The sTeX3 Package Collection*

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

2023-03-21

Contents

---

sTEX is – by now – relatively stable and ready to use for the general public. However, it is also actively being developed further and subject to ongoing research. Some of the features described in here might not fully work as expected, some are still experimental, there might occasionally be cryptic error messages, and in general bugs are expected.

We welcome all kinds of issues you might encounter at `https://github.com/slatex/sTeX`.

If you have questions or problems with sTEX, you can talk to us directly at
`https://matrix.to/#/#stex:fau.de`.

Chapter 1

Introduction & Setup

1.1 What is sTeX?

sTeX is a system for generating human-oriented documents in either PDF or HTML format, augmented with computer-actionable semantic information (conceptually) based on the OMDoc format and ontology.

At its core is the sTeX package for LaTeX, that allows for semantically marking up document fragments; in particular concepts, formulae and mathematical statements (such as definitions, theorems and proofs). Running `pdflatex` over sTeX-annotated documents formats them into normal-looking PDF.

But sTeX also comes with a conversion pipeline into semantically annotated HTML, which can host semantic added-value services that make the documents *active* (i.e. interactive and user-adaptive) – essentially turning LaTeX into a document format for (mathematical) knowledge management (MKM).

The sTeX system consists of the following components:

- The sTeX package,

- the RusTeX system for converting LaTeX documents to (semantically enriched) HTML,

- the MmT system for advanced knowledge management service and semantically informed backend for hosting the HTML with integrated added-value services,

- a collection of math archives of sTeX document fragments and symbols for reuse, available of `mathhub.info`, and

- the sTeX IDE: A VS Code extension that, besides the usual IDE functionalities like syntax highlighting, integrates RusTeX and MmT and allows for accessing the public math archives on `mathhub.info` to make the entire toolchain easily accessible to authors.

If PDF is all you are interested in, the sTeX package is all you *need*. Either way, however, we recommend using the sTeX IDE – it very much helps with semantically annotating your LaTeX documents.

1.2 The sTeX package

The sTeX package extends LaTeX with:

- A mechanism to declare symbols – concepts, functions, relations, variables, etc., which can be used and referenced in text or via semantic macros in mathematical formulae,

- a module system based on logical identifiers – modules bundle declarations, definitions, theorems, document snippets, and symbols for reuse, and

- an analogous organizational structure for developing documents modularly from individual fragments and sections.

The sTeX package has been designed to have minimal impact on other packages and document classes, or the actual document layout – formatting of semantic environments, symbol references and semantic macros can be fully customized.

1.3 What is Mmt?

Mmt is a software system and Scala API for generic knowledge management services. It is based on a version of the OMDoc ontology and document format (Mmt/OMDoc).

Among the services Mmt provides are compiling, building, converting and managing libraries, a built-in web-server for browsing content, various algorithms for generic computation, checking and translating expressions, and querying.

1.4 Math archives and the MathHub Directory

To make the most of sTeX, it is strongly encouraged to follow a workflow of small document fragments and modules to maximize reuse.

One considerable weakness of LaTeX is the way source files are referenced: they need to be either in a `texmf` directory, or else be referenced via file paths relative to the main `.tex`-file being compiled. This is highly inconvenient if we want to collaboratively develop many highly interrelated document fragments.

sTeX therefore adds an organizational layer on top of LaTeX's: **math archives** stored in a fixed `MathHub` directory anywhere on your hard drive. Referencing source files and modules is then done relative to the containing math archive, and is thus *independent* of user's individual setups or the current `.tex`-file.

The drawback of this approach is that sTeX needs to know the location of your `MathHub` directory. There are multiple ways to achieve that, but the simplest and recommended approach is to set an environment variable: Simply create a new directory `<path>/MathHub` somewhere on your hard drive and set the environment variable `MATHHUB` as the path to this new directory.

Alternatively, you can let the sTeX IDE do the work for you (see section 1.5).

For more on math archives, see section 5.2.
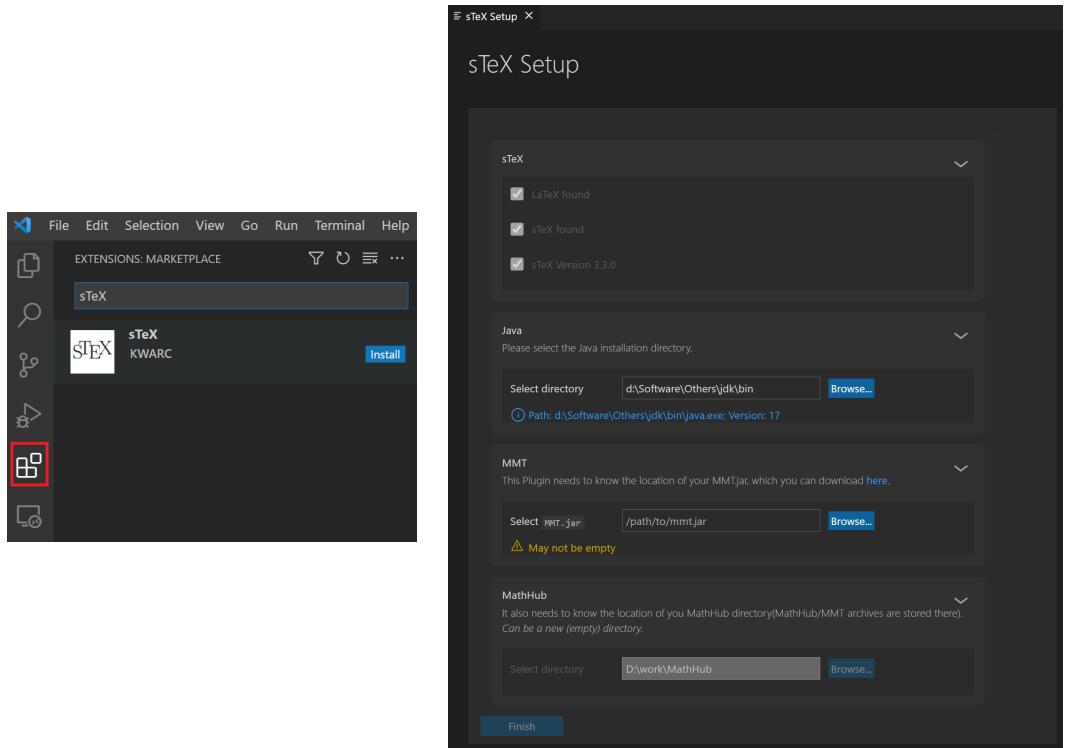
Figure 1: Installing the sTeX IDE

1.5   Setting Up the sTeX IDE

sTeX is based on LaTeX, and adds additional layers of presentational and functional markup to it. As a consequence the source files of sTeX documents look quite different from the resulting XHTML and PDF documents. Thus the best way of interacting the sTeX document collections is via an integrated development environment (IDE). In this tutorial we will use the sTeX plugin for the VS Code, which you should set up as a first step (this also sets up the necessary auxiliary software).

Setting up sTeX with the dedicated IDE is easy:

1. Download and install VS Code here: `https://code.visualstudio.com/download`

2. Start VS Code and navigate to the *Extensions*-tab on the left. Here you can search for Extensions in the VS Code marketplace. Look for the sTeX extension by *KWARC*, as in Figure 1 on the left.

