allows switching the order of arguments. For example, given a 2-ary semantic macro `\forevery` with exemplary notation `\forall #1. #2`, we can write

Example 8

```
\symdecl[args=2]{forevery}
\forevery*[2]{The proposition $P$}[ \comp{holds for every} ]*[1]{$x\in A$}
```

The proposition P holds for every $x \in A$

.

When using `*[n]`, after reading the provided (nth) argument, the "argument counter" automatically continues where we left off, so the `*[1]` in the above example can be omitted.

---

[4]EDNOTE: TODO

For a macro with arity > 0, we can refer to the operator *itself* semantically by suffixing the semantic macro with an exclamation point ! in either text or math mode. For that reason \notation (and thus \symdef) take an additional optional argument op=, which allows to assign a notation for the operator itself. e.g.

Example 9

```
\symdef[args=2,op={+}]{add}{#1 \comp+ #2}
The operator $\add!$ adds two elements, as in $\add ab$.
```

The operator $+$ adds two elements, as in $a + b$.

.

* is composable with ! for custom notations, as in:

Example 10

```
\mult![\comp{Multiplication}] (denoted by $\mult*![\comp\cdot]$) is defined by...
```

Multiplication (denoted by ·) is defined by...

.

The macro \comp as used everywhere above is responsible for highlighting, linking, and tooltips, and should be wrapped around the notation (or text) components that should be treated accordingly. While it is attractive to just wrap a whole notation, this would also wrap around e.g. the arguments themselves, so instead, the user is tasked with marking the notation components themself.

The precise behaviour of \comp is governed by the macro \@comp, which takes two arguments: The tex code of the text (unexpanded) to highlight, and the URI of the current symbol. \@comp can be safely redefined to customize the behaviour.

The starred variant \symdecl*{foo} does not introduce a semantic macro, but still declares a corresponding symbol. foo (like any other symbol, for that matter) can then be accessed via \STEXsymbol{foo} or (if foo was declared in a module Foo) via \STEXModule{Foo}?{foo}.

both \STEXsymbol and \STEXModule take any arbitrary ending segment of a full URI to determine which symbol or module is meant. e.g. \STEXsymbol{Foo?foo} is also valid, as are e.g. \STEXModule{path?Foo}?{foo} or \STEXsymbol{path?Foo?foo}

There's also a convient shortcut \symref{?foo}{some text} for \STEXsymbol{?foo}![some text]

Other Argument Types

So far, we have stated the arity of a semantic macro directly. This works if we only have "normal" (or more precisely: i-type) arguments. To make use of other argument types, instead of providing the arity numerically, we can provide it as a sequence of characters representing the argument types – e.g. instead of writing args=2, we can equivalently write args=ii, indicating that the macro takes two i-type arguments.

Besides i-type arguments, sTEX has two other types, which we will discuss now.

The first are *binding* (b-type) arguments, representing variables that are *bound* by the operator. This is the case for example in the above \forevery-macro: The first

15

argument is not actually an argument that the `forevery` "function" is "applied" to; rather, the first argument is a new variable (e.g. x) that is *bound* in the subsequent argument. More accurately, the macro should therefore have been implemented thusly:

```
\symdef[args=bi]{forevery}{\forall #1.\; #2}
```

`b`-type arguments are indistinguishable from `i`-type arguments within sTeX, but are treated very differently in OMDoc and by Mmt. More interesting *within* sTeX are `a`-type arguments, which represent (associative) arguments of flexible arity, which are provided as comma-separated lists. This allows e.g. better representing the `\mult`-macro above:

Example 11

```
\symdef[args=a]{mult}{#1}{#1 \comp\cdot #2}
$\mult{a,b,c,{d^e},f}$
```

$a \cdot b \cdot c \cdot d^e \cdot f$

˙As the example above shows, notations get a little more complicated for associative arguments. For every `a`-type argument, the `\notation`-macro takes an additional argument that declares how individual entries in an `a`-type argument list are aggregated. The first notation argument then describes how the aggregated expression is combined into the full representation.

For a more interesting example, consider a flexary operator for ordered sequences in ordered set, that taking arguments `{a,b,c}` and `\mathbb{R}` prints $a \leq b \leq c \in \mathbb{R}$. This operator takes two arguments (an `a`-type argument and an `i`-type argument), aggregates the individuals of the associative argument using `\leq`, and combines the result with `\in` and the second argument thusly:

Example 12

```
\symdef[args=ai]{numseq}{#1 \comp\in #2}{#1 \comp\leq #2}
$\numseq{a,b,c}{\mathbb R}$
```

$a \leq b \leq c \in \mathbb{R}$

.

Finally, `B`-type arguments combine the functionalities of `a` and `b`, i.e. they represent flexary binding operator arguments.
5 6

Precedences

Every notation has an (upwards) *operator precedence* and for each argument a (downwards) *argument precedence* used for automated bracketing. For example, a notation for a binary operator `\foo` could be declared like this:

```
\notation[prec=200;500x600]{foo}{#1 \comp{+} #2}
```

---

[5]EdNote: what about e.g. \int _x\int _y\int _z f dx dy dz?
[6]EdNote: "decompose" a-type arguments into fixed-arity operators?

assigning an operator precedence of 200, an argument precedence of 500 for the first argument, and an argument precedence of 600 for the second argument.

SₜₑX insert brackets thusly: Upon encountering a semantic macro (such as \foo), its operator precedence (e.g. 200) is compared to the current downwards precedence (initially \neginfprec). If the operator precedence is *larger* than the current downwards precedence, parentheses are inserted around the semantic macro.

Notations for symbols of arity 0 have a default precedence of \infprec, i.e. by default, parentheses are never inserted around constants. Notations for symbols with arity > 0 have a default operator precedence of 0. If no argument precedences are explicitly provided, then by default they are equal to the operator precedence.

Consequently, if some operator A should bind stronger than some operator B, then As operator precedence should be smaller than Bs argument precedences.

For example:

Example 13

```
\notation[prec=100]{plus}{#1 \comp{+} #2}
\notation[prec=50]{times}{#1 \comp{\cdot} #2}
$\plus{a}{\times{b}{c}}$ and $\times{a}{\plus{b}{c}}$
```

$a + b \cdot c$ and $a \cdot (b + c)$

.

8.1.2 Archives and Imports

Namespaces

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

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

- If \begin{module}{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[1].

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.

---

[1]which is internally attached to the module name instead, but a user need not worry about that.

Paths in Import-Statements

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

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

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

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

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

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

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

  Since this is less compatible with a modular development, using full URIs directly is discouraged.

Part II
Documentation

Chapter 9

sTeX-Basics

Both the sTeX package and class offer the following package options:

debug (⟨*log-prefix*⟩∗) Logs debugging information with the given prefixes to the terminal, or all if `all` is given.

showmods (⟨*boolean*⟩) Shows explicit module information at the document margins.

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

mathhub (⟨*directory*⟩) MathHub folder to search for repositories.

sms (⟨*boolean*⟩) use *persisted* mode (see ???).

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

9.1 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*⟩.

`\stex_add_to_sms:n`

Adds the provided code to the `.sms`-file of the document.

`\if@latexml`
`\latexml_if_p:`
`\latexml_if:T`
`\latexml_if:F`
`\latexml_if:TF`

LaTeX2e and LaTeX3 conditionals for LaTeXML.

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

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

`\stex_annotate_invisible:n` adds the attributes

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

`\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⟩}`.

`\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`.

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

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

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

`\MSC`	`\MSC{⟨msc⟩}`

Designates the *math subject classifier* of the current module / file.

Chapter 10

sTEX-MathHub

Code related to managing and using MathHub repositories, files, paths and related hooks and methods.

10.1 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.

10.1.1 Files, Paths, URIs

`\stex_path_from_string:Nn`
`\stex_path_from_string:(NV|cn|cV)`

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

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

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

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

`\stex_path_canonicalize:N`

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

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

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

---

`\c_stex_pwd_seq`
`\c_stex_pwd_str`
`\c_stex_mainfile_seq`
`\c_stex_mainfile_str`

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

`\g_stex_currentfile_seq`

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

Test 1

```
\ExplSyntaxOn
\def\cpath@print#1{
\stex_path_from_string:Nn \l_tmpb_seq { #1 }
\stex_path_to_string:NN \l_tmpb_seq \l_tmpa_str
\str_use:N \l_tmpa_str
}
\ExplSyntaxOff
\begin{center}
\begin{tabular}{|l|l|l|}\hline
path & canonicalized path & expected\\\hline
aaa & \cpath@print{aaa} & aaa \\
../../aaa & \cpath@print{../../aaa} & ../../aaa\\
aaa/bbb & \cpath@print{aaa/bbb} & aaa/bbb \\
aaa/.. & \cpath@print{aaa/..} &\\
../../aaa/bbb & \cpath@print{../../aaa/bbb} & ../../aaa/bbb\\
../aaa/../bbb & \cpath@print{../aaa/../bbb} & ../bbb \\
../aaa/bbb & \cpath@print{../aaa/bbb} & ../aaa/bbb\\
aaa/bbb/../ddd & \cpath@print{aaa/bbb/../ddd} & aaa/ddd\\
aaa/bbb/./ddd & \cpath@print{aaa/bbb/./ddd} & aaa/bbb/ddd\\
./ & \cpath@print{./} & \\
aaa/bbb/../.. & \cpath@print{aaa/bbb/../..} & \\\hline
\end{tabular}
\end{center}
```

path	canonicalized path	expected
aaa	aaa	aaa
../../aaa	../../aaa	../../aaa
aaa/bbb	aaa/bbb	aaa/bbb
aaa/..		
../../aaa/bbb	../../aaa/bbb	../../aaa/bbb
../aaa/../bbb	../bbb	../bbb
../aaa/bbb	../aaa/bbb	../aaa/bbb
aaa/bbb/../ddd	aaa/ddd	aaa/ddd
aaa/bbb/./ddd	aaa/bbb/ddd	aaa/bbb/ddd
./		
aaa/bbb/../..		

.

10.1.2 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 fields `id`, `ns` (namespace), `narr` (narrative namespace; currently not in use) and `deps` (dependencies; 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*⟩}.

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

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

`\inputref`
`\inputref:nn` `\inputref[⟨archive-ID⟩]{⟨filename⟩}`

`\inputs` the file ⟨*filename*⟩ in repository ⟨*archive-ID*⟩.

`\libinput` `\libinput{⟨filename⟩}`

Inputs ⟨*filename*⟩`.tex` from the `lib` folders in the current archive and the `meta-inf`-archive of the current archive group (if existent). Throws an error if no file by that name exists in either folder, includes both if both exist.

Test 2

```
\ExplSyntaxOn
\stex_require_repository:n { Foo/Bar }
id:~\prop_item:cn {c_stex_mathhub_Foo/Bar_manifest_prop} {id}\ \\
narr:~\prop_item:cn {c_stex_mathhub_Foo/Bar_manifest_prop} {narr}\ \\
ns:~\prop_item:cn {c_stex_mathhub_Foo/Bar_manifest_prop} {ns}\ \\
deps:~\prop_item:cn {c_stex_mathhub_Foo/Bar_manifest_prop} {deps}\ \\
\stex_require_repository:n { Bar/Foo }
\ExplSyntaxOff
```

```
id: Foo/Bar
narr:
ns: http://mathhub.info/tests/Foo/Bar
deps:
```

.

Chapter 11

sTeX-References

Code related to links and cross-references

11.1 Macros and Environments

Chapter 12

sTeX-Modules

Code related to Modules

12.1 Macros and Environments

`\l_stex_current_module_str`

All information of a module is stored as a property list. `\l_stex_current_module_str` always points to the current module (if existent).

Most importantly, the `content`-field stores all the code to execute on activation; i.e. when this module is being included.

Additionally, it stores:

- The *name* in field `name`,

- the *namespace* in field `ns`,

- this module's *language* in field `lang`,

- if a language module that translates some other modules, the *original* module in field `sig` (for signature),

- the *metatheory* in field `meta`,

- the URIs of all *imported modules* in field `imports`,

- the names of all *declarations* in field `constants`,

- the *file* this module was declared in in field `file`,

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

`\g_stex_module_files_prop`
`\g_stex_modules_in_file_seq`

A property list mapping file paths to the lists of all modules declared therein. `\g_stex_-modules_in_file_seq` always points to the current file(-stream - `\inputs` are considered the same file).

`\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 `content` field of the current module.

`\stex_add_constant_to_current_module:n`

Adds the declaration with the provided name to the `constants` field of the current module.

`\stex_add_import_to_current_module:n`

Adds the module with the provided full URI to the `imports` field of the current module.

`\stex_modules_compute_namespace:nN`   `\stex_modules_compute_namespace:nN`
{⟨*namespace*⟩} {⟨`path`⟩}

Computes the namespace for file ⟨*path*⟩ in repository with namespace ⟨*namespace*⟩ as follows:

If the file is `.../source/sub/file.tex` and the namespace `http://some.namespace/foo`, then the namespace of is `http://some.namespace/foo/sub/file`.

`\stex_modules_current_namespace:`

Computes the current namespace

Test 3

```
\ExplSyntaxOn
\stex_modules_current_namespace:
Namespace~1:\\ \l_stex_modules_ns_str \\
Faking~a~repository:\\
\stex_set_current_repository:n{Foo/Bar}
\seq_pop_right:NN \g_stex_currentfile_seq \testtemp
\edef\testtempb{\detokenize{source}}
\exp_args:NNo \seq_put_right:Nn \g_stex_currentfile_seq { \testtempb }
\edef\testtempb{\detokenize{test}}
\exp_args:NNo \seq_put_right:Nn \g_stex_currentfile_seq { \testtempb }
\exp_args:NNo \seq_put_right:Nn \g_stex_currentfile_seq { \testtemp }
\stex_modules_current_namespace:
Namespace~2:\\ \l_stex_modules_ns_str
\ExplSyntaxOff
```

.

12.1.1 The `module`-environment

module

\begin{module}[⟨*options*⟩]{⟨*name*⟩}
Opens a new module with name ⟨*name*⟩.
TODO document options.

\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.

\stex_modules_heading:

Takes care of the module header, if the `showmods` package option is true. This macro can be overridden for customization.

@module

\begin{@module}[⟨*options*⟩]{⟨*name*⟩}
Core functionality of the `module`-environment without a header.

Test 4

```
\ExplSyntaxOn
\stex__set__current__repository:n {Foo/Bar}
\seq_pop_right:NN \g_stex_currentfile_seq \l_tmpa_tl
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{tests} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{Foo} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{Bar} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{source} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{Foo.tex} }
\begin{@module}{Foo}
Module~path:~
\prop_item:cn {c_stex_module_\l_stex_current_module_str _prop} { ns }?
\prop_item:cn {c_stex_module_\l_stex_current_module_str _prop} { name }\\
Language:~\prop_item:cn {c_stex_module_\l_stex_current_module_str _prop} { lang }\\
Signature:~\prop_item:cn {c_stex_module_\l_stex_current_module_str _prop} { sig }\\
Metatheory:~\prop_item:cn {c_stex_module_\l_stex_current_module_str _prop} { meta }\\
\end{@module}
\ExplSyntaxOff
```

```
Module path: http://mathhub.info/tests/Foo/Bar?Foo
Language:
Signature:
Metatheory:
```

.

Test 5

```
 \ExplSyntaxOn
\stex_set_current_repository:n {Foo/Bar}
\stex_debug:nn{modules}{Test:~\stex_path_to_string:N \g_stex_currentfile_seq }
\seq_pop_right:NN \g_stex_currentfile_seq \l_tmpa_tl
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{tests} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{Foo} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{Bar} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{source} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{Foo.tex} }
\stex_debug:nn{modules}{Test:~\stex_path_to_string:N \g_stex_currentfile_seq }
\begin{module}[title=Foo Bar]{Bar}
Module~path:~
\prop_item:cn {c_stex_module_\l_stex_current_module_str _prop} { ns }?
\prop_item:cn {c_stex_module_\l_stex_current_module_str _prop} { name }\\
Language:~\prop_item:cn {c_stex_module_\l_stex_current_module_str _prop} { lang }\\
Signature:~\prop_item:cn {c_stex_module_\l_stex_current_module_str _prop} { sig }\\
Metatheory:~\prop_item:cn {c_stex_module_\l_stex_current_module_str _prop} { meta }\\
\end{module}
\ExplSyntaxOff
```

> **Module** 12.1.1[Bar]   (FooBar)
>   Module path: http://mathhub.info/tests/Foo/Bar/Foo?Bar
> Language:
> Signature:
> Metatheory:

.

\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.

Test 6

```
 \begin{module}{STEXModuleTest1}
\symdecl{foo}
\end{module}
\begin{module}{STEXModuleTest2}
\importmodule{STEXModuleTest1}
\symdecl{foo}
\end{module}
\begin{module}{STEXModuleTest3}
\importmodule{STEXModuleTest2}
\symdecl{foo}
\STEXModule{STEXModuleTest1}!\teststring
\teststring\\
\STEXModule{STEXModuleTest2}!\teststring
\teststring\\
\STEXModule{STEXModuleTest3}!\teststring
\teststring\\
\STEXModule{STEXModuleTest1}?{foo}[\comp{foo1}]\\
\STEXModule{STEXModuleTest2}?{foo}[\comp{foo2}]\\
\STEXModule{STEXModuleTest3}?{foo}[\comp{foo3}]\\
\end{module}
```

> **Module** 12.1.2[STEXModuleTest1]

> **Module** 12.1.3[STEXModuleTest2]

> **Module** 12.1.4[STEXModuleTest3]
> file://Users/kohlhase/localmh/ext/sTeX/doc/stextest?STEXModuleTest1
> file://Users/kohlhase/localmh/ext/sTeX/doc/stextest?STEXModuleTest2
> file://Users/kohlhase/localmh/ext/sTeX/doc/stextest?STEXModuleTest3
> foo1
> foo2
> foo3

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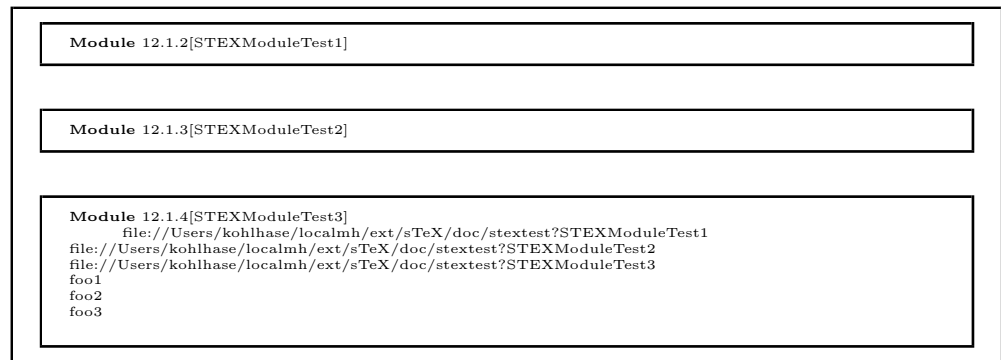
`\stex_activate_module:n` Activate the module with the provided URI; i.e. executes all macro code of the module's `content`-field (does nothing if the module is already activated in the current context) and adds the module to `\l_stex_all_modules_seq`.

Chapter 13

sTeX-Module Inheritance

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

13.1   Macros and Environments

13.1.1   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:

`\g_stex_smsmode_allowedmacros_tl`

Macros that are executed as is; i.e. with the category code scheme used in SMS mode.

`\g_stex_smsmode_allowedmacros_escape_tl`

Macros that are executed with the category codes restored.

Importantly, these macros need to call `\stex_smsmode_set_codes:` after reading all arguments. Note, that `\stex_smsmode_set_codes:` takes care of checking whether we are in SMS mode in the first place, so calling this function eagerly is unproblematic.

`\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_set_codes:` should be called at the end of the `\begin`-code. Since `\end`-statements take no arguments anyway, those are called with the SMS mode category code scheme active.

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

Tests whether SMS mode is currently active.

`\stex_smsmode_set_codes:`

Sets the current category code scheme to that of the SMS mode, if SMS mode is currently active and if necessary.

This method should be called at the end of every macro or `\begin` environment code that are allowed in SMS mode.

`\stex_in_smsmode:nn`

`\stex_in_smsmode:nn {⟨name⟩} {⟨code⟩}`

Executes ⟨*code*⟩ in SMS mode. ⟨*name*⟩ can be arbitrary, but should be distinct, since it allows for nesting `\stex_in_smsmode:nn` without spuriously terminating SMS mode.

Test 7

```
 \immediate\openout\testfile=./tests/sometest.tex
\immediate\write\testfile{\detokenize{\this is \a test}^^J}
\immediate\write\testfile{\detokenize{this \is a \test}}
\immediate\closeout\testfile
\ExplSyntaxOn
\stex_in_smsmode:nn { foo } {
\input{tests/sometest.tex}
}
\ExplSyntaxOff
```

.

13.1.2 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`.

Test 8

```
 \begin{module}{Foo}
\symdecl[name=foo, args=3]{bar}
\symdecl[args=bai]{foobar}
Meaning:~\present\bar\\
\end{module}
Meaning:~\present\bar\\
\begin{module}{Importtest}
\importmodule{Foo}
Meaning:~\present\bar\\
\end{module}
\begin{module}{Importtest2}
\importmodule{Importtest}
Meaning:~\present\bar\\
\end{module}
```

> **Module 13.1.1[Foo]**
> Meaning: »macro:->\stex_invoke_symbol:n {file://Users/kohlhase/localmh/ext/sTeX/doc/stextest?Foo?foo}«

Meaning: »macro:->\protect \bar «

> **Module 13.1.2[Importtest]**
> Meaning: »macro:->\stex_invoke_symbol:n {file://Users/kohlhase/localmh/ext/sTeX/doc/stextest?Foo?foo}«

> **Module 13.1.3[Importtest2]**
> Meaning: »macro:->\stex_invoke_symbol:n {file://Users/kohlhase/localmh/ext/sTeX/doc/stextest?Foo?foo}«

`\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

Test 9

```
 \begin{module}{UseTest1}
\symdecl{foo}
\end{module}
\begin{module}{UseTest2}
\usemodule{UseTest1}
\symdecl{bar}
Meaning:~\present\foo\\
\end{module}
\begin{module}{UseTest3}
\importmodule{UseTest2}
Meaning:~\present\foo\\
Meaning:~\present\bar\\

All modules: \ExplSyntaxOn
\seq_use:Nn \l_stex_all_modules_seq {,~} \\
All~symbols:~
\seq_use:Nn \l_stex_all_symbols_seq {,~}
\ExplSyntaxOff
\end{module}
```

Module 13.1.4[UseTest1]

Module 13.1.5[UseTest2]
          Meaning: »macro:->\stex_invoke_symbol:n {file://Users/kohlhase/localmh/ext/sTeX/doc/stextest?UseTest1?foo}«

Module 13.1.6[UseTest3]
          Meaning: »undefined«
Meaning: »macro:->\stex_invoke_symbol:n {file://Users/kohlhase/localmh/ext/sTeX/doc/stextest?UseTest2?bar}«

          All modules: http://mathhub.info/sTeX?Metatheory, file://Users/kohlhase/localmh/ext/sTeX/doc/stextest?UseTest3,
file://Users/kohlhase/localmh/ext/sTeX/doc/stextest?UseTest2
All symbols: http://mathhub.info/sTeX?Metatheory?isa, http://mathhub.info/sTeX?Metatheory?bind, http://mathhub.info/sTeX?Metatheo
http://mathhub.info/sTeX?Metatheory?fromto, http://mathhub.info/sTeX?Metatheory?apply, http://mathhub.info/sTeX?Metatheory?collec
http://mathhub.info/sTeX?Metatheory?seqtype, http://mathhub.info/sTeX?Metatheory?sequence-index, http://mathhub.info/sTeX?Metath
http://mathhub.info/sTeX?Metatheory?aseqfromto, http://mathhub.info/sTeX?Metatheory?aseqfromtovia, http://mathhub.info/sTeX?Meta
http://mathhub.info/sTeX?Metatheory?module-type, http://mathhub.info/sTeX?Metatheory?mathematical-structure,
http://mathhub.info/sTeX?Metatheory?isa, http://mathhub.info/sTeX?Metatheory?bind, http://mathhub.info/sTeX?Metatheory?dummyvar
http://mathhub.info/sTeX?Metatheory?fromto, http://mathhub.info/sTeX?Metatheory?apply, http://mathhub.info/sTeX?Metatheory?collec
http://mathhub.info/sTeX?Metatheory?seqtype, http://mathhub.info/sTeX?Metatheory?sequence-index, http://mathhub.info/sTeX?Metath
http://mathhub.info/sTeX?Metatheory?aseqfromto, http://mathhub.info/sTeX?Metatheory?aseqfromtovia, http://mathhub.info/sTeX?Meta
http://mathhub.info/sTeX?Metatheory?module-type, http://mathhub.info/sTeX?Metatheory?mathematical-structure,
file://Users/kohlhase/localmh/ext/sTeX/doc/stextest?UseTest2?bar

.

Test 10

```
 Circular dependencies:
\begin{module}{CircDep1}
\importmodule[Foo/Bar]{circular1?Circular1}
\importmodule[Bar/Foo]{circular2?Circular2}
\present\fooA\\
\present\fooB
\end{module}
```

> Circular dependencies:
>
> **Module** 13.1.7[CircDep1]
> »macro:->\stex_invoke_symbol:n {http://mathhub.info/tests/Foo/Bar/circular1?Circular1?fooA}«
> »macro:->\stex_invoke_symbol:n {http://mathhub.info/tests/Bar/Foo//circular2?Circular2?fooB}«

.

\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.

\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 content-field.

Chapter 14

sTeX-Symbols

Code related to symbol declarations and notations

14.1 Macros and Environments

`\symdecl`

`\symdecl[⟨args⟩]{⟨macroname⟩}`

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. 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 $0 \le n \le 9$, or a (more precise) sequence of the following characters:

  i a "normal" argument, e.g. `\symdecl[args=ii]{plus}` 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[args=a]{plus}` 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[args=bi]{forall}` 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),

Test 11

```
 \begin{module}{SymdeclTest}
\symdecl[name=foo, args=3]{bar}
\symdecl[name=foobar, args=iab]{bari}
\symdecl[def=\bar* abc]{bardef}
\ExplSyntaxOn
Meaning:~\present\bar\\
\stex__get_symbol:n { bar }
Result:~\l_stex_get_symbol_uri_str\\
Meaning:~\present\bardef\\
\ExplSyntaxOff
\end{module}
```

> **Module** 14.1.1[SymdeclTest]
>   Meaning: »macro:->\stex_invoke_symbol:n {file://Users/kohlhase/localmh/ext/sTeX/doc/stextest?SymdeclTest?foo}«
> Result: file://Users/kohlhase/localmh/ext/sTeX/doc/stextest?SymdeclTest?foo
> Meaning: »macro:->\stex_invoke_symbol:n {file://Users/kohlhase/localmh/ext/sTeX/doc/stextest?SymdeclTest?bardef}«

.

\l_stex_all_symbols_seq   Stores full URIs for all modules currently in scope.

\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)

Test 12

```
\begin{module}{NotationTest}
\importmodule{Foo}
\notation[foo, prec=500;20x20x20]{bar}{\comp\langle {#1 ^ {#2}}_{#3} \comp\rangle }
\notation[foo, prec=500;20x20x20]{foobar}{\comp\langle #1 \comp\mid [ #2 ]^{#3} \comp\rangle }{ {#1}_{\com
\end{module}
```

> **Module** 14.1.2[NotationTest]

.

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

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

Test 13

```
\begin{module}{SymdefTest}
\symdef[args=a, prec=50]{plus}{ #1 }{#1 \comp+ #2}
$\plus{a,b,c}$
\end{module}
```

> **Module** 14.1.3[SymdefTest]
> $a + b + c$

.

Chapter 15

sTEX-Terms

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

15.1 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⟩.

Maximal and minimal notation precedences.

\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 ⟨*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.

Test 14

```
\begin{module}{MathTest1}
\importmodule{Foo}
\notation[foo, prec=500;20x20x20]{bar}{\comp\langle {#1 ^ {#2}}_{#3} \comp\rangle }
$\bar abc$ and $\bar[foo] abc$.

\end{module}
```

> **Module** 15.1.1[MathTest1]
> ⟨$a^b{}_c$⟩ and ⟨$a^b{}_c$⟩.

.

Test 15

```
\begin{module}{MathTest2}
\importmodule{Foo}
\notation[foo, prec=500;20x20x20]{foobar}{\comp\langle #1 \comp\mid [ #2 ]^{#3} \comp\rangle }{ {#1}_{\com
$\foobar a{b,c,d,e,f}g$ and $\foobar[foo] a{b,c}g$ and $\foobar abc$

\symdecl[args=a]{plus}
\symdecl[args=a]{mult}
\notation[prec=50]{plus}{#1}{#1 \comp+ #2}
\notation[prec=100]{mult}{#1}{#1 \comp\cdot #2}
$\plus{a,\mult{b,c}}$ and $\mult{a,\plus{\frac ab,\frac ac}}$
\[\plus{a,\mult{b,c}}\text{ and }\mult{a,\plus{\frac ab,\frac ac}}\]
$\displaystyle \plus{a,\mult{b,c}}$ and
\withbrackets[]{$\displaystyle
\mult{a,\plus{\frac ab,\frac ac}}$}
\end{module}
```

> **Module** 15.1.2[MathTest2]
> ⟨$a \mid [b{:}_c{:}_{d}{:}_{e}{:}_f]^g$⟩ and ⟨$a \mid [b{:}_c]^g$⟩ and ⟨$a \mid [b]^c$⟩
>
> $a + (b \cdot c)$ and $a \cdot \frac{a}{b} + \frac{a}{c}$
>
> $$a + (b \cdot c) \text{ and } a \cdot \frac{a}{b} + \frac{a}{c}$$
>
> $a + (b \cdot c)$ and $a \cdot \dfrac{a}{b} + \dfrac{a}{c}$

.

\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 !.

> **Test 16**
>
> ```
> \begin{module}{TextTest}
> \importmodule{Foo}
>
> \bar[some ]a[ and some ]b[ and also some ]c[ here].
>
> $\bar*[\text{some }]a[\text{ and some }]b[\text{ and also some }]c[\text{ here}]$.
>
> $\bar!![\mathtt{bar}]$
>
> \bar*{a}*{b}[or just some ]c
>
> \bar![bar]
>
> \bar[or first ]*[2]{b}[, then ]*[3]{c}[, and finally ]a
>
> \end{module}
> ```
>
> ---
>
> **Module** 15.1.3[TextTest]
>   some a and some b and also some c here.
>   some a and some b and also some c here.
>   **bar**
>   or just some c
>   **bar**
>   or first b, then c, and finally a

.

\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

\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 16

sTEX-Structural Features

Code related to structural features

16.1 Macros and Environments

16.1.1 Structures

mathstructure TODO

Chapter 17

sTEX-Statements

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

17.1 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 18

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

18.1 Introduction

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

SТEX 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,for=sum-over-odds]
   {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}[display=flow] then we compute $1=1^2$\end{spfstep}
   \end{spfcase}
   \begin{spfcase}{$n=2$}
      \begin{sproofcomment}[display=flow]
        This case is not really necessary, but we do it for the
        fun of it (and to get more intuition).
      \end{sproofcomment}
      \begin{spfstep}[display=flow] 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).[7]

---

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

18.2 The User Interface

18.2.1 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).

18.2.2 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.

18.2.3 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

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

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

P.1.1 $n = 2$: This case is not really necessary, but we do it for the fun of it (and to get more intuition). We compute $1 + 3 = 2^2 = 4$ □

P.1.1 $n > 1$:

P.1.1.1 Now, we assume that the assertion is true for a certain $k \geq 1$, i.e. $\sum_{i=1}^{k} (2i - 1) = k^2$.

P.1.1.1 We have to show that we can derive the assertion for $n = k + 1$ from this assumption, i.e. $\sum_{i=1}^{k+1} (2i - 1) = (k + 1)^2$.

P.1.1.1 We obtain $\sum_{i=1}^{k+1} (2i - 1) = \sum_{i=1}^{k} (2i - 1) + 2(k + 1) - 1$ by splitting the sum

P.1.1.1 Thus we have $\sum_{i=1}^{k+1} (2i - 1) = k^2 + 2k + 1$ by inductive hypothesis.

P.1.1.1 We can simplify the right-hand side to $(k+1)^2$, which proves the assertion. □

P.1.1 We have considered all the cases, so we have proven the assertion. □

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

18.2.4 Proof Structure

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

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

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

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

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

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

18.2.5 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{}`.

18.2.6 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:8 support.[8] The proof step labels can be customized via the `\pstlabelstyle` macro:

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

Figure 1: Configuration Hooks for Semantic Proof Markup

\pstlabelstyle

`\pstlabelstyle{⟨style⟩}` sets the style; see Figure 2 for an overview of styles. Package writers can add additional styles by adding a macro `\pst@make@label@⟨style⟩` that takes 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 2 for examples.

style	example	configuration macro
`long`	0.8.1.5	`\def\pst@make@label@long#1#2{\@for\@I:=#1\do{\@I.}#2}`
`angles`	⟩⟩⟩5	`\def\pst@make@label@angles#1#2` `{\ensuremath{\@for\@I:=#1\do{\rangle}}#2}`
`short`	5	`\def\pst@make@label@short#1#2{#2}`
`empty`		`\def\pst@make@label@empty#1#2{}`

Figure 2: Configuration Proof Step Label Styles

18.3 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].

---

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

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 19

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.

19.1   Symbols

Part III

Extensions

Chapter 20

Tikzinput

20.1 Macros and Environments

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

Chapter 21

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.

21.1   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.[9]

21.2   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`.

21.2.1   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.

21.2.2   Document Structure

document
\documentkeys
id

The top-level `document` environment can be given key/value information by the `\documentkeys` macro in the preamble[2]. 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.

omgroup

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{module}{foo}
\symdef{bar}{B^a_r}
 ...
\begin{omgroup}[id=sec.barderiv,loadmodules]{Introducing $\protect\bar$ Derivations}
```

---

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

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

$\mathcal{S}T_{E}X$ 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`:

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

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`.

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

---

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

21.2.3   Ignoring Inputs

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].

21.2.4   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.[10]

EdN:10

21.2.5   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

---

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

56

$\texttt{\textbackslash ifSGvar}\{\langle\textit{vname}\rangle\}\{\langle\textit{val}\rangle\}\{\langle\textit{ctext}\rangle\}$ tests the content of the global variable $\langle\textit{vname}\rangle$, only if (after expansion) it is equal to $\langle\textit{val}\rangle$, the conditional text $\langle\textit{ctext}\rangle$ is formatted.

21.2.6 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{`$\langle\textit{something}\rangle$`}` writes $\langle\textit{something}\rangle$ in blue. The macros `\red \green`, `\cyan`, `\magenta`, `\brown`, `\yellow`, `\orange`, `\gray`, and finally `\black` are analogous.

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

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 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 22

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.

22.1 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`.

22.2 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.

22.2.1 Package Options

EdN:11   The `notesslides` class takes a variety of class options:[11]

slides
notes
- The options `slides` and `notes` switch between slides mode and notes mode (see Section 22.2.2).

• 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 22.2.4). 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`.

22.2.2 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.[4]

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

```
\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

---

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

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

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*`   `\inputref*` of the `\inputref` macro from [KGA20]: `\inputref*{foo}` is equivalent to `\begin{note}\inputref{foo}\end{note}`.