3. Having done so, upon opening any folder in VS Code containing a `.tex`-file the setup window will pop up, as in Figure 1 on the right.

   The IDE will attempt to determine your Java installation and your `MathHub` directory (if set via an environment variable). Alternatively, you can set the latter now.

4. Download the Mmt `.jar`-file at the link provided in the setup and select it. The IDE should then be able to determine your Mmt version.

And that's it. Click on *Finish* and your setup is finished. The extension will start and download R_US T_EX and some fundamental math archives for you automatically (an internet connection is required when finishing the setup).

Part I
Tutorial

The dynamic HTML version of this part can be found at

sTeX is a system for generating human-oriented documents in either PDF or HTML format, augmented with computer-actionable semantic information (conceptually) based on the OMDoc format and ontology.

In this part, we will give a broad but shallow introduction to sTeX, and what you can get out of it. Additionally, this serves as an introduction to the sTeX IDE, and we consequently assume that you have that one set up, as described in section 1.5.

Note that in PDFs, the specific highlighting of semantically annotated text is fully customizable (see chapter 9). In this document, we use this highlighting for notation components, this highlighting for symbol references, this highlighting for (local) variables and **this highlighting** for definienda; i.e. new concepts being introduced.

Chapter 2

The Basics

This document itself uses sTeX and serves as a direct example for the following. You can download its source files, the generated PDF files, and the generated HTML documents directly from within the IDE, by navigating to the sTeX tab in the menu on the left and finding **sTeX/Documentation** in the list of math archives and clicking the small "Install"-button next to it, see the screenshot on the left of Figure 2.

Once downloading is finished (this may take a while since dependencies are also downloaded), you can then browse the `.tex`-files in **sTeX/Documentation** directly from the math archives panel in the sTeX tab, as you can see in the right screenshot in Figure 2.

For example, you can now navigate to the file `tutorial/intro.en` to see the sources of this very chapter.

As a first example, consider the following document fragment from section 1.1:

> sTeX is a system for generating human-oriented documents in either PDF or HTML format, augmented with computer-actionable semantic information (conceptually) based on the OMDoc format and ontology.

If you were to look at the generated HTML from this fragment, you could hover over the highlighted words (sTeX, PDF, HTML, OMDoc) and get a little popup with their definitions (Figure 3). Neat, huh?

Here, in the PDF, hovering will only show you a unique identifier (MMT-URI) for the word, and link to a definition on the web. Still useful, but not quite as neat, of course.

A plain LaTeX-version of the above document fragment, without any sTeX markup, could look like this:

Example 1
Input:
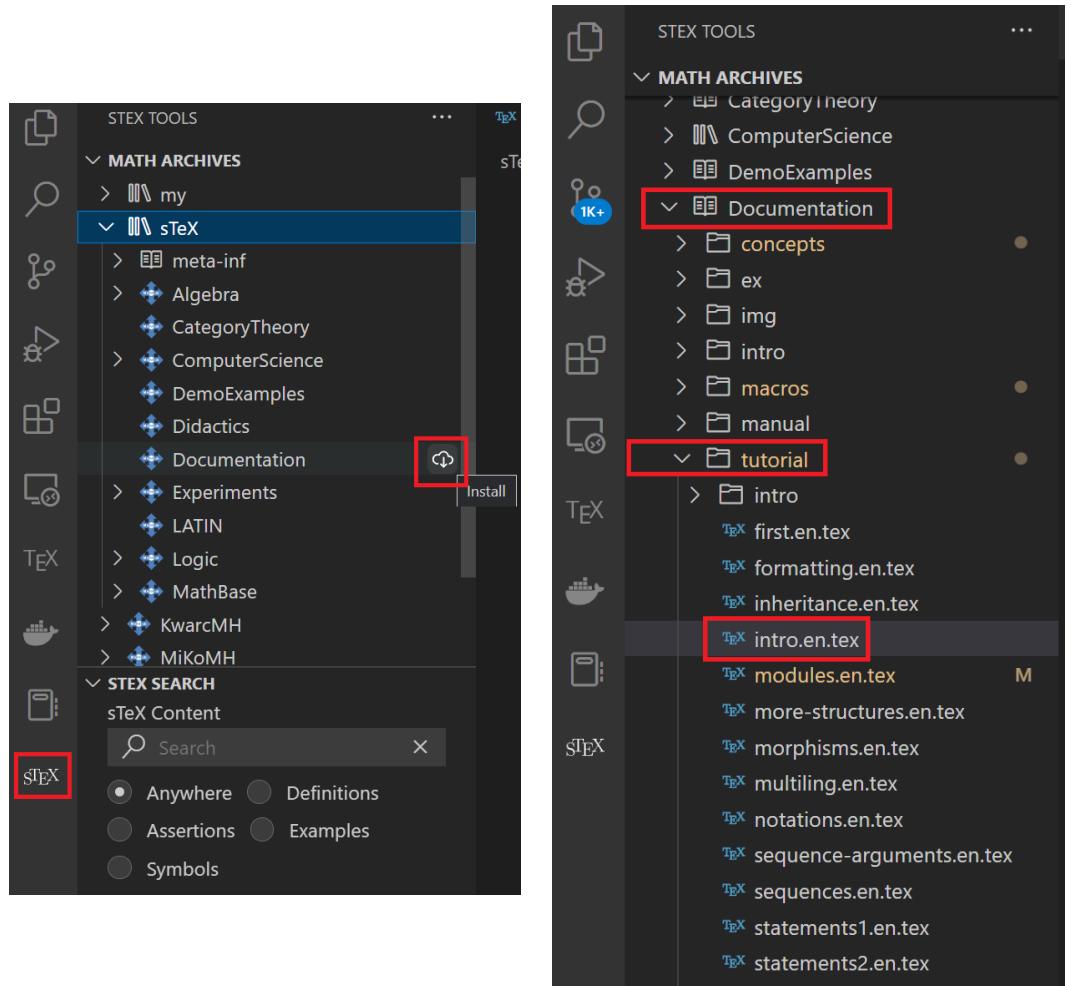
Figure 2: Installing Math Archives
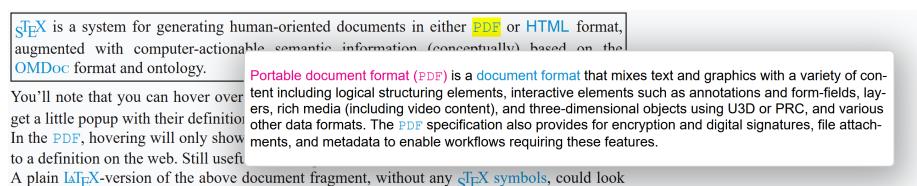
Figure 3: Definition on Hover

```
                    File [sTeX/Documentation]tutorial/intro/intro1plain.en.tex
 1 \documentclass{article}
 2 \usepackage{stex-logo}
 3 \begin{document}
 4
 5   \sTeX{} is a system for generating human-oriented documents
 6   in either \textsf{PDF} or \textsf{HTML} format, augmented
 7   with computer-actionable semantic information (conceptually)
 8   based on the \textsc{OMDoc} format and ontology.
 9
10 \end{document}
```

Output:

> sTeX is a system for generating human-oriented documents in either PDF or HTML format, augmented with computer-actionable semantic information (conceptually) based on the OMDoc format and ontology.

(Examples like the one above always show the file the source code is in, so if you have downloaded the `sTeX/Documentation` math archive you can toy around with it yourself)

If you save a file in the IDE (regardless of whether it has unsaved changes), a preview window will pop up, showing you the HTML generated from the `.tex`-file; see (Figure 4).
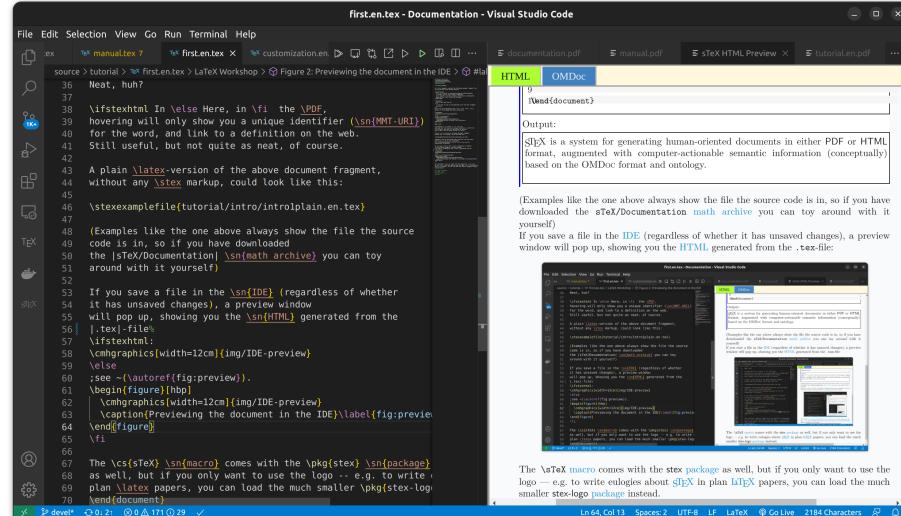
Figure 4: Previewing the document in the IDE

The \sTeX macro comes with the stex package as well, but if you only want to use the logo – e.g. to write eulogies about sTeX in plan LaTeX papers, you can load the much smaller stex-logo package instead.

2.1   Text symbols

The most central concept behind sTeX is that of a *symbol*:

> A **symbol** is a *named* concept that can be defined, documented and referenced.
> Examples for symbols are mathematical constants, functions, theorems, statements, principles – anything that has a (somewhat) precise meaning and can be referenced by name can be a symbol.

Before we explain how we can declare new symbols and associate them with definitions, notations and all that, let's assume an ideal world in which others have done that job already for us – after all, sTeX is all about *reuse*, and naturally, there are sTeX symbols for all of the above already. Let's start with the one for sTeX itself:

2.1.1   Using Modules & Search in the IDE

In the VS Code IDE, navigate to the sTeX-tab on the left. In the search panel, select the "`Symbols`" radio button and search for "`sTeX`". The second search result should be what we're looking for (Figure 5).

Search results are grouped into *local* and *remote* results. Local ones are the ones you already have in your local `MathHub` directory; remote ones you can download directly from within the IDE.

You can click the preview button to see the generated HTML for the document – the resulting window that pops up also has an OMDoc tab you can select, which (among other things) shows you the semantic macros provided by the respective module: In this case, it tells us that there is a *text symbol* named "sTeX" with semantic macro `\stex` in the module `mod/systems/tex?sTeX` that is in the `\sTeX/ComputerScience/Software` archive. It produces the presentation "sTeX" as we want (Figure 6).

> A **text symbol** is a symbol foo with an associated semantic macro `\foo`. The macro `\foo` is allowed in text or math mode and produces a predefined piece of text output annotated with foo.
> The variant `\fooname` produces the same output without annotation.

If we want to use the sTeX symbol in a document – which we have open in the IDE – we simply click on the `use` button, and the IDE will automatically insert the line `\usemodule[sTeX/ComputerScience/Software]{mod/systems/tex?sTeX}`, making all symbols in that module available to use – in particular, we can now use the `\stex` semantic macro instead of the plain, non-semantic `\sTeX` macro – that is, of course, after we include the `stex` package first.

> The `\usemodule` macro takes as *optional* argument the name of a math archive, and as a regular argument the path to an sTeX module (see section 7.5).

Analogously, we can also search for the PDF, HTML and OMDoc symbols, all of which are also text symbols and have the associated semantic macros `\PDF`, `\HTML` and `\omdoc`; the document should thus look like this:
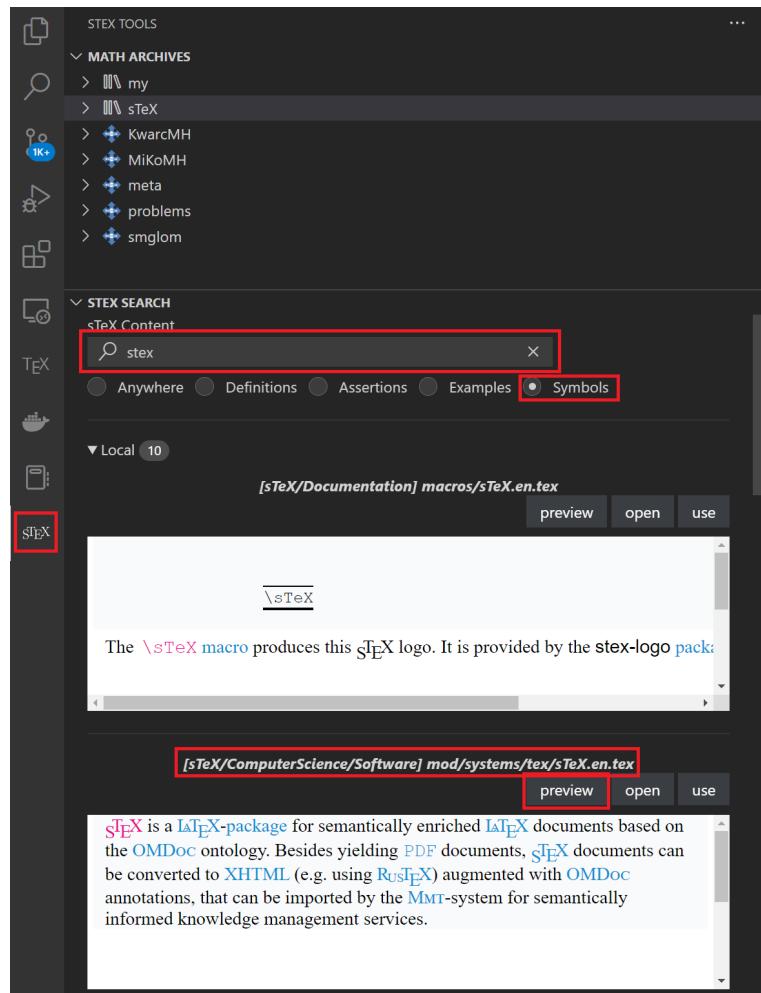
Figure 5: Search in the sTEX IDE

Figure 6: OMDoc Preview

Example 2

Input:

```
                    File [sTeX/Documentation]tutorial/intro/intro1stex.en.tex
 1 \documentclass{article}
 2 \usepackage{stex}
 3 \begin{document}
 4   \usemodule[sTeX/ComputerScience/Software]{mod/systems/tex?sTeX}
 5   \usemodule[sTeX/ComputerScience/Software]{mod/formats?PDF}
 6   \usemodule[sTeX/ComputerScience/Software]{mod/formats?HTML}
 7   \usemodule[sTeX/ComputerScience/Software]{mod/formats?OMDoc}
 8
 9   \stex is a system for generating human-oriented documents
10   in either \PDF or \HTML format, augmented
11   with computer-actionable semantic information (conceptually)
12   based on the \omdoc format and ontology.
13 \end{document}
```

Output:

> sTeX is a system for generating human-oriented documents in either PDF or HTML format, augmented with computer-actionable semantic information (conceptually) based on the OMDoc format and ontology.