There are some environments that tend to occur at the top-level of `note` environ-
`nparagraph`   ments. We make convenience versions of these: e.g. the `nparagraph` environment is just an `sparagraph` inside a `note` environment (but looks nicer in the source, since it avoids
`nomgroup`   one level of source indenting). Similarly, we have the `nomgroup`, `ndefinition`, `nexample`,
`ndefinition`   `nsproof`, and `nassertion` environments.
`nexample`
`nsproof`   ### 22.2.3 Header and Footer Lines of the Slides
`nassertion`

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

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

22.2.4 Frame Images

Sometimes, we want to integrate slides as images after all – e.g. because we already have a PowerPoint presentation, to which we want to add sTEXnotes. In this case we
`\frameimage`   can use `\frameimage[⟨opt⟩]{⟨path⟩}`, where ⟨opt⟩ are the options of `\includegraphics` from the `graphicx` package [CR99] and ⟨path⟩ is the file path (extension can be left off like in `\includegraphics`). We have added the `label` key that allows to give a frame
EdN:12   label that can be referenced like a regular `beamer` frame.[12]
`\mhframeimage`   The `\mhframeimage` macro is a variant of `\frameimage` with repository support. Instead of writing

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

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

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

---

[12]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}
```

22.2.5 Colors and Highlighting

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

22.2.6 Front Matter, Titles, etc.

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

22.2.8 Miscellaneous

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. 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 23

`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.

23.1   Introduction

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

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

23.2   The User Interface

23.2.1   Package Options

solutions
notes
hints
gnotes
pts
min
boxed
test
mh
showmeta

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

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

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

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

---

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

23.2.2 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{problem}[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{problem}
\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).

Problem0.0 ()
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.

23.2.3   Multiple Choice Blocks

mcb
\mcc

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
F
Ttext
Ftext
feedback

- `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

23.2.4   Including Problems

\includeproblem

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

title
min
pts

23.2.5   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 distribution 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.

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

```
\begin{problem}[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{problem}
```

Problem0.0 ()

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

1. def

2. function

3. fun

4. public static void

Problem0.0 ()

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 24

`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

24.1   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.

24.2   The User Interface

24.2.1   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**].

24.2.2   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).

24.2.3   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

24.2.4   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

24.3   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. 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: MatriculationNumber:

320101 General Computer Science (Fall 2010)

2022-02-11

You have 60 minutes (sharp) for the test;
Write the solutions to the sheet.
The estimated time for solving this exam is 58 minutes, leaving you 2 minutes for revising your exam.
You can reach 30 points if you solve all problems. You will only need 27 points for a perfect score, i.e. 3 points are bonus points.

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

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

prob.	0.0	0.0	0.0	1.1	2.1	2.2	2.3	3.1	3.2	3.3	Sum	grade
total				4	4	6	6	4	4	2	30	
reached												

To be used for grading, do not write here

good luck

Example 8: A generated test heading.

Part IV

Implementation

Chapter 25

sTEX
-Basics Implementation

25.1 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}{2021/08/01}{1.9}{bla}
7  \LoadClass[border=1px,varwidth]{standalone}
8  \setlength\textwidth{15cm}
9
10 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
11 \ProcessOptions
12
13 \RequirePackage{stex}
14 ⟨/cls⟩
```

25.2 Preliminaries

```
15 ⟨∗package⟩
16
17 %%%%%%%%%%%%%   basics.dtx   %%%%%%%%%%%%%
18
19 \RequirePackage{expl3,l3keys2e,ltxcmds}
20 \ProvidesExplPackage{stex}{2021/08/01}{1.9}{bla}
21 \RequirePackage{expl-keystr-compat}
22
23 %\RequirePackage{morewrites}
24 %\RequirePackage{amsmath}
25
```

Package options:

```
26 \keys_define:nn { stex } {
27   debug      .clist_set:N  = \c_stex_debug_clist ,
28   showmods   .bool_set:N   = \c_stex_showmods_bool ,
29   lang       .clist_set:N  = \c_stex_languages_clist ,
30   mathhub    .tl_set_x:N   = \mathhub ,
31   sms        .bool_set:N   = \c_stex_persist_mode_bool ,
32   image      .bool_set:N   = \c_tikzinput_image_bool,
33   unknown    .code:n       = {}
34 }
35 \ProcessKeysOptions { stex }
```

\stex The sTeXlogo:
\sTeX

```
36 \protected\def\stex{%
37   \@ifundefined{texorpdfstring}%
38   {\let\texorpdfstring\@firstoftwo}%
39   {}%
40   \texorpdfstring{\raisebox{-.5ex}S\kern-.5ex\TeX}{sTeX}\xspace%
41 }
42 \def\sTeX{\stex}
```

(*End definition for* `\stex` *and* `\sTeX`*. These functions are documented on page 20.*)

25.3 Messages and logging

```
43 ⟨@@=stex_log⟩
```

Warnings and error messages

```
44 \msg_new:nnn{stex}{error/unknownlanguage}{
45   Unknown~language:~#1
46 }
47 \msg_new:nnn{stex}{warning/nomathhub}{
48   MATHHUB~system~variable~not~found~and~no~
49   \detokenize{\mathhub}-value~set!
50 }
51 \msg_new:nnn{stex}{error/deactivated-macro}{
52   The~\detokenize{#1}~command~is~only~allowed~in~#2!
53 }
```

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

```
54 \cs_new_protected:Nn \stex_debug:nn {
55   \clist_if_in:NnTF \c_stex_debug_clist { all } {
56     \exp_args:Nnnx\msg_set:nnn{stex}{debug / #1}{
57       \\Debug~#1:~#2\\
58     }
59     \msg_none:nn{stex}{debug / #1}
60   }{
61     \clist_if_in:NnT \c_stex_debug_clist { #1 } {
62       \exp_args:Nnnx\msg_set:nnn{stex}{debug / #1}{
63         \\Debug~#1:~#2\\
64       }
65       \msg_none:nn{stex}{debug / #1}
66     }
67   }
68 }
```

(*End definition for* `\stex_debug:nn`*. This function is documented on page 20.*)

Redirecting messages:

```
69 \clist_if_in:NnTF \c_stex_debug_clist {all} {
70   \msg_redirect_module:nnn{ stex }{ none }{ term }
71 }{
72   \clist_map_inline:Nn \c_stex_debug_clist {
73     \msg_redirect_name:nnn{ stex }{ debug / ##1 }{ term }
74   }
75 }
76
77 \stex_debug:nn{log}{debug~mode~on}
```

25.4   Persistence

```
78 ⟨@@=stex_persist⟩
```

`\c__stex_persist_sms_iow`   File variable used for the sms-File

```
79 \iow_new:N \c__stex_persist_sms_iow
80 \AddToHook{begindocument}{
81   \bool_if:NTF \c_stex_persist_mode_bool {
82     \ExplSyntaxOn \input{\jobname.sms} \ExplSyntaxOff
83   } {
84 %    \iow_open:Nn \c__stex_persist_sms_iow {\jobname.sms}
85   }
86 }
87 \AddToHook{enddocument}{
88   \bool_if:NF \c_stex_persist_mode_bool {
89 %    \iow_close:N \c__stex_persist_sms_iow
90   }
91 }
```

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

`\stex_add_to_sms:n`   Adds the provided code to the `.sms`-file of the document.

```
92 \cs_new_protected:Nn \stex_add_to_sms:n {
93   \bool_if:NF \c_stex_persist_mode_bool {
94 %    \iow_now:Nn \c__stex_persist_sms_iow { #1 }
95   }
96 }
```

(*End definition for* `\stex_add_to_sms:n`*. This function is documented on page 20.*)

25.5   HTML Annotations

```
97 ⟨@@=stex_annotate⟩
98 \RequirePackage{rustex}
```

We add the namespace abbreviation `ns:stex="http://kwarc.info/ns/sTeX"` to RᵤₛTₑX:

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

`\if@latexml`   Conditionals for LᴬTᴇXML:
`\latexml_if_p:`
`\latexml_if:`*TF*

```
100 \ifcsname if@latexml\endcsname\else
```

```
101     \expandafter\newif\csname if@latexml\endcsname\@latexmlfalse
102   \fi
103
104 \prg_new_conditional:Nnn \latexml_if: {p, T, F, TF} {
105     \if@latexml
106       \prg_return_true:
107     \else:
108       \prg_return_false:
109     \fi:
110 }
```

(*End definition for* \if@latexml *and* \latexml_if:TF. *These functions are documented on page* 20.)

\l__stex_annotate_arg_tl
\c__stex_annotate_emptyarg_tl

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

```
111 \tl_new:N \l__stex_annotate_arg_tl
112 \tl_const:Nx \c__stex_annotate_emptyarg_tl {
113     \rustex_if:TF {
114       \rustex_direct_HTML:n { \c_ampersand_str lrm; }
115     }{~}
116 }
```

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

__stex_annotate_checkempty:n

```
117 \cs_new_protected:Nn \__stex_annotate_checkempty:n {
118     \tl_set:Nn \l__stex_annotate_arg_tl { #1 }
119     \tl_if_empty:NT \l__stex_annotate_arg_tl {
120       \tl_set_eq:NN \l__stex_annotate_arg_tl \c__stex_annotate_emptyarg_tl
121     }
122 }
```

(*End definition for* __stex_annotate_checkempty:n.)

\l_stex_html_do_output_bool
\stex_if_do_html:

Whether to (locally) produce HTML output

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

(*End definition for* \l_stex_html_do_output_bool *and* \stex_if_do_html:. *These functions are documented on page* **??**.)

\stex_suppress_html:n

Whether to (locally) produce HTML output

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

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

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

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

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

```
187    }{
188      \par\rustex_annotate_HTML_end:
189    }
190  }{
191    \latexml_if:TF {
192      \cs_new_protected:Nn \stex_annotate:nnn {
193        \__stex_annotate_checkempty:n { #3 }
194        \mode_if_math:TF {
195          \cs:w latexml@annotate@math\cs_end:{#1}{#2}{
196            \tl_use:N \l__stex_annotate_arg_tl
197          }
198        }{
199          \cs:w latexml@annotate@text\cs_end:{#1}{#2}{
200            \tl_use:N \l__stex_annotate_arg_tl
201          }
202        }
203      }
204      \cs_new_protected:Nn \stex_annotate_invisible:n {
205        \__stex_annotate_checkempty:n { #1 }
206        \mode_if_math:TF {
207          \cs:w latexml@invisible@math\cs_end:{
208            \tl_use:N \l__stex_annotate_arg_tl
209          }
210        } {
211          \cs:w latexml@invisible@text\cs_end:{
212            \tl_use:N \l__stex_annotate_arg_tl
213          }
214        }
215      }
216      \cs_new_protected:Nn \stex_annotate_invisible:nnn {
217        \__stex_annotate_checkempty:n { #3 }
218        \cs:w latexml@annotate@invisible\cs_end:{#1}{#2}{
219          \tl_use:N \l__stex_annotate_arg_tl
220        }
221      }
222      \NewDocumentEnvironment{stex_annotate_env} { m m } {
223        \par\begin{latexml@annotateenv}{#1}{#2}
224      }{
225        \par\end{latexml@annotateenv}
226      }
227    }{
228      \cs_new_protected:Nn \stex_annotate:nnn {#3}
229      \cs_new_protected:Nn \stex_annotate_invisible:n {}
230      \cs_new_protected:Nn \stex_annotate_invisible:nnn {}
231      \NewDocumentEnvironment{stex_annotate_env} { m m } {}{}
232    }
233  }
```

(*End definition for* \stex_annotate:nnn *,* \stex_annotate_invisible:n *, and* \stex_annotate_invisible:nnn*. These functions are documented on page 21.*)

25.6   Languages

```
234  ⟨@@=stex_language⟩
```

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

```
235 \prop_const_from_keyval:Nn \c_stex_languages_prop {
236   en = english ,
237   de = ngerman ,
238   ar = arabic ,
239   bg = bulgarian ,
240   ru = russian ,
241   fi = finnish ,
242   ro = romanian ,
243   tr = turkish ,
244   fr = french
245 }
246
247 \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop {
248   english   = en ,
249   ngerman   = de ,
250   arabic    = ar ,
251   bulgarian = bg ,
252   russian   = ru ,
253   finnish   = fi ,
254   romanian  = ro ,
255   turkish   = tr ,
256   french    = fr
257 }
258 % todo: chinese simplified (zhs)
259 %        chinese traditional (zht)
```

(*End definition for* \c_stex_languages_prop *and* \c_stex_language_abbrevs_prop. *These variables are documented on page 21.*)

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

```
260 \clist_if_empty:NF \c_stex_languages_clist {
261   \clist_clear:N \l_tmpa_clist
262   \clist_map_inline:Nn \c_stex_languages_clist {
263     \prop_get:NnNTF \c_stex_languages_prop { #1 } \l_tmpa_str {
264       \clist_put_right:No \l_tmpa_clist \l_tmpa_str
265     } {
266       \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
267     }
268   }
269   \stex_debug:nn{lang} {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
270   \RequirePackage[\clist_use:Nn \l_tmpa_clist,]{babel}
271 }
```

25.7   Activating/Deactivating Macros

```
272 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
273   \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
274   \def#1{
275     \msg_error:nnnn{stex}{error/deactivated-macro}{#1}{#2}
276   }
277 }
```

(End definition for `\stex_deactivate_macro:Nn`. *This function is documented on page 21.)*

`\stex_reactivate_macro:N`

```
278 \cs_new_protected:Nn \stex_reactivate_macro:N {
279     \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname
280 }
```

(End definition for `\stex_reactivate_macro:N`. *This function is documented on page 21.)*

`\stex_do_aftergroup:nn`

```
281 ⟨@@=stex_aftergroup⟩
282 \tl_new:N \l__stex_aftergroup_tl
283 \cs_new_protected:Nn \stex_do_aftergroup:n {
284     \int_compare:nNnTF \l_stex_module_group_depth_int = \currentgrouplevel {
285         #1
286     }{
287         #1
288         \expandafter \tl_gset:Nn \expandafter \l__stex_aftergroup_tl \expandafter { \l__stex_aft
289         \aftergroup\__stex_aftergroup_do:
290     }
291 }
292 \cs_new_protected:Nn \__stex_aftergroup_do: {
293     \int_compare:nNnTF \l_stex_module_group_depth_int = \currentgrouplevel {
294         \l__stex_aftergroup_tl
295         \tl_clear:N \l__stex_aftergroup_tl
296     }{
297         \l__stex_aftergroup_tl
298         \aftergroup\__stex_aftergroup_do:
299     }
300 }
```

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

```
301 ⟨/package⟩
```

79

Chapter 26

STEX -MathHub Implementation

```
302 ⟨*package⟩
303
304 %%%%%%%%%%%%   mathhub.dtx   %%%%%%%%%%%%
305
306 ⟨@@=stex_path⟩
```

Warnings and error messages

```
307 \msg_new:nnn{stex}{error/norepository}{
308   No~archive~#1~found~in~#2
309 }
310 \msg_new:nnn{stex}{error/notinarchive}{
311   Not~currently~in~an~archive,~but~\detokenize{#1}~
312   needs~one!
313 }
314 \msg_new:nnn{stex}{error/nofile}{
315   \detokenize{#1}~could~not~find~file~#2
316 }
317 \msg_new:nnn{stex}{error/twofiles}{
318   \detokenize{#1}~found~two~candidates~for~#2
319 }
```

26.1   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
\stex_path_from_string:NV
\stex_path_from_string:cn
\stex_path_from_string:cV

```
320 \cs_new_protected:Nn \stex_path_from_string:Nn {
321   \str_set:Nx \l_tmpa_str { #2 }
322   \str_if_empty:NTF \l_tmpa_str {
323     \seq_clear:N #1
324   }{
325     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
326     \sys_if_platform_windows:T{
327       \seq_clear:N \l_tmpa_tl
```

80

```
328        \seq_map_inline:Nn #1 {
329          \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
330          \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
331        }
332        \seq_set_eq:NN #1 \l_tmpa_tl
333      }
334      \stex_path_canonicalize:N #1
335    }
336 }
337 \cs_generate_variant:Nn \stex_path_from_string:Nn
338    { NV, cn, cV }
```

(*End definition for* `\stex_path_from_string:Nn`. *This function is documented on page* *22*.)

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

```
339 \cs_new_protected:Nn \stex_path_to_string:NN {
340    \exp_args:NNe \str_set:Nn #2 { \seq_use:Nn #1 / }
341 }
342
343 \cs_new:Nn \stex_path_to_string:N {
344    \seq_use:Nn #1 /
345 }
```

(*End definition for* `\stex_path_to_string:NN` *and* `\stex_path_to_string:N`. *These functions are documented on page* *22*.)

`\c__stex_path_dot_str`
`\c__stex_path_up_str`

`.` and `..`, respectively.

```
346 \str_const:Nn \c__stex_path_dot_str {.}
347 \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.

```
348 \cs_new_protected:Nn \stex_path_canonicalize:N {
349    \seq_if_empty:NF #1 {
350      \seq_clear:N \l_tmpa_seq
351      \seq_get_left:NN #1 \l_tmpa_tl
352      \str_if_empty:NT \l_tmpa_tl {
353        \seq_put_right:Nn \l_tmpa_seq {}
354      }
355    \seq_map_inline:Nn #1 {
356      \str_set:Nn \l_tmpa_tl { ##1 }
357      \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_dot_str {} {
358        \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
359          \seq_if_empty:NTF \l_tmpa_seq {
360            \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
361              \c__stex_path_up_str
362            }
363          }{
364            \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
365            \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
366              \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
367                \c__stex_path_up_str
368              }
```

```
369            }{
370              \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
371            }
372          }
373        }{
374          \str_if_empty:NF \l_tmpa_tl {
375            \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
376          }
377        }
378      }
379    }
380    \seq_gset_eq:NN #1 \l_tmpa_seq
381  }
382 }
```

(End definition for `\stex_path_canonicalize:N`. *This function is documented on page 22.)*

```
383 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
384   \seq_if_empty:NTF #1 {
385     \prg_return_false:
386   }{
387     \seq_get_left:NN #1 \l_tmpa_tl
388     \str_if_empty:NTF \l_tmpa_tl {
389       \prg_return_true:
390     }{
391       \prg_return_false:
392     }
393   }
394 }
```

(End definition for `\stex_path_if_absolute:NTF`. *This function is documented on page 22.)*

26.2   PWD and kpsewhich

```
395 \str_new:N\l_stex_kpsewhich_return_str
396 \cs_new_protected:Nn \stex_kpsewhich:n {
397   \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
398   \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
399   \tl_trim_spaces:N \l_stex_kpsewhich_return_str
400 }
```

(End definition for `\stex_kpsewhich:n`. *This function is documented on page 22.)*
    We determine the PWD

```
401 \sys_if_platform_windows:TF{
402   \stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
403 }{
404   \stex_kpsewhich:n{-var-value~PWD}
405 }
406
```

```
407  \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
408  \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
409  \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* *22*.)

26.3   File Hooks and Tracking

```
410  ⟨@@=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 SLₑX-purposes.

`\g__stex_files_stack`   keeps track of file changes

```
411  \seq_gclear_new:N\g__stex_files_stack
```

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

`\c_stex_mainfile_seq`
`\c_stex_mainfile_str`
```
412  \str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
413  \stex_path_from_string:Nn \c_stex_mainfile_seq
414     \c_stex_mainfile_str
```

(*End definition for* `\c_stex_mainfile_seq` *and* `\c_stex_mainfile_str`. *These variables are documented on page* *22*.)

`\g_stex_currentfile_seq`   Hooks for file inputs that push/pop `\g__stex_files_stack` to update `\c_stex_-mainfile_seq`.

```
415  \seq_gclear_new:N\g_stex_currentfile_seq
416  \AddToHook{file/before}{
417     \stex_path_from_string:Nn\g_stex_currentfile_seq{\CurrentFilePath}
418     \stex_path_if_absolute:NTF\g_stex_currentfile_seq{
419        \exp_args:NNe\seq_put_right:Nn\g_stex_currentfile_seq{\CurrentFile}
420     }{
421        \stex_path_from_string:Nn\g_stex_currentfile_seq{
422           \c_stex_pwd_str/\CurrentFilePath/\CurrentFile
423        }
424     }
425     \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
426     \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
427  }
428  \AddToHook{file/after}{
429     \seq_if_empty:NF\g__stex_files_stack{
430        \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
431     }
432     \seq_if_empty:NTF\g__stex_files_stack{
433        \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
434     }{
435        \seq_get:NN\g__stex_files_stack\l_tmpa_seq
436        \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
437     }
438  }
```

26.4 MathHub Repositories

```
439 ⟨@@=stex_mathhub⟩
```

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

```
440 \str_if_empty:NTF\mathhub{
441   \stex_kpsewhich:n{-var-value~MATHHUB}
442   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
443
444   \str_if_empty:NTF\c_stex_mathhub_str{
445     \msg_warning:nn{stex}{warning/nomathhub}
446   }{
447     \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
448     \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
449   }
450 }{
451   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
452   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
453     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
454       \c_stex_pwd_str/\mathhub
455     }
456   }
457   \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
458   \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
459 }
```

(*End definition for* `\mathhub`, `\c_stex_mathhub_seq`*, and* `\c_stex_mathhub_str`*. These variables are documented on page* *23*.)

`\__stex_mathhub_do_manifest:n`

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

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

`\l__stex_mathhub_manifest_file_seq`

```
476 \str_new:N\l__stex_mathhub_manifest_file_seq
```

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

`\__stex_mathhub_find_manifest:N` Attempts to find the `MANIFEST.MF` in some file path and stores its path in `\l__stex_-mathhub_manifest_file_seq`:

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

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

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

```
512 \ior_new:N \c__stex_mathhub_manifest_ior
```

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

`\__stex_mathhub_parse_manifest:n` Stores the entries in manifest file in the corresponding property list:

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

```
519          \l_tmpb_seq \c_colon_str \l_tmpa_str
520        \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
521          \exp_args:NNe \str_set:Nn \l_tmpb_tl {
522            \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
523          }
524          \exp_args:No \str_case:nnTF \l_tmpa_tl {
525            {id} {
526              \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
527                { id } \l_tmpb_tl
528            }
529            {narration-base} {
530              \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
531                { narr } \l_tmpb_tl
532            }
533            {url-base} {
534              \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
535                { docurl } \l_tmpb_tl
536            }
537            {source-base} {
538              \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
539                { ns } \l_tmpb_tl
540            }
541            {ns} {
542              \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
543                { ns } \l_tmpb_tl
544            }
545            {dependencies} {
546              \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
547                { deps } \l_tmpb_tl
548            }
549          }{}{}
550        }{}
551      }
552      \ior_close:N \c__stex_mathhub_manifest_ior
553  }
```

(End definition for __stex_mathhub_parse_manifest:n.*)*

```
554  \cs_new_protected:Nn \stex_set_current_repository:n {
555    \stex_require_repository:n { #1 }
556    \prop_set_eq:Nc \l_stex_current_repository_prop {
557      c_stex_mathhub_#1_manifest_prop
558    }
559  }
```

(End definition for \stex_set_current_repository:n. *This function is documented on page* 24.*)*

```
560  \cs_new_protected:Nn \stex_require_repository:n {
561    \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
562      \stex_debug:nn{mathhub}{Opening~archive:~#1}
563      \__stex_mathhub_do_manifest:n { #1 }
564      \exp_args:Nx \stex_add_to_sms:n {
565        \prop_const_from_keyval:cn { c_stex_mathhub_#1_manifest_prop } {
```

```
566        id  = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } {  id  } ,
567        ns  = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } {  ns  } ,
568        narr = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { narr } ,
569        deps = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { deps }
570      }
571    }
572  }
573 }
```

(*End definition for* `\stex_require_repository:n`. *This function is documented on page* *24.*)

`\l_stex_current_repository_prop`   Current MathHub repository

```
574 %\prop_new:N \l_stex_current_repository_prop
575
576 \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
577 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
578    \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
579 } {
580    \__stex_mathhub_parse_manifest:n { main }
581    \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
582      \l_tmpa_str
583    \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str _manifest_prop }
584      \c_stex_mathhub_main_manifest_prop
585    \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
586    \stex_debug:nn{mathhub}{Current~repository:~
587      \prop_item:Nn \l_stex_current_repository_prop {id}
588    }
589 }
```

(*End definition for* `\l_stex_current_repository_prop`. *This variable is documented on page* *23.*)

`\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.

```
590 \cs_new_protected:Nn \stex_in_repository:nn {
591    \str_set:Nx \l_tmpa_str { #1 }
592    \cs_set:Npn \l_tmpa_cs ##1 { #2 }
593    \str_if_empty:NTF \l_tmpa_str {
594      \prop_if_exist:NTF \l_stex_current_repository_prop {
595        \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
596        \exp_args:Ne \l_tmpa_cs{
597          \prop_item:Nn \l_stex_current_repository_prop { id }
598        }
599      }{
600        \l_tmpa_cs{}
601      }
602    }{
603      \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}
604      \stex_require_repository:n \l_tmpa_str
605      \str_set:Nx \l_tmpa_str { #1 }
606      \exp_args:Nne \use:nn {
607        \stex_set_current_repository:n \l_tmpa_str
608        \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
609      }{
610        \stex_debug:nn{mathhub}{switching~back~to:~
```

```
611         \prop_if_exist:NTF \l_stex_current_repository_prop {
612           \prop_item:Nn \l_stex_current_repository_prop { id }:~
613           \meaning\l_stex_current_repository_prop
614         }{
615           no~repository
616         }
617       }
618       \prop_if_exist:NTF \l_stex_current_repository_prop {
619        \stex_set_current_repository:n {
620        \prop_item:Nn \l_stex_current_repository_prop { id }
621        }
622       }{
623         \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
624       }
625     }
626   }
627 }
```

(End definition for `\stex_in_repository:nn`*. This function is documented on page 24.)*

`\inputref`
`\stex_inputref:nn`
`\mhinput\stex_mhinput:nn`

```
628 \newif \ifinputref \inputreffalse
629
630 \cs_new_protected:Nn \stex_mhinput:nn {
631   \stex_in_repository:nn {#1} {
632     \ifinputref
633       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
634     \else
635       \inputreftrue
636       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
637       \inputreffalse
638     \fi
639   }
640 }
641 \NewDocumentCommand \mhinput { O{} m}{
642   \stex_mhinput:nn{ #1 }{ #2 }
643 }
644
645 \cs_new_protected:Nn \stex_inputref:nn {
646   \stex_in_repository:nn {#1} {
647     \bool_lazy_any:nTF {
648       {\rustex_if_p:} {\latexml_if_p:}
649     } {
650       \str_clear:N \l_tmpa_str
651       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
652         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
653       }
654       \stex_annotate_invisible:nnn{inputref}{
655         \l_tmpa_str / #2
656       }{}
657     }{
658       \begingroup
659         \inputreftrue
660         \input{ \c_stex_mathhub_str / ##1 / source / #2 }
```

88

```
661        \endgroup
662      }
663    }
664  }
665
666  \NewDocumentCommand \inputref { O{} m}{
667    \stex_inputref:nn{ #1 }{ #2 }
668  }
669
670  \cs_new_protected:Nn \stex_mhbibresource:nn {
671    \stex_in_repository:nn {#1} {
672      \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
673    }
674  }
675  \newcommand\addmhbibresource[2][]{
676    \stex_mhbibresource:nn{ #1 }{ #2 }
677  }
```

(*End definition for* \inputref *,* \stex_inputref:nn *, and* \mhinput\stex_mhinput:nn*. These functions are documented on page* *24.*)

\mhpath

```
678    \def \mhpath #1 #2 {
679      \exp_args:Ne \str_if_eq:nnTF{#1}{}{
680        \c_stex_mathhub_str /
681          \prop_item:Nn \l_stex_current_repository_prop { id }
682          / source / #2
683      }{
684        \c_stex_mathhub_str / #1 / source / #2
685      }
686    }
```

(*End definition for* \mhpath*. This function is documented on page* *24.*)

\libinput

```
687  \cs_new_protected:Npn \libinput #1 {
688    \prop_if_exist:NF \l_stex_current_repository_prop {
689      \msg_error:nnn{stex}{error/notinarchive}\libinput
690    }
691    \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
692      \msg_error:nnn{stex}{error/notinarchive}\libinput
693    }
694    \bool_set_false:N \l_tmpa_bool
695    \tl_clear:N \l_tmpa_tl
696    \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
697    \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
698    \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str
699    \seq_pop_left:NNT \l_tmpb_seq \l_tmpb_str {
700      \seq_put_right:No \l_tmpa_seq \l_tmpb_str
701      \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
702        / meta-inf / lib / #1.tex}{
703          \bool_set_true:N \l_tmpa_bool
704          \tl_put_right:Nx \l_tmpa_tl {
705            \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
706            / meta-inf / lib / #1.tex}
```

89

```
707            }
708          }{}
709      }
710      \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
711        / \l_tmpa_str / lib / #1.tex
712      }{
713        \bool_set_true:N \l_tmpa_bool
714        \tl_put_right:Nx \l_tmpa_tl {
715          \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
716          / \l_tmpa_str / lib / #1.tex}
717        }
718      }{}
719      \bool_if:NF \l_tmpa_bool {
720        \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
721      }
722      \l_tmpa_tl
723  }
```

(End definition for `\libinput`*. This function is documented on page 24.)*

`\libusepackage`

```
724  \cs_new_protected:Npn \libusepackage #1 {
725      \prop_if_exist:NF \l_stex_current_repository_prop {
726        \msg_error:nnn{stex}{error/notinarchive}\libusepackage
727      }
728      \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
729        \msg_error:nnn{stex}{error/notinarchive}\libusepackage
730      }
731      \bool_set_false:N \l_libusepackage_bool
732      \tl_clear:N \l_tmpa_tl
733      \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
734      \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
735      \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str
736      \seq_pop_left:NNT \l_tmpb_seq \l_tmpb_str {
737        \seq_put_right:No \l_tmpa_seq \l_tmpb_str
738        \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
739          / meta-inf / lib / #1.sty}{
740            \bool_set_true:N \l_libusepackage_bool
741            \tl_put_right:Nx \l_tmpa_tl {
742              \exp_not:N \usepackage { \stex_path_to_string:N \l_tmpa_seq
743              / meta-inf / lib / #1.sty}
744            }
745        }{}
746      }
747      \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
748        / \l_tmpa_str / lib / #1.sty
749      }{
750        \bool_if:NT \l_libusepackage_bool {
751          \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#1.sty}
752        }
753        \bool_set_true:N \l_libusepackage_bool
754        \tl_put_right:Nx \l_tmpa_tl {
755          \exp_not:N \usepackage { \stex_path_to_string:N \l_tmpa_seq
756          / \l_tmpa_str / lib / #1.sty}
```

```
757      }
758    }{}
759    \bool_if:NF \l_libusepackage_bool {
760      \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#1.sty}
761    }
762    \l_tmpa_tl
763  }
```

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

```
764  ⟨/package⟩
```

Chapter 27

sTeX
-References Implementation

```
765  ⟨*package⟩
766
767  %%%%%%%%%%%%   references.dtx   %%%%%%%%%%%%
768
769  %\RequirePackage{hyperref}
770  %\RequirePackage{cleveref}
771  ⟨@@=stex_refs⟩
```

Warnings and error messages

```
772
773  \iow_new:N \c__stex_refs_refs_iow
774  \AddToHook{begindocument}{
775    \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
776  }
777  \AddToHook{enddocument}{
778    \iow_close:N \c__stex_refs_refs_iow
779  }
780
781  \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
782
783  \NewDocumentCommand \STEXreftitle { m } {
784    \tl_gset:Nx \g__stex_refs_title_tl { #1 }
785  }
```

27.1   Document URIs and URLs

```
786  \seq_new:N \g__stex_refs_all_refs_seq
787
788  \str_new:N \l_stex_current_docns_str
789
790  \cs_new_protected:Nn \stex_get_document_uri: {
791    \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
792    \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
793    \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
794    \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
```

```
795    \seq_put_right:No \l_tmpa_seq \l_tmpb_str
796
797    \str_clear:N \l_tmpa_str
798    \prop_if_exist:NT \l_stex_current_repository_prop {
799      \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
800        \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
801      }
802    }
803
804    \str_if_empty:NTF \l_tmpa_str {
805      \str_set:Nx \l_stex_current_docns_str {
806        file:/\stex_path_to_string:N \l_tmpa_seq
807      }
808    }{
809      \bool_set_true:N \l_tmpa_bool
810      \bool_while_do:Nn \l_tmpa_bool {
811        \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
812        \exp_args:No \str_case:nnTF { \l_tmpb_str } {
813          {source} { \bool_set_false:N \l_tmpa_bool }
814        }{}{
815          \seq_if_empty:NT \l_tmpa_seq {
816            \bool_set_false:N \l_tmpa_bool
817          }
818        }
819      }
820
821      \seq_if_empty:NTF \l_tmpa_seq {
822        \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
823      }{
824        \str_set:Nx \l_stex_current_docns_str {
825          \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
826        }
827      }
828    }
829 }
830 \str_new:N \l_stex_current_docurl_str
831 \cs_new_protected:Nn \stex_get_document_url: {
832    \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
833    \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
834    \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
835    \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
836    \seq_put_right:No \l_tmpa_seq \l_tmpb_str
837
838    \str_clear:N \l_tmpa_str
839    \prop_if_exist:NT \l_stex_current_repository_prop {
840      \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
841        \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
842          \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
843        }
844      }
845    }
846
847    \str_if_empty:NTF \l_tmpa_str {
848      \str_set:Nx \l_stex_current_docurl_str {
```

```
849      file:/\stex_path_to_string:N \l_tmpa_seq
850    }
851  }{
852    \bool_set_true:N \l_tmpa_bool
853    \bool_while_do:Nn \l_tmpa_bool {
854      \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
855      \exp_args:No \str_case:nnTF { \l_tmpb_str } {
856        {source} { \bool_set_false:N \l_tmpa_bool }
857      }{}{
858        \seq_if_empty:NT \l_tmpa_seq {
859          \bool_set_false:N \l_tmpa_bool
860        }
861      }
862    }
863
864    \seq_if_empty:NTF \l_tmpa_seq {
865      \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
866    }{
867      \str_set:Nx \l_stex_current_docurl_str {
868        \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
869      }
870    }
871  }
872 }
```

27.2 Setting Reference Targets

```
873 \str_const:Nn \c__stex_refs_url_str{URL}
874 \str_const:Nn \c__stex_refs_ref_str{REF}
875 % @currentlabel -> number
876 % @currentlabelname -> title
877 % @currentHref -> name.number <- id of some kind
878 % \theH# -> \arabic{section}
879 % \the#  -> number
880 % \hyper@makecurrent{#}
881 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
882   \stex_get_document_uri:
883   \str_set:Nx \l_tmpa_str { #1 }
884   \str_if_empty:NT \l_tmpa_str {
885     \int_zero:N \l_tmpa_int
886     \bool_set_true:N \l_tmpa_bool
887     \bool_while_do:Nn \l_tmpa_bool {
888       \cs_if_exist:cTF {
889         sref_\l_stex_current_docns_str?? REF_\int_use:N \l_tmpa_int _type
890       }{
891         \int_incr:N \l_tmpa_int
892       }{
893         \str_set:Nx \l_tmpa_str { REF_\int_use:N \l_tmpa_int }
894         \bool_set_false:N \l_tmpa_bool
895       }
896     }
897   }
898   \str_set:Nx \l_tmpa_str {
899     \l_stex_current_docns_str??\l_tmpa_str
```

```
900    }
901    \seq_gput_right:No \g__stex_refs_all_refs_seq \l_tmpa_str
902    \stex_if_smsmode:TF {
903      \stex_get_document_url:
904      \str_gset_eq:cN {sref_url_\l_tmpa_str _str}\l_stex_current_docurl_str
905      \str_gset_eq:cN {sref_\l_tmpa_str _type}\c__stex_refs_url_str
906    }{
907      \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=~\expandafter{\@currentlabel\iffalse}{
908      \exp_args:Nx\label{sref_\l_tmpa_str}
909
910      \exp_args:NNNx\immediate\write\@auxout{\stexauxadddocref{\l_tmpa_str}}
911      \str_gset:cx {sref_\l_tmpa_str _type}\c__stex_refs_ref_str
912    }
913  }
914  \cs_new_protected:Npn \stexauxadddocref #1 {
915    \str_set:Nx \l_tmpa_str {#1}
916    \str_gset_eq:cN{sref_\l_tmpa_str _type}\c__stex_refs_ref_str
917    \seq_gput_right:Nx \g__stex_refs_all_refs_seq {\l_tmpa_str}
918  }
919  \cs_new_protected:Nn \stex_ref_new_sym_target:n {
920    \stex_get_document_uri:
921    \stex_if_smsmode:TF {
922      \stex_get_document_url:
923      \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str
924      \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
925
926    }{
927      \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=~\expandafter{\@currentlabel\iffalse}{
928      \exp_args:Nx\label{sref_sym_#1}
929
930      \exp_args:NNNx\immediate\write\@auxout{\stexauxadddocref{sym_#1}}
931      \str_gset:cx {sref_sym_#1_type}\c__stex_refs_ref_str
932    }
933  }
```

27.3   Using References