Now, our generated HTML looks a lot more interesting, with highlighting, pop-ups on hover and all that. Notably however, if we compile the file with `pdflatex`, it looks pretty much exactly as before – except for (optional/configurable) colors.

That's because we haven't told sTeX what to do with semantic annotations yet – and by default, it does not do anything fancy, except for wrapping them in an `\emph`. We can customize how we want sTeX to highlight various semantic text fragments (see chapter 9). A default highlighting schema is provided in the stex-highlighting package – including that will

16
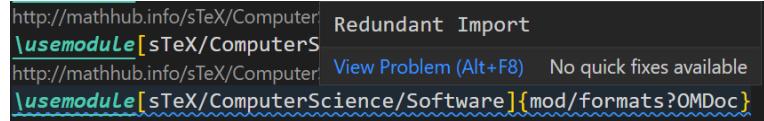
Figure 7: Redundant Imports
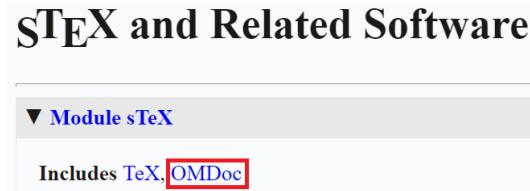
Figure 8: Includes in the OMDoc Preview

- highlight semantically annotated text in this color,

- show the MMT-URI of the corresponding symbol in a tooltip on hovering over the text,

- make the text link to the place the symbol is being defined in the current document (if it is), or, alternatively,

- make it link to an external resource, if one is known. In our case, they link to stexmmt.mathhub.info/:sTeX, where the HTML for all the symbols we use in this document are hosted.

Note that in the IDE, the \usemodule-statement for OMDoc is underlined in blue (Figure 7) – VS Code is letting us know, that this \usemodule statement is *redundant*. That is because the sTeX module we imported earlier already imports the OMDoc module; as such we have all macros therein available already. If we look at the sTeX module in the VS Code preview window again, we can see that (Figure 8).

We can consequently safely delete the \usemodule again.

2.2 Symbol References

Let's continue with the next paragraph of section 1.1; for now unannotated:

Example 3
Input:

17

```
                    File [sTeX/Documentation]tutorial/intro/intro2plain.en.tex
 1 \documentclass{article}
 2 \usepackage{stex}
 3 \begin{document}
 4
 5   At its core is the \sTeX{} package for \LaTeX, that allows for
 6   semantically marking up document fragments; in particular
 7   concepts, formulae and mathematical statements (such as
 8   definitions, theorems and proofs). Running \texttt{pdflatex}
 9   over \sTeX-annotated documents formats them into normal-looking
10   \textsf{PDF}.
11
12 \end{document}
```

Output:

> At its core is the sTeX package for LaTeX, that allows for semantically marking up
> document fragments; in particular concepts, formulae and mathematical statements
> (such as definitions, theorems and proofs). Running `pdflatex` over sTeX-annotated
> documents formats them into normal-looking PDF.

We already know how to annotate "sTeX" and "PDF"; and if we use the search field in the IDE again, we can also find a text symbol for "LaTeX". But if we look at the documentation, we will note that *more* is highlighted:

> At its core is the sTeX package for LaTeX, that allows for semantically marking up
> document fragments; in particular concepts, formulae and mathematical statements
> (such as definitions, theorems and proofs). Running `pdflatex` over sTeX-annotated
> documents formats them into normal-looking PDF.

The "`package`"-symbol can be found in the LaTeX module too, and searching for the keywords "formula" and "mathematics" will yield the symbols "`well-formed formula`" and "`mathematics`", but they are not *text symbols* and "`mathematics`" and "`package`" do not even have a semantic macro – and the one for "`well-formed formula`" would not work outside of math mode.

Text symbols are special in that way – they are intended for symbols that have a specific formatting associated (such as LaTeX, OMDoc, or HTML, which we prefer to typeset as sans serif). For those settings, it makes sense to associate that formatting with a semantic macro that does the typesetting for us.

Symbols *without* a text macro can be referenced with the `\symname` macro: `\symname{package}` prints the *name* of the "`package`"-symbol and annotates it accordingly, without any special formatting – in particular it is compatible with being in `\emph`, `\textbf` and similar macros. That takes care of *one* of the missing annotations.

More generally, the `\symref` macro can be used to annotate arbitrary text with a symbol: `\symref{mathematics}{mathematical}` associates the text `mathematical` with the symbol "`mathematics`"; thus, we get "mathematical" and similarly "formulae".

> sTeX In general, any macro that expects a symbol name can be given either
>
> 1. the *name* of the symbol,

Since `\symname` and `\symref` are a lot to type for something that should ideally be used as often as possible, the macros `\sn` and `\sr` exist as well and behave exactly the same way. We also provide some convenience abbreviations for `\sn`; namely `\Sn` (capitalizes the first letter of the symbol name), `\sns` (adds an "`s`" at the end, for the most common pluralization of a name), and `\Sns` (both).

Using all of the above, our annotated fragment now looks like this:

Example 4

Input:

```
                    File [sTeX/Documentation]tutorial/intro/intro2stex.en.tex
 5 \usemodule[sTeX/ComputerScience/Software]{mod/systems/tex?sTeX}
 6 \usemodule[sTeX/Logic/General]{mod/syntax?Formula}
 7 \usemodule[sTeX/MathBase/General]{mod?Mathematics}
 8 \usemodule[sTeX/ComputerScience/Software]{mod/formats?PDF}
 9
10   At its core is the \stex \sn{package} for \latex, that allows for
11   semantically marking up document fragments; in particular
12   concepts, \sr{wff}{formulae} and \sr{mathematics}{mathematical}
13   statements (such as definitions, theorems and proofs). Running
14   \texttt{pdflatex} over \stex-annotated documents formats them
15   into normal-looking \PDF.
```

Output:

At its core is the sTeX package for LaTeX, that allows for semantically marking up document fragments; in particular concepts, formulae and mathematical statements (such as definitions, theorems and proofs). Running `pdflatex` over sTeX-annotated documents formats them into normal-looking PDF.

There's only one problem: *the document does not compile*, with an error `Undefined control sequence`. The reason being that *some* macro in the module `Formula` uses the `\text` macro. We can fix that by using the `amsfonts` package of course, but this points to a more general problem; namely that modules can make use of various LaTeX packages for typesetting symbols.

Good practice suggests putting those packages into a *prelude* per math archive, which we can then import from anywhere, using the `\libinput` macro. For more on that, see section 5.3.