```
934  \str_new:N \l__stex_refs_indocument_str
935  \keys_define:nn { stex / sref } {
936    linktext      .tl_set:N  = \l__stex_refs_linktext_tl ,
937    fallback      .tl_set:N  = \l__stex_refs_fallback_tl ,
938    pre           .tl_set:N  = \l__stex_refs_pre_tl ,
939    post          .tl_set:N  = \l__stex_refs_post_tl ,
940    %indoc         .str_set_x:N  = \l__stex_refs_repo_str ,
941  }
942
943  \bool_new:N \c__stex_refs_hyperref_bool
944  \bool_set_false:N \c__stex_refs_hyperref_bool
945  \AddToHook{begindocument}{
946    \@ifpackageloaded{hyperref}{
947      \bool_set_true:N \c__stex_refs_hyperref_bool
948    }{}
949  }
950
```

```
951
952 \cs_new_protected:Nn \__stex_refs_args:n {
953   \tl_clear:N \l__stex_refs_linktext_tl
954   \tl_clear:N \l__stex_refs_fallback_tl
955   \tl_clear:N \l__stex_refs_pre_tl
956   \tl_clear:N \l__stex_refs_post_tl
957   \str_clear:N \l__stex_refs_repo_str
958   \keys_set:nn { stex / sref } { #1 }
959 }
960
961 \NewDocumentCommand \sref { O{} m}{
962   \__stex_refs_args:n { #1 }
963   \str_if_empty:NTF \l__stex_refs_indocument_str {
964     \str_set:Nn \l_tmpa_str { #2 }
965     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
966     \tl_set:Nn \l_tmpa_tl {
967       \l__stex_refs_fallback_tl
968     }
969     \seq_map_inline:Nn \g__stex_refs_all_refs_seq {
970       \str_set:Nn \l_tmpb_str { ##1 }
971       \str_if_eq:eeT { \l_tmpa_str } {
972         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int }{ -1 }
973       } {
974         \seq_map_break:n {
975           \tl_set:Nn \l_tmpa_tl {
976             % doc uri in \l_tmpb_str
977             \str_set:Nx \l_tmpa_str {\use:c{sref_\l_tmpb_str _type}}
978             \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
979               % reference
980               \cs_if_exist:cTF{autoref}{
981                 \l__stex_refs_pre_tl\autoref{sref_\l_tmpb_str}\l__stex_refs_post_tl
982               }{
983                 \l__stex_refs_pre_tl\ref{sref_\l_tmpb_str}\l__stex_refs_post_tl
984               }
985             }{
986               % URL
987               \if_bool:N \c__stex_refs_hyperref_bool {
988                 \exp_args:Nx \href{\use:c{sref_url_\l_tmpb_str _str}}{\l__stex_refs_fallback
989               }{
990                 \l__stex_refs_fallback_tl
991               }
992             }
993           }
994         }
995       }
996     }
997     \l_tmpa_tl
998   }{
999     % TODO
1000   }
1001 }
1002
1003 \NewDocumentCommand \srefsym { O{} m}{
1004   \stex_get_symbol:n { #2 }
```

```
1005     \__stex_refs_args:n { #1 }
1006     \str_if_empty:NTF \l__stex_refs_indocument_str {
1007       \tl_set:Nn \l_tmpa_tl {
1008         \l__stex_refs_fallback_tl
1009       }
1010       \tl_if_exist:cT{sref_sym_\l_stex_get_symbol_uri_str _type}{
1011         \tl_set:Nn \l_tmpa_tl {
1012           % doc uri in \l_tmpb_str
1013           \str_set:Nx \l_tmpa_str {\use:c{sref_sym_\l_stex_get_symbol_uri_str _type}}
1014           \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1015             % reference
1016             \cs_if_exist:cTF{autoref}{
1017               \l__stex_refs_pre_tl\autoref{sref_sym_\l_stex_get_symbol_uri_str}\l__stex_refs_p
1018             }{
1019               \l__stex_refs_pre_tl\ref{sref_sym_\l_stex_get_symbol_uri_str}\l__stex_refs_post_
1020             }
1021           }{
1022             % URL
1023             \if_bool:N \c__stex_refs_hyperref_bool {
1024               \exp_args:Nx \href{\use:c{sref_sym_url_\l_stex_get_symbol_uri_str _str}}{\l__ste
1025             }{
1026               \l__stex_refs_fallback_tl
1027             }
1028           }
1029         }
1030       }
1031       \l_tmpa_tl
1032     }{
1033       % TODO
1034     }
1035   }
1036
1037   \cs_new_protected:Npn \srefsymuri #1 #2 {
1038     \hyperref[sref_sym_#1]{#2}
1039   }
1040
1041   ⟨/package⟩
```

97

Chapter 28

sTEX
-Modules Implementation

1042 ⟨∗package⟩

1043

1044 %%%%%%%%%%%%   modules.dtx   %%%%%%%%%%%%

1045

1046 ⟨@@=stex_modules⟩

Warnings and error messages

1047 \msg_new:nnn{stex}{error/unknownmodule}{

1048   No~module~#1~found

1049 }

1050 \msg_new:nnn{stex}{error/syntax}{

1051   Syntax~error:~#1

1052 }

1053 \msg_new:nnn{stex}{error/siglanguage}{

1054   Module~#1~declares~signature~#2,~but~does~not~

1055   declare~its~language

1056 }

1057

1058 \msg_new:nnn{stex}{error/conclictingmodules}{

1059   Comflicting~imports~for~module~#1

1060 }

`\l_stex_current_module_str`   The current module:

1061 \str_new:N \l_stex_current_module_str

(*End definition for* `\l_stex_current_module_str`*. This variable is documented on page 26.*)

`\l_stex_all_modules_seq`   Stores all available modules

1062 \seq_new:N \l_stex_all_modules_seq

(*End definition for* `\l_stex_all_modules_seq`*. This variable is documented on page 26.*)

`\stex_if_in_module_p:`
`\stex_if_in_module:TF`   1063 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {

1064   \str_if_empty:NTF \l_stex_current_module_str

1065   \prg_return_false: \prg_return_true:

1066 }

(End definition for `\stex_if_in_module:TF`*. This function is documented on page 27.)*

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

```
1067 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1068   \prop_if_exist:cTF { c_stex_module_#1_prop }
1069     \prg_return_true: \prg_return_false:
1070 }
```

(End definition for `\stex_if_module_exists:nTF`*. This function is documented on page 27.)*

`\stex_add_to_current_module:n`
`\STEXexport`

Only allowed within modules:

```
1071 \cs_new_protected:Nn \stex_add_to_current_module:n {
1072   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1073 }
1074 \cs_new_protected:Npn \STEXexport {
1075   \begingroup
1076   \newlinechar=-1\relax
1077   \endlinechar=-1\relax
1078   %\catcode'\ = 9\relax
1079   \expandafter\endgroup\STEXexport:n
1080 }
1081 \cs_new_protected:Nn \STEXexport:n {
1082   \ignorespaces #1
1083   \stex_add_to_current_module:n { \ignorespaces #1 }
1084   \stex_smsmode_set_codes:
1085 }
1086 \stex_deactivate_macro:Nn \STEXexport {module~environments}
```

(End definition for `\stex_add_to_current_module:n` *and* `\STEXexport`*. These functions are documented on page 27.)*

`\stex_add_constant_to_current_module:n`

```
1087 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1088   \str_set:Nx \l_tmpa_str { #1 }
1089   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
1090 }
1091
1092 %\cs_new_protected:Nn \stex_add_field_to_current_module:n {
1093 %  \str_set:Nx \l_tmpa_str { #1 }
1094 %  \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _fields} { \l_tmpa_str }
1095 %}
```

(End definition for `\stex_add_constant_to_current_module:n`*. This function is documented on page 27.)*

`\stex_collect_imports:n`

```
1096 \cs_new_protected:Nn \stex_collect_imports:n {
1097   \seq_clear:N \l_stex_collect_imports_seq
1098   \__stex_modules_collect_imports:n {#1}
1099 }
1100 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1101   \seq_map_inline:cn {c_stex_module_#1_imports} {
1102     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1103       \__stex_modules_collect_imports:n { ##1 }
1104     }
```

```
1105    }
1106    \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1107      \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1108    }
1109 }
```

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

\stex_add_import_to_current_module:n

```
1110 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1111    \str_set:Nx \l_tmpa_str { #1 }
1112    \exp_args:Nno
1113    \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1114      \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1115    }
1116 }
```

(*End definition for* \stex_add_import_to_current_module:n. *This function is documented on page* *27*.)

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

```
1117 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
1118    \str_set:Nx \l_tmpa_str { #1 }
1119    \seq_set_eq:NN \l_tmpa_seq #2
1120    % split off file extension
1121    \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1122    \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1123    \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1124    \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1125
1126    \bool_set_true:N \l_tmpa_bool
1127    \bool_while_do:Nn \l_tmpa_bool {
1128      \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1129      \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1130        {source} { \bool_set_false:N \l_tmpa_bool }
1131      }{}{
1132        \seq_if_empty:NT \l_tmpa_seq {
1133          \bool_set_false:N \l_tmpa_bool
1134        }
1135      }
1136    }
1137
1138    \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
1139    \str_if_empty:NTF \l_stex_modules_subpath_str {
1140      \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
1141    }{
1142      \str_set:Nx \l_stex_modules_ns_str {
1143        \l_tmpa_str/\l_stex_modules_subpath_str
1144      }
1145    }
1146 }
```

(*End definition for* \stex_modules_compute_namespace:nN. *This function is documented on page* *27*.)

Stores its return values in:

```
1147 \str_new:N \l_stex_modules_ns_str
1148 \str_new:N \l_stex_modules_subpath_str
```

(*End definition for* \l_stex_modules_ns_str *and* \l_stex_modules_subpath_str*. These variables are documented on page* **??***.*)

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

```
1149 \cs_new_protected:Nn \stex_modules_current_namespace: {
1150   \str_clear:N \l_stex_modules_subpath_str
1151   \prop_if_exist:NTF \l_stex_current_repository_prop {
1152     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1153     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1154   }{
1155     % split off file extension
1156     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1157     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1158     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1159     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1160     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1161     \str_set:Nx \l_stex_modules_ns_str {
1162       file:/\stex_path_to_string:N \l_tmpa_seq
1163     }
1164   }
1165 }
```

(*End definition for* \stex_modules_current_namespace:*. This function is documented on page* *27**.*)

28.1   The module environment

module arguments:

```
1166 \keys_define:nn { stex / module } {
1167   title         .str_set_x:N  = \l_stex_module_title_str ,
1168   ns            .str_set_x:N  = \l_stex_module_ns_str ,
1169   lang          .str_set_x:N  = \l_stex_module_lang_str ,
1170   sig           .str_set_x:N  = \l_stex_module_sig_str ,
1171   creators      .str_set_x:N  = \l_stex_module_creators_str ,
1172   contributors  .str_set_x:N  = \l_stex_module_contributors_str ,
1173   meta          .str_set_x:N  = \l_stex_module_meta_str ,
1174   srccite       .str_set_x:N  = \l_stex_module_srccite_str
1175 }
1176
1177 \cs_new_protected:Nn \__stex_modules_args:n {
1178   \str_clear:N \l_stex_module_title_str
1179   \str_clear:N \l_stex_module_ns_str
1180   \str_clear:N \l_stex_module_lang_str
1181   \str_clear:N \l_stex_module_sig_str
1182   \str_clear:N \l_stex_module_creators_str
1183   \str_clear:N \l_stex_module_contributors_str
1184   \str_clear:N \l_stex_module_meta_str
1185   \str_clear:N \l_stex_module_srccite_str
1186   \keys_set:nn { stex / module } { #1 }
```

101

```
1187 }
1188
1189 % module parameters here? In the body?
1190
```

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

```
1191 \cs_new_protected:Nn \stex_module_setup:nn {
1192   \str_set:Nx \l_stex_module_name_str { #2 }
1193   \__stex_modules_args:n { #1 }
```

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

Are we in a nested module?

```
1194   \stex_if_in_module:TF {
1195     % Nested module
1196     \prop_get:cnN {c_stex_module_\l_stex_current_module_str _prop}
1197       { ns } \l_stex_module_ns_str
1198     \str_set:Nx \l_stex_module_name_str {
1199       \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
1200         { name } / \l_stex_module_name_str
1201     }
1202   }{
1203     % not nested:
1204     \str_if_empty:NT \l_stex_module_ns_str {
1205       \stex_modules_current_namespace:
1206       \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1207       \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1208         / {\l_stex_module_ns_str}
1209       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1210       \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1211         \str_set:Nx \l_stex_module_ns_str {
1212           \stex_path_to_string:N \l_tmpa_seq
1213         }
1214       }
1215     }
1216   }
```

Next, we determine the language of the module:

```
1217   \str_if_empty:NT \l_stex_module_lang_str {
1218     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1219     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1220     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1221     \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1222     \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1223       \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1224         inferred~from~file~name}
1225       \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1226     }
1227   }
1228
1229   \str_if_empty:NF \l_stex_module_lang_str {
1230     \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1231       \l_tmpa_str {
1232         \ltx@ifpackageloaded{babel}{
1233           \exp_args:Nx \selectlanguage { \l_tmpa_str }
```

```
1234        }{}
1235      } {
1236        \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
1237      }
1238   }
```

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

```
1239   \str_if_empty:NTF \l_stex_module_sig_str {
1240     \exp_args:Nnx \prop_gset_from_keyval:cn {
1241       c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1242     } {
1243       name      = \l_stex_module_name_str ,
1244       ns        = \l_stex_module_ns_str ,
1245       file      = \exp_not:o { \g_stex_currentfile_seq } ,
1246       lang      = \l_stex_module_lang_str ,
1247       sig       = \l_stex_module_sig_str ,
1248       meta      = \l_stex_module_meta_str
1249     }
1250     \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1251     \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _fields}
1252     \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1253     \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1254     \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
```

We load the metatheory:

```
1255     \str_if_empty:NT \l_stex_module_meta_str {
1256       \str_set:Nx \l_stex_module_meta_str {
1257         \c_stex_metatheory_ns_str ? Metatheory
1258       }
1259     }
1260     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1261       \bool_set_true:N \l_stex_in_meta_bool
1262       \exp_args:Nx \stex_add_to_current_module:n {
1263         \bool_set_true:N \l_stex_in_meta_bool
1264         \stex_activate_module:n {\l_stex_module_meta_str}
1265         \bool_set_false:N \l_stex_in_meta_bool
1266       }
1267       \stex_activate_module:n {\l_stex_module_meta_str}
1268       \bool_set_false:N \l_stex_in_meta_bool
1269     }
1270   }{
1271     \str_if_empty:NT \l_stex_module_lang_str {
1272       \msg_error:nnxx{stex}{error/siglanguage}{
1273         \l_stex_module_ns_str?\l_stex_module_name_str
1274       }{\l_stex_module_sig_str}
1275     }
1276
1277     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1278     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1279     \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1280     \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1281     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1282     \str_set:Nx \l_tmpa_str {
```

```
1283        \stex_path_to_string:N \l_tmpa_seq /
1284        \l_tmpa_str . \l_stex_module_sig_str .tex
1285      }
1286      \IfFileExists \l_tmpa_str {
1287        \exp_args:No \stex_in_smsmode:nn { \l_tmpa_str } {
1288          \seq_clear:N \l_stex_all_modules_seq
1289          \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1290          \input { \l_tmpa_str }
1291        }
1292      }{
1293        \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1294      }
1295      \stex_if_smsmode:F {
1296        \stex_activate_module:n {
1297          \l_stex_module_ns_str ? \l_stex_module_name_str
1298        }
1299      }
1300      \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1301    }
1302  }
```

(*End definition for* `\stex_module_setup:nn`*. This function is documented on page* *28*.)

module   The module environment.

__stex_modules_begin_module:nn   implements \begin{module}

```
1303  \int_new:N \l_stex_module_group_depth_int
1304  \cs_new_protected:Nn \__stex_modules_begin_module:nn {
1305    \stex_reactivate_macro:N \STEXexport
1306    \stex_reactivate_macro:N \importmodule
1307    \stex_reactivate_macro:N \symdecl
1308    \stex_reactivate_macro:N \notation
1309    \stex_reactivate_macro:N \symdef
1310    \stex_module_setup:nn{#1}{#2}
1311
1312    \stex_debug:nn{modules}{
1313      New~module:\\
1314      Namespace:~\l_stex_module_ns_str\\
1315      Name:~\l_stex_module_name_str\\
1316      Language:~\l_stex_module_lang_str\\
1317      Signature:~\l_stex_module_sig_str\\
1318      Metatheory:~\l_stex_module_meta_str\\
1319      File:~\stex_path_to_string:N \g_stex_currentfile_seq
1320    }
1321
1322    \seq_put_right:Nx \l_stex_all_modules_seq {
1323      \l_stex_module_ns_str ? \l_stex_module_name_str
1324    }
1325
1326  % \seq_gput_right:Nx  \g_stex_modules_in_file_seq
1327  %      { \l_stex_module_ns_str ? \l_stex_module_name_str }
1328
1329
1330    \stex_if_smsmode:TF {
```

```
1331      \stex_smsmode_set_codes:
1332    } {
1333      \begin{stex_annotate_env} {theory} {
1334        \l_stex_module_ns_str ? \l_stex_module_name_str
1335      }
1336
1337      \stex_annotate_invisible:nnn{header}{} {
1338        \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1339        \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1340        \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1341          \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1342        }
1343      }
1344    }
1345    \int_set:Nn \l_stex_module_group_depth_int {\currentgrouplevel}
1346    % TODO: Inherit metatheory for nested modules?
1347 }
1348 \iffalse \end{stex_annotate_env} \fi %^^A make syntax highlighting work again
```

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

`\__stex_modules_end_module:`   implements `\end{module}`

```
1349 \cs_new_protected:Nn \__stex_modules_end_module: {
1350 %   \str_set:Nx \l_tmpa_str {
1351 %     c_stex_module_
1352 %     \prop_item:Nn \l_stex_current_module_prop { ns } ?
1353 %     \prop_item:Nn \l_stex_current_module_prop { name }
1354 %     _prop
1355 %   }
1356   %^^A \prop_new:c { \l_tmpa_str }
1357 %   \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
1358   \stex_debug:nn{modules}{Closing~module~\prop_item:cn {c_stex_module_\l_stex_current_module
1359 }
```

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

`@module`   The core environment, with no header

```
1360 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1361 \NewDocumentEnvironment { @module } { O{} m } {
1362   \par
1363   \__stex_modules_begin_module:nn{#1}{#2}
1364 } {
1365   \__stex_modules_end_module:
1366   \stex_if_smsmode:TF {
1367 %     \exp_args:Nx \stex_add_to_sms:n {
1368 %       \prop_gset_from_keyval:cn {
1369 %         c_stex_module_
1370 %         \prop_item:Nn \l_stex_current_module_prop { ns } ?
1371 %         \prop_item:Nn \l_stex_current_module_prop { name }
1372 %         _prop
1373 %       } {
1374 %       name      = \prop_item:cn { \l_tmpa_str } { name } ,
1375 %       ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
1376 %       file      = \prop_item:cn { \l_tmpa_str } { file } ,
```

```
1377 %            lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
1378 %            sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
1379 %            meta      = \prop_item:cn { \l_tmpa_str } { meta }
1380 %          }
1381 %      }
1382   }{
1383     \end{stex_annotate_env}
1384   }
1385 }
```

\stex_modules_heading: Code for document headers

```
1386 \cs_if_exist:NTF \thesection {
1387   \newcounter{module}[section]
1388 }{
1389   \newcounter{module}
1390 }
1391
1392 \bool_if:NT \c_stex_showmods_bool {
1393   \latexml_if:F { \RequirePackage{mdframed} }
1394 }
1395
1396 \cs_new_protected:Nn \stex_modules_heading: {
1397   \stepcounter{module}
1398   \par
1399   \bool_if:NT \c_stex_showmods_bool {
1400     \noindent{\textbf{Module} ~
1401       \cs_if_exist:NT \thesection {\thesection.}
1402       \themodule ~ [\l_stex_module_name_str]
1403     }
1404     \str_if_empty:NTF \l_stex_module_title_str {
1405     }{
1406       \quad(\l_stex_module_title_str)\hfill
1407     }\par
1408   }
1409   \edef\@currentlabel{Module~\thesection.\themodule~[\l_stex_module_name_str]}
1410   % TODO
1411   \stex_ref_new_doc_target:n \l_stex_module_name_str
1412 }
```

(*End definition for* \stex_modules_heading:. *This function is documented on page* 28.)
     Finally:

```
1413 \NewDocumentEnvironment { module } { O{} m } {
1414   \bool_if:NT \c_stex_showmods_bool {
1415     \begin{mdframed}
1416   }
1417   \begin{@module}[#1]{#2}
1418   \stex_modules_heading:
1419 }{
1420   \end{@module}
1421   \bool_if:NT \c_stex_showmods_bool {
1422     \end{mdframed}
1423   }
1424 }
```

28.2   Invoking modules

```
1425 \NewDocumentCommand \STEXModule { m } {
1426   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1427   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1428   \tl_set:Nn \l_tmpa_tl {
1429     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1430   }
1431   \seq_map_inline:Nn \l_stex_all_modules_seq {
1432     \str_set:Nn \l_tmpb_str { ##1 }
1433     \str_if_eq:eeT { \l_tmpa_str } {
1434       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1435     } {
1436       \seq_map_break:n {
1437         \tl_set:Nn \l_tmpa_tl {
1438           \stex_invoke_module:n { ##1 }
1439         }
1440       }
1441     }
1442   }
1443   \l_tmpa_tl
1444 }
1445
1446 \cs_new_protected:Nn \stex_invoke_module:n {
1447   \stex_debug:nn{modules}{Invoking~module~#1}
1448   \peek_charcode_remove:NTF ! {
1449     \__stex_modules_invoke_uri:nN { #1 }
1450   } {
1451     \peek_charcode_remove:NTF ? {
1452       \__stex_modules_invoke_symbol:nn { #1 }
1453     } {
1454       \msg_error:nnx{stex}{error/syntax}{
1455         ?~or~!~expected~after~
1456         \c_backslash_str STEXModule{#1}
1457       }
1458     }
1459   }
1460 }
1461
1462 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1463   \str_set:Nn #2 { #1 }
1464 }
1465
1466 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1467   \stex_invoke_symbol:n{#1?#2}
1468 }
```

(*End definition for* \STEXModule *and* \stex_invoke_module:n. *These functions are documented on page* *29*.)

```
1469 \bool_new:N \l_stex_in_meta_bool
1470 \bool_set_false:N \l_stex_in_meta_bool
```

107

```
1471 \cs_new_protected:Nn \stex_activate_module:n {
1472   \stex_debug:nn{modules}{Activating~module~#1}
1473   \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
1474     \msg_error:nnn{stex}{error/conclictingmodules}{ #1 }
1475   }
1476   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1477     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1478     \use:c{ c_stex_module_#1_code }
1479   }
1480 }
```

(*End definition for* `\stex_activate_module:n`*. This function is documented on page* *30*.)

```
1481 ⟨/package⟩
```

Chapter 29

SТEX
-Module Inheritance
Implementation

```
1482 ⟨*package⟩
1483
1484 %%%%%%%%%%%%   inheritance.dtx   %%%%%%%%%%%%
1485
```

29.1   SMS Mode

```
1486 ⟨@@=stex_smsmode⟩
```

```
1487 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1488 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1489 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1490
1491 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1492    \makeatletter
1493    \makeatother
1494    \ExplSyntaxOn
1495    \ExplSyntaxOff
1496 }
1497
1498 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1499    \symdef
1500    \importmodule
1501    \notation
1502    \symdecl
1503    \STEXexport
1504 }
1505
1506 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1507    \tl_to_str:n {
1508       module,
1509       @module
```

109

```
1510    }
1511 }
```

(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 31.)*

`\stex_if_smsmode_p:`
`\stex_if_smsmode:TF`

```
1512 \bool_new:N \g__stex_smsmode_bool
1513 \bool_set_false:N \g__stex_smsmode_bool
1514 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1515    \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1516 }
```

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

`\__stex_smsmode_if_catcodes_p:`
`\__stex_smsmode_if_catcodes:TF`

Checks whether the SMS mode category code scheme is active.

```
1517 \bool_new:N \g__stex_smsmode_catcode_bool
1518 \bool_set_false:N \g__stex_smsmode_catcode_bool
1519 \prg_new_conditional:Nnn \__stex_smsmode_if_catcodes: { p, T, F, TF } {
1520    \bool_if:NTF \g__stex_smsmode_catcode_bool
1521       \prg_return_true: \prg_return_false:
1522 }
```

(End definition for `\__stex_smsmode_if_catcodes:TF`.)*

`\stex_smsmode_set_codes:`

```
1523 \cs_new_protected:Nn \stex_smsmode_set_codes: {
1524    \stex_if_smsmode:T {
1525       \__stex_smsmode_if_catcodes:F {
1526          \bool_gset_true:N \g__stex_smsmode_catcode_bool
1527          \exp_after:wN \char_gset_active_eq:NN
1528             \c_backslash_str \__stex_smsmode_cs:
1529          \tex_global:D \char_set_catcode_active:N \\
1530          \tex_global:D \char_set_catcode_other:N $
1531          \tex_global:D \char_set_catcode_other:N ^
1532          \tex_global:D \char_set_catcode_other:N _
1533          \tex_global:D \char_set_catcode_other:N &
1534          \tex_global:D \char_set_catcode_other:N ##
1535       }
1536    }
1537 } \iffalse $ \fi % to make syntax highlighting work again
```

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

`\__stex_smsmode_unset_codes:`

Sets category code scheme back from the one used in SMS mode.

```
1538 \cs_new_protected:Nn \__stex_smsmode_unset_codes: {
1539    \__stex_smsmode_if_catcodes:T {
1540       \bool_gset_false:N \g__stex_smsmode_catcode_bool
1541       \exp_after:wN \tex_global:D \exp_after:wN
1542          \char_set_catcode_escape:N \c_backslash_str
1543       \tex_global:D \char_set_catcode_math_toggle:N $
1544       \tex_global:D \char_set_catcode_math_superscript:N ^
1545       \tex_global:D \char_set_catcode_math_subscript:N _
1546       \tex_global:D \char_set_catcode_alignment:N &
1547       \tex_global:D \char_set_catcode_parameter:N ##
1548    }
1549 } \iffalse $ \fi % to make syntax highlighting work again
```

110

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

`\stex_in_smsmode:nn`

```
1550 \cs_new_protected:Nn \stex_in_smsmode:nn {
1551   \vbox_set:Nn \l_tmpa_box {
1552     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1553     \bool_gset_true:N \g__stex_smsmode_bool
1554     \stex_smsmode_set_codes:
1555     #2
1556     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1557     \stex_if_smsmode:F {
1558       \__stex_smsmode_unset_codes:
1559     }
1560   }
1561   \box_clear:N \l_tmpa_box
1562 }
```

(End definition for `\stex_in_smsmode:nn`*. This function is documented on page 32.)*

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

```
1563 \cs_new_protected:Nn \__stex_smsmode_cs: {
1564   \str_clear:N \l_tmpa_str
1565   \peek_analysis_map_inline:n {
1566     % #1: token (one expansion)
1567     % #2: charcode
1568     % #3 catcode
1569     \token_if_eq_charcode:NNTF ##3 B {
1570       % token is a letter
1571       \exp_args:NNo \str_put_right:Nn \l_tmpa_str { ##1 }
1572     } {
1573       \str_if_empty:NTF \l_tmpa_str {
1574         % we don't allow (or need) single non-letter CSs
1575         % for now
1576         \peek_analysis_map_break:
1577       }{
1578         \str_if_eq:onTF \l_tmpa_str { begin } {
1579           \peek_analysis_map_break:n {
1580             \exp_after:wN \__stex_smsmode_checkbegin:n ##1
1581           }
1582         } {
1583           \str_if_eq:onTF \l_tmpa_str { end } {
1584             \peek_analysis_map_break:n {
1585               \exp_after:wN \__stex_smsmode_checkend:n ##1
1586             }
1587           } {
1588             \tl_set:Nn \l_tmpa_tl { \use:c{\l_tmpa_str} }
1589             \exp_args:NNNo \exp_args:NNo \tl_if_in:NnTF
1590               \g_stex_smsmode_allowedmacros_tl
1591                 { \use:c{\l_tmpa_str} } {
1592                 \stex_debug:nn{modules}{Executing~1:~\l_tmpa_str}
1593                 \peek_analysis_map_break:n {
1594                   \exp_after:wN \l_tmpa_tl ##1
1595                 }
```

111

```
1596                    } {
1597                      \exp_args:NNNo \exp_args:NNo \tl_if_in:NnTF
1598                      \g_stex_smsmode_allowedmacros_escape_tl
1599                        { \use:c{\l_tmpa_str} } {
1600                        \__stex_smsmode_unset_codes:
1601                        \stex_debug:nn{modules}{Executing~2:~\l_tmpa_str}
1602                        % TODO \__stex_smsmode_rescan_cs:
1603 %                       \int_compare:nNnTF {##2} = {92} {
1604 %                         \peek_analysis_map_break:n {
1605 %                           \__stex_smsmode_unset_codes:
1606 %                           \__stex_smsmode_rescan_cs:
1607 %                         }
1608 %                       } {
1609                          \peek_analysis_map_break:n {
1610                            \exp_after:wN \l_tmpa_tl ##1
1611                          }
1612 %                       }
1613                      } {
1614                        \int_compare:nNnTF {##2} = {92} {
1615                          \peek_analysis_map_break:n { \__stex_smsmode_cs: }
1616                        }{
1617                          \peek_analysis_map_break:n { \exp_after:wN\relax ##1 }
1618                        }
1619                      }
1620                    }
1621                  }
1622                }
1623              }
1624            }
1625          }
1626 }
```

(*End definition for* __stex_smsmode_cs:.)

__stex_smsmode_rescan_cs: If the last token gobbled by \stex_smsmode_cs: happened to be a \, we need to rescan the cs name and reinsert it into the input stream:

```
1627 \cs_new_protected:Nn \__stex_smsmode_rescan_cs: {
1628   \str_clear:N \l_tmpb_str
1629   \peek_analysis_map_inline:n {
1630     \token_if_eq_charcode:NNTF ##3 B {
1631       % token is a letter
1632       \exp_args:NNo \str_put_right:Nn \l_tmpb_str { ##1 }
1633     } {
1634       \peek_analysis_map_break:n {
1635         \exp_after:wN \use:c \exp_after:wN {
1636           \exp_after:wN \l_tmpa_str\exp_after:wN
1637         } \use:c { \l_tmpb_str \exp_after:wN } ##1
1638       }
1639     }
1640   }
1641 }
```

(*End definition for* __stex_smsmode_rescan_cs:.)

__stex_smsmode_checkbegin:n called on \begin; checks whether the environment being opened is allowed in SMS mode.

```
1642 \cs_new_protected:Nn \__stex_smsmode_checkbegin:n {
1643   \str_set:Nn \l_tmpa_str { #1 }
1644   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1645     \__stex_smsmode_unset_codes:
1646     \begin{#1}
1647   }
1648 }
```

(*End definition for* __stex_smsmode_checkbegin:n.)

__stex_smsmode_checkend:n called on \end; checks whether the environment being opened is allowed in SMS mode.

```
1649 \cs_new_protected:Nn \__stex_smsmode_checkend:n {
1650   \str_set:Nn \l_tmpa_str { #1 }
1651   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1652     \end{#1}
1653   }
1654 }
```

(*End definition for* __stex_smsmode_checkend:n.)

29.2   Inheritance

```
1655 ⟨@@=stex_importmodule⟩
```

\stex_import_module_uri:nn

```
1656 \cs_new_protected:Nn \stex_import_module_uri:nn {
1657   \str_set:Nx \l_stex_import_archive_str { #1 }
1658   \str_set:Nn \l_stex_import_path_str { #2 }
1659
1660   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1661   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1662   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1663
1664   \stex_modules_current_namespace:
1665   \bool_lazy_all:nTF {
1666     {\str_if_empty_p:N \l_stex_import_archive_str}
1667     {\str_if_empty_p:N \l_stex_import_path_str}
1668     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1669   }{
1670     \str_set_eq:NN \l_stex_import_path_str \l_stex_modules_subpath_str
1671     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1672   }{
1673     \str_if_empty:NT \l_stex_import_archive_str {
1674       \prop_if_exist:NT \l_stex_current_repository_prop {
1675         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1676       }
1677     }
1678     \str_if_empty:NTF \l_stex_import_archive_str {
1679       \str_if_empty:NF \l_stex_import_path_str {
1680         \str_set:Nx \l_stex_import_ns_str {
1681           \l_stex_module_ns_str / \l_stex_import_path_str
1682         }
1683       }
```

```
1684        }{
1685          \stex_require_repository:n \l_stex_import_archive_str
1686          \prop_get:cnN { c_stex_mathhub_\l_stex_import_archive_str _manifest_prop } { ns }
1687            \l_stex_import_ns_str
1688          \str_if_empty:NF \l_stex_import_path_str {
1689            \str_set:Nx \l_stex_import_ns_str {
1690              \l_stex_import_ns_str / \l_stex_import_path_str
1691            }
1692          }
1693        }
1694      }
1695  }
```

(End definition for `\stex_import_module_uri:nn`. *This function is documented on page 34.)*

`\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`.