For now, suffice it to say that we can import all packages required for the module `Formula` from the math archive `sTeX/Logic/General` by adding the line

19

```
\libinput[sTeX/Logic/General]{preamble}
```

before the `\begin{document}`.

2.3   Modules and Simple Symbol Declarations

Consider again the first two paragraphs of section 1.1:

> sTeX is a system for generating human-oriented documents in either PDF or HTML
> format, augmented with computer-actionable semantic information (conceptually)
> based on the OMDoc format and ontology.
>   At its core is the sTeX package for LaTeX, that allows for semantically marking
> up document fragments; in particular concepts, formulae and mathematical state-
> ments (such as definitions, theorems and proofs). Running `pdflatex` over sTeX-
> annotated documents formats them into normal-looking PDF.

Firstly, note that the first paragraph would be perfectly suitable to serve as a pop-up
definition on hover for the sTeX symbol. Secondly, what if all the symbols used in the
above *didn't* already exist?

In this section, we will describe how to make your own symbols and collect them as
reusable fragments in modules and math archives from scratch.

We start by creating a new math archive. In the IDE, switch to the sTeX-tab on the
left and click the "`New sTeX Archive`" button (Figure 9). You will then be asked for the

Figure 9: New Math Archive in the IDE

name of the archive, a namespace for its content, and a url-base, where the content is
supposedly going to end up online. You can safely keep the defaults for the latter two.
In the following, we assume that your archive is named `my/archive`.

The IDE will then create the following files and directories in your MathHub direc-
tory:

```
- my
  - archive
    - lib
      - preamble.tex
    - META-INF
      - MANIFEST.MF
    - source
      - helloworld.tex
```

. . . and open the file `helloworld.tex` with the content

```
1  \documentclass{stex}
2  \libinput{preamble}
3  \begin{document}
4  % A first sTeX document
5  \end{document}
```
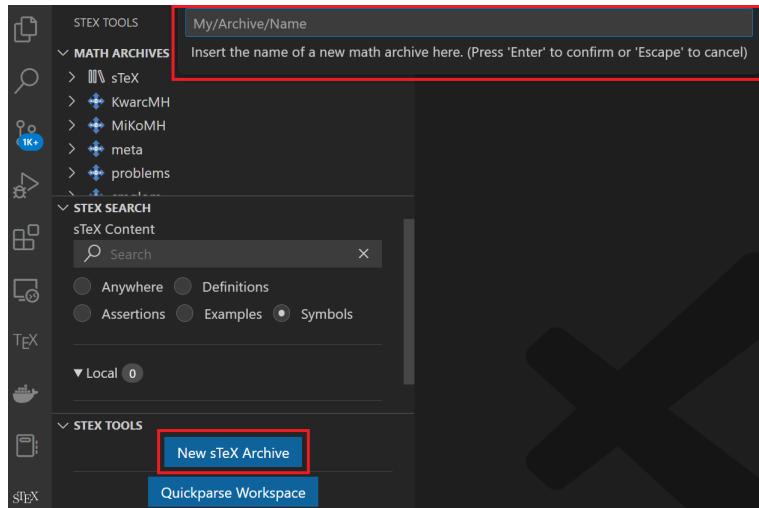
You can now reference any newly created content in you new archive using for example `\usemodule[my/archive]{...}`.

Let's start with the "L^ATEX" symbol. Rename the file `helloworld.tex` to something more meaningful, for example `latex.en.tex` – the `.en` will be picked up on by sTEX to signify that the fragment will be in *english* (see subsection 7.1.1).

What we want to achieve in this file is the following:

> **TEX** is a document typesetting software developed by Donald Knuth, with a focus on mathematical formulae. It is based on a powerful and extensible **macro** expansion engine.
>
> **L^ATEX** is a (nowadays) default collection of TEX macros developed by Leslie Lamport. Among other things, L^ATEX introduces **environments**, a distinction between preamble and document content, **packages** to bundle and distribute macro definitions, and **document classes**: special packages that govern the global layout of a document.

In particular, in the HTML the two paragraphs above should be shown when hovering over the symbols they define (as indicated by the magenta definiendum highlighting). So we need symbols and semantic macros, for: TEX, macro, L^ATEX, environment, package and document class.

Symbol declarations are only allowed within modules:

> sTEX   A **module** is a *named* block that bundles symbol declarations for subsequent reuse. A module is introduced with the `smodule`-environment.

Let's name our module `LaTeX`. We then wrap the contents of our document in a `smodule` environment:

```
\begin{document}
  \begin{smodule}{LaTeX}
    ...
  \end{smodule}
\end{document}
```

Note, that the IDE immediately picks up on this and displays the full MMT-URI of our new module over the `\begin{smodule}{LaTeX}` (Figure 10) –

Figure 10: VS Code Code Lense

From this, we can glimpse that the namespace of the module is `http://mathhub.info/my/archive/latex`. This implies, that to use the module somewhere else, we will have to type `\usemodule[my/archive]{latex?LaTeX}` – the `latex`-part pointing to the *file* and `LaTeX` referring to the actual module.

If we rename the file to `LaTeX.en.tex`, we notice that the namespace changes to `http://mathhub.info/my/archive`, allowing us to now use it with `\usemodule[my/archive]{LaTeX}` directly. That's because the module name `LaTeX` and the file name `LaTeX` match now (see section 7.5, Figure 11).

Figure 11: VS Code Code Lense

> Note that "`LaTeX`" and "`latex`" only differ in capitalization – if your file system is case-insensitive (as e.g. MacOS's was until quite recently), this distinction gets murky, but remains very important especially if you want to share your math archive with others!
>
> It is therefore *highly recommended* to treat file names as case-sensitive either way.

Within the module, we can now declare new symbols using the `\symdecl`-macro. We start with those that are not text symbols:

```
\symdecl*{macro}
\symdecl*{environment}
\symdecl*{package}
\symdecl*{document class}
```

The * after the `\symdecl` indicates, that we do not want a semantic macro for the symbol – otherwise, it would generate one with the same name as the symbol itself and "pollute the macro space", so to speak.

The symbols TeX and LaTeX, however, have a definite way of being typeset associated with them, which can be produced using the standard `\TeX` and `\LaTeX` macros. So let's make them text symbols, using the `\textsymdecl` macro:

```
\textsymdecl{tex}{\TeX}
\textsymdecl{latex}{\LaTeX}
```

The first argument being the name of the generated macro (i.e. `\tex` and `\latex`) and the second one specifying the output to produce.

2.4   Documenting Symbols

We can now use the two new macros, \symname/\sn, \symref/\sr etc. to mark up the above two paragraphs. But the IDE also makes us aware of the symbols not yet being documented, via squiggly blue lines(Figure 12).


```
Symbol is not documented
Consider adding an {sdefinition} or {sparagraph}[type=symdoc]
for this symbol
View Problem (Alt+F8)    No quick fixes available
\symdecl*{macro}
```
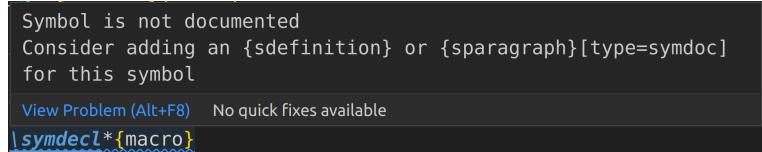
Figure 12: Undocumented Symbols

Among other things, this means that the system does not yet know what to show a reader when hovering over the symbol in the HTML. The IDE also recommends two ways to fix that: The sdefinition or sparagraph environments.