```
1696  \str_new:N \l_stex_import_name_str
1697  \str_new:N \l_stex_import_archive_str
1698  \str_new:N \l_stex_import_path_str
1699  \str_new:N \l_stex_import_ns_str
```

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

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

```
1700  \cs_new_protected:Nn \stex_import_require_module:nnnn {
1701    \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1702
1703      % archive
1704      \str_set:Nx \l_tmpa_str { #2 }
1705      \str_if_empty:NTF \l_tmpa_str {
1706        \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1707      } {
1708        \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1709        \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1710        \seq_put_right:Nn \l_tmpa_seq { source }
1711      }
1712
1713      % path
1714      \str_set:Nx \l_tmpb_str { #3 }
1715      \str_if_empty:NTF \l_tmpb_str {
1716        \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1717
1718        \ltx@ifpackageloaded{babel} {
1719          \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1720            { \languagename } \l_tmpb_str {
1721              \msg_error:nnx{stex}{error/unknownlanguage}{\languagename}
1722            }
1723        } {
1724          \str_clear:N \l_tmpb_str
1725        }
1726
1727        \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1728        \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1729          \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
```

```
1730       }{
1731         \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1732         \IfFileExists{ \l_tmpa_str.tex }{
1733           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1734         }{
1735           % try english as default
1736           \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1737           \IfFileExists{ \l_tmpa_str.en.tex }{
1738             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1739           }{
1740             \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1741           }
1742         }
1743       }

1744
1745     } {
1746       \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1747       \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq

1748
1749       \ltx@ifpackageloaded{babel} {
1750         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1751           { \languagename } \l_tmpb_str {
1752             \msg_error:nnx{stex}{error/unknownlanguage}{\languagename}
1753         }
1754       } {
1755         \str_clear:N \l_tmpb_str
1756       }

1757
1758       \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str

1759
1760       \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1761       \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1762         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1763       }{
1764         \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1765         \IfFileExists{ \l_tmpa_str/#4.tex }{
1766           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1767         }{
1768           % try english as default
1769           \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1770           \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1771             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1772           }{
1773             \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1774             \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1775               \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1776             }{
1777               \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1778               \IfFileExists{ \l_tmpa_str.tex }{
1779                 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1780               }{
1781                 % try english as default
1782                 \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1783                 \IfFileExists{ \l_tmpa_str.en.tex }{
```

```
1784                    \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1785                  }{
1786                    \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1787                  }
1788                }
1789              }
1790            }
1791          }
1792        }
1793      }

1794

1795      \exp_args:No \stex_in_smsmode:nn { \g__stex_importmodule_file_str } {
1796        \seq_clear:N \l_stex_all_modules_seq
1797        \str_clear:N \l_stex_current_module_str
1798        \str_set:Nx \l_tmpb_str { #2 }
1799        \str_if_empty:NF \l_tmpb_str {
1800          \stex_set_current_repository:n { #2 }
1801        }
1802        \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1803        \input { \g__stex_importmodule_file_str }
1804      }

1805

1806      \stex_if_module_exists:nF { #1 ? #4 } {
1807        \msg_error:nnx{stex}{error/unknownmodule}{
1808          #1?#4~(in~file~\g__stex_importmodule_file_str)
1809        }
1810      }
1811    }
1812    \stex_activate_module:n { #1 ? #4 }
1813 }
```

(*End definition for* \stex_import_require_module:nnnn. *This function is documented on page 34.*)

\importmodule

```
1814 \NewDocumentCommand \importmodule { O{} m } {
1815    \stex_import_module_uri:nn { #1 } { #2 }
1816    \stex_debug:nn{modules}{Importing~module:~
1817      \l_stex_import_ns_str ? \l_stex_import_name_str
1818    }
1819    \stex_if_smsmode:F {
1820      \stex_import_require_module:nnnn
1821      { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1822      { \l_stex_import_path_str } { \l_stex_import_name_str }
1823      \stex_annotate_invisible:nnn
1824        {import} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1825    }
1826    \exp_args:Nx \stex_add_to_current_module:n {
1827      \stex_import_require_module:nnnn
1828      { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1829      { \l_stex_import_path_str } { \l_stex_import_name_str }
1830    }
1831    \exp_args:Nx \stex_add_import_to_current_module:n {
1832      \l_stex_import_ns_str ? \l_stex_import_name_str
1833    }
```

```
1834      \stex_smsmode_set_codes:
1835    }
1836  \stex_deactivate_macro:Nn \importmodule {module~environments}
```

(*End definition for* \importmodule. *This function is documented on page* *32.*)

\usemodule

```
1837  \NewDocumentCommand \usemodule { O{} m } {
1838    \stex_if_smsmode:F {
1839      \stex_import_module_uri:nn { #1 } { #2 }
1840      \stex_import_require_module:nnnn
1841      { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1842      { \l_stex_import_path_str } { \l_stex_import_name_str }
1843      \stex_annotate_invisible:nnn
1844        {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1845    }
1846    \stex_smsmode_set_codes:
1847  }
```

(*End definition for* \usemodule. *This function is documented on page* *33.*)

```
1848  ⟨/package⟩
```

Chapter 30

SᴛᴇX
-Symbols Implementation

⟨∗package⟩

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

Warnings and error messages

30.1   Symbol Declarations

⟨@@=stex_symdecl⟩

\l_stex_all_symbols_seq   Stores all available symbols

\seq_new:N \l_stex_all_symbols_seq

(*End definition for* \l_stex_all_symbols_seq. *This variable is documented on page 36.*)

\STEXsymbol

\NewDocumentCommand \STEXsymbol { m } {
\stex_get_symbol:n { #1 }
\exp_args:No
\stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
}

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

symdecl arguments:

\keys_define:nn { stex / symdecl } {
name   .str_set_x:N = \l_stex_symdecl_name_str ,
local   .bool_set:N = \l_stex_symdecl_local_bool ,
args   .str_set_x:N = \l_stex_symdecl_args_str ,
type   .tl_set:N   = \l_stex_symdecl_type_tl ,
align   .str_set:N   = \l_stex_symdecl_align_str , % TODO(?)
gfc   .str_set:N   = \l_stex_symdecl_gfc_str , % TODO(?)
specializes .str_set:N   = \l_stex_symdecl_specializes_str , % TODO(?)
def   .tl_set:N   = \l_stex_symdecl_definiens_tl
}

```
1871
1872  \bool_new:N \l_stex_symdecl_make_macro_bool
1873
1874  \cs_new_protected:Nn \__stex_symdecl_args:n {
1875    \str_clear:N \l_stex_symdecl_name_str
1876    \str_clear:N \l_stex_symdecl_args_str
1877    \bool_set_false:N \l_stex_symdecl_local_bool
1878    \tl_clear:N \l_stex_symdecl_type_tl
1879    \tl_clear:N \l_stex_symdecl_definiens_tl
1880
1881    \keys_set:nn { stex / symdecl } { #1 }
1882  }
```

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

```
1883
1884  \NewDocumentCommand \symdecl { s O{} m } {
1885    \__stex_symdecl_args:n { #2 }
1886    \IfBooleanTF #1 {
1887      \bool_set_false:N \l_stex_symdecl_make_macro_bool
1888    } {
1889      \bool_set_true:N \l_stex_symdecl_make_macro_bool
1890    }
1891    \stex_symdecl_do:n { #3 }
1892    \stex_smsmode_set_codes:
1893  }
1894  \stex_deactivate_macro:Nn \symdecl {module~environments}
```

(*End definition for* \symdecl. *This function is documented on page 35.*)

\stex_symdecl_do:n

```
1895  \cs_new_protected:Nn \stex_symdecl_do:n {
1896    \stex_if_in_module:F {
1897      % TODO throw error? some default namespace?
1898    }
1899
1900    \str_if_empty:NT \l_stex_symdecl_name_str {
1901      \str_set:Nx \l_stex_symdecl_name_str { #1 }
1902    }
1903
1904    \prop_if_exist:cT { l_stex_symdecl_
1905        \l_stex_current_module_str ?
1906        \l_stex_symdecl_name_str
1907      _prop
1908    }{
1909      % TODO throw error (beware of circular dependencies)
1910    }
1911
1912    \prop_clear:N \l_tmpa_prop
1913    \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
1914    \seq_clear:N \l_tmpa_seq
1915    \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1916    \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1917
```

119

```latex
1918    \exp_args:No \stex_add_constant_to_current_module:n {
1919      \l_stex_symdecl_name_str
1920    }
1921
1922    % arity/args
1923    \int_zero:N \l_tmpb_int
1924
1925    \bool_set_true:N \l_tmpa_bool
1926    \str_map_inline:Nn \l_stex_symdecl_args_str {
1927      \token_case_meaning:NnF ##1 {
1928        0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1929        {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1930        {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1931        {\tl_to_str:n a} {
1932          \bool_set_false:N \l_tmpa_bool
1933          \int_incr:N \l_tmpb_int
1934        }
1935        {\tl_to_str:n B} {
1936          \bool_set_false:N \l_tmpa_bool
1937          \int_incr:N \l_tmpb_int
1938        }
1939      }{
1940        \msg_set:nnn{stex}{error/wrongargs}{
1941          args~value~in~symbol~declaration~for~
1942          \l_stex_current_module_str ?
1943          \l_stex_symdecl_name_str ~
1944          needs~to~be~
1945          i,~a,~b~or~B,~but~##1~given
1946        }
1947        \msg_error:nn{stex}{error/wrongargs}
1948      }
1949    }
1950    \bool_if:NTF \l_tmpa_bool {
1951      % possibly numeric
1952      \str_if_empty:NTF \l_stex_symdecl_args_str {
1953        \prop_put:Nnn \l_tmpa_prop { args } {}
1954        \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1955      }{
1956        \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1957        \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1958        \str_clear:N \l_tmpa_str
1959        \int_step_inline:nn \l_tmpa_int {
1960          \str_put_right:Nn \l_tmpa_str i
1961        }
1962        \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1963      }
1964    } {
1965      \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1966      \prop_put:Nnx \l_tmpa_prop { arity }
1967        { \str_count:N \l_stex_symdecl_args_str }
1968    }
1969    \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
1970
1971
```

```
1972   % semantic macro
1973
1974   \bool_if:NT \l_stex_symdecl_make_macro_bool {
1975     \exp_args:Nx \stex_do_aftergroup:n {
1976       \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1977         \l_stex_current_module_str ? \l_stex_symdecl_name_str
1978       }}
1979     }
1980
1981     \bool_if:NF \l_stex_symdecl_local_bool {
1982       \exp_args:Nx \stex_add_to_current_module:n {
1983         \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1984           \l_stex_current_module_str ? \l_stex_symdecl_name_str
1985         } }
1986       }
1987     }
1988   }
1989
1990   % add to all symbols
1991
1992   \bool_if:NF \l_stex_symdecl_local_bool {
1993     \exp_args:Nx \stex_add_to_current_module:n {
1994       \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1995         \l_stex_current_module_str ? \l_stex_symdecl_name_str
1996       }
1997     }
1998 %     \exp_args:Nx \stex_add_field_to_current_module:n {
1999 %       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2000 %     }
2001   }
2002
2003   \stex_debug:nn{symbols}{New~symbol:~
2004     \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2005     Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2006     Args:~\prop_item:Nn \l_tmpa_prop { args }
2007   }
2008
2009   % circular dependencies require this:
2010
2011   \prop_if_exist:cF {
2012     l_stex_symdecl_
2013     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2014     _prop
2015   } {
2016     \prop_set_eq:cN {
2017       l_stex_symdecl_
2018       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2019       _prop
2020     } \l_tmpa_prop
2021   }
2022
2023   \seq_clear:c {
2024     l_stex_symdecl_
2025     \l_stex_current_module_str ? \l_stex_symdecl_name_str
```

```
2026        _notations
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
2052    \stex_if_smsmode:TF {
2053      \bool_if:NF \l_stex_symdecl_local_bool {
2054 %       \exp_args:Nx \stex_add_to_sms:n {
2055 %         \prop_set_from_keyval:cn {
2056 %           l_stex_symdecl_
2057 %           \l_stex_current_module_str ? \l_stex_symdecl_name_str
2058 %           _prop
2059 %         } {
2060 %           name      = \prop_item:Nn \l_tmpa_prop { name }        ,
2061 %           module    = \prop_item:Nn \l_tmpa_prop { module }      ,
2062 %           local     = \prop_item:Nn \l_tmpa_prop { local }       ,
2063 %           type      = \prop_item:Nn \l_tmpa_prop { type }        ,
2064 %           args      = \prop_item:Nn \l_tmpa_prop { args }        ,
2065 %           arity     = \prop_item:Nn \l_tmpa_prop { arity }       ,
2066 %           assocs    = \prop_item:Nn \l_tmpa_prop { assocs }
2067 %         }
2068 %         \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2069 %           \l_stex_current_module_str ? \l_stex_symdecl_name_str
2070 %         }
2071 %       }
2072      }
2073    }{
2074      \exp_args:Nx \stex_do_aftergroup:n {
2075          \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2076          \l_stex_current_module_str ? \l_stex_symdecl_name_str
2077        }
2078      }
2079      \stex_if_do_html:T {
```

```
2080        \stex_annotate_invisible:nnn {symdecl} {
2081          \l_stex_current_module_str ? \l_stex_symdecl_name_str
2082        } {
2083          \tl_if_empty:NF \l_stex_symdecl_type_tl {\stex_annotate_invisible:nnn{type}{}{$\l_st
2084          \stex_annotate_invisible:nnn{args}{}{
2085            \prop_item:Nn \l_tmpa_prop { args }
2086          }
2087          \stex_annotate_invisible:nnn{macroname}{#1}{}
2088          \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2089            \stex_annotate_invisible:nnn{definiens}{}
2090              {$\l_stex_symdecl_definiens_tl$}
2091          }
2092        }
2093      }
2094    }
2095 }
```

(*End definition for* `\stex_symdecl_do:n`*. This function is documented on page 36.*)

```
2096 \str_new:N \l_stex_get_symbol_uri_str
2097
2098 \cs_new_protected:Nn \stex_get_symbol:n {
2099   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2100     \__stex_symdecl_get_symbol_from_cs:n { #1 }
2101   }{
2102     % argument is a string
2103     % is it a command name?
2104     \cs_if_exist:cTF { #1 }{
2105       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2106       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2107       \str_if_empty:NTF \l_tmpa_str {
2108         \exp_args:Nx \cs_if_eq:NNTF {
2109           \tl_head:N \l_tmpa_tl
2110         } \stex_invoke_symbol:n {
2111           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
2112         }{
2113           \__stex_symdecl_get_symbol_from_string:n { #1 }
2114         }
2115       } {
2116         \__stex_symdecl_get_symbol_from_string:n { #1 }
2117       }
2118     }{
2119       % argument is not a command name
2120       \__stex_symdecl_get_symbol_from_string:n { #1 }
2121       % \l_stex_all_symbols_seq
2122     }
2123   }
2124 }
2125
2126 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2127   \str_set:Nn \l_tmpa_str { #1 }
2128   \bool_set_false:N \l_tmpa_bool
2129   \stex_if_in_module:T {
```

123

```
2130      \exp_args:Nno \seq_if_in:cnT {c_stex_module_\l_stex_current_module_str _constants} { \l_
2131        \bool_set_true:N \l_tmpa_bool
2132        \str_set:Nx \l_stex_get_symbol_uri_str {
2133          \l_stex_current_module_str ? #1
2134        }
2135      }
2136    }
2137    \bool_if:NF \l_tmpa_bool {
2138      \tl_set:Nn \l_tmpa_tl {
2139        \msg_set:nnn{stex}{error/unknownsymbol}{
2140          No~symbol~#1~found!
2141        }
2142        \msg_error:nn{stex}{error/unknownsymbol}
2143      }
2144      \str_set:Nn \l_tmpa_str { #1 }
2145      \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2146      \seq_map_inline:Nn \l_stex_all_symbols_seq {
2147        \str_set:Nn \l_tmpb_str { ##1 }
2148        \str_if_eq:eeT { \l_tmpa_str } {
2149          \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
2150        } {
2151          \seq_map_break:n {
2152            \tl_set:Nn \l_tmpa_tl {
2153              \str_set:Nn \l_stex_get_symbol_uri_str {
2154                ##1
2155              }
2156            }
2157          }
2158        }
2159      }
2160      \l_tmpa_tl
2161    }
2162  }
2163
2164  \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
2165    \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2166      { \tl_tail:N \l_tmpa_tl }
2167    \tl_if_single:NTF \l_tmpa_tl {
2168      \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2169        \exp_after:wN \str_set:Nn \exp_after:wN
2170          \l_stex_get_symbol_uri_str \l_tmpa_tl
2171      }{
2172        % TODO
2173        % tail is not a single group
2174      }
2175    }{
2176      % TODO
2177      % tail is not a single group
2178    }
2179  }
```

(*End definition for* `\stex_get_symbol:n`. *This function is documented on page 36.*)

30.2 Notations

⟨@@=stex_notation⟩

notation arguments:

```
2181 \keys_define:nn { stex / notation } {
2182   lang    .tl_set_x:N  = \l__stex_notation_lang_str ,
2183   variant .tl_set_x:N  = \l__stex_notation_variant_str ,
2184   prec    .str_set_x:N = \l__stex_notation_prec_str ,
2185   op      .tl_set:N    = \l__stex_notation_op_tl ,
2186   primary .bool_set:N  = \l__stex_notation_primary_bool ,
2187   primary .default:n   = {true} ,
2188   unknown .code:n      = \str_set:Nx
2189       \l__stex_notation_variant_str \l_keys_key_str
2190 }
2191
2192 \cs_new_protected:Nn \_stex_notation_args:n {
2193   \str_clear:N \l__stex_notation_lang_str
2194   \str_clear:N \l__stex_notation_variant_str
2195   \str_clear:N \l__stex_notation_prec_str
2196   \tl_clear:N \l__stex_notation_op_tl
2197   \bool_set_false:N \l__stex_notation_primary_bool
2198
2199   \keys_set:nn { stex / notation } { #1 }
2200 }
```

\notation

```
2201 \NewDocumentCommand \notation { O{} m } {
2202   \_stex_notation_args:n { #1 }
2203   \tl_clear:N \l_stex_symdecl_definiens_tl
2204   \stex_get_symbol:n { #2 }
2205   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
2206 }
2207 \stex_deactivate_macro:Nn \notation {module~environments}
```

(*End definition for* \notation. *This function is documented on page 36.*)

\stex_notation_do:nn

```
2208 \cs_new_protected:Nn \stex_notation_do:nn {
2209   \let\l_stex_current_symbol_str\relax
2210   \prop_set_eq:Nc \l_tmpa_prop {
2211     l_stex_symdecl_ #1 _prop
2212   }
2213
2214   \prop_clear:N \l_tmpb_prop
2215   \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
2216   \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
2217   \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str
2218
2219   % precedences
2220   \seq_clear:N \l_tmpb_seq
2221   \exp_args:NNno
2222   \str_if_empty:NTF \l__stex_notation_prec_str {
2223     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2224     \int_compare:nNnTF \l_tmpa_str = 0 {
```

125

```
    \exp_args:NNnx
    \prop_put:Nno \l_tmpb_prop { opprec }
      { \neginfprec }
  }{
    \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
  }
} {
  \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
    \exp_args:NNnx
    \prop_put:Nno \l_tmpb_prop { opprec }
      { \neginfprec }
    \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
    \int_step_inline:nn { \l_tmpa_str } {
      \exp_args:NNx
      \seq_put_right:Nn \l_tmpb_seq { \infprec }
    }
  }{
    \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
    \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
      \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str
      \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
        \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
          \l_tmpa_seq {\tl_to_str:n{x} } { \l_tmpa_str }
        \seq_map_inline:Nn \l_tmpa_seq {
          \seq_put_right:Nn \l_tmpb_seq { ##1 }
        }
      }
      \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
    }{
      \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
      \int_compare:nNnTF \l_tmpa_str = 0 {
        \exp_args:NNnx
        \prop_put:Nno \l_tmpb_prop { opprec }
          { \infprec }
      }{
        \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
      }
    }
  }
}

\seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
\int_step_inline:nn { \l_tmpa_str } {
  \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
    \exp_args:NNx
    \seq_put_right:Nn \l_tmpb_seq {
      \prop_item:Nn \l_tmpb_prop { opprec }
    }
  }
}

\prop_put:Nno \l_tmpb_prop { argprecs } \l_tmpb_seq
\tl_clear:N \l_tmpa_tl
```

```
2279    \int_compare:nNnTF \l_tmpa_str = 0 {
2280      \exp_args:NNe
2281      \cs_set:Npn \l__stex_notation_macrocode_cs {
2282        \_stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2283          { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2284          { \prop_item:Nn \l_tmpb_prop { opprec } }
2285          { \exp_not:n { #2 } }
2286      }
2287      \__stex_notation_final:
2288    }{
2289      \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
2290      \str_if_in:NnTF \l_tmpb_str b {
2291        \exp_args:Nne \use:nn
2292        {
2293        \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2294        \cs_set:Npn \l_tmpa_str } { {
2295          \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2296            { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2297            { \prop_item:Nn \l_tmpb_prop { opprec } }
2298            { \exp_not:n { #2 } }
2299        }}
2300      }{
2301        \str_if_in:NnTF \l_tmpb_str B {
2302          \exp_args:Nne \use:nn
2303          {
2304          \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2305          \cs_set:Npn \l_tmpa_str } { {
2306            \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2307              { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2308              { \prop_item:Nn \l_tmpb_prop { opprec } }
2309              { \exp_not:n { #2 } }
2310          } }
2311        }{
2312          \exp_args:Nne \use:nn
2313          {
2314          \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2315          \cs_set:Npn \l_tmpa_str } { {
2316            \_stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2317              { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2318              { \prop_item:Nn \l_tmpb_prop { opprec } }
2319              { \exp_not:n { #2 } }
2320          } }
2321        }
2322      }
2323
2324      \int_zero:N \l_tmpa_int
2325      \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2326      \prop_get:NnN \l_tmpb_prop { argprecs } \l_tmpa_seq
2327      \__stex_notation_arguments:
2328    }
2329 }
```

(*End definition for* \stex_notation_do:nn. *This function is documented on page* *37.*)

__stex_notation_arguments: Takes care of annotating the arguments in a notation macro

```
2330 \cs_new_protected:Nn \__stex_notation_arguments: {
2331   \int_incr:N \l_tmpa_int
2332   \str_if_empty:NTF \l_tmpa_str {
2333     \__stex_notation_final:
2334   }{
2335     \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2336     \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2337     \str_if_eq:VnTF \l_tmpb_str a {
2338       \__stex_notation_argument_assoc:n
2339     }{
2340       \str_if_eq:VnTF \l_tmpb_str B {
2341         \__stex_notation_argument_assoc:n
2342       }{
2343         \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2344         \tl_put_right:Nx \l_tmpa_tl {
2345           { \_stex_term_math_arg:nnn
2346             { \int_use:N \l_tmpa_int }
2347             { \l_tmpb_str }
2348             { ####\int_use:N \l_tmpa_int }
2349           }
2350         }
2351         \__stex_notation_arguments:
2352       }
2353     }
2354   }
2355 }
```

(*End definition for* __stex_notation_arguments:.)

__stex_notation_argument_assoc:n

```
2356 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
2357   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2358   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
2359   \tl_put_right:Nx \l_tmpa_tl {
2360     { \_stex_term_math_assoc_arg:nnnn
2361       { \int_use:N \l_tmpa_int }
2362       { \l_tmpb_str }
2363       \exp_args:No \exp_not:n
2364       {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
2365       { ####\int_use:N \l_tmpa_int }
2366     }
2367   }
2368   \__stex_notation_arguments:
2369 }
```

(*End definition for* __stex_notation_argument_assoc:n.)

__stex_notation_final: Called after processing all notation arguments

```
2370 \cs_new_protected:Nn \__stex_notation_final: {
2371   \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
2372   \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
2373   \prop_get:NnN \l_tmpb_prop { argprecs } \l_tmpa_seq
2374   \exp_args:Nne \use:nn
```

```
{
\cs_generate_from_arg_count:cNnn {
    stex_notation_ \l_tmpa_str \c_hash_str
    \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
    _cs
  }
  \cs_set:Npn \l_tmpb_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_tmpa_tl }
} }

\tl_if_empty:NF \l__stex_notation_op_tl {
  \cs_set:cpx {
    stex_op_notation_ \l_tmpa_str \c_hash_str
    \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
    _cs
  } {
    \_stex_term_oms:nnn {
      \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
      \l__stex_notation_lang_str
    }{
      \l_tmpa_str
    }{ \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_tmpa_str \c_hash_str
    \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
    _cs
  } \cs_set:Npn {\l_tmpb_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_tmpa_tl }
  }
  \tl_if_empty:NF \l__stex_notation_op_tl {
    \cs_set:cpn {
      stex_op_notation_ \l_tmpa_str \c_hash_str
      \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
      _cs
    } {
      \_stex_term_oms:nnn {
        \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
        \l__stex_notation_lang_str
      }{
        \l_tmpa_str
      }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } } }
    }
  }
}
```

```
2429    \seq_put_right:cx {
2430      l_stex_symdecl_
2431        \prop_item:Nn \l_tmpb_prop { symbol }
2432      _notations
2433    } {
2434      \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2435    }
2436
2437    \stex_debug:nn{symbols}{
2438      Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2439      ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
2440      Operator~precedence:~
2441        \prop_item:Nn \l_tmpb_prop { opprec }^^J
2442      Argument~precedences:~
2443        \seq_use:Nn \l_tmpa_seq {,~}^^J
2444      Notation: \cs_meaning:c {
2445        stex_notation_ \l_tmpa_str \c_hash_str
2446        \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2447        _cs
2448      }
2449    }
2450
2451    \prop_set_eq:cN {
2452      l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2453        \c_hash_str \l__stex_notation_lang_str _prop
2454    } \l_tmpb_prop
2455
2456    \exp_args:Ne
2457    \stex_add_to_current_module:n {
2458      \seq_put_right:cn {
2459        l_stex_symdecl_
2460          \prop_item:Nn \l_tmpb_prop { symbol }
2461        _notations
2462      } {
2463        \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2464      }
2465      \prop_set_from_keyval:cn {
2466        l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2467          \c_hash_str \l__stex_notation_lang_str _prop
2468      } {
2469        symbol    = \prop_item:Nn \l_tmpb_prop { symbol }     ,
2470        language  = \prop_item:Nn \l_tmpb_prop { language }   ,
2471        variant   = \prop_item:Nn \l_tmpb_prop { variant }    ,
2472        opprec    = \prop_item:Nn \l_tmpb_prop { opprec }     ,
2473        argprecs  = \prop_item:Nn \l_tmpb_prop { argprecs }   ,
2474      }
2475    }
2476
2477    \stex_if_smsmode:TF {
2478      \stex_smsmode_set_codes:
2479 %      \exp_args:Nx \stex_add_to_sms:n {
2480 %        \prop_set_from_keyval:cn {
2481 %          l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2482 %            \c_hash_str \l__stex_notation_lang_str _prop
```

```
2483 %        } {
2484 %          symbol   = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2485 %          language = \prop_item:Nn \l_tmpb_prop { language }    ,
2486 %          variant  = \prop_item:Nn \l_tmpb_prop { variant }     ,
2487 %          opprec   = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2488 %          argprecs = \prop_item:Nn \l_tmpb_prop { argprecs }    ,
2489 %        }
2490 %      }
2491   }{

2493      % HTML annotations
2494      \stex_if_do_html:T {
2495        \stex_annotate_invisible:nnn { notation }
2496        { \prop_item:Nn \l_tmpb_prop { symbol } } {
2497          \stex_annotate_invisible:nnn { notationfragment }
2498            { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
2499          \prop_get:NnN \l_tmpb_prop { argprecs } \l_tmpa_seq
2500          \stex_annotate_invisible:nnn { precedence }
2501            { \prop_item:Nn \l_tmpb_prop { opprec };
2502              \seq_use:Nn \l_tmpa_seq { x }
2503            }{}

2505          \int_zero:N \l_tmpa_int
2506          \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2507          \tl_clear:N \l_tmpa_tl
2508          \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
2509            \int_incr:N \l_tmpa_int
2510            \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2511            \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2512            \str_if_eq:VnTF \l_tmpb_str a {
2513              \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2514                \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2515                \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2516              } }
2517            }{
2518              \str_if_eq:VnTF \l_tmpb_str B {
2519                \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2520                  \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2521                  \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2522                } }
2523              }{
2524                \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2525                  \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2526                } }
2527              }
2528            }
2529          }
2530          \stex_annotate_invisible:nnn { notationcomp }{}{
2531            \str_set:Nx \l_stex_current_symbol_str {\prop_item:Nn \l_tmpb_prop { symbol }}
2532            $ \exp_args:Nno \use:nn { \use:c {
2533              stex_notation_ \l_stex_current_symbol_str
2534              \c_hash_str \l__stex_notation_variant_str
2535              \c_hash_str \l__stex_notation_lang_str _cs
2536            } } { \l_tmpa_tl } $
```

131

```
2537              }
2538            }
2539          }
2540        }
2541 }
```

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

`\setnotation`

```
2542 \keys_define:nn { stex / setnotation } {
2543   lang    .tl_set_x:N  = \l__stex_notation_lang_str ,
2544   variant .tl_set_x:N  = \l__stex_notation_variant_str ,
2545   unknown .code:n       = \str_set:Nx
2546       \l__stex_notation_variant_str \l_keys_key_str
2547 }
2548
2549 \cs_new_protected:Nn \_stex_setnotation_args:n {
2550   \str_clear:N \l__stex_notation_lang_str
2551   \str_clear:N \l__stex_notation_variant_str
2552   \keys_set:nn { stex / setnotation } { #1 }
2553 }
2554
2555 \NewDocumentCommand \setnotation {m m} {
2556   \stex_get_symbol:n { #1 }
2557   \_stex_setnotation_args:n { #2 }
2558   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_\l_stex_get_symbol_uri_str _notations }
2559     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2560       \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _notation
2561         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2562       \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _notation
2563         { \c_hash_str }
2564       \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _notations
2565         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2566       \exp_args:Nx \stex_add_to_current_module:n {
2567         \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _notati
2568           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2569         \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _notation
2570           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2571         \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _notati
2572           { \c_hash_str }
2573       }
2574       \stex_debug:nn {notations}{
2575         Setting~default~notation~
2576         {\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str}~for~
2577         \l_stex_get_symbol_uri_str \\
2578         \expandafter\meaning\csname
2579         l_stex_symdecl_\l_stex_get_symbol_uri_str _notations\endcsname
2580       }
2581     }{
2582       % todo throw error
2583     }
2584 }
2585
```

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

132

\symdef

```
2586 \keys_define:nn { stex / symdef } {
2587   name    .str_set_x:N = \l_stex_symdecl_name_str ,
2588   local   .bool_set:N  = \l_stex_symdecl_local_bool ,
2589   args    .str_set_x:N = \l_stex_symdecl_args_str ,
2590   type    .tl_set:N    = \l_stex_symdecl_type_tl ,
2591   def     .tl_set:N    = \l_stex_symdecl_definiens_tl ,
2592   op      .tl_set:N    = \l__stex_notation_op_tl ,
2593   lang    .str_set_x:N = \l__stex_notation_lang_str ,
2594   variant .str_set_x:N = \l__stex_notation_variant_str ,
2595   prec    .str_set_x:N = \l__stex_notation_prec_str ,
2596   unknown .code:n      = \str_set:Nx
2597       \l__stex_notation_variant_str \l_keys_key_str
2598 }
2599
2600 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2601   \str_clear:N \l_stex_symdecl_name_str
2602   \str_clear:N \l_stex_symdecl_args_str
2603   \bool_set_false:N \l_stex_symdecl_local_bool
2604   \tl_clear:N \l_stex_symdecl_type_tl
2605   \tl_clear:N \l_stex_symdecl_definiens_tl
2606   \str_clear:N \l__stex_notation_lang_str
2607   \str_clear:N \l__stex_notation_variant_str
2608   \str_clear:N \l__stex_notation_prec_str
2609   \tl_clear:N \l__stex_notation_op_tl
2610
2611   \keys_set:nn { stex / symdef } { #1 }
2612 }
2613
2614 \NewDocumentCommand \symdef { O{} m } {
2615   \__stex_notation_symdef_args:n { #1 }
2616   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2617   \stex_symdecl_do:n { #2 }
2618   \exp_args:Nx \stex_notation_do:nn {
2619     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2620   }
2621 }
2622 \stex_deactivate_macro:Nn \symdef {module~environments}
```

(*End definition for* \symdef. *This function is documented on page 37.*)

```
2623 ⟨/package⟩
```

Chapter 31

SₜₑX
-Terms Implementation

```
2624  ⟨*package⟩
2625
2626  %%%%%%%%%%%%    terms.dtx    %%%%%%%%%%%%
2627
2628  ⟨@@=stex_terms⟩
```

Warnings and error messages

```
2629  \msg_new:nnn{stex}{error/nonotation}{
2630    Symbol~#1~invoked,~but~has~no~notation#2!
2631  }
2632  \msg_new:nnn{stex}{error/notationarg}{
2633    Error~in~parsing~notation~#1
2634  }
2635  \msg_new:nnn{stex}{error/noop}{
2636    Symbol~#1~has~no~operator~notation~for~notation~#2
2637  }
2638
```

31.1   Symbol Invokations

Arguments:

```
2639  \keys_define:nn { stex / terms } {
2640    lang    .tl_set_x:N = \l__stex_terms_lang_str ,
2641    variant .tl_set_x:N = \l__stex_terms_variant_str ,
2642    unknown .code:n      = \str_set:Nx
2643        \l__stex_terms_variant_str \l_keys_key_str
2644  }
2645
2646  \cs_new_protected:Nn \__stex_terms_args:n {
2647    \str_clear:N \l__stex_terms_lang_str
2648    \str_clear:N \l__stex_terms_variant_str
2649    \str_clear:N \l__stex_terms_prec_str
2650    \tl_clear:N \l__stex_terms_op_tl
2651
2652    \keys_set:nn { stex / terms } { #1 }
```

```
2653 }
```

\stex_invoke_symbol:n Invokes a semantic macro

```
2654 \cs_new_protected:Nn \stex_invoke_symbol:n {
2655   \if_mode_math:
2656     \exp_after:wN \__stex_terms_invoke_math:n
2657   \else:
2658     \exp_after:wN \__stex_terms_invoke_text:n
2659   \fi: { #1 }
2660 }
```

(*End definition for* \stex_invoke_symbol:n. *This function is documented on page 38.*)

__stex_terms_invoke_math:n

```
2661 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2662   \peek_charcode_remove:NTF ! {
2663     \peek_charcode:NTF [ {
2664       \__stex_terms_invoke_op:nw { #1 }
2665     }{
2666       \peek_charcode_remove:NTF ! {
2667         \peek_charcode:NTF [ {
2668           \__stex_terms_invoke_op_custom:nw
2669         }{
2670           % TODO throw error
2671         }
2672       }{
2673         \__stex_terms_invoke_op:nw { #1 } []
2674       }
2675     }
2676   }{
2677     \peek_charcode_remove:NTF * {
2678       \__stex_terms_invoke_text:n { #1 }
2679     }{
2680       \peek_charcode:NTF [ {
2681         \__stex_terms_invoke_math:nw { #1 }
2682       }{
2683         \__stex_terms_invoke_math:nw { #1 } []
2684       }
2685     }
2686   }
2687 }
```

(*End definition for* __stex_terms_invoke_math:n.)

__stex_terms_invoke_op_custom:nw

```
2688 \cs_new_protected:Npn \__stex_terms_invoke_op_custom:nw  #1 [#2] {
2689   \_stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
2690     \stex_highlight_term:nn{#1}{#2}
2691   }
2692 }
```

(*End definition for* __stex_terms_invoke_op_custom:nw.)

```
2693 \cs_new_protected:Npn \__stex_terms_invoke_op:nw  #1 [#2] {
2694   \__stex_terms_args:n { #2 }
2695   \cs_if_exist:cTF {
2696     stex_op_notation_ #1 \c_hash_str
2697     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2698   }{
2699     \csname stex_op_notation_ #1 \c_hash_str
2700       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2701     \endcsname
2702   }{
2703     \msg_error:nnxx{stex}{error/noop}{#1}{\l__stex_terms_variant_str \c_hash_str \l__stex_te
2704   }
2705 }
```

(*End definition for* `\__stex_terms_invoke_op:nw`.)

```
2706 \cs_new_protected:Npn \__stex_terms_invoke_math:nw  #1 [#2] {
2707   \__stex_terms_args:n { #2 }
2708   \seq_if_empty:cTF {
2709     l_stex_symdecl_ #1 _notations
2710   } {
2711     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
2712   } {
2713     \seq_if_in:cxTF {
2714       l_stex_symdecl_ #1 _notations
2715     }
2716     { \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str }{
2717     \str_set:Nn \l_stex_current_symbol_str { #1 }
2718     \use:c{
2719       stex_notation_ #1 \c_hash_str
2720       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2721       _cs
2722     }
2723     }{
2724       \str_if_empty:NTF \l__stex_terms_variant_str {
2725       \str_if_empty:NTF \l__stex_terms_lang_str {
2726         \seq_get_left:cN {
2727           l_stex_symdecl_ #1 _notations
2728         } \l_tmpa_str
2729         \str_set:Nn \l_stex_current_symbol_str { #1 }
2730         \use:c{
2731           stex_notation_ #1 \c_hash_str \l_tmpa_str
2732           _cs
2733         }
2734       }{
2735         \msg_error:nnxx{stex}{error/nonotation}{#1}{
2736           ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2737         }
2738       }
2739       }{
2740         \msg_error:nnxx{stex}{error/nonotation}{#1}{
2741           ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
```

136

```
2742            }
2743          }
2744        }
2745      }
2746  }
```

(*End definition for* `\__stex_terms_invoke_math:nw`.)

`\__stex_terms_invoke_text:n`

```
2747  \cs_new_protected:Nn \__stex_terms_invoke_text:n {
2748    \peek_charcode_remove:NTF ! {
2749      \stex_term_custom:nn { #1 } { }
2750    }{
2751      \prop_set_eq:Nc \l_tmpa_prop {
2752        l_stex_symdecl_ #1 _prop
2753      }
2754      \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2755      \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2756    }
2757  }
```

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

31.2 Terms

Precedences:

\infprec
\neginfprec
`\l__stex_terms_downprec`

```
2758  \tl_const:Nx \infprec {\int_use:N \c_max_int}
2759  \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2760  \int_new:N \l__stex_terms_downprec
2761  \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* .)

Bracketing:

`\l__stex_terms_left_bracket_str`
`\l__stex_terms_right_bracket_str`

```
2762  \tl_set:Nn \l__stex_terms_left_bracket_str (
2763  \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

```
2764  \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
2765    \bool_if:NTF \l__stex_terms_brackets_done_bool {
2766      \bool_set_false:N \l__stex_terms_brackets_done_bool
2767      #2
2768    } {
2769      \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
2770        \bool_if:NTF \l_stex_inparray_bool { #2 }{
2771          \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
2772          \dobrackets { #2 }
2773        }
```

```
2774        }{ #2 }
2775      }
2776    }
```

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

`\dobrackets`

```
2777  \bool_new:N \l__stex_terms_brackets_done_bool
2778  %\RequirePackage{scalerel}
2779  \cs_new_protected:Npn \dobrackets #1 {
2780    %\ThisStyle{\if D\m@switch
2781    %   \exp_args:Nnx \use:nn
2782    %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
2783    %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
2784    %  \else
2785        \exp_args:Nnx \use:nn
2786        {
2787          \bool_set_true:N \l__stex_terms_brackets_done_bool
2788          \int_set:Nn \l__stex_terms_downprec \infprec
2789          \l__stex_terms_left_bracket_str
2790          #1
2791        }
2792        {
2793          \bool_set_false:N \l__stex_terms_brackets_done_bool
2794          \l__stex_terms_right_bracket_str
2795          \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2796        }
2797    %\fi}
2798  }
```

(*End definition for* `\dobrackets`. *This function is documented on page 39.*)

`\withbrackets`

```
2799  \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2800    \exp_args:Nnx \use:nn
2801    {
2802      \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
2803      \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
2804      #3
2805    }
2806    {
2807      \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
2808        {\l__stex_terms_left_bracket_str}
2809      \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
2810        {\l__stex_terms_right_bracket_str}
2811    }
2812  }
```

(*End definition for* `\withbrackets`. *This function is documented on page 39.*)

`\STEXinvisible`

```
2813  \cs_new_protected:Npn \STEXinvisible #1 {
2814    \stex_annotate_invisible:n { #1 }
2815  }
```

(End definition for `\STEXinvisible`*. This function is documented on page 40.)*

OMDoc terms:

```
2816 \cs_new_protected:Nn \_stex_term_oms:nnn {
2817   \stex_annotate:nnn{ OMID }{ #2 }{
2818     \stex_highlight_term:nn { #1 } { #3 }
2819   }
2820 }
2821
2822 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2823   \__stex_terms_maybe_brackets:nn { #3 }{
2824     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2825   }
2826 }
```

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

```
2827 \cs_new_protected:Nn \_stex_term_oma:nnn {
2828   \stex_annotate:nnn{ OMA }{ #2 }{
2829     \stex_highlight_term:nn { #1 } { #3 }
2830   }
2831 }
2832
2833 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2834   \__stex_terms_maybe_brackets:nn { #3 }{
2835     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2836   }
2837 }
```

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

```
2838 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2839   \stex_annotate:nnn{ OMBIND }{ #2 }{
2840     \stex_highlight_term:nn { #1 } { #3 }
2841   }
2842 }
2843
2844 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2845   \__stex_terms_maybe_brackets:nn { #3 }{
2846     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2847   }
2848 }
```

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

```
2849 \cs_new_protected:Nn \_stex_term_arg:nn {
2850   \stex_unhighlight_term:n {
2851     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2852   }
2853 }
```

139

```
2854  \cs_new_protected:Nn \_stex_term_math_arg:nnn {
2855    \exp_args:Nnx \use:nn
2856      { \int_set:Nn \l__stex_terms_downprec { #2 }
2857        \_stex_term_arg:nn { #1 }{ #3 }
2858      }
2859      { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2860  }
```

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

```
2861  \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
2862    \clist_set:Nn \l_tmpa_clist{ #4 }
2863    \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
2864      \tl_set:Nn \l_tmpa_tl { #4 }
2865    }{
2866      \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
2867      \clist_reverse:N \l_tmpa_clist
2868      \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
2869
2870      \clist_map_inline:Nn \l_tmpa_clist {
2871        \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
2872          \exp_args:Nno
2873          \l_tmpa_cs { ##1 } \l_tmpa_tl
2874        }
2875      }
2876
2877    }
2878    \exp_args:Nnno
2879    \_stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
2880  }
```

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

```
2881  \cs_new_protected:Nn \stex_term_custom:nn {
2882    \str_set:Nn \l__stex_terms_custom_uri { #1 }
2883    \str_set:Nn \l_tmpa_str { #2 }
2884    \tl_clear:N \l_tmpa_tl
2885    \int_zero:N \l_tmpa_int
2886    \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2887    \__stex_terms_custom_loop:
2888  }
```

(*End definition for* `\stex_term_custom:nn`. *This function is documented on page 40.*)

```
2889  \cs_new_protected:Nn \__stex_terms_custom_loop: {
2890    \bool_set_false:N \l_tmpa_bool
2891    \bool_while_do:nn {
2892      \str_if_eq_p:ee X {
2893        \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2894      }
2895    }{
2896      \int_incr:N \l_tmpa_int
```

140

```
2897    }
2898
2899    \peek_charcode:NTF [ {
2900      % notation/text component
2901      \__stex_terms_custom_component:w
2902    } {
2903      \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2904        % all arguments read => finish
2905        \__stex_terms_custom_final:
2906      } {
2907        % arguments missing
2908        \peek_charcode_remove:NTF * {
2909          % invisible, specific argument position or both
2910          \peek_charcode:NTF [ {
2911            % visible specific argument position
2912            \__stex_terms_custom_arg:wn
2913          } {
2914            % invisible
2915            \peek_charcode_remove:NTF * {
2916              % invisible specific argument position
2917              \__stex_terms_custom_arg_inv:wn
2918            } {
2919              % invisible next argument
2920              \__stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2921            }
2922          }
2923        } {
2924          % next normal argument
2925          \__stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
2926        }
2927      }
2928    }
2929 }
```

(End definition for __stex_terms_custom_loop:.)

__stex_terms_custom_arg_inv:wn

```
2930 \cs_new_protected:Npn \__stex_terms_custom_arg_inv:wn [ #1 ] #2 {
2931    \bool_set_true:N \l_tmpa_bool
2932    \__stex_terms_custom_arg:wn [ #1 ] { #2 }
2933 }
```

(End definition for __stex_terms_custom_arg_inv:wn.)

__stex_terms_custom_arg:wn

```
2934 \cs_new_protected:Npn \__stex_terms_custom_arg:wn [ #1 ] #2 {
2935    \str_set:Nx \l_tmpb_str {
2936      \str_item:Nn \l_tmpa_str { #1 }
2937    }
2938    \str_case:VnTF \l_tmpb_str {
2939      { X } {
2940        \msg_error:nnx{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2941      }
2942      { i } { \__stex_terms_custom_set_X:n { #1 } }
2943      { b } { \__stex_terms_custom_set_X:n { #1 } }
```

141

```
2944        { a } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
2945        { B } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
2946     }{}{
2947       \msg_error:nnx{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2948     }
2949
2950     \bool_if:nTF \l_tmpa_bool {
2951       \tl_put_right:Nx \l_tmpa_tl {
2952         \stex_annotate_invisible:n {
2953           \_stex_term_arg:nn { \int_eval:n { #1 } }
2954             \exp_not:n { { #2 } }
2955         }
2956       }
2957     } {
2958       \tl_put_right:Nx \l_tmpa_tl {
2959         \_stex_term_arg:nn { \int_eval:n { #1 } }
2960           \exp_not:n { { #2 } }
2961       }
2962     }
2963
2964     \__stex_terms_custom_loop:
2965 }
```

(End definition for __stex_terms_custom_arg:wn.)

`\__stex_terms_custom_set_X:n`

```
2966 \cs_new_protected:Nn \__stex_terms_custom_set_X:n {
2967     \str_set:Nx \l_tmpa_str {
2968       \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
2969       X
2970       \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
2971     }
2972 }
```

(End definition for __stex_terms_custom_set_X:n.)

`\__stex_terms_custom_component:`

```
2973 \cs_new_protected:Npn \__stex_terms_custom_component:w [ #1 ] {
2974     \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
2975     \__stex_terms_custom_loop:
2976 }
```

(End definition for __stex_terms_custom_component:.)

`\__stex_terms_custom_final:`

```
2977 \cs_new_protected:Nn \__stex_terms_custom_final: {
2978     \int_compare:nNnTF \l_tmpb_int = 0 {
2979       \exp_args:Nnno \_stex_term_oms:nnn
2980     }{
2981       \str_if_in:NnTF \l_tmpa_str {b} {
2982         \exp_args:Nnno \_stex_term_ombind:nnn
2983       } {
2984         \exp_args:Nnno \_stex_term_oma:nnn
2985       }
2986     }
```

```
2987        { \l__stex_terms_custom_uri } { \l__stex_terms_custom_uri } { \l_tmpa_tl }
2988  }
```

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

`\symref`
`\symname`

```
2989  \NewDocumentCommand \symref { m m }{
2990    \let\compemph_uri_prev:\compemph@uri
2991    \let\compemph@uri\symrefemph@uri
2992    \STEXsymbol{#1}![#2]
2993    \let\compemph@uri\compemph_uri_prev:
2994  }
2995
2996  \keys_define:nn { stex / symname } {
2997    post     .str_set_x:N   = \l_stex_symname_post_str
2998  }
2999
3000  \cs_new_protected:Nn \stex_symname_args:n {
3001    \str_clear:N \l_stex_symname_post_str
3002    \keys_set:nn { stex / symname } { #1 }
3003  }
3004
3005  \NewDocumentCommand \symname { O{} m }{
3006    \stex_symname_args:n { #1 }
3007    \stex_get_symbol:n { #2 }
3008    \str_set:Nx \l_tmpa_str {
3009      \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3010    }
3011    \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3012
3013    \let\compemph_uri_prev:\compemph@uri
3014    \let\compemph@uri\symrefemph@uri
3015    \exp_args:NNx \use:nn
3016    \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }![
3017      \l_tmpa_str \l_stex_symname_post_str
3018    ] }
3019    \let\compemph@uri\compemph_uri_prev:
3020  }
```

(*End definition for* `\symref` *and* `\symname`. *These functions are documented on page* *38.*)

31.3   Notation Components

```
3021  ⟨@@=stex_notationcomps⟩
```

`\stex_highlight_term:nn`

```
3022
3023  \str_new:N \l_stex_current_symbol_str
3024  \cs_new_protected:Nn \stex_highlight_term:nn {
3025    \exp_args:Nnx
3026    \use:nn {
3027      \str_set:Nx \l_stex_current_symbol_str { #1 }
3028      #2
3029    } {
```

```
3030        \str_set:Nx \exp_not:N \l_stex_current_symbol_str
3031          { \l_stex_current_symbol_str }
3032    }
3033 }
3034
3035 \cs_new_protected:Nn \stex_unhighlight_term:n {
3036 %  \latexml_if:TF {
3037 %    #1
3038 %  } {
3039 %    \rustex_if:TF {
3040 %      #1
3041 %    } {
3042        #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
3043 %    }
3044 %  }
3045 }
```

(End definition for `\stex_highlight_term:nn`*. This function is documented on page 40.)*

```
3046 \cs_new_protected:Npn \comp #1 {
3047    \str_if_empty:NF \l_stex_current_symbol_str {
3048        \rustex_if:TF {
3049            \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
3050        }{
3051            \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3052        }
3053    }
3054 }
3055
3056 \cs_new_protected:Npn \compemph@uri #1 #2 {
3057    \compemph{ #1 }
3058 }
3059
3060
3061 \cs_new_protected:Npn \compemph #1 {
3062    #1
3063 }
3064
3065 \cs_new_protected:Npn \defemph@uri #1 #2 {
3066    \defemph{#1}
3067 }
3068
3069 \cs_new_protected:Npn \defemph #1 {
3070    \textbf{#1}
3071 }
3072
3073 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3074    \symrefemph{#1}
3075 }
3076
3077 \cs_new_protected:Npn \symrefemph #1 {
3078    \textbf{#1}
3079 }
```

(End definition for `\comp` *and others. These functions are documented on page* *40.)*

`\ellipses`

```
3080 \NewDocumentCommand \ellipses {} { \ldots }
```

(End definition for `\ellipses`*. This function is documented on page* *40.)*

`\parray`
`\prmatrix`
`\parrayline`
`\parraylineh`
`\parraycell`

```
3081 \bool_new:N \l_stex_inparray_bool
3082 \bool_set_false:N \l_stex_inparray_bool
3083 \NewDocumentCommand \parray { m m } {
3084     \begingroup
3085     \bool_set_true:N \l_stex_inparray_bool
3086     \begin{array}{#1}
3087        #2
3088     \end{array}
3089     \endgroup
3090 }
3091
3092 \NewDocumentCommand \prmatrix { m } {
3093     \begingroup
3094     \bool_set_true:N \l_stex_inparray_bool
3095     \begin{matrix}
3096        #1
3097     \end{matrix}
3098     \endgroup
3099 }
3100
3101 \def \maybephline {
3102     \bool_if:NT \l_stex_inparray_bool {\hline}
3103 }
3104
3105 \def \parrayline #1 #2 {
3106    #1 #2 \bool_if:NT \l_stex_inparray_bool {\\}
3107 }
3108
3109 \def \pmrow #1 { \parrayline{}{ #1 } }
3110
3111 \def \parraylineh #1 #2 {
3112    #1 #2 \bool_if:NT \l_stex_inparray_bool {\\\hline}
3113 }
3114
3115 \def \parraycell #1 {
3116    #1 \bool_if:NT \l_stex_inparray_bool {&}
3117 }
```

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

```
3118 ⟨/package⟩
```

Chapter 32

STEX
-Structural Features
Implementation

```
3119 ⟨∗package⟩
3120
3121 %%%%%%%%%%%%    features.dtx    %%%%%%%%%%%%
3122
3123 ⟨@@=stex_features⟩
```

Warnings and error messages

```
3124 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3125   Symbol~#1~can~not~be~assigned~in~copymodule~#2
3126 }
3127 \msg_new:nnn{stex}{error/interpretmodule/nodefiniens}{
3128   Symbol~#1~not~assigned~in~interpretmodule~#2
3129 }
3130
```

32.1   Imports with modification

```
3131 \cs_new_protected:Nn \stex_get_symbol_in_copymodule:n {
3132   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3133     \__stex_features_get_symbol_from_cs:n { #1 }
3134   }{
3135     % argument is a string
3136     % is it a command name?
3137     \cs_if_exist:cTF { #1 }{
3138       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3139       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3140       \str_if_empty:NTF \l_tmpa_str {
3141         \exp_args:Nx \cs_if_eq:NNTF {
3142           \tl_head:N \l_tmpa_tl
3143         } \stex_invoke_symbol:n {
3144           \exp_args:No \__stex_features_get_symbol_from_cs:n { \use:c { #1 } }
3145         }{
3146           \__stex_features_get_symbol_from_string:n { #1 }
```

```
3147              }
3148          } {
3149            \__stex_features_get_symbol_from_string:n { #1 }
3150          }
3151      }{
3152        % argument is not a command name
3153        \__stex_features_get_symbol_from_string:n { #1 }
3154        % \l_stex_all_symbols_seq
3155      }
3156    }
3157  }
3158
3159  \cs_new_protected:Nn \__stex_features_get_symbol_from_string:n {
3160    \str_set:Nn \l_tmpa_str { #1 }
3161    \bool_set_false:N \l_tmpa_bool
3162    \bool_if:NF \l_tmpa_bool {
3163      \tl_set:Nn \l_tmpa_tl {
3164        \msg_set:nnn{stex}{error/unknownsymbol}{
3165          No~symbol~#1~found!
3166        }
3167        \msg_error:nn{stex}{error/unknownsymbol}
3168      }
3169      \str_set:Nn \l_tmpa_str { #1 }
3170      \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3171      \seq_map_inline:Nn \l__stex_features_copymodule_fields_seq {
3172        \str_set:Nn \l_tmpb_str { ##1 }
3173        \str_if_eq:eeT { \l_tmpa_str } {
3174          \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3175        } {
3176          \seq_map_break:n {
3177            \tl_set:Nn \l_tmpa_tl {
3178              \str_set:Nn \l_stex_get_symbol_uri_str {
3179                ##1
3180              }
3181              \__stex_features_get_symbol_check:
3182            }
3183          }
3184        }
3185      }
3186      \l_tmpa_tl
3187    }
3188  }
3189
3190  \cs_new_protected:Nn \__stex_features_get_symbol_from_cs:n {
3191    \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3192      { \tl_tail:N \l_tmpa_tl }
3193    \tl_if_single:NTF \l_tmpa_tl {
3194      \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3195        \exp_after:wN \str_set:Nn \exp_after:wN
3196          \l_stex_get_symbol_uri_str \l_tmpa_tl
3197        \__stex_features_get_symbol_check:
3198      }{
3199        % TODO
3200        % tail is not a single group
```

```
3201          }
3202     }{
3203        % TODO
3204        % tail is not a single group
3205     }
3206  }
3207
3208  \cs_new_protected:Nn \__stex_features_get_symbol_check: {
3209     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq {?} \l_stex_get_symbol_uri_str
3210     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 3 {
3211        \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
3212        \str_set:Nx \l_tmpa_str {\seq_use:Nn \l_tmpa_seq ?}
3213        \seq_if_in:NoF \l__stex_features_copymodule_modules_seq \l_tmpa_str {
3214           \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3215              \l_stex_current_copymodule_name_str\\Allowed:~\seq_use:Nn \l__stex_features_copymodu
3216           }
3217        }
3218     }{
3219        \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3220           \l_stex_current_copymodule_name_str~(inexplicably)
3221        }
3222     }
3223  }
3224
3225  \cs_new_protected:Nn \stex_copymodule_start:nnnn {
3226     \stex_import_module_uri:nn { #1 } { #2 }
3227     \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3228     \stex_import_require_module:nnnn
3229        { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3230        { \l_stex_import_path_str } { \l_stex_import_name_str }
3231     \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3232     \seq_set_eq:NN \l__stex_features_copymodule_modules_seq \l_stex_collect_imports_seq
3233     \seq_clear:N \l__stex_features_copymodule_fields_seq
3234     \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3235        \seq_map_inline:cn {c_stex_module_##1_constants}{
3236           \exp_args:NNx \seq_put_right:Nn \l__stex_features_copymodule_fields_seq {
3237              ##1 ? ####1
3238           }
3239        }
3240     }
3241     \seq_clear:N \l_tmpa_seq
3242     \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3243        name     = \l_stex_current_copymodule_name_str ,
3244        module   = \l_stex_current_module_str ,
3245        from     = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3246        includes = \l_tmpa_seq ,
3247        fields   = \l_tmpa_seq
3248     }
3249     \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3250        as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3251     \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_features_copymodule_modules_seq
3252     \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_features_copymodule_fields_seq {,~}
3253     \stex_if_smsmode:TF {
3254        \stex_smsmode_set_codes:
```

```
3255    } {
3256      \begin{stex_annotate_env} {#4} {
3257        \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3258      }
3259      \stex_annotate_invisible:nnn{from}{\l_stex_import_ns_str ?\l_stex_import_name_str}{}
3260    }
3261    \bool_set_eq:NN \l__stex_features_oldhtml_bool \l_stex_html_do_output_bool
3262    \bool_set_false:N \l_stex_html_do_output_bool
3263 }
3264 \cs_new_protected:Nn \stex_copymodule_end:n {
3265    \def \l_tmpa_cs ##1 ##2 {#1}
3266    \bool_set_eq:NN \l_stex_html_do_output_bool \l__stex_features_oldhtml_bool
3267    \tl_clear:N \l_tmpa_tl
3268    \prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3269    \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3270      \seq_map_inline:cn {c_stex_module_##1_constants}{\stex_annotate:nnn{assignment} {##1?###
3271        \l_tmpa_cs{##1}{####1}
3272        \str_if_exist:cTF {l__stex_features_copymodule_##1?####1_name_str} {
3273          \tl_put_right:Nx \l_tmpa_tl {
3274            \prop_set_from_keyval:cn {
3275              l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_#
3276            }{
3277              \exp_after:wN \prop_to_keyval:N \csname
3278                l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule
3279              \endcsname
3280            }
3281            \seq_clear:c {
3282              l_stex_symdecl_
3283              \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_name_s
3284              _notations
3285            }
3286          }
3287          \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_features_copymodule_##1?####1_nam
3288          \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \use:c{l__stex_features_
3289          \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3290            \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?###
3291            \tl_put_right:Nx \l_tmpa_tl {
3292              \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3293                \stex_invoke_symbol:n {
3294                  \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_na
3295                }
3296              }
3297            }
3298          }
3299        }{
3300          \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
3301          \prop_put:Nnx \l_tmpa_prop { name }{ \l_stex_current_copymodule_name_str / ####1 }
3302          \prop_put:Nnx \l_tmpa_prop { module }{ \l_stex_current_module_str }
3303          \tl_put_right:Nx \l_tmpa_tl {
3304            \prop_set_from_keyval:cn {
3305              l_stex_symdecl_\l_stex_current_module_str ? \l_stex_current_copymodule_name_str
3306            }{
3307              \prop_to_keyval:N \l_tmpa_prop
3308            }
```

149

```
3309            \seq_clear:c {
3310              l_stex_symdecl_
3311              \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3312              _notations
3313            }
3314          }
3315          \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \l_stex_current_copymodu
3316          \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3317            \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?###
3318            \tl_put_right:Nx \l_tmpa_tl {
3319              \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3320                \stex_invoke_symbol:n {
3321                  \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3322                }
3323              }
3324            }
3325          }
3326        }
3327        \tl_if_exist:cT {l__stex_features_copymodule_##1?####1_def_tl}{
3328          \stex_annotate_invisible:nnn{definiens}{}{$\use:c{l__stex_features_copymodule_##1?##
3329        }
3330        % todo notations
3331      }}
3332    }
3333    \prop_put:Nno \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3334    \tl_put_left:Nx \l_tmpa_tl {
3335      \prop_set_from_keyval:cn {
3336        l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _pro
3337      }{
3338        \prop_to_keyval:N \l_stex_current_copymodule_prop
3339      }
3340    }
3341    \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl
3342    \stex_debug:nn{copymodule}{result:\meaning \l_tmpa_tl}
3343    \exp_args:Nx \stex_do_aftergroup:n {
3344        \exp_args:No \exp_not:n \l_tmpa_tl
3345    }
3346    \stex_if_smsmode:F {
3347      \end{stex_annotate_env}
3348    }
3349 }
3350
3351 \NewDocumentEnvironment {copymodule} { O{} m m}{
3352   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ structure }
3353   \stex_deactivate_macro:Nn \symdecl {module~environments}
3354   \stex_deactivate_macro:Nn \symdef {module~environments}
3355   \stex_deactivate_macro:Nn \notation {module~environments}
3356   \stex_reactivate_macro:N \assign
3357   \stex_reactivate_macro:N \renamedecl
3358   \stex_reactivate_macro:N \donotcopy
3359 }{
3360   \stex_copymodule_end:n {}
3361 }
3362
```

```
3363  \NewDocumentEnvironment {interpretmodule} { O{} m m}{
3364      \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ realization }
3365      \stex_deactivate_macro:Nn \symdecl {module~environments}
3366      \stex_deactivate_macro:Nn \symdef {module~environments}
3367      \stex_deactivate_macro:Nn \notation {module~environments}
3368      \stex_reactivate_macro:N \assign
3369      \stex_reactivate_macro:N \renamedecl
3370      \stex_reactivate_macro:N \donotcopy
3371  }{
3372      \stex_copymodule_end:n {
3373          \tl_if_exist:cF {
3374              l__stex_features_copymodule_##1?##2_def_tl
3375          }{
3376              \msg_error:nnxx{stex}{error/interpretmodule/nodefiniens}{
3377                  ##1?##2
3378              }{\l_stex_current_copymodule_name_str}
3379          }
3380      }
3381  }
3382
3383  \NewDocumentCommand \donotcopy { O{} m}{
3384      \stex_import_module_uri:nn { #1 } { #2 }
3385      \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3386      \seq_map_inline:Nn \l_stex_collect_imports_seq {
3387          \seq_remove_all:Nn \l__stex_features_copymodule_modules_seq { ##1 }
3388          \seq_map_inline:cn {c_stex_module_##1_constants}{
3389              \seq_remove_all:Nn \l__stex_features_copymodule_fields_seq { ##1 ? ####1 }
3390              \bool_lazy_any_p:nT {
3391                  { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_name_str}}
3392                  { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_macroname_str}}
3393                  { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_def_tl}}
3394              }{
3395                  % TODO throw error
3396              }
3397          }
3398      }
3399
3400      \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
3401      \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_ns_str ?\l_stex_import_name_str }
3402      \prop_put:Nnx \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
3403  }
3404
3405  \NewDocumentCommand \assign { m m }{
3406      \stex_get_symbol_in_copymodule:n {#1}
3407      \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
3408      \tl_set:cn {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _def_tl}{#2}
3409  }
3410
3411  \keys_define:nn { stex / renamedecl } {
3412      name            .str_set_x:N  = \l_stex_renamedecl_name_str
3413  }
3414  \cs_new_protected:Nn \__stex_features_renamedecl_args:n {
3415      \str_clear:N \l_stex_renamedecl_name_str
3416
```

```
3417    \keys_set:nn { stex / renamedecl } { #1 }
3418  }
3419
3420  \NewDocumentCommand \renamedecl { O{} m m}{
3421    \__stex_features_renamedecl_args:n { #1 }
3422    \stex_get_symbol_in_copymodule:n {#2}
3423    \stex_debug:nn{renamedecl}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
3424    \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
3425    \str_if_empty:NTF \l_stex_renamedecl_name_str {
3426      \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3427        \l_stex_get_symbol_uri_str
3428      } }
3429    } {
3430      \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex_r
3431      \stex_debug:nn{renamedecl}{@~\l_stex_current_module_str ? \l_stex_renamedecl_name_str}
3432      \prop_set_eq:cc {l_stex_symdecl_
3433        \l_stex_current_module_str ? \l_stex_renamedecl_name_str
3434        _prop
3435      }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
3436      \seq_set_eq:cc {l_stex_symdecl_
3437        \l_stex_current_module_str ? \l_stex_renamedecl_name_str
3438        _notations
3439      }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
3440      \prop_put:cnx {l_stex_symdecl_
3441        \l_stex_current_module_str ? \l_stex_renamedecl_name_str
3442        _prop
3443      }{ name }{ \l_stex_renamedecl_name_str }
3444      \prop_put:cnx {l_stex_symdecl_
3445        \l_stex_current_module_str ? \l_stex_renamedecl_name_str
3446        _prop
3447      }{ module }{ \l_stex_current_module_str }
3448      \exp_args:NNx \seq_put_left:Nn \l__stex_features_copymodule_fields_seq {
3449        \l_stex_current_module_str ? \l_stex_renamedecl_name_str
3450      }
3451      \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3452        \l_stex_current_module_str ? \l_stex_renamedecl_name_str
3453      } }
3454    }
3455  }
3456  %\NewDocumentCommand \notation_in_copymodules: { O{} m } {
3457  %  \_stex_notation_args:n { #1 }
3458  %  \tl_clear:N \l_stex_symdecl_definiens_tl
3459  %  \stex_get_symbol_in_copymodule:n { #2 }
3460  %  \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
3461  %  % todo
3462  %}
3463  \stex_deactivate_macro:Nn \assign {copymodules}
3464  \stex_deactivate_macro:Nn \renamedecl {copymodules}
3465  \stex_deactivate_macro:Nn \donotcopy {copymodules}
3466
3467
3468  \seq_new:N \l_stex_implicit_morphisms_seq
3469  \NewDocumentCommand \implicitmorphism { O{} m m}{
3470    \stex_import_module_uri:nn { #1 } { #2 }
```

```
3471    \stex_debug:nn{implicits}{
3472      Implicit~morphism:~
3473      \l_stex_module_ns_str ? \l__stex_features_name_str
3474    }
3475    \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
3476      \l_stex_module_ns_str ? \l__stex_features_name_str
3477    }{
3478      \msg_error:nnn{stex}{error/conflictingmodules}{
3479        \l_stex_module_ns_str ? \l__stex_features_name_str
3480      }
3481    }
3482
3483    % TODO
3484
3485
3486
3487    \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
3488      \l_stex_module_ns_str ? \l__stex_features_name_str
3489    }
3490  }
3491
```

32.2 The feature environment

```
3492
3493  \NewDocumentEnvironment{structural@feature}{ m m m }{
3494    \stex_if_in_module:F {
3495      \msg_set:nnn{stex}{error/nomodule}{
3496        Structural~Feature~has~to~occur~in~a~module:\\
3497        Feature~#2~of~type~#1\\
3498        In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
3499      }
3500      \msg_error:nn{stex}{error/nomodule}
3501    }
3502
3503    \str_set:Nx \l_stex_module_name_str {
3504      \prop_item:Nn \l_stex_current_module_prop
3505        { name } / #2 - feature
3506    }
3507
3508    \str_set:Nx \l_stex_module_ns_str {
3509      \prop_item:Nn \l_stex_current_module_prop
3510        { ns }
3511    }
3512
3513
3514    \str_clear:N \l_tmpa_str
3515    \seq_clear:N \l_tmpa_seq
3516    \tl_clear:N \l_tmpa_tl
3517    \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
3518      origname  = #2,
3519      name      = \l_stex_module_name_str ,
3520      ns        = \l_stex_module_ns_str ,
```

```
3521      imports   = \exp_not:o { \l_tmpa_seq } ,
3522      constants = \exp_not:o { \l_tmpa_seq } ,
3523      content   = \exp_not:o { \l_tmpa_tl }   ,
3524      file      = \exp_not:o { \g_stex_currentfile_seq } ,
3525      lang      = \l_stex_module_lang_str ,
3526      sig       = \l_tmpa_str ,
3527      meta      = \l_tmpa_str ,
3528      feature   = #1 ,
3529    }
3530
3531    \stex_if_smsmode:TF {
3532      \stex_smsmode_set_codes:
3533    } {
3534      \begin{stex_annotate_env}{ feature:#1 }{}
3535        \stex_annotate_invisible:nnn{header}{}{ #3 }
3536    }
3537 }{
3538    \str_set:Nx \l_tmpa_str {
3539      c_stex_feature_
3540      \prop_item:Nn \l_stex_current_module_prop { ns } ?
3541      \prop_item:Nn \l_stex_current_module_prop { name }
3542      _prop
3543    }
3544    \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
3545    \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
3546    \stex_if_smsmode:TF {
3547      \exp_args:Nx \stex_add_to_sms:n {
3548        \prop_gset_from_keyval:cn {
3549          c_stex_feature_
3550          \prop_item:Nn \l_stex_current_module_prop { ns } ?
3551          \prop_item:Nn \l_stex_current_module_prop { name }
3552          _prop
3553        } {
3554        origname  = #2,
3555        name      = \prop_item:cn { \l_tmpa_str } { name } ,
3556        ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
3557        imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
3558        constants = \prop_item:cn { \l_tmpa_str } { constants } ,
3559        content   = \prop_item:cn { \l_tmpa_str } { content } ,
3560        file      = \prop_item:cn { \l_tmpa_str } { file } ,
3561        lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
3562        sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
3563        meta      = \prop_item:cn { \l_tmpa_str } { meta } ,
3564        feature   = \prop_item:cn { \l_tmpa_str } { feature }
3565      }
3566    }
3567    } {
3568        \end{stex_annotate_env}
3569    }
3570 }
3571
```

32.3 Features

```
3572
3573 \prop_new:N \l_stex_all_structures_prop
3574
3575 \keys_define:nn { stex / features / structure } {
3576   name          .str_set_x:N  = \l__stex_features_structure_name_str ,
3577 }
3578
3579 \cs_new_protected:Nn \__stex_features_structure_args:n {
3580   \str_clear:N \l__stex_features_structure_name_str
3581   \keys_set:nn { stex / features / structure } { #1 }
3582 }
3583
3584 %\stex_new_feature:nnnn { structure } { O{} m } {
3585 %  \__stex_features_structure_args:n { ##1 }
3586 %  \str_if_empty:NT \l__stex_features_structure_name_str {
3587 %    \str_set:Nx \l__stex_features_structure_name_str { ##2 }
3588 %  }
3589 %} {
3590 %
3591 %}
3592
3593 \NewDocumentEnvironment{mathstructure}{ O{} m }{
3594   \__stex_features_structure_args:n { #1 }
3595   \str_if_empty:NT \l__stex_features_structure_name_str {
3596     \str_set:Nx \l__stex_features_structure_name_str { #2 }
3597   }
3598   \exp_args:Nnnx
3599   \begin{structural@feature}{ structure }
3600     { \l__stex_features_structure_name_str }{}
3601   \seq_clear:N \l_tmpa_seq
3602   \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
3603
3604 }{
3605     \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
3606     \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
3607     \str_set:Nx \l_tmpa_str {
3608       \prop_item:Nn \l_stex_current_module_prop { ns } ?
3609       \prop_item:Nn \l_stex_current_module_prop { name }
3610     }
3611     \seq_map_inline:Nn \l_tmpa_seq {
3612       \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
3613     }
3614     \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
3615     \exp_args:Nnx
3616     \AddToHookNext { env / mathstructure / after }{
3617       \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}{
3618         \_stex_term_math_oms:nnnn { \l_tmpa_str }{}{O}{}
3619       }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
3620       \STEXexport {
3621         \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3622           {\prop_item:Nn \l_stex_current_module_prop { origname }}
```

155

```
3623              {\l_tmpa_str}
3624            \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3625              {#2}{\l_tmpa_str}
3626 %          \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3627 %            \prop_item:Nn \l_stex_current_module_prop { origname },
3628 %            \l_tmpa_str
3629 %          }
3630 %          \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3631 %            #2,\l_tmpa_str
3632 %          }
3633 %          \tl_set:cx { #2 } {
3634 %              \stex_invoke_structure:n { \l_tmpa_str }
3635          }
3636      }
3637
3638    \end{structural@feature}
3639    % \g_stex_last_feature_prop
3640 }
```

```
3641 \seq_new:N \l__stex_features_structure_field_seq
3642 \str_new:N \l__stex_features_structure_field_str
3643 \str_new:N \l__stex_features_structure_def_tl
3644 \prop_new:N \l__stex_features_structure_prop
3645 \NewDocumentCommand \instantiate { m O{} m }{
3646    \stex_smsmode_set_codes:
3647    \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
3648    \prop_set_eq:Nc \l__stex_features_structure_prop {
3649      c_stex_feature_\l_tmpa_str _prop
3650    }
3651    \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
3652    \seq_map_inline:Nn \l__stex_features_structure_field_seq {
3653      \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }
3654      \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3655        \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
3656        \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
3657          {!} \l_tmpa_tl
3658        \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3659          \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
3660          \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3661          \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3662        }{
3663          \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
3664          \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3665          \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
3666            \l_tmpa_tl
3667          \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3668            \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
3669            \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3670          }{
3671            \tl_clear:N \l_tmpb_tl
3672          }
3673        }
3674      }{
```

```
3675        \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
3676        \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3677          \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
3678          \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
3679          \tl_clear:N \l_tmpa_tl
3680        }{
3681          % TODO throw error
3682        }
3683      }
3684      % \l_tmpa_str: name
3685      % \l_tmpa_tl: definiens
3686      % \l_tmpb_tl: notation
3687      \tl_if_empty:NT \l__stex_features_structure_field_str {
3688        % TODO throw error
3689      }
3690      \str_clear:N \l_tmpb_str
3691
3692      \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3693      \seq_map_inline:Nn \l_tmpa_seq {
3694        \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
3695        \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
3696        \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
3697          \seq_map_break:n {
3698            \str_set:Nn \l_tmpb_str { ####1 }
3699          }
3700        }
3701      }
3702      \prop_get:cnN { l_stex_symdecl_ \l_tmpb_str _prop } {args}
3703        \l_tmpb_str
3704
3705      \tl_if_empty:NTF \l_tmpb_tl {
3706        \tl_if_empty:NF \l_tmpa_tl {
3707          \exp_args:Nx \use:n {
3708            \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fe
3709          }
3710        }
3711      }{
3712        \tl_if_empty:NTF \l_tmpa_tl {
3713          \exp_args:Nx \use:n {
3714            \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\e
3715          }
3716
3717        }{
3718          \exp_args:Nx \use:n {
3719            \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
3720            \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
3721          }
3722        }
3723      }
3724 %    \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
3725 %    \prop_item:Nn \l_stex_current_module_prop {name} ?
3726 %    #3/\l__stex_features_structure_field_str
3727 %    \par
3728 %    \expandafter\present\csname
```

157