Ignoring the former for now, which is more useful for mathematical concepts, we can use the following to mark up the first paragraph:

```
\begin{sparagraph}[style=symdoc,for={tex,macro}]
  \tex is a document typesetting
  software developed by Donald Knuth, with a focus on
  mathematical formulae. It is based on a powerful
  and extensible \sn{macro} expansion engine.
\end{sparagraph}
```

In general, the sparagraph environment can be used to mark up arbitrary paragraphs semantically, but the style=symdoc option tells sTeX to use this paragraph as a documentation for the symbols provided in the for= option. And indeed, doing so makes the squiggly blue lines in the IDE under \textsymdecl{tex}{TeX} and \symdecl*{macro} disappear.

We just used the semantic macro \stex and the \sn macro to mark up the fragment – but we can do better. Both concepts are being *introduced* in the above paragraph, and we can let sTeX know that that is the case:

Within an sparagraph environment with style=symdoc (or an sdefinition environment), we can mark up *definienda*, meaning the terms *being defined*, explicitly. Analogously to \symname and \symref, we have the macros \definame and \definiendum for that purpose.

Note that the \tex macro induced by the text symbol above already marks up the "TeX" it produces, so wrapping it in another \definiendum would be redundant. However, every text symbol also generates a *second* macro with the suffix name that generates a non-marked-up version of the same presentation. In other words, we get the macro \texname for free, that produces "TeX" (of course, we could just as well use the \TeX macro, but that one you probably know already).

Furthermore, every \definiendum or \definame automatically adds the symbol being referenced to the internal for=-list of the sparagraph environment, obviating the need to list it explicitly.

As such, we can produce a better markup like this:

```
\begin{sparagraph}[style=symdoc]
  \definiendum{tex}{\texname} is a document typesetting
```

23

```
    software developed by Donald Knuth, with a focus on
    mathematical formulae. It is based on a powerful
    and extensible \definame{macro} expansion engine.
\end{sparagraph}
```

Exercise

In your archive `my/archive`, create additional files that produce the following outputs:

Mathematics.en.tex

To do **mathematics** is to be, at once, touched by fire and bound by reason. This is no contradiction. Logic forms a narrow channel through which intuition flows with vastly augmented force.

– Jordan Ellenberg

PDF.en.tex

Portable Document Format (PDF) is a document format that mixes text and graphics with a variety of content.

HTML.en.tex

The **HyperText Markup Language (HTML)** is a representation format for web-pages.

OMDoc.en.tex

OMDoc is a document format for representing mathematical documents with their flexiformal semantics.

---

such that the following file compiles and shows the above snippets on hover:

sTeX.en.tex

```
1  \documentclass{stex}
2  \libinput{preamble}
3  \begin{document}
4  \begin{smodule}{sTeX}
5    \usemodule{OMDoc}
6    \usemodule{PDF}
7    \usemodule{HTML}
8    \textsymdecl{stex}{\sTeX}
9    \begin{sparagraph}[style=symdoc]
10     \definiendum{stex}{\stexname} is a system for generating
11     documents in either \PDF or \HTML format, augmented with
12     computer-actionable semantic information (conceptually)
13     based on the \OMDoc format and ontology.
14   \end{sparagraph}
15 \end{smodule}
16 \end{document}
```

sTeX is a system for generating documents in either PDF or HTML format, augmented with computer-actionable semantic information (conceptually) based on the OMDoc format and ontology.

The preamble of every file should only be

```
\documentclass{stex}
\libinput{preamble}
```

and the macros \OMDoc, \PDF, \HTML should produce \textsc{OMDoc}, \textsf{PDF} and \textsf{HTML}, respectively (but with semantic annotations of course).

Solution: Can be found in [sTeX/Documentation]source/tutorial/solution

---

2.5   Sectioning and Reusing Document Fragments

We know now how to import and reuse the symbols of some module (using \usemodule). What about the actual document *content*?

Assume we want to write a new article that includes all of the fragments in my/archive we made so far, in a file all.en.tex in the same math archive:

```
1  \documentclass{article}
2  \usepackage{stex}
3  \libinput{preamble}
4  \begin{document}
5    \author{Me}
6    \title{The \texttt{my/archive} Archive}
7    \maketitle
8    \tableofcontents
9    ...
10 \end{document}
```

In there, we want sections as follows:

```
- 1 Preliminaries
  (Mathematics)
  - 1.1 Document Formats
    (PDF)
    (HTML)
    (OMDoc)
- 2 \TeX and Friends
  (LaTeX)
  (sTeX)
```

We could of course do the following:

```
\section{Preliminaries}
  \input{Mathematics.en}
  \subsection{Document Formats}
    \input{PDF.en}
    \input{HTML.en}
    \input{OMDoc.en}
\section{\TeX and Friends}
  \input{LaTeX.en}
  \input{sTeX.en}
```

...but this approach has two drawbacks:

Firstly, we need to manually keep track of the section levels, by explicitly writing \section, \subsection etc. This is fine as long as we are just interested in this particular article. But what if we want to *reuse* the article's content in another document at some point? The section levels might be entirely different then – e.g. we might want the "Preliminaries" section to be a subsection instead.

Secondly, the `\input` macro considers the file name/path provided to be either *absolute* or relative to the *current `tex` file being compiled* – which means that the `\input{Mathematics.en}` only works for files in the same directory as `Mathematics.en.tex`.

In short: using `\section`, `\chapter` etc. explicitly, and `\input` to reuse fragments, breaks reusability.

Instead of using `\section` and `\subsection`, sTeX therefore provides the sfragment environment.

`\begin{sfragment}{Foo}...\end{sfragment}` inserts a sectioning header depending on the current section level and availability. These are: `\part`, `\chapter`, `\section`, `\subsection`, `\subsubsection`, `\paragraph` and `\subparagraph`. This allows us to do the following instead:

```
\begin{sfragment}{Preliminaries}
  \input{Mathematics.en}
  \begin{sfragment}{Document Formats}
    \input{PDF.en}
    \input{HTML.en}
    \input{OMDoc.en}
  \end{sfragment}
\end{sfragment}
\begin{sfragment}{\TeX and Friends}
  \input{LaTeX.en}
  \input{sTeX.en}
\end{sfragment}
```

The only problem remaining now is that if we do this, sTeX will insert a `\part` for the first `sfragment`. If we want the "top-level" sectioning level to be `\section` instead, we can insert a `\setsectionlevel{section}` in the preamble.

As a more reuse-friendly replacement of `\input`, sTeX provides the `\inputref` macro. Using that has two advantages: Firstly, its argument is relative to some (optionally provided, or the current) math archive and is thus independent of the specific location of the file relative to the currently being compiled `.tex`-file. Secondly, when converting to HTML, it will *not* "copy" the referenced file's content in its entirety (as `\input` would), but instead dynamically insert the already existent (if so) HTML of the referenced file, avoiding content duplication and having to process the file all over again.