```
3729 %       l_stex_symdecl_
3730 %         \prop_item:Nn \l_stex_current_module_prop {ns} ?
3731 %         \prop_item:Nn \l_stex_current_module_prop {name} ?
3732 %         #3/\l__stex_features_structure_field_str
3733 %         _prop
3734 %       \endcsname
3735   }
3736
3737   \tl_clear:N \l__stex_features_structure_def_tl
3738
3739   \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3740   \seq_map_inline:Nn \l_tmpa_seq {
3741     \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3742     \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3743     \exp_args:Nx \use:n {
3744       \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
3745
3746       }
3747     }
3748
3749     \prop_if_exist:cF {
3750       l_stex_symdecl_
3751       \prop_item:Nn \l_stex_current_module_prop {ns} ?
3752       \prop_item:Nn \l_stex_current_module_prop {name} ?
3753       #3/\l_tmpa_str
3754       _prop
3755     }{
3756       \prop_get:cnN { l_stex_symdecl_ ##1 _prop } {args}
3757         \l_tmpb_str
3758       \exp_args:Nx \use:n {
3759         \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
3760       }
3761     }
3762   }
3763
3764   \symdecl*[type={\STEXsymbol{module-type}{
3765     \_stex_term_math_oms:nnnn {
3766       \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3767       \prop_item:Nn \l__stex_features_structure_prop {name}
3768     }{}{0}{}
3769 }}]{#3}
3770
3771 % TODO: -> sms file
3772
3773   \tl_set:cx{ #3 }{
3774     \stex_invoke_structure:nnn {
3775       \prop_item:Nn \l_stex_current_module_prop {ns} ?
3776       \prop_item:Nn \l_stex_current_module_prop {name} ? #3
3777     } {
3778       \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3779       \prop_item:Nn \l__stex_features_structure_prop {name}
3780     }
3781   }
3782
```

158

```
3783 }
```

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

`\stex_invoke_structure:nnn`

```
3784 % #1: URI of the instance
3785 % #2: URI of the instantiated module
3786 \cs_new_protected:Nn \stex_invoke_structure:nnn {
3787   \tl_if_empty:nTF{ #3 }{
3788     \prop_set_eq:Nc \l__stex_features_structure_prop {
3789       c_stex_feature_ #2 _prop
3790     }
3791     \tl_clear:N \l_tmpa_tl
3792     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3793     \seq_map_inline:Nn \l_tmpa_seq {
3794       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3795       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3796       \cs_if_exist:cT {
3797         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
3798       }{
3799         \tl_if_empty:NF \l_tmpa_tl {
3800           \tl_put_right:Nn \l_tmpa_tl {,}
3801         }
3802         \tl_put_right:Nx \l_tmpa_tl {
3803           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
3804         }
3805       }
3806     }
3807     \exp_args:No \mathstruct \l_tmpa_tl
3808   }{
3809     \stex_invoke_symbol:n{#1/#3}
3810   }
3811 }
```

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

```
3812 ⟨/package⟩
```

Chapter 33

sTeX
-Statements Implementation

⟨∗package⟩

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

\protected\def\ignorespacesandpars{

\begingroup\catcode13=10\relax

\@ifnextchar\par{

\endgroup\expandafter\ignorespacesandpars\@gobble

}{

\endgroup

}

}

⟨@@=stex_statements⟩

Warnings and error messages

\titleemph

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

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

33.1 Definitions

definiendum

\keys_define:nn {stex / definiendum }{

post   .tl_set:N   = \l__stex_statements_definiendum_post_tl,

root   .str_set_x:N = \l__stex_statements_definiendum_root_str,

gfa   .str_set_x:N = \l__stex_statements_definiendum_gfa_str

}

\cs_new_protected:Nn __stex_statements_definiendum_args:n {

\str_clear:N \l__stex_statements_definiendum_root_str

\tl_clear:N \l__stex_statements_definiendum_post_tl

\str_clear:N \l__stex_statements_definiendum_gfa_str

```
3838        \keys_set:nn { stex / definiendum }{ #1 }
3839    }
3840    \NewDocumentCommand \definiendum { O{} m m} {
3841        \__stex_statements_definiendum_args:n { #1 }
3842        \stex_get_symbol:n { #2 }
3843        \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3844        \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
3845            \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
3846                \tl_set:Nn \l_tmpa_tl { #3 }
3847            } {
3848                \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
3849                \tl_set:Nn \l_tmpa_tl {
3850                    \l__stex_statements_definiendum_root_str\l__stex_statements_definiendum_post_tl
3851                }
3852            }
3853        } {
3854            \tl_set:Nn \l_tmpa_tl { #3 }
3855        }
3856
3857        % TODO root
3858        \rustex_if:TF {
3859            \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
3860        } {
3861            \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
3862        }
3863    }
3864    \stex_deactivate_macro:Nn \definiendum {definition~environments}
```

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

definame

```
3865
3866    \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3867
3868    \NewDocumentCommand \definame { O{} m } {
3869        \__stex_statements_definiendum_args:n { #1 }
3870        % TODO: root
3871        \stex_get_symbol:n { #2 }
3872        \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3873        \str_set:Nx \l_tmpa_str {
3874            \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3875        }
3876        \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3877        \rustex_if:TF {
3878            \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3879                \l_tmpa_str\l__stex_statements_definiendum_post_tl
3880            }
3881        } {
3882            \defemph@uri {
3883                \l_tmpa_str\l__stex_statements_definiendum_post_tl
3884            } { \l_stex_get_symbol_uri_str }
3885        }
3886    }
3887    \stex_deactivate_macro:Nn \definame {definition~environments}
```

```
3888
3889 \NewDocumentCommand \Definame { O{} m } {
3890   \__stex_statements_definiendum_args:n { #1 }
3891   \stex_get_symbol:n { #2 }
3892   \str_set:Nx \l_tmpa_str {
3893     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3894   }
3895   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3896   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3897   \rustex_if:TF {
3898     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3899       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3900     }
3901   } {
3902     \defemph@uri {
3903       \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
3904     } { \l_stex_get_symbol_uri_str }
3905   }
3906 }
3907 \stex_deactivate_macro:Nn \Definame {definition~environments}
3908
3909 \NewDocumentCommand \Symname { O{} m }{
3910   \stex_symname_args:n { #1 }
3911   \stex_get_symbol:n { #2 }
3912   \str_set:Nx \l_tmpa_str {
3913     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3914   }
3915   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3916   \let\compemph_uri_prev:\compemph@uri
3917   \let\compemph@uri\symrefemph@uri
3918   \exp_args:NNx \use:nn
3919   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }![
3920     \exp_after:wN \stex_capitalize:n \l_tmpa_str
3921       \l_stex_symname_post_str
3922   ] }
3923   \let\compemph@uri\compemph_uri_prev:
3924 }
```

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

sdefinition

```
3925
3926 \keys_define:nn {stex / sdefinition }{
3927   type    .str_set_x:N  = \sdefinitiontype,
3928   id      .str_set_x:N  = \sdefinitionid,
3929   title   .tl_set:N     = \sdefinitiontitle
3930 }
3931 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
3932   \str_clear:N \sdefinitiontype
3933   \str_clear:N \sdefinitionid
3934   \tl_clear:N \sdefinitiontitle
3935   \keys_set:nn { stex / sdefinition }{ #1 }
3936 }
3937
```

```
3938  \NewDocumentEnvironment{sdefinition}{O{}}{
3939    \__stex_statements_sdefinition_args:n{ #1 }
3940    \stex_reactivate_macro:N \definiendum
3941    \stex_reactivate_macro:N \definame
3942    \stex_reactivate_macro:N \Definame
3943    \stex_smsmode_set_codes:
3944    \stex_if_smsmode:F {
3945      \exp_args:Nnnx
3946      \begin{stex_annotate_env}{definition}{}
3947      \str_if_empty:NF \sdefinitiontype {
3948        \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{}
3949      }
3950    }
3951    \clist_set:No \l_tmpa_clist \sdefinitiontype
3952    \tl_clear:N \l_tmpa_tl
3953    \clist_map_inline:Nn \l_tmpa_clist {
3954      \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
3955        \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
3956      }
3957    }
3958    \tl_if_empty:NTF \l_tmpa_tl {
3959      \__stex_statements_sdefinition_start:
3960    }{
3961      \l_tmpa_tl
3962    }
3963    \stex_ref_new_doc_target:n \sdefinitionid
3964  }{
3965    \clist_set:No \l_tmpa_clist \sdefinitiontype
3966    \tl_clear:N \l_tmpa_tl
3967    \clist_map_inline:Nn \l_tmpa_clist {
3968      \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
3969        \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
3970      }
3971    }
3972    \tl_if_empty:NTF \l_tmpa_tl {
3973      \__stex_statements_sdefinition_end:
3974    }{
3975      \l_tmpa_tl
3976    }
3977    \stex_if_smsmode:F {
3978      \end{stex_annotate_env}
3979    }
3980  }
```

\stexpatchdefinition

```
3981  \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
3982    \par\noindent\titleemph{Definition\tl_if_empty:NF \sdefinitiontitle {
3983      ~(\sdefinitiontitle)
3984    }~}
3985  }
3986  \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\par\medskip}
3987
3988  \newcommand\stexpatchdefinition[3][] {
3989      \str_set:Nx \l_tmpa_str{ #1 }
```

163

```
3990      \str_if_empty:NTF \l_tmpa_str {
3991          \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
3992          \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
3993      }{
3994          \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2
3995          \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
3996      }
3997  }
```

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

\inlinedef   inline:
```
3998  \NewDocumentCommand \inlinedef { m } {
3999      \begingroup
4000      \stex_reactivate_macro:N \definiendum
4001      \stex_reactivate_macro:N \definame
4002      \stex_ref_new_doc_target:n{}
4003      #1
4004      \endgroup
4005  }
```

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

33.2   Assertions

sassertion
```
4006
4007  \keys_define:nn {stex / sassertion }{
4008      type    .str_set_x:N  = \sassertiontype,
4009      id      .str_set_x:N  = \sassertionid,
4010      title   .tl_set:N     = \sassertiontitle ,
4011      name    .str_set_x:N  = \sassertionname
4012  }
4013  \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
4014      \str_clear:N \sassertiontype
4015      \str_clear:N \sassertionid
4016      \str_clear:N \sassertionname
4017      \tl_clear:N \sassertiontitle
4018      \keys_set:nn { stex / sassertion }{ #1 }
4019  }
4020
4021  %\tl_new:N \g__stex_statements_aftergroup_tl
4022
4023  \NewDocumentEnvironment{sassertion}{O{}}{
4024      \__stex_statements_sassertion_args:n{ #1 }
4025      \stex_smsmode_set_codes:
4026      \stex_if_smsmode:F {
4027          \exp_args:Nnnx
4028          \begin{stex_annotate_env}{assertion}{}
4029          \str_if_empty:NF \sassertiontype {
4030              \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
4031          }
4032      }
```

```
4033      \clist_set:No \l_tmpa_clist \sassertiontype
4034      \tl_clear:N \l_tmpa_tl
4035      \clist_map_inline:Nn \l_tmpa_clist {
4036        \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
4037          \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
4038        }
4039      }
4040      \tl_if_empty:NTF \l_tmpa_tl {
4041        \__stex_statements_sassertion_start:
4042      }{
4043        \l_tmpa_tl
4044      }
4045      \stex_ref_new_doc_target:n \sassertionid
4046    }{
4047      \clist_set:No \l_tmpa_clist \sassertiontype
4048      \tl_clear:N \l_tmpa_tl
4049      \clist_map_inline:Nn \l_tmpa_clist {
4050        \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
4051          \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
4052        }
4053      }
4054      \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
4055      \tl_if_empty:NTF \l_tmpa_tl {
4056        \__stex_statements_sassertion_end:
4057      }{
4058        \l_tmpa_tl
4059      }
4060      \stex_if_smsmode:F {
4061        \end{stex_annotate_env}
4062      }
4063  }
```

\stexpatchassertion

```
4064
4065  \cs_new_protected:Nn \__stex_statements_sassertion_start: {
4066    \par\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {
4067      (\sassertiontitle)
4068    }~}
4069  }
4070  \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
4071
4072  \newcommand\stexpatchassertion[3][] {
4073      \str_set:Nx \l_tmpa_str{ #1 }
4074      \str_if_empty:NTF \l_tmpa_str {
4075        \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
4076        \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
4077      }{
4078        \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
4079        \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
4080      }
4081  }
```

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

\inlineass inline:

```
4082 \keys_define:nn {stex / inlineass }{
4083   type    .str_set_x:N  = \sassertiontype,
4084   id      .str_set_x:N  = \sassertionid,
4085   name    .str_set_x:N  = \sassertionname
4086 }
4087 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
4088   \str_clear:N \sassertiontype
4089   \str_clear:N \sassertionid
4090   \str_clear:N \sassertionname
4091   \tl_clear:N \sassertiontitle
4092   \keys_set:nn { stex / inlineass }{ #1 }
4093 }
4094 \NewDocumentCommand \inlineass { O{} m } {
4095   \begingroup
4096   \__stex_statements_inlineass_args:n{ #1 }
4097   \stex_ref_new_doc_target:n \sassertionid
4098   \stex_annotate:nnn{assertion}{}{
4099     \str_if_empty:NF \sassertiontype {
4100       \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
4101     }
4102     #1
4103   }
4104   \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
4105   \endgroup
4106 }
```

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

33.3   Examples

sexample

```
4107
4108 \keys_define:nn {stex / sexample }{
4109   type    .str_set_x:N  = \exampletype,
4110   id      .str_set_x:N  = \sexampleid,
4111   title   .tl_set:N     = \sexampletitle,
4112   for     .clist_set:N  = \sexamplefor,
4113 }
4114 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
4115   \str_clear:N \sexampletype
4116   \str_clear:N \sexampleid
4117   \tl_clear:N \sexampletitle
4118   \clist_clear:N \sexamplefor
4119   \keys_set:nn { stex / sexample }{ #1 }
4120 }
4121
4122 \NewDocumentEnvironment{sexample}{O{}}{
4123   \__stex_statements_sexample_args:n{ #1 }
4124   \stex_smsmode_set_codes:
4125   \stex_if_smsmode:F {
4126     \seq_clear:N \l_tmpa_seq
4127     \clist_map_inline:Nn \sexamplefor {
```

```
4128        \str_if_eq:nnF{ ##1 }{}{
4129          \stex_get_symbol:n { ##1 }
4130          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4131            \l_stex_get_symbol_uri_str
4132          }
4133        }
4134      }
4135      \exp_args:Nnnx
4136      \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
4137      \str_if_empty:NF \sexampletype {
4138        \stex_annotate_invisible:nnn{type}{\sexampletype}{}
4139      }
4140    }
4141    \stex_ref_new_doc_target:n \sexampleid
4142    \clist_set:No \l_tmpa_clist \sexampletype
4143    \tl_clear:N \l_tmpa_tl
4144    \clist_map_inline:Nn \l_tmpa_clist {
4145      \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
4146        \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
4147      }
4148    }
4149    \tl_if_empty:NTF \l_tmpa_tl {
4150      \__stex_statements_sexample_start:
4151    }{
4152      \l_tmpa_tl
4153    }
4154  }{
4155    \clist_set:No \l_tmpa_clist \sexampletype
4156    \tl_clear:N \l_tmpa_tl
4157    \clist_map_inline:Nn \l_tmpa_clist {
4158      \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
4159        \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
4160      }
4161    }
4162    \tl_if_empty:NTF \l_tmpa_tl {
4163      \__stex_statements_sexample_end:
4164    }{
4165      \l_tmpa_tl
4166    }
4167    \stex_if_smsmode:F {
4168      \end{stex_annotate_env}
4169    }
4170  }
```

\stexpatchexample

```
4171
4172  \cs_new_protected:Nn \__stex_statements_sexample_start: {
4173    \par\noindent\titleemph{Example~\tl_if_empty:NF \sexampletitle {
4174      (\sexampletitle)
4175    }~}
4176  }
4177  \cs_new_protected:Nn \__stex_statements_sexample_end: {\par\medskip}
4178
4179  \newcommand\stexpatchexample[3][] {
```

```
4180      \str_set:Nx \l_tmpa_str{ #1 }
4181      \str_if_empty:NTF \l_tmpa_str {
4182        \tl_set:Nn \__stex_statements_sexample_start: { #2 }
4183        \tl_set:Nn \__stex_statements_sexample_end: { #3 }
4184      }{
4185        \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
4186        \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
4187      }
4188  }
```

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

\inlineex   inline:

```
4189  \NewDocumentCommand \inlineex { m } {
4190    \begingroup
4191    \stex_ref_new_doc_target:n{}
4192    #1
4193    \endgroup
4194  }
```

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

33.4   Logical Paragraphs

sparagraph

```
4195  \keys_define:nn { stex / sparagraph} {
4196    id      .str_set_x:N  = \sparagraphid ,
4197    title   .tl_set:N     = \l_stex_sparagraph_title_tl ,
4198    type    .str_set_x:N  = \sparagraphtype ,
4199    for     .str_set_x:N  = \sparagraphfor ,
4200    from    .tl_set_x:N   = \sparagraphfrom ,
4201    start   .tl_set:N     = \l_stex_sparagraph_start_tl ,
4202    name    .str_set:N    = \sparagraphname
4203  }
4204
4205  \cs_new_protected:Nn \stex_sparagraph_args:n {
4206    \tl_clear:N \l_stex_sparagraph_title_tl
4207    \tl_clear:N \sparagraphfrom
4208    \tl_clear:N \l_stex_sparagraph_start_tl
4209    \str_clear:N \sparagraphid
4210    \str_clear:N \sparagraphtype
4211    \str_clear:N \sparagraphfor
4212    \str_clear:N \sparagraphname
4213    \keys_set:nn { stex / sparagraph }{ #1 }
4214  }
4215  \newif\if@in@omtext\@in@omtextfalse
4216
4217  \NewDocumentEnvironment {sparagraph} { O{} } {
4218    \stex_sparagraph_args:n { #1 }
4219    \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4220      \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
4221    }{
4222      \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
```

```
4223     }
4224     \@in@omtexttrue
4225     \stex_smsmode_set_codes:
4226     \stex_if_smsmode:F {
4227       \exp_args:Nnnx
4228       \begin{stex_annotate_env}{paragraph}{}
4229       \str_if_empty:NF \sparagraphtype {
4230         \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}
4231       }
4232     }
4233     \clist_set:No \l_tmpa_clist \sparagraphtype
4234     \tl_clear:N \l_tmpa_tl
4235     \clist_map_inline:Nn \l_tmpa_clist {
4236       \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
4237         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
4238       }
4239     }
4240     \tl_if_empty:NTF \l_tmpa_tl {
4241       \__stex_statements_sparagraph_start:
4242     }{
4243       \l_tmpa_tl
4244     }
4245     \stex_ref_new_doc_target:n \sparagraphid
4246     \ignorespacesandpars
4247   }{
4248     \clist_set:No \l_tmpa_clist \sparagraphtype
4249     \tl_clear:N \l_tmpa_tl
4250     \clist_map_inline:Nn \l_tmpa_clist {
4251       \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
4252         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
4253       }
4254     }
4255     \str_if_empty:NF \sparagraphname { \symdecl*{\sparagraphname} }
4256     \tl_if_empty:NTF \l_tmpa_tl {
4257       \__stex_statements_sparagraph_end:
4258     }{
4259       \l_tmpa_tl
4260     }
4261     \stex_if_smsmode:F {
4262       \end{stex_annotate_env}
4263     }
4264 }
```

\stexpatchparagraph

```
4265
4266 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
4267   \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4268     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
4269       \titleemph{\l_stex_sparagraph_title_tl}:~
4270     }
4271   }{
4272     \titleemph{\l_stex_sparagraph_start_tl}~
4273   }
4274 }
```

169

```
4275 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
4276
4277 \newcommand\stexpatchparagraph[3][] {
4278     \str_set:Nx \l_tmpa_str{ #1 }
4279     \str_if_empty:NTF \l_tmpa_str {
4280        \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
4281        \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
4282     }{
4283        \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
4284        \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3
4285     }
4286 }
```

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

symboldoc

```
4287 \NewDocumentEnvironment{symboldoc}{ m }{
4288    \seq_set_split:Nnn \l_tmpa_seq , { #1 }
4289    \seq_clear:N \l_tmpb_seq
4290    \seq_map_inline:Nn \l_tmpa_seq {
4291      \str_if_eq:nnF{ ##1 }{}{
4292         \stex_get_symbol:n { ##1 }
4293         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
4294           \l_stex_get_symbol_uri_str
4295         }
4296      }
4297    }
4298    \par
4299    \exp_args:Nnnx
4300    \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
4301 }{
4302    \end{stex_annotate_env}
4303 }
```

```
4304 ⟨/package⟩
```

Chapter 34

The Implementation

34.1 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).[13]

```
4305 ⟨∗package⟩
4306 ⟨@@=stex_sproof⟩
4307
4308 %%%%%%%%%%%%    sproof.dtx    %%%%%%%%%%%%
4309
```

34.2 Proofs

We first define some keys for the `proof` environment.

```
4310 \keys_define:nn { stex / spf } {
4311    id          .str_set_x:N  = \l__stex_sproof_spf_id_str,
4312    display     .tl_set:N     = \l__stex_sproof_spf_display_tl,
4313    for         .tl_set:N     = \l__stex_sproof_spf_for_tl ,
4314    from        .tl_set:N     = \l__stex_sproof_spf_from_tl ,
4315    proofend    .tl_set:N     = \l__stex_sproof_spf_proofend_tl,
4316    type        .tl_set:N     = \l__stex_sproof_spf_type_tl,
4317    title       .tl_set:N     = \l__stex_sproof_spf_title_tl,
4318    continues   .tl_set:N     = \l__stex_sproof_spf_continues_tl,
4319    functions   .tl_set:N     = \l__stex_sproof_spf_functions_tl,
4320    method      .tl_set:N     = \l__stex_sproof_spf_method_tl
4321 }
4322 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
4323 \str_clear:N \l__stex_sproof_spf_id_str
4324 \tl_clear:N \l__stex_sproof_spf_display_tl
4325 \tl_clear:N \l__stex_sproof_spf_for_tl
4326 \tl_clear:N \l__stex_sproof_spf_from_tl
4327 \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
4328 \tl_clear:N \l__stex_sproof_spf_type_tl
4329 \tl_clear:N \l__stex_sproof_spf_title_tl
```

---

[13]EDNOTE: need an implementation for LaTeXML

```
4330 \tl_clear:N \l__stex_sproof_spf_continues_tl
4331 \tl_clear:N \l__stex_sproof_spf_functions_tl
4332 \tl_clear:N \l__stex_sproof_spf_method_tl
4333 \keys_set:nn { stex / spf }{ #1 }
4334 }
```

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

```
4335 \def\spf@flow{flow}
```

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

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[6] 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.

```
4336 \newcount\count_ten
4337 \newenvironment{pst@with@label}[1]{
4338     \edef\pst@label{#1}
4339     \advance\count_ten by 1\relax
4340     \count_ten=1
4341 }{
4342     \advance\count_ten by -1\relax
4343 }
```

\the@pst@label   \the@pst@label evaluates to the current step label.

```
4344 \def\the@pst@label{
4345     \pst@make@label\pst@label{\number\count_ten}\l__stex_sproof_pstlabel_postfix_tl
4346 }
```

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

\setpstlabelstyle   \setpstlabelstyle{metaKey-Val pairs} makes the labeling style customizable. \setpstlabelstyle{pr
will change the labeling style from **P.1.2.3** to **Pr-1-2-3†**. \setpstlabelstyledefault
will set the labeling style back to default.

```
4347 \keys_define:nn { stex / pstlabel }{
4348     prefix      .tl_set:N   = \l__stex_sproof_pstlabel_prefix_tl,
4349     delimiter   .tl_set:N   = \l__stex_sproof_pstlabel_delimiter_tl,
4350     postfix     .tl_set:N   = \l__stex_sproof_pstlabel_postfix_tl
4351 }
4352 \cs_new_protected:Nn \__stex_sproof_pstlabel_args:n {
```

---

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

```
4353    \tl_set:Nn \l__stex_sproof_pstlabel_prefix_tl {P}
4354    \tl_set:Nn \l__stex_sproof_pstlabel_delimiter_tl {.}
4355    \tl_clear:N \l__stex_sproof_pstlabel_postfix_tl
4356 }
4357 \__stex_sproof_pstlabel_args:n {}
4358 \newcommand\setpstlabelstyle[1]{
4359    \__stex_sproof_pstlabel_args:n {#1}
4360 }
4361 \newcommand\setpstlabelstyledefault{%
4362    \__stex_sproof_pstlabel_args:n{prefix=P,delimiter=.,postfix={}}
4363 }
```

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

\pstlabelstyle   `\pstlabelstyle` just sets the `\pst@make@label` macro according to the style.

```
4364    \ExplSyntaxOff
4365    \def\pst@make@label@long#1#2{\@for\@I:=#1\do{\expandafter\expandafter\expandafter\@I\csname
4366    \def\pst@make@label@angles#1#2{\ensuremath{\@for\@I:=#1\do{\rangle}}#2}
4367    \def\pst@make@label@short#1#2{#2}
4368    \def\pst@make@label@empty#1#2{}
4369    \ExplSyntaxOn
4370    \def\pstlabelstyle#1{%
4371       \def\pst@make@label{\use:c{pst@make@label@#1}}%
4372    }%
4373    \pstlabelstyle{long}%
```

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

\next@pst@label   `\next@pst@label` increments the step label at the current level.

```
4374    \def\next@pst@label{%
4375       \global\advance\count\count10 by 1%
4376    }%
```

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

\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

```
4377    \def\sproof@box{
4378       \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
4379    }
4380    \def\spf@proofend{\sproof@box}
4381    \def\sproofend{
4382       \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
4383          \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
4384       }
4385    }
4386    \def\sProofEndSymbol#1{\def\sproof@box{#1}}
```

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

spf@*@kw

```
4387    \def\spf@proofsketch@kw{Proof Sketch}
4388    \def\spf@proof@kw{Proof}
4389    \def\spf@step@kw{Step}
```

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

For the other languages, we set up triggers

```
4390  \AddToHook{begindocument}{
4391    \ltx@ifpackageloaded{babel}{
4392      \makeatletter
4393      \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4394      \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4395        \input{sproof-ngerman.ldf}
4396      }
4397      \clist_if_in:NnT \l_tmpa_clist {finnish}{
4398        \input{sproof-finnish.ldf}
4399      }
4400      \clist_if_in:NnT \l_tmpa_clist {french}{
4401        \input{sproof-french.ldf}
4402      }
4403      \clist_if_in:NnT \l_tmpa_clist {russian}{
4404        \input{sproof-russian.ldf}
4405      }
4406      \makeatother
4407    }{}
4408  }
```

spfsketch

```
4409  \newcommand\spfsketch[2][]{
4410    \__stex_sproof_spf_args:n{#1}
4411    \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4412      \titleemph{
4413        \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {
4414          \spf@proofsketch@kw
4415        }{
4416          \l__stex_sproof_spf_type_tl
4417        }
4418      }:
4419    }
4420    {~#2}
4421    %\sref@label@id{this \ifx\spf@type\@empty\spf@proofsketch@kw\else\spf@type\fi}
4422    \sproofend
4423  }
```

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

spfeq This is very similar to \spfsketch, but uses a computation array[14][15]

```
4424  \newenvironment{spfeq}[2][]{
4425    \__stex_sproof_spf_args:n{#1}
4426    %\sref@target
4427    \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4428      \titleemph{
4429        \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {
4430          \spf@proof@kw
4431        }{
```

---

[14]EDNOTE: This should really be more like a tabular with an ensuremath in it. or invoke text on the last column

[15]EDNOTE: document above

174

```
4432        \l__stex_sproof_spf_type_tl
4433      }
4434    }:
4435  }
4436  {~#2}
4437  \begin{displaymath}\begin{array}{rcll}
4438 }{
4439    \end{array}\end{displaymath}
4440 }
```

(*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.

```
4441 \newenvironment{spf@proof}[2][]{
4442    \__stex_sproof_spf_args:n{#1}
4443    %\sref@target
4444    \count_ten=10
4445    \par\noindent
4446    \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4447      \titleemph{
4448        \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {
4449          \spf@proof@kw
4450        }{
4451          \l__stex_sproof_spf_type_tl
4452        }
4453      }:
4454    }
4455    {~#2}
4456    %\sref@label@id{this \ifx\spf@type\@empty\spf@proof@kw\else\spf@type\fi}
4457    \def\pst@label{}
4458    \newcount\pst@count% initialize the labeling mechanism
4459    \begin{description}\begin{pst@with@label}{\l__stex_sproof_pstlabel_prefix_tl}
4460 }{
4461    \end{pst@with@label}\end{description}
4462 }
4463 \newenvironment{sproof}[2][]{\begin{spf@proof}[#1]{#2}}{\sproofend\end{spf@proof}}
4464 \newenvironment{sProof}[2][]{\begin{spf@proof}[#1]{#2}}{\end{spf@proof}}
```

\spfidea

```
4465 \newcommand\spfidea[2][]{
4466    \__stex_sproof_spf_args:n{#1}
4467    \titleemph{
4468      \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {Proof~Idea}{
4469        \l__stex_sproof_spf_type_tl
4470      }:
4471    }~#2
4472    \sproofend
4473 }
```

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

The next two environments (proof steps) and comments, are mostly semantical, they take `KeyVal` arguments that specify their semantic role. In draft mode, they read these

values and show them. If the surrounding proof had `display=flow`, then no new `\item` is generated, otherwise it is. In any case, the proof step number (at the current level) is incremented.

spfstep [16]

```
4474 \newenvironment{spfstep}[1][]{
4475   \__stex_sproof_spf_args:n{#1}
4476   \@in@omtexttrue
4477   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4478     \item[\the@pst@label]
4479   }
4480   \tl_if_empty:NF \l__stex_sproof_spf_title_tl {
4481     {(\titleemph{\l__stex_sproof_spf_title_tl})\enspace}
4482   }
4483   %\sref@label@id{\pst@label}
4484   \ignorespacesandpars
4485 }{
4486   \next@pst@label\ignorespacesandpars
4487 }
```

sproofcomment

```
4488 \newenvironment{sproofcomment}[1][]{
4489   \__stex_sproof_spf_args:n{#1}
4490   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4491     \item[\the@pst@label]
4492   }
4493 }{
4494   \next@pst@label
4495 }
```

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.

```
4496 \newenvironment{subproof}[2][]{
4497   \__stex_sproof_spf_args:n{#1}
4498   \def\@test{#2}
4499   \ifx\@test\empty\else
4500     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4501       \item[\the@pst@label]
4502     }{#2}
4503   \fi
4504   \begin{pst@with@label}{\pst@label,\number\count_ten}
4505 }{
4506   \end{pst@with@label}\next@pst@label
4507 }
```

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

```
4508 \newenvironment{spfcases}[2][]{
4509   \def\@test{#1}
4510   \ifx\@test\empty
4511     \begin{subproof}[method=by-cases]{#2}
```

---

[16]EDNOTE: MK: labeling of steps does not work yet.

```
4512      \else
4513        \begin{subproof}[#1,method=by-cases]{#2}
4514      \fi
4515 }{
4516      \end{subproof}
4517 }
```

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

```
4518 \newenvironment{spfcase}[2][]{
4519      \__stex_sproof_spf_args:n{#1}
4520      \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4521        \item[\the@pst@label]
4522      }
4523      \def\@test{#2}
4524      \ifx\@test\@empty
4525      \else
4526        {\titleemph{#2}:~}
4527      \fi
4528      \begin{pst@with@label}{\pst@label,\number\count_ten}
4529 }{
4530      \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4531        \sproofend
4532      }
4533      \end{pst@with@label}
4534      \next@pst@label
4535 }
```

spfcase   similar to `spfcase`, takes a third argument.

```
4536 \newcommand\spfcasesketch[3][]{
4537      \__stex_sproof_spf_args:n{#1}
4538      \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4539        \item[\the@pst@label]
4540      }
4541      \def\@test{#2}
4542      \ifx\@test\@empty
4543      \else
4544        {\titleemph{#2}:~}
4545      \fi#3
4546      \next@pst@label
4547 }%
```

34.3   Justifications

We define the actions that are undertaken, when the keys for justifications are encoun-
tered. Here this is very simple, we just define an internal macro with the value, so that
we can use it later.

```
4548 \keys_define:nn { stex / just }{
4549      id        .str_set_x:N  = \l__stex_sproof_just_id_str,
4550      method    .tl_set:N     = \l__stex_sproof_just_method_tl,
4551      premises  .tl_set:N     = \l__stex_sproof_just_premises_tl,
4552      args      .tl_set:N     = \l__stex_sproof_just_args_tl
4553 }
```

The next three environments and macros are purely semantic, so we ignore the keyval arguments for now and only display the content.[17]

justification

4554 `\newenvironment{justification}[1][]{}{}`

\premise

4555 `\newcommand\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.

4556 `\newcommand\justarg[2][]{#2}`
4557 ⟨/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.

---

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

Chapter 35

ST_EX -Others Implementation

4558 ⟨∗package⟩
4559
4560 %%%%%%%%%%%%   others.dtx   %%%%%%%%%%%%
4561
4562 ⟨@@=stex_others⟩

Warnings and error messages
4563   % None

\MSC Math subject classifier

4564 \NewDocumentCommand \MSC {m} {
4565   % TODO
4566 }

(*End definition for* \MSC. *This function is documented on page 21.*)

Patching tikzinput, if loaded

4567 \@ifpackageloaded{tikzinput}{
4568   \RequirePackage{stex-tikzinput}
4569 }{}

4570 ⟨/package⟩

Chapter 36

sTeX -Metatheory Implementation

```
4571 ⟨∗package⟩
4572 ⟨@@=stex_modules⟩
4573
4574 %%%%%%%%%%%%  metatheory.dtx  %%%%%%%%%%%%
4575
4576 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
4577 \begingroup
4578 \stex_module_setup:nn{
4579   ns=\c_stex_metatheory_ns_str,
4580   meta=NONE
4581 }{Metatheory}
4582 \stex_reactivate_macro:N \symdecl
4583 \stex_reactivate_macro:N \notation
4584 \stex_reactivate_macro:N \symdef
4585 \ExplSyntaxOff
4586 \csname stex_suppress_html:n\endcsname{
4587   % is-a (a:A, a \in A, a is an A, etc.)
4588   \symdecl[args=ai]{isa}
4589   \notation[typed]{isa}{#1 \comp{:} #2}{#1 \comp, #2}
4590   \notation[in]{isa}{#1 \comp\in #2}{#1 \comp, #2}
4591   \notation[pred]{isa}{#2\comp(#1 \comp)}{#1 \comp, #2}
4592
4593   % bind (\forall, \Pi, \lambda etc.)
4594   \symdecl[args=Bi]{bind}
4595   \notation[forall]{bind}{\comp\forall #1.\;#2}{#1 \comp, #2}
4596   \notation[Pi]{bind}{\comp\prod_{#1}#2}{#1 \comp, #2}
4597   \notation[depfun]{bind}{\comp( #1 \comp)\;\to\;} #2}{#1 \comp, #2}
4598
4599   % dummy variable
4600   \symdecl{dummyvar}
4601   \notation[underscore]{dummyvar}{\comp\_}
4602   \notation[dot]{dummyvar}{\comp\cdot}
4603   \notation[dash]{dummyvar}{\comp{{\rm --}}}
4604
4605   %fromto (function space, Hom-set, implication etc.)
```

```
4606     \symdecl[args=ai]{fromto}
4607     \notation[xarrow]{fromto}{#1 \comp\to #2}{#1 \comp\times #2}
4608     \notation[arrow]{fromto}{#1 \comp\to #2}{#1 \comp\to #2}

4609
4610     % mapto (lambda etc.)
4611     %\symdecl[args=Bi]{mapto}
4612     %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
4613     %\notation[lambda]{mapto}{\comp\lambda #1 \comp.\; #2}{#1 \comp, #2}
4614     %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp.\; #2}{#1 \comp, #2}

4615
4616     % function/operator application
4617     \symdecl[args=ia]{apply}
4618     \notation[prec=0;0x\infprec,parens]{apply}{#1 \comp( #2 \comp)}{#1 \comp, #2}
4619     \notation[prec=0;0x\infprec,lambda]{apply}{#1 \; #2 }{#1 \; #2}

4620
4621     % ``type'' of all collections (sets,classes,types,kinds)
4622     \symdecl{collection}
4623     \notation[U]{collection}{\comp{\mathcal{U}}}
4624     \notation[set]{collection}{\comp{\textsf{Set}}}

4625
4626     % sequences
4627     \symdecl[args=1]{seqtype}
4628     \notation[kleene]{seqtype}{#1^{\comp\ast}}

4629
4630     \symdef[args=2,li,prec=nobrackets]{sequence-index}{#1_{#2}}
4631     \notation[ui,prec=nobrackets]{sequence-index}{#1^{#2}}

4632
4633     %\symdef[args=3,li]{sequence-from-to}{#1_{#2}\comp{,\ellipses,}#1_{#3}}
4634     %\notation[ui]{sequence-from-to}{#1^{#2}\comp{,\ellipses,}#1^{#3}}
4635     % ^ superceded by \aseqfromto and \livar/\uivar

4636
4637     \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{,\ellipses}}{#1\comp,#2}
4638     \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{,\ellipses,}#2}{#1\comp,#2}
4639     \symdef[args=aii,prec=nobrackets]{aseqfromtovia}{#1\comp{,\ellipses,}#2\comp{,\ellipses,}#

4640
4641     % letin (``let'', local definitions, variable substitution)
4642     \symdecl[args=bii]{letin}
4643     \notation[let]{letin}{\comp{{\rm let}}\;#1\comp{=}#2\;\comp{{\rm in}}\;#3}
4644     \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
4645     \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}

4646
4647     % structures
4648     \symdecl*[args=1]{module-type}
4649     \notation{module-type}{\mathtt{MOD} #1}
4650     \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
4651     \notation[angle,prec=nobrackets]{mathstruct}{\comp\langle #1 \comp\rangle}{#1 \comp, #2}

4652
4653 }
4654     \ExplSyntaxOn
4655     \stex_add_to_current_module:n{
4656       \let\nappa\apply
4657       \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
4658       \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
4659       \def\livar{\csname sequence-index\endcsname[li]}
```

181

```
4660        \def\uivar{\csname sequence-index\endcsname[ui]}
4661        \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
4662        \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
4663        \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
4664      }
4665    \__stex_modules_end_module:
4666    \endgroup
4667    ⟨/package⟩
```

Chapter 37

Tikzinput Implementation

```
4668  ⟨∗package⟩
4669
4670  %%%%%%%%%%%%  tikzinput.dtx  %%%%%%%%%%%%
4671
4672  \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
4673  \RequirePackage{l3keys2e}
4674
4675  \keys_define:nn { tikzinput } {
4676    image    .bool_set:N   = \c_tikzinput_image_bool,
4677    image    .default:n    = false ,
4678    unknown    .code:n        = {}
4679  }
4680
4681  \ProcessKeysOptions { tikzinput }
4682
4683  \bool_if:NTF \c_tikzinput_image_bool {
4684    \RequirePackage{graphicx}
4685
4686    \providecommand\usetikzlibrary[]{}
4687    \newcommand\tikzinput[2][]{\includegraphics[#1]{#2}}
4688  }{
4689    \RequirePackage{tikz}
4690    \RequirePackage{standalone}
4691
4692    \newcommand \tikzinput [2] [] {
4693      \setkeys{Gin}{#1}
4694      \ifx \Gin@ewidth \Gin@exclamation
4695        \ifx \Gin@eheight \Gin@exclamation
4696          \input { #2 }
4697        \else
4698          \resizebox{!}{ \Gin@eheight }{
4699            \input { #2 }
4700          }
4701        \fi
4702      \else
4703        \ifx \Gin@eheight \Gin@exclamation
4704          \resizebox{ \Gin@ewidth }{!}{
4705            \input { #2 }
```

```
4706                }
4707            \else
4708                \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
4709                    \input { #2 }
4710                }
4711            \fi
4712        \fi
4713        }
4714    }
4715
4716    \newcommand \ctikzinput [2] [] {
4717        \begin{center}
4718            \tikzinput [#1] {#2}
4719        \end{center}
4720    }
4721
4722    \@ifpackageloaded{stex}{
4723        \RequirePackage{stex-tikzinput}
4724    }{}
4725
4726  ⟨/package⟩
4727  ⟨∗stex⟩
4728    \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
4729    \RequirePackage{stex}
4730    \RequirePackage{tikzinput}
4731
4732    \newcommand\mhtikzinput[2][]{%
4733        \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
4734        \stex_in_repository:nn\Gin@mhrepos{
4735            \tikzinput[#1]{\mhpath{##1}{#2}}
4736        }
4737    }
4738    \newcommand\cmhtikzinput[2][]{\begin{center}\mhtikzinput[#1]{#2}\end{center}}
4739  ⟨/stex⟩
```

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

Chapter 38

document-structure.sty Implementation

38.1 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.