In general `\inputref[some/archive]{file/path}` inputs the file `file/path.tex` in the archive `some/archive`. As the `\input`-ed files in the example above are in the same archive anyway, we can simply substitute the `\input`s by `\inputref`s and call it a day.

Finally, we can make two more minor changes:

1. The *title* of our document is only supposed to be there, if we compile the document directly – if we were to `\inputref` our file into a "driver file" `all.en.tex`, the title and the table of contents should be omitted.

   We can achieve this using the `\ifinputref` conditional: by wrapping the header in an `\ifinputref \else...\fi`, it will only be processed if the file is *not* being loaded using `\inputref`. `\ifinputref` is a "classic" TeX conditional and is treated as such in both PDF and HTML compilation. A smarter macro to use is `\IfInputref`, which takes two arguments for the *true* and *false* cases, respectively.

Additionally, when compiling to HTML, *both* arguments to `\IfInputref` will be processed, and the backend will decide which of the two to present when serving a document.

2. The table of contents should also be omitted in HTML mode. To achieve that, we can use the `\ifstexhtml` conditional, which is *true* if the document is being compiled to HTML, and *false* if compiled to PDF.

> ⚠ Note, that since *both* arguments of `\IfInputref` are processed, they should *not* open TeX groups or environments!

In summary, we can modify our document to do the following:

```
\IfInputref{}{
    \author{Me}
    \title{The \texttt{my/archive} Archive}
    \maketitle
    \ifstexhtml \else \tableofcontents \fi
}
```

The final `all.en.tex` can be found in `[sTeX/Documentation]tutorial/solution/all.en.tex`.

2.6   Building and Exporting HTML

So far we know how to write sTeX documents, (we assume) how to build PDF files from them (via `pdflatex` of course), and on saving documents the IDE will preview the generated HTML. But if we do that with our new `all.en.tex`, we get presented with Figure 13 Where did all of our fragments go?
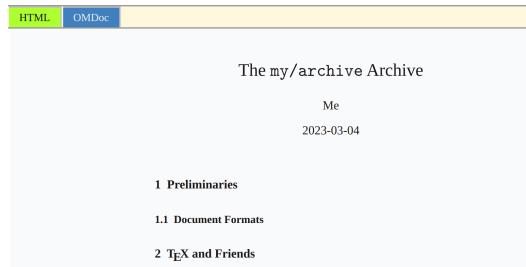
Figure 13: Missing Fragments in the HTML Preview

Well, they don't exist yet as HTML. The HTML Preview window in the IDE is really just that: A *preview.* But when using `\inputref`, it has to find the HTML of the `\inputref`ed fragment *somewhere.* Meaning: we have to compile all of the fragments we used to HTML first. Individually, we can compile the currently open file in VS Code using the button in Figure 14.

This will do the following:

1. Run `pdflatex` over the file three times.

2. Store the resulting `.pdf` in `[archive]/export/pdf/<filepath>.pdf`.

Figure 14: The `Build PDF/XHTML/OMDoc` Button

3. Convert the file to HTML and store it in `[archive]/xhtml/<filepath>.xhtml`.

4. Extract all the semantics and store them as OMDoc in `[archive]/content/...`, `[archive]/narration/...` and `[archive]/relational/...`.

5. Construct a search index in `[archive]/export/lucene/...`.

Doing all of this for every individual file *in hindsight* would of course be a huge hassle. We can therefore just compile the full archive, folders in an archive, or whole *groups* of archives via right-clicking an element in the `Math Archives` viewer in the sTeX tab (Figure 15).
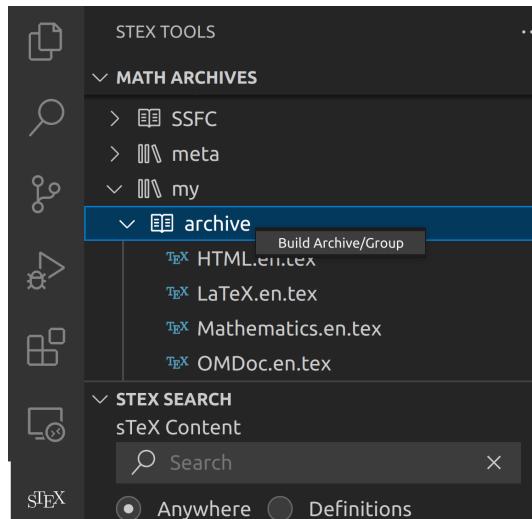
Figure 15: Building Archives in the IDE

Once that's done, saving `all.en.tex` again yields the correct HTML in the preview window.

At this point, it should be noted that you can't actually just open the HTML files exported to `[archive]/xhtml` in your browser and get all of the expected functionality – that shouldn't be too surprising. Features like the fancy pop-up windows require a semantically informed backend infrastructure, in the form of the MMT system. However, MMT *can* dump a standalone version for you. Let's do that now:

With our `all.en.tex` file open and everything built as above, click the `Export Standalone HTML`-button in the IDE (see Figure 16).

In the dialog box that opens now, select an **empty** directory and MMT will dump a standalone version of our `all.en.tex` document there. You will still not be able to

Figure 16: Exporting HTML in the IDE

open it in the browser directly, because most browser forbid javascript modules on the `file://` protocol, but opening the file via `http` will yield the desired result, and you can now upload the directory's content to wherever you might want to use it.

If you want to test this, a quick and easy way to do so is to use VS Code: You can install the `Live Server` extension, open the directory and click the `Go Live` button on the lower right of the window, which will start a small web-server in the selected directory and open its `index.html` in the browser for you.

Chapter 3

Mathematical Concepts

So far, we have seen how to declare and reference symbols generate semantic macros for text symbols, collect them in modules and document them properly.

But where sTeX really shines is when it comes to mathematics and related subject areas: semantic macros are significantly more useful when used for generating symbolic notations in math mode, and by associating symbols with (flexi-)formal semantics, sTeX can even *check* that your content is (to some degree) formally correct, or at least well-formed.

Alos sTeX provides specialized functionality for mathematical statements: the text fragments marked as Definition, Theorem, Proof that are iconic to mathematical documents.

The example snippets in this chapter can be found in the math archive `sTeX/MathTutorial`. If you downloaded the `sTeX/Documentation` archive in the sTeX IDE, you already have that archive. If not, you can download it from within the IDE, as described in chapter 2.

3.1   Simple Symbol Declarations

We will start with symbols and semantic macros for mathematical concepts and objects and their contribution to mathematical formulae.

3.1.1   Semantic Macros and Notations

Let us start with a very fundamental concept; namely equality. As you should by now know, declaring a new symbol requires a module, so let's open a new one and use `\symdecl`:

```
\begin{smodule}{Equality}
  \symdecl{equal}
\end{smodule}
```

As mentioned in section 2.3, the starred variant `\symdecl*` does not create a semantic macro, so presumably, the variant without a `*` *does*. And indeed, we now have a macro `\equal`, which however will produce errors if we try to use it. That's because we haven't told sTeX what to do with it yet.

A **semantic macro** is a LaTeX-macro that allows for referencing a symbol itself, or – in the case of e.g. a function – the *application* of a symbol to (one or multiple) *arguments*; primarily by invoking a symbol's notation in *math mode*.