```
4740 ⟨∗cls⟩
4741 ⟨@@=document_structure⟩
4742 \ProvidesExplClass{document-structure}{2022/02/10}{3.0}{Modular Document Structure Class}
4743 \RequirePackage{l3keys2e,expl-keystr-compat}
```

38.2 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

```
4744 \keys_define:nn{ document-structure / pkg }{
4745   class       .str_set_x:N  = \c_document_structure_class_str,
4746   minimal     .bool_set:N   = \c_document_structure_minimal_bool,
4747   report      .code:n       = {
4748     \ClassWarning{document-structure}{the option 'report' is deprecated, use 'class=report',
4749     \str_set:Nn \c_document_structure_class_str {report}
4750   },
4751   book        .code:n       = {
4752     \ClassWarning{document-structure}{the option 'book' is deprecated, use 'class=book', ins
4753     \str_set:Nn \c_document_structure_class_str {book}
4754   },
4755   bookpart    .code:n       = {
4756     \ClassWarning{document-structure}{the option 'bookpart' is deprecated, use 'class=book,t
4757     \str_set:Nn \c_document_structure_class_str {book}
4758     \str_set:Nn \c_document_structure_topsect_str {chapter}
4759   },
```

```
4760    docopt      .str_set_x:N  = \c_document_structure_docopt_str,
4761    unknown     .code:n       = {
4762      \PassOptionsToPackage{ \CurrentOption }{ document-structure }
4763    }
4764  }
4765  \ProcessKeysOptions{ document-structure / pkg }
4766  \str_if_empty:NT \c_document_structure_class_str {
4767    \str_set:Nn \c_document_structure_class_str {article}
4768  }
4769  \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
4770    {\c_document_structure_class_str}
4771
```

38.3   Beefing up the `document` environment

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

```
4772  \RequirePackage{document-structure}
4773  \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
picked up by LATEXML.[18]

```
4774  \keys_define:nn { document-structure / document }{
4775    id .str_set_x:N = \c_document_structure_document_id_str
4776  }
4777  \let\__document_structure_orig_document=\document
4778  \renewcommand{\document}[1][]{
4779    \keys_set:nn{ document-structure / document }{ #1 }
4780    \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
4781    \__document_structure_orig_document
4782  }
```

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

```
4783  }
4784  ⟨/cls⟩
```

38.4   Implementation: document-structure Package

```
4785  ⟨*package⟩
4786  \ProvidesExplPackage{document-structure}{2022/02/10}{3.0}{Modular Document Structure}
4787  \RequirePackage{expl-keystr-compat,l3keys2e}
```

38.5   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).

---

[18]EdNote: faking documentkeys for now. @HANG, please implement

```
4788
4789 \keys_define:nn{ document-structure / pkg }{
4790   class      .str_set_x:N  = \c_document_structure_class_str,
4791   topsect    .str_set_x:N  = \c_document_structure_topsect_str,
4792 % showignores .bool_set:N   = \c_document_structure_showignores_bool,
4793 }
4794 \ProcessKeysOptions{ document-structure / pkg }
4795 \str_if_empty:NT \c_document_structure_class_str {
4796   \str_set:Nn \c_document_structure_class_str {article}
4797 }
4798 \str_if_empty:NT \c_document_structure_topsect_str {
4799   \str_set:Nn \c_document_structure_topsect_str {section}
4800 }
```

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

```
4801 \RequirePackage{xspace}
4802 \RequirePackage{comment}
4803 \AddToHook{begindocument}{
4804 \ltx@ifpackageloaded{babel}{
4805    \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4806    \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4807      \makeatletter\input{omdoc-ngerman.ldf}\makeatother
4808    }
4809  }{}
4810 }
```

\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.

```
4811 \int_new:N \l_document_structure_section_level_int
4812 \str_case:VnF \c_document_structure_topsect_str {
4813   {part}{
4814     \int_set:Nn \l_document_structure_section_level_int {0}
4815   }
4816   {chapter}{
4817     \int_set:Nn \l_document_structure_section_level_int {1}
4818   }
4819 }{
4820   \str_case:VnF \c_document_structure_class_str {
4821     {book}{
4822       \int_set:Nn \l_document_structure_section_level_int {0}
4823     }
4824     {report}{
4825       \int_set:Nn \l_document_structure_section_level_int {0}
4826     }
4827   }{
4828     \int_set:Nn \l_document_structure_section_level_int {2}
4829   }
4830 }
```

38.6 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.[19]

EdN:19

```
4831 \def\current@section@level{document}%
4832 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
4833 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

\skipomgroup

```
4834 \cs_new_protected:Npn \skipomgroup {
4835   \ifcase\l_document_structure_section_level_int
4836   \or\stepcounter{part}
4837   \or\stepcounter{chapter}
4838   \or\stepcounter{section}
4839   \or\stepcounter{subsection}
4840   \or\stepcounter{subsubsection}
4841   \or\stepcounter{paragraph}
4842   \or\stepcounter{subparagraph}
4843   \fi
4844 }
```

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

blindomgroup

```
4845 \newcommand\at@begin@blindomgroup[1]{}
4846 \newenvironment{blindomgroup}
4847 {
4848   \int_incr:N\l_document_structure_section_level_int
4849   \at@begin@blindomgroup\l_document_structure_section_level_int
4850 }{}
```

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

```
4851 \newcommand\omgroup@nonum[2]{
4852   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
4853   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
4854 }
```

(*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.

```
4855 \newcommand\omgroup@num[2]{
```

---

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

```
4856    \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
4857      \@nameuse{#1}{#2}
4858    }{
4859      \cs_if_exist:NTF\rdfmeta@sectioning{
4860        \@nameuse{rdfmeta@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
4861      }{
4862        \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
4863      }
4864    }
4865  %\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\omgroup@id
4866 }
```

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

<div style="text-align:left">omgroup</div>

```
4867 \keys_define:nn { document-structure / omgroup }{
4868   id            .str_set_x:N = \l__document_structure_omgroup_id_str,
4869   date          .str_set_x:N = \l__document_structure_omgroup_date_str,
4870   creators      .clist_set:N = \l__document_structure_omgroup_creators_clist,
4871   contributors  .clist_set:N = \l__document_structure_omgroup_contributors_clist,
4872   srccite       .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
4873   type          .tl_set:N    = \l__document_structure_omgroup_type_tl,
4874   short         .tl_set:N    = \l__document_structure_omgroup_short_tl,
4875   display       .tl_set:N    = \l__document_structure_omgroup_display_tl,
4876   intro         .tl_set:N    = \l__document_structure_omgroup_intro_tl,
4877   loadmodules   .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
4878 }
4879 \cs_new_protected:Nn \__document_structure_omgroup_args:n {
4880   \str_clear:N \l__document_structure_omgroup_id_str
4881   \str_clear:N \l__document_structure_omgroup_date_str
4882   \clist_clear:N \l__document_structure_omgroup_creators_clist
4883   \clist_clear:N \l__document_structure_omgroup_contributors_clist
4884   \tl_clear:N \l__document_structure_omgroup_srccite_tl
4885   \tl_clear:N \l__document_structure_omgroup_type_tl
4886   \tl_clear:N \l__document_structure_omgroup_short_tl
4887   \tl_clear:N \l__document_structure_omgroup_display_tl
4888   \tl_clear:N \l__document_structure_omgroup_intro_tl
4889   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
4890   \keys_set:nn { document-structure / omgroup } { #1 }
4891 }
```

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.

```
4892 \newif\if@mainmatter\@mainmattertrue
4893 \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.

```
4894 \keys_define:nn { document-structure / sectioning }{
4895   name    .str_set_x:N  = \l__document_structure_sect_name_str    ,
4896   ref     .str_set_x:N  = \l__document_structure_sect_ref_str     ,
4897   clear   .bool_set:N   = \l__document_structure_sect_clear_bool  ,
4898   num     .bool_set:N   = \l__document_structure_sect_num_bool    ,
4899 }
```

```
4900  \cs_new_protected:Nn \__document_structure_sect_args:n {
4901    \str_clear:N \l__document_structure_sect_name_str
4902    \str_clear:N \l__document_structure_sect_ref_str
4903    \bool_set_false:N \l__document_structure_sect_clear_bool
4904    \bool_set_false:N \l__document_structure_sect_num_bool
4905    \keys_set:nn { document-structure / sectioning } { #1 }
4906  }
4907  \newcommand\omdoc@sectioning[3][]{
4908    \__document_structure_sect_args:n {#1 }
4909    \let\omdoc@sect@name\l__document_structure_sect_name_str
4910    \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
4911    \if@mainmatter% numbering not overridden by frontmatter, etc.
4912      \bool_if:NTF \l__document_structure_sect_num_bool {
4913        \omgroup@num{#2}{#3}
4914      }{
4915        \omgroup@nonum{#2}{#3}
4916      }
4917      \def\current@section@level{\omdoc@sect@name}
4918    \else
4919      \omgroup@nonum{#2}{#3}
4920    \fi
4921  }% 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.

```
4922  \newcommand\omgroup@redefine@addtocontents[1]{%
4923  %\edef\__document_structureimport{#1}%
4924  %\@for\@I:=\__document_structureimport\do{%
4925  %\edef\@path{\csname module@\@I  @path\endcsname}%
4926  %\@ifundefined{tf@toc}\relax%
4927  %     {\protected@write\tf@toc{}{\string\@requiremodules{\@path}}}}}
4928  %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
4929  %\def\addcontentsline##1##2##3{%
4930  %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}
4931  %\else% hyperref.sty not loaded
4932  %\def\addcontentsline##1##2##3{%
4933  %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}{
4934  %\fi
4935  }% 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.

```
4936  \int_new:N \l_document_structure_omgroup_level_int
4937  \newenvironment{omgroup}[2][]% keys, title
4938  {
4939    \__document_structure_omgroup_args:n { #1 }%\sref@target%
```

If the `loadmodules` key is set on `\begin{omgroup}`, we redefine the `\addcontetsline` macro that determines how the sectioning commands below construct the entries for the table of contents.

```
4940    \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
4941      \omgroup@redefine@addtocontents{
4942        %\@ifundefined{module@id}\used@modules%
4943        %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}
```

```
4944          }
4945        }
```

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

```
4946    \int_incr:N \l_document_structure_omgroup_level_int
4947    \int_incr:N\l_document_structure_section_level_int
4948    \ifcase\l_document_structure_section_level_int
4949      \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
4950      \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
4951      \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
4952      \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
4953      \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
4954      \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#
4955      \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{paragr
4956    \fi
4957    \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
4958    \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
4959 }% for customization
4960 {}
```

and finally, we localize the sections

```
4961 \newcommand\omdoc@part@kw{Part}
4962 \newcommand\omdoc@chapter@kw{Chapter}
4963 \newcommand\omdoc@section@kw{Section}
4964 \newcommand\omdoc@subsection@kw{Subsection}
4965 \newcommand\omdoc@subsubsection@kw{Subsubsection}
4966 \newcommand\omdoc@paragraph@kw{paragraph}
4967 \newcommand\omdoc@subparagraph@kw{subparagraph}
```

38.7   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

```
4968 \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).

```
4969 \cs_if_exist:NTF\frontmatter{
4970    \let\__document_structure_orig_frontmatter\frontmatter
4971    \let\frontmatter\relax
4972 }{
4973    \tl_set:Nn\__document_structure_orig_frontmatter{
4974      \clearpage
4975      \@mainmatterfalse
4976      \pagenumbering{roman}
4977    }
4978 }
```

```
4979 \cs_if_exist:NTF\backmatter{
4980   \let\__document_structure_orig_backmatter\backmatter
4981   \let\backmatter\relax
4982 }{
4983   \tl_set:Nn\__document_structure_orig_backmatter{
4984     \clearpage
4985     \@mainmatterfalse
4986     \pagenumbering{roman}
4987   }
4988 }
```

Using these, we can now define the `frontmatter` and `backmatter` environments

we use the \orig@frontmatter macro defined above and \mainmatter if it exists, otherwise we define it.

```
4989 \newenvironment{frontmatter}{
4990   \__document_structure_orig_frontmatter
4991 }{
4992   \cs_if_exist:NTF\mainmatter{
4993     \mainmatter
4994   }{
4995     \clearpage
4996     \@mainmattertrue
4997     \pagenumbering{arabic}
4998   }
4999 }
```

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

```
5000 \newenvironment{backmatter}{
5001   \__document_structure_orig_backmatter
5002 }{
5003   \cs_if_exist:NTF\mainmatter{
5004     \mainmatter
5005   }{
5006     \clearpage
5007     \@mainmattertrue
5008     \pagenumbering{arabic}
5009   }
5010 }
```

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

```
5011 \@mainmattertrue\pagenumbering{arabic}
```

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

```
5012 \def \c__document_structure_document_str{document}
5013 \newcommand\afterprematurestop{}
5014 \def\prematurestop@endomgroup{
5015   \unless\ifx\@currenvir\c__document_structure_document_str
5016     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafte
5017     \expandafter\prematurestop@endomgroup
5018   \fi
5019 }
```

```
5020  \providecommand\prematurestop{
5021    \message{Stopping~sTeX~processing~prematurely}
5022    \prematurestop@endomgroup
5023    \afterprematurestop
5024    \end{document}
5025  }
```

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

38.8   Global Variables

\setSGvar   set a global variable

```
5026  \RequirePackage{etoolbox}
5027  \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}
```

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

\useSGvar   use a global variable

```
5028  \newrobustcmd\useSGvar[1]{%
5029    \@ifundefined{sTeX@Gvar@#1}
5030    {\PackageError{document-structure}
5031      {The sTeX Global variable #1 is undefined}
5032      {set it with \protect\setSGvar}}
5033  \@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.

```
5034  \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
5035    \@ifundefined{sTeX@Gvar@#1}
5036    {\PackageError{document-structure}
5037      {The sTeX Global variable #1 is undefined}
5038      {set it with \protect\setSGvar}}
5039    {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}
```

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

```

# Chapter 39

# NotesSlides – Implementation

## 39.1  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.

```
5040 ⟨∗cls⟩
5041 ⟨@@=notesslides⟩
5042 \ProvidesExplClass{notesslides}{2022/02/10}{3.0}{notesslides Class}
5043 \RequirePackage{l3keys2e,expl-keystr-compat}
5044
5045 \keys_define:nn{notesslides / cls}{
5046   class   .code:n   = {
5047   \PassOptionsToClass{\CurrentOption}{omdoc}
5048   \str_if_eq:nnT{#1}{book}{
5049   \PassOptionsToPackage{defaulttopsec=part}{notesslides}
5050   }
5051   \str_if_eq:nnT{#1}{report}{
5052   \PassOptionsToPackage{defaulttopsec=part}{notesslides}
5053   }
5054   },
5055   notes   .bool_set:N = \c__notesslides_notes_bool ,
5056   slides .code:n   = { \bool_set_false:N \c__notesslides_notes_bool },
5057   unknown .code:n   = {
5058   \PassOptionsToClass{\CurrentOption}{omdoc}
5059   \PassOptionsToClass{\CurrentOption}{beamer}
5060   \PassOptionsToPackage{\CurrentOption}{notesslides}
5061   }
5062 }
5063 \ProcessKeysOptions{ notesslides / cls }
5064 \bool_if:NTF \c__notesslides_notes_bool {
5065   \PassOptionsToPackage{notes=true}{notesslides}
5066 }{
5067   \PassOptionsToPackage{notes=false}{notesslides}
5068 }
5069 ⟨/cls⟩
```

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

```
5070 ⟨*package⟩
5071 \ProvidesExplPackage{notesslides}{2022/02/10}{3.0}{notesslides Package}
5072 \RequirePackage{l3keys2e,expl-keystr-compat}
5073
5074 \keys_define:nn{notesslides / pkg}{
5075   topsect         .str_set_x:N  = \c__notesslides_topsect_str,
5076   defaulttopsect  .str_set_x:N  = \c__notesslides_defaulttopsec_str,
5077   notes           .bool_set:N   = \c__notesslides_notes_bool ,
5078   slides          .code:n       = { \bool_set_false:N \c__notesslides_notes_bool },
5079   sectocframes    .bool_set:N   = \c__notesslides_sectocframes_bool ,
5080   frameimages     .bool_set:N   = \c__notesslides_frameimages_bool ,
5081   fiboxed         .bool_set:N   = \c__notesslides_fiboxed_bool ,
5082   noproblems      .bool_set:N   = \c__notesslides_noproblems_bool,
5083   unknown         .code:n       = {
5084     \PassOptionsToClass{\CurrentOption}{stex}
5085     \PassOptionsToClass{\CurrentOption}{tikzinput}
5086   }
5087 }
5088 \ProcessKeysOptions{ notesslides / pkg }
5089 \newif\ifnotes
5090 \bool_if:NTF \c__notesslides_notes_bool {
5091   \notestrue
5092 }{
5093   \notesfalse
5094 }
5095
```

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

```
5096 \str_if_empty:NTF \c__notesslides_topsect_str {
5097   \str_set_eq:NN \__notesslidestopsect \c__notesslides_defaulttopsec_str
5098 }{
5099   \str_set_eq:NN \__notesslidestopsect \c__notesslides_topsect_str
5100 }
5101 ⟨/package⟩
```

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

```
5102 ⟨*cls⟩
5103 \bool_if:NTF \c__notesslides_notes_bool {
5104   \LoadClass{document-structure}
5105 }{
5106   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
5107   \newcounter{Item}
5108   \newcounter{paragraph}
5109   \newcounter{subparagraph}
5110   \newcounter{Hfootnote}
5111   \RequirePackage{document-structure}
5112 }
```

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

```
5113 \RequirePackage{notesslides}
5114 ⟨/cls⟩
```

195

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.

```
5115 ⟨∗package⟩
5116 \bool_if:NT \c__notesslides_notes_bool {
5117    \RequirePackage{a4wide}
5118    \RequirePackage{marginnote}
5119    \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
5120    \RequirePackage{mdframed}
5121    \RequirePackage[noxcolor,noamsthm]{beamerarticle}
5122    \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
5123 }
5124 \RequirePackage{stex-tikzinput}
5125 \RequirePackage{etoolbox}
5126 \RequirePackage{amssymb}
5127 \RequirePackage{amsmath}
5128 \RequirePackage{comment}
5129 \RequirePackage{textcomp}
5130 \RequirePackage{url}
5131 \RequirePackage{graphicx}
5132 \RequirePackage{pgf}
```

39.2   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

EdN:20 notes version loads `beamernotestheme`⟨*theme*⟩`.sty`.[20]

```
5133 \bool_if:NT \c__notesslides_notes_bool {
5134    \renewcommand\usetheme[2][]{\usepackage[#1]{beamernotestheme#2}}
5135 }
```

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

```
5136 \newcounter{slide}
5137 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
5138 \newlength{\slideheight}\setlength{\slideheight}{9cm}
```

note   The `note` environment is used to leave out text in the `slides` mode. It does not have a counterpart in OMDoc. So for course notes, we define the `note` environment to be a no-operation otherwise we declare the `note` environment as a comment via the `comment` package.

```
5139 \bool_if:NTF \c__notesslides_notes_bool {
5140    \renewenvironment{note}{\ignorespaces}{}
5141 }{
5142    \excludecomment{note}
5143 }
```

---

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

We 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.

```
5144 \bool_if:NT \c__notesslides_notes_bool {
5145   \newlength{\slideframewidth}
5146   \setlength{\slideframewidth}{1.5pt}
```

frame   We first define the keys.

```
5147   \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
5148     \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
5149       \bool_set_true:N #1
5150     }{
5151       \bool_set_false:N #1
5152     }
5153   }
5154   \keys_define:nn{notesslides / frame}{
5155     label              .str_set_x:N  = \l__notesslides_frame_label_str,
5156     allowframebreaks   .code:n       = {
5157       \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
5158     },
5159     allowdisplaybreaks .code:n       = {
5160       \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
5161     },
5162     fragile            .code:n       = {
5163       \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
5164     },
5165     shrink             .code:n       = {
5166       \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
5167     },
5168     squeeze            .code:n       = {
5169       \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
5170     },
5171     t                  .code:n       = {
5172       \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
5173     },
5174   }
5175   \cs_new_protected:Nn \__notesslides_frame_args:n {
5176     \str_clear:N \l__notesslides_frame_label_str
5177     \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
5178     \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
5179     \bool_set_true:N \l__notesslides_frame_fragile_bool
5180     \bool_set_true:N \l__notesslides_frame_shrink_bool
5181     \bool_set_true:N \l__notesslides_frame_squeeze_bool
5182     \bool_set_true:N \l__notesslides_frame_t_bool
5183     \keys_set:nn { notesslides / frame }{ #1 }
5184   }
```

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

```
5185   \renewenvironment{frame}[1][]{
5186     \__notesslides_frame_args:n{#1}
5187     \sffamily
5188     \stepcounter{slide}
5189     \def\@currentlabel{\theslide}
5190     \str_if_empty:NF \l__notesslides_frame_label_str {
5191       \label{\l__notesslides_frame_label_str}
```

```
5192        }
```

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

```
5193        \def\itemize@level{outer}
5194        \def\itemize@outer{outer}
5195        \def\itemize@inner{inner}
5196        \renewcommand\newpage{\addtocounter{framenumber}{1}}
5197        \newcommand\metakeys@show@keys[2]{\marginnote{{\scriptsize ##2}}}
5198        \renewenvironment{itemize}{
5199          \ifx\itemize@level\itemize@outer
5200            \def\itemize@label{$\rhd$}
5201          \fi
5202          \ifx\itemize@level\itemize@inner
5203            \def\itemize@label{$\scriptstyle\rhd$}
5204          \fi
5205          \begin{list}
5206          {\itemize@label}
5207          {\setlength{\labelsep}{.3em}
5208           \setlength{\labelwidth}{.5em}
5209           \setlength{\leftmargin}{1.5em}
5210          }
5211          \edef\itemize@level{\itemize@inner}
5212        }{
5213          \end{list}
5214        }
```

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

```
5215        \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth
5216      }{
5217        \medskip\miko@slidelabel\end{mdframed}
5218      }
```

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

\frametitle

```
5219        \renewcommand{\frametitle}[1]{{\Large\bf\sf\color{blue}{#1}}\medskip}
5220      }
```

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

\pause   [21]

```
5221      \bool_if:NT \c__notesslides_notes_bool {
5222        \newcommand\pause{}
5223      }
```

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

nparagraph

```
5224      \bool_if:NTF \c__notesslides_notes_bool {
5225        \newenvironment{nparagraph}[1][]{\begin{sparagraph}[#1]}{\end{sparagraph}}
5226      }{
5227        \excludecomment{nparagraph}
5228      }
```

---

[21] EDNOTE: MK: fake it in notes mode for now

nomgroup

```
5229 \bool_if:NTF \c__notesslides_notes_bool {
5230   \newenvironment{nomgroup}[2][]{\begin{omgroup}[#1]{#2}}{\end{omgroup}}
5231 }{
5232   \excludecomment{nomgroup}
5233 }
```

ndefinition

```
5234 \bool_if:NTF \c__notesslides_notes_bool {
5235   \newenvironment{ndefinition}[1][]{\begin{sdefinition}[#1]}{\end{sdefinition}}
5236 }{
5237   \excludecomment{ndefinition}
5238 }
```

nassertion

```
5239 \bool_if:NTF \c__notesslides_notes_bool {
5240   \newenvironment{nassertion}[1][]{\begin{sassertion}[#1]}{\end{sassertion}}
5241 }{
5242   \excludecomment{nassertion}
5243 }
```

nsproof

```
5244 \bool_if:NTF \c__notesslides_notes_bool {
5245   \newenvironment{nproof}[2][]{\begin{sproof}[#1]{#2}}{\end{sproof}}
5246 }{
5247   \excludecomment{nproof}
5248 }
```

nexample

```
5249 \bool_if:NTF \c__notesslides_notes_bool {
5250   \newenvironment{nexample}[1][]{\begin{example}[#1]}{\end{example}}
5251 }{
5252   \excludecomment{nexample}
5253 }
```

nparagraph

```
5254 \bool_if:NTF \c__notesslides_notes_bool {
5255   \newenvironment{nparagraph}[1][]{\begin{sparagraph}[#1]}{\end{sparagraph}}
5256 }{
5257   \excludecomment{nparagraph}
5258 }
```

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

```
5259 \def\inputref@preskip{\smallskip}
5260 \def\inputref@postskip{\medskip}
```

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

\inputref*

```
5261 \let\orig@inputref\inputref
5262 \def\inputref{\@ifstar\ninputref\orig@inputref}
5263 \newcommand\ninputref[2][]{
5264   \bool_if:NT \c__notesslides_notes_bool {
```

```
5265        \orig@inputref[#1]{#2}
5266    }
5267 }
```

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

39.3   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*⟩}.

```
5268 \newlength{\slidelogoheight}
5269
5270 \bool_if:NTF \c__notesslides_notes_bool {
5271    \setlength{\slidelogoheight}{.4cm}
5272 }{
5273    \setlength{\slidelogoheight}{1cm}
5274 }
5275 \newsavebox{\slidelogo}
5276 \sbox{\slidelogo}{\sTeX}
5277 \newrobustcmd\setslidelogo[1]{
5278    \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
5279 }
```

(*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.

```
5280 \def\source{Michael Kohlhase}% customize locally
5281 \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.

```
5282 \def\copyrightnotice{\footnotesize\copyright :\hspace{.3ex}{\source}}
5283 \newsavebox{\cclogo}
5284 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}
5285 \newif\ifcchref\cchreffalse
5286 \AtBeginDocument{
5287    \@ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
5288 }
5289 \def\licensing{
5290    \ifcchref
5291      \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
5292    \else
5293      {\usebox{\cclogo}}
5294    \fi
5295 }
```

```
5296 \newrobustcmd{\setlicensing}[2][]{
5297   \def\@url{#1}
5298   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
5299   \ifx\@url\@empty
5300     \def\licensing{{\usebox{\cclogo}}}
5301   \else
5302     \def\licensing{
5303       \ifcchref
5304       \href{#1}{\usebox{\cclogo}}
5305       \else
5306       {\usebox{\cclogo}}
5307       \fi
5308     }
5309   \fi
5310 }
```

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

\slidelabel   Now, we set up the slide label for the `article` mode.[22]

```
5311 \newrobustcmd\miko@slidelabel{
5312   \vbox to \slidelogoheight{
5313     \vss\hbox to \slidewidth
5314     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
5315   }
5316 }
```

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

39.4   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.

```
5317 \def\Gin@mhrepos{}
5318 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
5319 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
5320 \newrobustcmd\frameimage[2][]{
5321   \stepcounter{slide}
5322   \bool_if:NT \c__notesslides_frameimages_bool {
5323     \def\Gin@ewidth{}\setkeys{Gin}{#1}
5324     \bool_if:NF \c__notesslides_notes_bool { \vfill }
5325     \begin{center}
5326       \bool_if:NTF \c__notesslides_fiboxed_bool {
5327         \fbox{
5328           \ifx\Gin@ewidth\@empty
5329             \ifx\Gin@mhrepos\@empty
5330               \mhgraphics[width=\slidewidth,#1]{#2}
5331             \else
5332               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5333             \fi
5334           \else% Gin@ewidth empty
5335             \ifx\Gin@mhrepos\@empty
5336               \mhgraphics[#1]{#2}
```

---

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

```
5337              \else
5338                \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5339              \fi
5340          \fi% Gin@ewidth empty
5341        }
5342      }{
5343        \ifx\Gin@ewidth\@empty
5344          \ifx\Gin@mhrepos\@empty
5345            \mhgraphics[width=\slidewidth,#1]{#2}
5346          \else
5347            \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5348          \fi
5349          \ifx\Gin@mhrepos\@empty
5350            \mhgraphics[#1]{#2}
5351          \else
5352            \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5353          \fi
5354        \fi% Gin@ewidth empty
5355      }
5356      \end{center}
5357      \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
5358      \bool_if:NF \c__notesslides_notes_bool { \vfill }
5359    }
5360  } % ifmks@sty@frameimages
```

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

39.5   Colors and Highlighting

We first specify sans serif fonts as the default.

```
5361  \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

```
5362  \AddToHook{begindocument}{
5363    \definecolor{green}{rgb}{0,.5,0}
5364    \definecolor{purple}{cmyk}{.3,1,0,.17}
5365  }
```

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`.

```
5366  % \def\STpresent#1{\textcolor{blue}{#1}}
5367  \def\defemph#1{{\textcolor{magenta}{#1}}}
5368  \def\symrefemph#1{{\textcolor{cyan}{#1}}}
5369  \def\compemph#1{{\textcolor{blue}{#1}}}
5370  \def\titleemph#1{{\textcolor{blue}{#1}}}
5371  \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.

```
5372  \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
5373  \def\smalltextwarning{
5374    \pgfuseimage{miko@small@dbend}
5375    \xspace
5376  }
5377  \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}
5378  \newrobustcmd\textwarning{
5379    \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
5380    \xspace
5381  }
5382  \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}
5383  \newrobustcmd\bigtextwarning{
5384    \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
5385    \xspace
5386  }
```

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

```
5387  \newrobustcmd\putgraphicsat[3]{
5388    \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
5389  }
5390  \newrobustcmd\putat[2]{
5391    \begin{picture}(0,0)\put(#1){#2}\end{picture}
5392  }
```

39.6 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`.

```
5393  \bool_if:NT \c__notesslides_sectocframes_bool {
5394    \str_if_eq:VnTF \__notesslidestopsect{part}{
5395      \newcounter{chapter}\counterwithin*{section}{chapter}
5396    }{
5397      \str_if_eq:VnT\__notesslidestopsect{chapter}{
5398        \newcounter{chapter}\counterwithin*{section}{chapter}
5399      }
5400    }
5401  }
```

\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

```
5402  \def\part@prefix{}
5403  \@ifpackageloaded{document-structure}{}{
5404    \str_case:VnF \__notesslidestopsect {
5405      {part}{
5406        \int_set:Nn \l_document_structure_section_level_int {0}
5407        \def\thesection{\arabic{chapter}.\arabic{section}}
5408        \def\part@prefix{\arabic{chapter}.}
5409      }
```

```
5410      {chapter}{
5411        \int_set:Nn \l_document_structure_section_level_int {1}
5412        \def\thesection{\arabic{chapter}.\arabic{section}}
5413        \def\part@prefix{\arabic{chapter}.}
5414      }
5415    }{
5416      \int_set:Nn \l_document_structure_section_level_int {2}
5417      \def\part@prefix{}
5418    }
5419  }
5420
5421  \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.

```
5422    \renewenvironment{omgroup}[2][]{
5423      \__document_structure_omgroup_args:n { #1 }
5424      \int_incr:N \l_document_structure_omgroup_level_int
5425      \int_incr:N \l_document_structure_section_level_int
5426      \bool_if:NT \c__notesslides_sectocframes_bool {
5427        \stepcounter{slide}
5428        \begin{frame}[noframenumbering]
5429        \vfill\Large\centering
5430        \red{
5431          \ifcase\l_document_structure_section_level_int\or
5432            \stepcounter{part}
5433            \def\__notesslideslabel{\omdoc@part@kw~\Roman{part}}
5434            \def\currentsectionlevel{\omdoc@part@kw}
5435          \or
5436            \stepcounter{chapter}
5437            \def\__notesslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
5438            \def\currentsectionlevel{\omdoc@chapter@kw}
5439          \or
5440            \stepcounter{section}
5441            \def\__notesslideslabel{\part@prefix\arabic{section}}
5442            \def\currentsectionlevel{\omdoc@section@kw}
5443          \or
5444            \stepcounter{subsection}
5445            \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
5446            \def\currentsectionlevel{\omdoc@subsection@kw}
5447          \or
5448            \stepcounter{subsubsection}
5449            \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
5450            \def\currentsectionlevel{\omdoc@subsubsection@kw}
5451          \or
5452            \stepcounter{paragraph}
5453            \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
5454            \def\currentsectionlevel{\omdoc@paragraph@kw}
5455          \else
5456            \def\__notesslideslabel{}
5457            \def\currentsectionlevel{\omdoc@paragraph@kw}
```

204

```
5458        \fi% end ifcase
5459        \__notesslideslabel%\sref@label@id\__notesslideslabel
5460        \quad #2%
5461      }%
5462      \vfill%
5463      \end{frame}%
5464    }
5465    \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str%
5466  }{}
5467 }
```

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

```
5468 \def\inserttheorembodyfont{\normalfont}
5469 %\bool_if:NF \c__notesslides_notes_bool {
5470 %  \defbeamertemplate{theorem begin}{miko}
5471 %  {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
5472 %    \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
5473 %    \inserttheorempunctuation\inserttheorembodyfont\xspace}
5474 %  \defbeamertemplate{theorem end}{miko}{}
```

and we set it as the default one.

```
5475 %  \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.

```
5476 %  \expandafter\def\csname Parent2\endcsname{}
5477 %}
5478
5479 \AddToHook{begindocument}{ % this does not work for some reasone
5480   \setbeamertemplate{theorems}[ams style]
5481 }
5482 \bool_if:NT \c__notesslides_notes_bool {
5483   \renewenvironment{columns}[1][]{%
5484     \par\noindent%
5485     \begin{minipage}%
5486     \slidewidth\centering\leavevmode%
5487   }{%
5488     \end{minipage}\par\noindent%
5489   }%
5490   \newsavebox\columnbox%
5491   \renewenvironment<>{column}[2][]{%
5492     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
5493   }{%
5494     \end{minipage}\end{lrbox}\usebox\columnbox%
5495   }%
5496 }
5497 \bool_if:NTF \c__notesslides_noproblems_bool {
5498   \newenvironment{problems}{}{}
5499 }{
5500   \excludecomment{problems}
5501 }
```

39.7 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.

```
5502 \gdef\printexcursions{}
5503 \newcommand\excursionref[2]{% label, text
5504   \bool_if:NT \c__notesslides_notes_bool {
5505     \begin{sparagraph}[title=Excursion]
5506       #2 \sref[fallback=the appendix]{#1}.
5507     \end{sparagraph}
5508   }
5509 }
5510 \newcommand\activate@excursion[2][]{
5511   \gappto\printexcursions{\inputref[#1]{#2}}
5512 }
5513 \newcommand\excursion[4][]{% repos, label, path, text
5514   \bool_if:NT \c__notesslides_notes_bool {
5515     \activate@excursion[#1]{#3}\excursionref{#2}{#4}
5516   }
5517 }
```

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

\excursiongroup

```
5518 \keys_define:nn{notesslides / excursiongroup }{
5519   id        .str_set_x:N = \l__notesslides_excursion_id_str,
5520   intro     .tl_set:N    = \l__notesslides_excursion_intro_tl,
5521   mhrepos   .str_set_x:N = \l__notesslides_excursion_mhrepos_str
5522 }
5523 \cs_new_protected:Nn \__notesslides_excursion_args:n {
5524   \tl_clear:N \l__notesslides_excursion_intro_tl
5525   \str_clear:N \l__notesslides_excursion_id_str
5526   \str_clear:N \l__notesslides_excursion_mhrepos_str
5527   \keys_set:nn {notesslides / excursiongroup }{ #1 }
5528 }
5529 \newcommand\excursiongroup[1][]{
5530   \__notesslides_excursion_args:n{ #1 }
5531   \ifdefempty\printexcursions{}% only if there are excursions
5532   {\begin{note}
5533     \begin{omgroup}[#1]{Excursions}%
5534       \ifdefempty\l__notesslides_excursion_intro_tl{}{
5535         \inputref[\l__notesslides_excursion_mhrepos_str]{
5536           \l__notesslides_excursion_intro_tl
5537         }
5538       }
5539       \printexcursions%
5540     \end{omgroup}
5541   \end{note}}
5542 }
5543 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
5544 ⟨/package⟩
```

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

Chapter 40

The Implementation

40.1 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.

```
5545 ⟨∗package⟩
5546 ⟨@@=problems⟩
5547 \ProvidesExplPackage{problem}{2019/03/20}{1.3}{Semantic Markup for Problems}
5548 \RequirePackage{l3keys2e,expl-keystr-compat}
5549
5550 \keys_define:nn { problem / pkg }{
5551   notes     .default:n   = { true },
5552   notes     .bool_set:N  = \c__problems_notes_bool,
5553   gnotes    .default:n   = { true },
5554   gnotes    .bool_set:N  = \c__problems_gnotes_bool,
5555   hints     .default:n   = { true },
5556   hints     .bool_set:N  = \c__problems_hints_bool,
5557   solutions .default:n   = { true },
5558   solutions .bool_set:N  = \c__problems_solutions_bool,
5559   pts       .default:n   = { true },
5560   pts       .bool_set:N  = \c__problems_pts_bool,
5561   min       .default:n   = { true },
5562   min       .bool_set:N  = \c__problems_min_bool,
5563   boxed     .default:n   = { true },
5564   boxed     .bool_set:N  = \c__problems_boxed_bool,
5565   unknown   .code:n      = {}
5566 }
5567 \def\solutionstrue{
5568   \bool_set_true:N \c__problems_solutions_bool
5569 }
5570 \def\solutionsfalse{
5571   \bool_set_false:N \c__problems_solutions_bool
5572 }
5573
5574 \ProcessKeysOptions{ problem / pkg }
```

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

```
5575  \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.

```
5576  \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.

```
5577  \def\prob@problem@kw{Problem}
5578  \def\prob@solution@kw{Solution}
5579  \def\prob@hint@kw{Hint}
5580  \def\prob@note@kw{Note}
5581  \def\prob@gnote@kw{Grading}
5582  \def\prob@pt@kw{pt}
5583  \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
5584  \AddToHook{begindocument}{
5585    \ltx@ifpackageloaded{babel}{
5586        \makeatletter
5587        \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5588        \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5589          \input{problem-ngerman.ldf}
5590        }
5591        \clist_if_in:NnT \l_tmpa_clist {finnish}{
5592          \input{problem-finnish.ldf}
5593        }
5594        \clist_if_in:NnT \l_tmpa_clist {french}{
5595          \input{problem-french.ldf}
5596        }
5597        \clist_if_in:NnT \l_tmpa_clist {russian}{
5598          \input{problem-russian.ldf}
5599        }
5600        \makeatother
5601    }{}
5602  }
```

40.2   Problems and Solutions

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

```
5603  \keys_define:nn{ problem / problem }{
5604    id      .str_set_x:N  = \l__problems_prob_id_str,
5605    pts     .tl_set:N     = \l__problems_prob_pts_tl,
5606    min     .tl_set:N     = \l__problems_prob_min_tl,
5607    title   .tl_set:N     = \l__problems_prob_title_tl,
5608    refnum  .int_set:N    = \l__problems_prob_refnum_int
5609  }
5610  \cs_new_protected:Nn \__problems_prob_args:n {
5611    \str_clear:N \l__problems_prob_id_str
```

```
5612    \tl_clear:N \l__problems_prob_pts_tl
5613    \tl_clear:N \l__problems_prob_min_tl
5614    \tl_clear:N \l__problems_prob_title_tl
5615    \int_zero_new:N \l__problems_prob_refnum_int
5616    \keys_set:nn { problem / problem }{ #1 }
5617    \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
5618        \let\l__problems_inclprob_refnum_int\undefined
5619    }
5620 }
```

Then we set up a counter for problems.

\numberproblemsin

```
5621 \newcounter{problem}
5622 \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.

```
5623 \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

```
5624 \newcommand\prob@number{
5625    \int_if_exist:NTF \l__problems_inclprob_refnum_int {
5626        \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
5627    }{
5628        \int_if_exist:NTF \l__problems_prob_refnum_int {
5629            \prob@label{\int_use:N \l__problems_prob_refnum_int }
5630        }{
5631            \prob@label\theproblem
5632        }
5633    }
5634 }
```

(*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.

```
5635 \newcommand\prob@title[3]{%
5636    \tl_if_exist:NTF \l__problems_inclprob_title_tl {
5637        #2 \l__problems_inclprob_title_tl #3
5638    }{
5639        \tl_if_exist:NTF \l__problems_prob_title_tl {
5640            #2 \l__problems_prob_title_tl #3
5641        }{
5642            #1
5643        }
5644    }
5645 }
```

(*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.

```
5646 \def\prob@heading{
5647   \prob@problem@kw~\prob@number\prob@title{~}{~(}{)\strut}
5648   %\sref@label@id{\prob@problem@kw~\prob@number}{}
5649 }
```

(*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.

problem

```
5650 \newenvironment{problem}[1][]{
5651   \__problems_prob_args:n{#1}%\sref@target%
5652   \@in@omtexttrue% we are in a statement (for inline definitions)
5653   \stepcounter{problem}\record@problem
5654   \def\current@section@level{\prob@problem@kw}
5655   \par\noindent\textbf\prob@heading\show@pts\show@min\\\ignorespacesandpars
5656 }%
5657 {\smallskip}
5658 \bool_if:NT \c__problems_boxed_bool {
5659   \surroundwithmdframed{problem}
5660 }
```

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

```
5661 \def\record@problem{
5662   \protected@write\@auxout{}
5663   {
5664     \string\@problem{\prob@number}
5665     {
5666       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
5667         \l__problems_inclprob_pts_tl
5668       }{
5669         \l__problems_prob_pts_tl
5670       }
5671     }%
5672     {
5673       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
5674         \l__problems_inclprob_min_tl
5675       }{
5676         \l__problems_prob_min_tl
5677       }
5678     }
5679   }
5680 }
```

(*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).

```
5681 \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.

```
5682 \keys_define:nn { problem / solution }{
5683   id             .str_set_x:N  = \l__problems_solution_id_str ,
5684   for            .tl_set:N     = \l__problems_solution_for_tl ,
5685   height         .dim_set:N    = \l__problems_solution_height_dim ,
5686   creators       .clist_set:N  = \l__problems_solution_creators_clist ,
5687   contributors   .clist_set:N  = \l__problems_solution_contributors_clist ,
5688   srccite        .tl_set:N     = \l__problems_solution_srccite_tl
5689 }
5690 \cs_new_protected:Nn \__problems_solution_args:n {
5691   \str_clear:N \l__problems_solution_id_str
5692   \tl_clear:N \l__problems_solution_for_tl
5693   \tl_clear:N \l__problems_solution_srccite_tl
5694   \clist_clear:N \l__problems_solution_creators_clist
5695   \clist_clear:N \l__problems_solution_contributors_clist
5696   \dim_zero:N \l__problems_solution_height_dim
5697   \keys_set:nn { problem / solution }{ #1 }
5698 }
```

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

```
5699 \newcommand\@startsolution[1][]{
5700   \__problems_solution_args:n { #1 }
5701   \@in@omtexttrue% we are in a statement.
5702   \bool_if:NF \c__problems_boxed_bool { \hrule }
5703   \smallskip\noindent
5704   {\textbf\prob@solution@kw :\enspace}
5705   \begin{small}
5706   \def\current@section@level{\prob@solution@kw}
5707   \ignorespacesandpars
5708 }
```

\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.

```
5709 \newcommand\startsolutions{
5710   \specialcomment{solution}{\@startsolution}{
5711     \bool_if:NF \c__problems_boxed_bool {
5712       \hrule\medskip
5713     }
5714     \end{small}%
5715   }
5716   \bool_if:NT \c__problems_boxed_bool {
5717     \surroundwithmdframed{solution}
5718   }
5719 }
```

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

\stopsolutions

```
5720 \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.

```
5721 \bool_if:NTF \c__problems_solutions_bool {
5722   \startsolutions
5723 }{
5724   \stopsolutions
5725 }
```

exnote

```
5726 \bool_if:NTF \c__problems_notes_bool {
5727   \newenvironment{exnote}[1][]{
5728     \par\smallskip\hrule\smallskip
5729     \noindent\textbf{\prob@note@kw : }\small
5730   }{
5731     \smallskip\hrule
5732   }
5733 }{
5734   \excludecomment{exnote}
5735 }
```

hint

```
5736 \bool_if:NTF \c__problems_notes_bool {
5737   \newenvironment{hint}[1][]{
5738     \par\smallskip\hrule\smallskip
5739     \noindent\textbf{\prob@hint@kw :~ }\small
5740   }{
5741     \smallskip\hrule
5742   }
5743   \newenvironment{exhint}[1][]{
5744     \par\smallskip\hrule\smallskip
5745     \noindent\textbf{\prob@hint@kw :~ }\small
5746   }{
5747     \smallskip\hrule
5748   }
5749 }{
5750   \excludecomment{hint}
5751   \excludecomment{exhint}
5752 }
```

gnote

```
5753 \bool_if:NTF \c__problems_notes_bool {
5754   \newenvironment{gnote}[1][]{
5755     \par\smallskip\hrule\smallskip
5756     \noindent\textbf{\prob@gnote@kw : }\small
5757   }{
5758     \smallskip\hrule
5759   }
5760 }{
5761   \excludecomment{gnote}
5762 }
```

40.3 Multiple Choice Blocks

mcb ²³

```
5763  \newenvironment{mcb}{
5764    \begin{enumerate}
5765  }{
5766    \end{enumerate}
5767  }
```

we define the keys for the mcc macro

```
5768  \cs_new_protected:Nn \__problems_do_yes_param:Nn {
5769    \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
5770      \bool_set_true:N #1
5771    }{
5772      \bool_set_false:N #1
5773    }
5774  }
5775  \keys_define:nn { problem / mcc }{
5776    id          .str_set_x:N  = \l__problems_mcc_id_str ,
5777    feedback   .tl_set:N     = \l__problems_mcc_feedback_tl ,
5778    T           .default:n    = { true } ,
5779    T           .bool_set:N   = \l__problems_mcc_t_bool ,
5780    F           .default:n    = { true } ,
5781    F           .bool_set:N   = \l__problems_mcc_f_bool ,
5782    Ttext       .code:n       = {
5783      \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
5784    } ,
5785    Ftext       .code:n       = {
5786      \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
5787    }
5788  }
5789  \cs_new_protected:Nn \l__problems_mcc_args:n {
5790    \str_clear:N \l__problems_mcc_id_str
5791    \tl_clear:N \l__problems_mcc_feedback_tl
5792    \bool_set_true:N \l__problems_mcc_t_bool
5793    \bool_set_true:N \l__problems_mcc_f_bool
5794    \bool_set_true:N \l__problems_mcc_Ttext_bool
5795    \bool_set_false:N \l__problems_mcc_Ftext_bool
5796    \keys_set:nn { problem / mcc }{ #1 }
5797  }
```

\mcc

```
5798  \newcommand\mcc[2][]{
5799    \l__problems_mcc_args:n{ #1 }
5800    \item #2
5801    \bool_if:NT \c__problems_solutions_bool {
5802      \\
5803      \bool_if:NT \l__problems_mcc_t_bool {
5804        % TODO!
5805        % \ifcsstring{mcc@T}{T}{}{\mcc@Ttext}%
5806      }
5807      \bool_if:NT \l__problems_mcc_f_bool {
```

---

²³EDNOTE: MK: maybe import something better here from a dedicated MC package

213

```
5808        % TODO!
5809        % \ifcsstring{mcc@F}{F}{}{\mcc@Ftext}%
5810      }
5811      \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
5812        !
5813      }{
5814        \l__problems_mcc_feedback_tl
5815      }
5816    }
5817  } %solutions
```

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

40.4   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.

```
5818
5819  \keys_define:nn{ problem / inclproblem }{
5820  % id      .str_set_x:N  = \l__problems_inclprob_id_str,
5821    pts     .tl_set:N     = \l__problems_inclprob_pts_tl,
5822    min     .tl_set:N     = \l__problems_inclprob_min_tl,
5823    title   .tl_set:N     = \l__problems_inclprob_title_tl,
5824    refnum  .int_set:N    = \l__problems_inclprob_refnum_int,
5825    mhrepos .str_set_x:N   = \l__problems_inclprob_mhrepos_str
5826  }
5827  \cs_new_protected:Nn \__problems_inclprob_args:n {
5828  % \str_clear:N \l__problems_prob_id_str
5829    \tl_clear:N \l__problems_inclprob_pts_tl
5830    \tl_clear:N \l__problems_inclprob_min_tl
5831    \tl_clear:N \l__problems_inclprob_title_tl
5832    \int_zero_new:N \l__problems_inclprob_refnum_int
5833    \str_clear:N \l__problems_inclprob_mhrepos_str
5834    \keys_set:nn { problem / inclproblem }{ #1 }
5835    \tl_if_empty:NT \l__problems_inclprob_pts_tl {
5836      \let\l__problems_inclprob_pts_tl\undefined
5837    }
5838    \tl_if_empty:NT \l__problems_inclprob_min_tl {
5839      \let\l__problems_inclprob_min_tl\undefined
5840    }
5841    \tl_if_empty:NT \l__problems_inclprob_title_tl {
5842      \let\l__problems_inclprob_title_tl\undefined
5843    }
5844    \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
5845      \let\l__problems_inclprob_refnum_int\undefined
5846    }
5847  }
5848
5849  \cs_new_protected:Nn \__problems_inclprob_clear: {
5850  % \str_clear:N \l__problems_prob_id_str
5851    \let\l__problems_inclprob_pts_tl\undefined
5852    \let\l__problems_inclprob_min_tl\undefined
```

```
5853    \let\l__problems_inclprob_title_tl\undefined
5854    \let\l__problems_inclprob_refnum_int\undefined
5855    \let\l__problems_inclprob_mhrepos_str\undefined
5856 }
5857
5858 \newcommand\includeproblem[2][]{
5859    \__problems_inclprob_args:n{ #1 }
5860    \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
5861       \input{#2}
5862    }{
5863       \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
5864          \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
5865       }
5866    }
5867    \__problems_inclprob_clear:
5868 }
```

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

40.5   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.

```
5869 \AddToHook{enddocument}{
5870    \bool_if:NT \c__problems_pts_bool {
5871       \message{Total:~\arabic{pts}~points}
5872    }
5873    \bool_if:NT \c__problems_min_bool {
5874       \message{Total:~\arabic{min}~minutes}
5875    }
5876 }
```

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

```
5877 \def\pts#1{
5878    \bool_if:NT \c__problems_pts_bool {
5879       \marginpar{#1~\prob@pt@kw}
5880    }
5881 }
5882 \def\min#1{
5883    \bool_if:NT \c__problems_min_bool {
5884       \marginpar{#1~\prob@min@kw}
5885    }
5886 }
```

`\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.

```
5887 \newcounter{pts}
5888 \def\show@pts{
5889    \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
5890       \bool_if:NT \c__problems_pts_bool {
5891          \marginpar{\l__problems_inclprob_pts_tl;\prob@pt@kw\smallskip}
5892          \addtocounter{pts}{\l__problems_inclprob_pts_tl}
```

```
5893        }
5894      }{
5895        \tl_if_exist:NT \l__problems_prob_pts_tl {
5896          \bool_if:NT \c__problems_pts_bool {
5897            \marginpar{\l__problems_prob_pts_tl;\prob@pt@kw\smallskip}
5898            \addtocounter{pts}{\l__problems_prob_pts_tl}
5899          }
5900        }
5901      }
5902    }
```

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

and now the same for the minutes

`\show@min`

```
5903  \newcounter{min}
5904  \def\show@min{
5905    \tl_if_exist:NTF \l__problems_inclprob_min_tl {
5906      \bool_if:NT \c__problems_min_bool {
5907        \marginpar{\l__problems_inclprob_pts_tl;min}
5908        \addtocounter{min}{\l__problems_inclprob_min_tl}
5909      }
5910    }{
5911      \tl_if_exist:NT \l__problems_prob_min_tl {
5912        \bool_if:NT \c__problems_min_bool {
5913          \marginpar{\l__problems_prob_min_tl;min}
5914          \addtocounter{min}{\l__problems_prob_min_tl}
5915        }
5916      }
5917    }
5918  }
5919  ⟨/package⟩
```

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

Chapter 41

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.

41.1   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.

```
5920 ⟨@@=hwexam⟩
5921 ⟨*cls⟩
5922 \ProvidesExplClass{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
5923 \RequirePackage{l3keys2e,expl-keystr-compat}
5924 \DeclareOption*{
5925   \PassOptionsToClass{\CurrentOption}{omdoc}
5926   \PassOptionsToPackage{\CurrentOption}{stex}
5927   \PassOptionsToPackage{\CurrentOption}{hwexam}
5928   \PassOptionsToPackage{\CurrentOption}{tikzinput}
5929 }
5930 \ProcessOptions
```

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

```
5931 \LoadClass{omdoc}
5932 \RequirePackage{stex}
5933 \RequirePackage{hwexam}
5934 \RequirePackage{tikzinput}
5935 \RequirePackage{graphicx}
5936 \RequirePackage{a4wide}
5937 \RequirePackage{amssymb}
5938 \RequirePackage{amstext}
5939 \RequirePackage{amsmath}
```

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

```
5940  \newcommand\assig@default@type{\hwexam@assignment@kw}
5941  \def\document@hwexamtype{\assig@default@type}
5942  ⟨@@=document_structure⟩
5943  \keys_define:nn { document-structure / document }{
5944  id .str_set_x:N = \c_document_structure_document_id_str,
5945  hwexamtype .tl_set:N = \document@hwexamtype
5946  }
5947  ⟨@@=hwexam⟩
5948  ⟨/cls⟩
```

Chapter 42

Implementation: The hwexam Package

42.1 Package Options

The first step is to declare (a few) package options that handle whether certain information is printed or not. Some come with their own conditionals that are set by the options, the rest is just passed on to the `problems` package.

```
5949 ⟨∗package⟩
5950 \ProvidesExplPackage{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
5951 \RequirePackage{l3keys2e,expl-keystr-compat}
5952
5953 \newif\iftest\testfalse
5954 \DeclareOption{test}{\testtrue}
5955 \newif\ifmultiple\multiplefalse
5956 \DeclareOption{multiple}{\multipletrue}
5957 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
5958 \ProcessOptions
```

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

```
5959 \RequirePackage{keyval}[1997/11/10]
5960 \RequirePackage{problem}
```

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

```
5961 \newcommand\hwexam@assignment@kw{Assignment}
5962 \newcommand\hwexam@given@kw{Given}
5963 \newcommand\hwexam@due@kw{Due}
5964 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
5965 blank~for~extra~space}%
5966 \newcommand\correction@probs@kw{prob.}%
5967 \newcommand\correction@pts@kw{total}%
5968 \newcommand\correction@reached@kw{reached}%
5969 \newcommand\correction@sum@kw{Sum}%
5970 \newcommand\correction@grade@kw{grade}%
5971 \newcommand\correction@forgrading@kw{To~be~used~for~grading,~do~not~write~here}
```

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

For the other languages, we set up triggers

```
5972  \AddToHook{begindocument}{
5973  \ltx@ifpackageloaded{babel}{
5974  \makeatletter
5975  \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5976  \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5977    \input{hwexam-ngerman.ldf}
5978  }
5979  \clist_if_in:NnT \l_tmpa_clist {finnish}{
5980    \input{hwexam-finnish.ldf}
5981  }
5982  \clist_if_in:NnT \l_tmpa_clist {french}{
5983    \input{hwexam-french.ldf}
5984  }
5985  \clist_if_in:NnT \l_tmpa_clist {russian}{
5986    \input{hwexam-russian.ldf}
5987  }
5988  \makeatother
5989  }{}
5990  }
5991
```

42.2 Assignments

Then we set up a counter for problems and make the problem counter inherited from `problem.sty` depend on it. Furthermore, we specialize the `\prob@label` macro to take the assignment counter into account.

```
5992  \newcounter{assignment}
5993  \numberproblemsin{assignment}
5994  \renewcommand\prob@label[1]{\arabic{assignment}.#1}
```

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

```
5995  \keys_define:nn { hwexam / assignment } {
5996  id    .str_set_x:N = \l__hwexam_assign_id_str,
5997  number   .int_set:N  = \l__hwexam_assign_number_int,
5998  title   .tl_set:N  = \l__hwexam_assign_title_tl,
5999  type   .tl_set:N  = \l__hwexam_assign_type_tl,
6000  given .tl_set:N  = \l__hwexam_assign_given_tl,
6001  due .tl_set:N  = \l__hwexam_assign_due_tl,
6002  loadmodules .code:n  = {
6003  \bool_set_true:N \l__hwexam_assign_loadmodules_bool
6004  }
6005  }
6006  \cs_new_protected:Nn \__hwexam_assignment_args:n {
6007  \str_clear:N \l__hwexam_assign_id_str
6008  \int_set:Nn \l__hwexam_assign_number_int {-1}
6009  \tl_clear:N \l__hwexam_assign_title_tl
6010  \tl_clear:N \l__hwexam_assign_type_tl
6011  \tl_clear:N \l__hwexam_assign_given_tl
6012  \tl_clear:N \l__hwexam_assign_due_tl
6013  \bool_set_false:N \l__hwexam_assign_loadmodules_bool
```

```
6014  \keys_set:nn { hwexam / assignment }{ #1 }
6015  }
```

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.

```
6016  \newcommand\given@due[2]{
6017  \bool_lazy_all:nF {
6018  {\tl_if_empty_p:V \l__hwexam_inclassign_given_tl}
6019  {\tl_if_empty_p:V \l__hwexam_assign_given_tl}
6020  {\tl_if_empty_p:V \l__hwexam_inclassign_due_tl}
6021  {\tl_if_empty_p:V \l__hwexam_assign_due_tl}
6022  }{ #1 }
6023
6024  \tl_if_empty:NTF \l__hwexam_inclassign_given_tl {
6025  \tl_if_empty:NF \l__hwexam_assign_given_tl {
6026  \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
6027  }
6028  }{
6029  \hwexam@given@kw\xspace\l__hwexam_inclassign_given_tl
6030  }
6031
6032  \bool_lazy_or:nnF {
6033  \bool_lazy_and_p:nn {
6034  \tl_if_empty_p:V \l__hwexam_inclassign_due_tl
6035  }{
6036  \tl_if_empty_p:V \l__hwexam_assign_due_tl
6037  }
6038  }{
6039  \bool_lazy_and_p:nn {
6040  \tl_if_empty_p:V \l__hwexam_inclassign_due_tl
6041  }{
6042  \tl_if_empty_p:V \l__hwexam_assign_due_tl
6043  }
6044  }{ ,~ }
6045
6046  \tl_if_empty:NTF \l__hwexam_inclassign_due_tl {
6047  \tl_if_empty:NF \l__hwexam_assign_due_tl {
6048  \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
6049  }
6050  }{
6051  \hwexam@due@kw\xspace \l__hwexam_inclassign_due_tl
6052  }
6053
6054  \bool_lazy_all:nF {
6055  { \tl_if_empty_p:V \l__hwexam_inclassign_given_tl }
6056  { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
6057  { \tl_if_empty_p:V \l__hwexam_inclassign_due_tl }
6058  { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
6059  }{ #2 }
6060  }
```

`\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.

```
6061 \newcommand\assignment@title[3]{
6062 \tl_if_empty:NTF \l__hwexam_inclassign_title_tl {
6063 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
6064 #1
6065 }{
6066 #2\l__hwexam_assign_title_tl#3
6067 }
6068 }{
6069 #2\l__hwexam_inclassign_title_tl#3
6070 }
6071 }
```

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

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

```
6072 \newcommand\assignment@number{
6073 \int_compare:nNnTF \l__hwexam_inclassign_number_int = {-1} {
6074 \int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
6075 \int_use:N \l__hwexam_assign_number_int
6076 }
6077 }{
6078 \int_use:N \l__hwexam_inclassign_number_int
6079 }
6080 }
```

(*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.

```
6081 \newenvironment{assignment}[1][]{
6082 \__hwexam_assignment_args:n { #1 }
6083 %\sref@target
6084 \let\__hwexamnum\l__hwexam_assign_number_int
6085 \int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
6086 \stepcounter{assignment}
6087 }{
6088 \setcounter{assignment}{\int_use:N\__hwexamnum}
6089 }
6090 \setcounter{problem}{0}
6091 \def\current@section@level{\document@hwexamtype}
6092 %\sref@label@id{\document@hwexamtype \thesection}
6093 \begin{@assignment}
6094 }{
6095 \end{@assignment}
6096 }
```

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

```
6097  \def\__hwexamasstitle{
6098  \protect\document@hwexamtype~\arabic{assignment}
6099  \assignment@title{}{\;(}{)\;} -- \given@due{}{}
6100  }
6101  \ifmultiple
6102  \newenvironment{@assignment}{
6103  \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
6104  \begin{omgroup}[loadmodules]{\__hwexamasstitle}
6105  }{
6106  \begin{omgroup}{\__hwexamasstitle}
6107  }
6108  }{
6109  \end{omgroup}
6110  }
```

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

```
6111  \else
6112  \newenvironment{@assignment}{
6113  \begin{center}\bf
6114  \Large\@title\strut\\
6115  \document@hwexamtype~\arabic{assignment}\assignment@title{\;}{:\;}{}\\}
6116  \large\given@due{--\;}{\;--}
6117  \end{center}
6118  }{}
6119  \fi% multiple
```

42.3   Including Assignments

`\in*assignment` This macro is essentially a glorified `\include` statement, it just sets some internal macros first that overwrite the local points Importantly, it resets the `inclassig` keys after the input.

```
6120  \keys_define:nn { hwexam / inclassignment } {
6121  %id   .str_set_x:N = \l__hwexam_assign_id_str,
6122  number   .int_set:N  = \l__hwexam_inclassign_number_int,
6123  title  .tl_set:N   = \l__hwexam_inclassign_title_tl,
6124  type   .tl_set:N   = \l__hwexam_inclassign_type_tl,
6125  given .tl_set:N  = \l__hwexam_inclassign_given_tl,
6126  due .tl_set:N  = \l__hwexam_inclassign_due_tl,
6127  mhrepos   .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
6128  }
6129  \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
6130  \int_set:Nn \l__hwexam_inclassign_number_int {-1}
6131  \tl_clear:N \l__hwexam_inclassign_title_tl
6132  \tl_clear:N \l__hwexam_inclassign_type_tl
6133  \tl_clear:N \l__hwexam_inclassign_given_tl
6134  \tl_clear:N \l__hwexam_inclassign_due_tl
6135  \str_clear:N \l__hwexam_inclassign_mhrepos_str
6136  \keys_set:nn { hwexam / inclassignment }{ #1 }
6137  }
6138  \__hwexam_inclassignment_args:n {}
6139
6140  \newcommand\inputassignment[2][]{
```

```
6141 \__hwexam_inclassignment_args:n { #1 }
6142 \str_if_empty:NTF \l__hwexam_inclassign_mhrepos_str {
6143 \input{#2}
6144 }{
6145 \stex_in_repository:nn{\l__hwexam_inclassign_mhrepos_str}{
6146 \input{\mhpath{\l__hwexam_inclassign_mhrepos_str}{#2}}}
6147 }
6148 }
6149 \__hwexam_inclassignment_args:n {}
6150 }
6151 \newcommand\includeassignment[2][]{
6152 \newpage
6153 \inputassignment[#1]{#2}
6154 }
```

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

42.4   Typesetting Exams

\quizheading

```
6155 \ExplSyntaxOff
6156 \newcommand\quizheading[1]{%
6157 \def\@tas{#1}%
6158 \large\noindent NAME: \hspace{8cm}  MAILBOX:\\[2ex]%
6159 \ifx\@tas\@empty\else%
6160 \noindent TA:~\@for\@I:=\@tas\do{{\Large$\Box$}\@I\hspace*{1em}}\\[2ex]%
6161 \fi%
6162 }
6163 \ExplSyntaxOn
```

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

\testheading

```
6164 \keys_define:nn { hwexam / testheading } {
6165 min   .tl_set:N  = \l__hwexam_testheading_min_tl,
6166 duration .tl_set:N  = \__hwexam_testheading_duration_tl,
6167 reqpts .tl_set:N  = \l__hwexam_testheading_reqpts_tl
6168 }
6169 \cs_new_protected:Nn \__hwexam_testheading_args:n {
6170 \tl_clear:N \l__hwexam_testheading_min_tl
6171 \tl_clear:N \l__hwexam_testheading_duration_tl
6172 \tl_clear:N \l__hwexam_testheading_reqpts_tl
6173 \keys_set:nn { hwexam / testheading }{ #1 }
6174 }
6175 \newenvironment{testheading}[1][]{
6176 \__hwexam_testheading_args:n{ #1 }
6177 \noindent\large{}Name:~\hfill
6178 Matriculation Number:\hspace*{2cm}\strut\\[1ex]
6179 \begin{center}
6180 \Large\textbf{\@title}\\[1ex]
6181 \large\@date\\[3ex]
6182 \end{center}
6183 \textbf{You~have~
```

```
6184    \tl_if_empty:NTF \l__hwexam_testheading_duration_tl {
6185    {\l__hwexam_testheading_min_tl}~minutes
6186    }{
6187    {\l__hwexam_testheading_duration_tl}
6188    }~
6189    (sharp)~for~the~test
6190    };\\
6191    Write~the~solutions~to~the~sheet.
6192    \par\noindent
6193    \newcount\check@time\check@time=\l__hwexam_testheading_min_tl
6194    \advance\check@time by -\theassignment@totalmin
6195    The~estimated~time~for~solving~this~exam~is~
6196    {\theassignment@totalmin}~minutes,~
6197    leaving~you~{\the\check@time}~minutes~for~revising~
6198    your~exam.
6199
6200    \par\noindent
6201    \newcount\bonus@pts\bonus@pts=\theassignment@totalpts
6202    \advance\bonus@pts by -\l__hwexam_testheading_reqpts_tl
6203    You~can~reach~{\theassignment@totalpts}~points~if~you~
6204    solve~all~problems.~You~will~only~need~
6205    {\l__hwexam_testheading_reqpts_tl}~points~for~a~perfect~score,~
6206    i.e.\ {\the\bonus@pts}~points~are~bonus~points.
6207    \vfill
6208    \begin{center}
6209        {
6210    \Large\em You~have~ample~time,~so~take~it~slow~
6211        and~avoid~rushing~to~mistakes!\\[2ex]
6212        Different~problems~test~different~skills~and~
6213    knowledge,~so~do~not~get~stuck~on~one~problem.
6214    }
6215    \vfill\par\resizebox{\textwidth}{!}{\correction@table}\\[3ex]
6216    \end{center}
6217    }{
6218    \newpage
6219    }
```

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

\testspace

```
6220    \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}
```

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

\testnewpage

```
6221    \newcommand\testnewpage{\iftest\newpage\fi}
```

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

\testemptypage

```
6222    \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.

```
6223 ⟨@@=problems⟩
6224 \renewcommand\@problem[3]{
6225 \stepcounter{assignment@probs}
6226 \def\__problemspts{#2}
6227 \ifx\__problemspts\@empty\else
6228 \addtocounter{assignment@totalpts}{#2}
6229 \fi
6230 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\f
6231 \xdef\correction@probs{\correction@probs & #1}%
6232 \xdef\correction@pts{\correction@pts & #2}
6233 \xdef\correction@reached{\correction@reached &}
6234 }
6235 ⟨@@=hwexam⟩
```

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

\correction@table This macro generates the correction table

```
6236 \newcounter{assignment@probs}
6237 \newcounter{assignment@totalpts}
6238 \newcounter{assignment@totalmin}
6239 \def\correction@probs{\correction@probs@kw}%
6240 \def\correction@pts{\correction@pts@kw}%
6241 \def\correction@reached{\correction@reached@kw}%
6242 \def\after@correction@table{}%
6243 \stepcounter{assignment@probs}
6244 \newcommand\correction@table{
6245 \resizebox{\textwidth}{!}{%
6246 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
6247 &\multicolumn{\theassignment@probs}{c||}%|
6248 {\footnotesize\correction@forgrading@kw} &\\\hline
6249 \correction@probs & \correction@sum@kw & \correction@grade@kw\\\hline
6250 \correction@pts &\theassignment@totalpts & \\\hline
6251 \correction@reached & & \\[.7cm]\hline
6252 \end{tabular}}
6253 \ifx\after@correction@table\@empty\else\strut\par\noindent\after@correction@table\fi}
6254 ⟨/package⟩
```

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

42.5   Leftovers

at some point, we may want to reactivate the logos font, then we use

```
here we define the logos that characterize the assignment
\font\bierfont=../assignments/bierglas
\font\denkerfont=../assignments/denker
\font\uhrfont=../assignments/uhr
\font\warnschildfont=../assignments/achtung

\newcommand\bierglas{{\bierfont\char65}}
\newcommand\denker{{\denkerfont\char65}}
```

```
\newcommand\uhr{{\uhrfont\char65}}
\newcommand\warnschild{{\warnschildfont\char 65}}
\newcommand\hardA{\warnschild}
\newcommand\longA{\uhr}
\newcommand\thinkA{\denker}
\newcommand\discussA{\bierglas}
```