The command `\symdecl{macroname}` declares a new symbol with name `macroname` and a semantic macro `\macroname`. In the case where we want the name and the semantic macro to be distinct, the command `\symdecl{macroname}[name=some name]` declares the name of the symbol to be `some name` instead.

The starred variant `\symdecl*{name}` declares the concept with the given name, but does not generate a semantic macro.

So let's provide equality with a notation. As a first step, we should let sTeX know that "equal" takes two arguments. We might also want to shorten the semantic macro to e.g. `\eq`, without changing the name. Hence:

```
\symdecl{eq}[name=equal,args=2]
```

Next, we add an infix notation with the `\notation` macro:

```
\notation{eq}{#1 = #2}
```

That seems like a lot to write, so for the very common case where we want to declare a symbol with a semantic macro and a notation all at once, the `\symdef` macro does all three by combining the optional and mandatory argument of `\symdecl` and `\notation`:

```
\symdef{eq}[name=equal,args=2]{#1 = #2}
```

and indeed, we can now use the `\eq` macro in math mode to invoke our new notation: `$\eq{a}{b}$` now yields $a = b$ – notably without any highlighting (and hover interaction in the HTML) though. Since our semantic macro takes *arguments*, which should be differently highlighted, we need to let our notation know which parts of the notation are highlightable components.

We can do so with the `\comp` and `\maincomp` macros:

The `\comp`-macro marks components to be highlighted in a notation for a symbol taking (one or more) arguments.

This is necessary because it is (nearly) impossible for LaTeX to figure out, which parts of a notation to highlight and which not on its own – in particular, the highlighting should stop for the *arguments* of a semantic macro.

Additionally, the `\maincomp` macro can be used to mark (at most) one notation component to represent the *primary* component of the notation.

Notations that do not take arguments, as well as operator notations, are automatically wrapped in `\maincomp`.

In our case, this applies only to the "=", symbol, so:

```
\symdef{eq}[name=equal,args=2]{#1 \mathrel{\maincomp{=}} #2}
```

You may be wondering about the role of the `\mathrel` macro in the example above: TeX determines spacing/kerning in math mode by assigning a *class* to every character. Both individual characters and whole subexpressions can be assigned one of these classes using dedicated macros. These are:

class	T_EX macro	examples
ordinary (default class)	\mathord	$\alpha\ i\ \diamond$
large operator	\mathop	$\sum \prod \int$
opening	\mathopen	$(\ [\ \langle$
closing	\mathclose	$)\ ]\ \rangle$
binary relation	\mathrel	$\leq\ >\ =$
binary operator	\mathbin	$+\ \cdot\ \circ$
punctuation	\mathpunct	$,\ ;$

T_EX "forgets" the class of an expression if it is wrapped in a `\comp` macro. It is therefore a good idea to wrap any occurence of a `\comp` in the corresponding T_EX macro for the desired class (e.g. `\mathrel{\comp{\leq}}`).

Having done so, we can now type `$\eq{a}{b}$` to get $a = b$. Thanks to using `\maincomp`, we now also have an operator notation, which we can invoke using `$\eq!$`, yielding $=$.

What if we want to add more notations? Say we want to be able to invoke equality to get the variant notation $a \equiv b$ ()without changing the intended meaning). If we want to be able to choose one of several notations, we should give the notation an *identifier*.

Let's again modify our earlier notation by adding the identifier `eq` to the optional arguments of `\symdef`, like so:

```
\symdef{eq}[name=equal,args=2,eq]{#1 \mathrel{\maincomp{=}} #2}
```

We can now invoke the specific notation provided here by writing `$\eq[eq]{a}{b}$` to the same effect. But we can also add more notations using the `\notation` macro:

```
\notation{eq}[equiv]{#1 \mathrel{\maincomp{\equiv}} #2}
```

which we can now invoke with `$\eq[equiv]{a}{b}$`, yielding $a \equiv b$.

By default, the *first* notation provided for a given symbol is considered the *default notation*, which is invoked if the semantic macro is used without an optional argument – hence, `$\eq{a}{b}$` still yields $a = b$.

If we use the starred variant of the `\notation` macro, the notation is set as the new default. Hence, had we done

```
\notation*{eq}[equiv]{#1 \mathrel{\maincomp{\equiv}} #2}
```

then `$\eq{a}{b}$` would now yield $a \equiv b$.

Any already existing notation can be set as default using the `\setnotation` macro; e.g. instead of using `\notation*`, we could also do

```
\notation{eq}[equiv]{#1 \mathrel{\maincomp{\equiv}} #2}
\setnotation{eq}{equiv}
```

Exercise
Implement the symbol "equal" as above in a new module "Equality" and add a documentation such that hovering over the symbol in the HTML yields the following snippet:

> Two objects a, b are considered **equal** (written $a = b$ or $a \equiv b$), if there is no property that distinguishes them.

Solution: Can be found in [sTeX/MathTutorial]/mod/Equality1.en.tex

3.1.2 Types and Variables

You might have noticed – after you save the file – that the expressions `$\eq{a}{b}$` and `$\eq[equiv]{a}{b}$` are underlined in yellow in the IDE and have a warning attached to them (Figure 17). If we click on the `Invalid Unit` link in the error message, we get

Figure 17: Type Checking Warning

a somewhat cryptic stacktrace-like window (Figure 18). The reason being, that Mmt
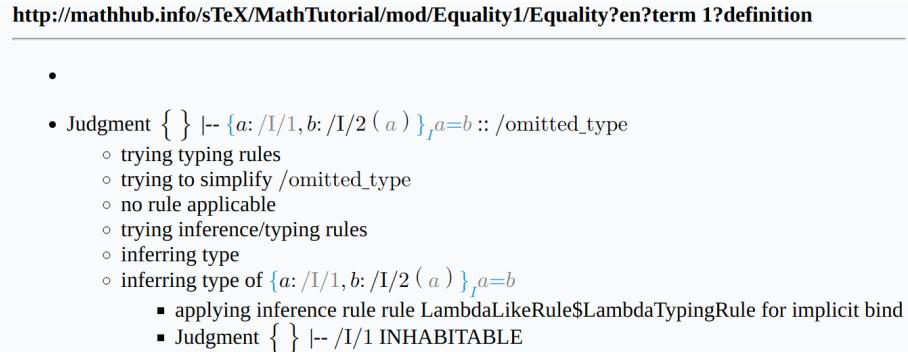
Figure 18: Type Checking Proof Tree

actually tries to formally verify *everything we write using semantic macros!* It does so, by attempting to infer the *type* of an expression – success implies that the expression is in fact well-typed.

If the former paragraph is difficult to comprehend for you, don't worry – you'll likely pick up on things as we go along. For now, sufice it to say that we can assign "*types*" to symbols, and the Mmt system is smart enough to use those to check that what we're writing actually "makes sense"; for example, $a + b$ makes perfect sense if $+$ is addition and a and b are numbers, or elements of a vector space, but not if a and b are, say, triangles.

> sTeX Every symbol or variable can be assigned a **type**, signifying what "kind of object" the symbol represents, or what (primary) set it is contained in.

33

In order to *formally verify* a mathematical statement, we have to rely on a set of *rules* that determine what is or isn't a valid statement. There are many systems of such rules with very different flavours, called **(logical) foundations**.

The most commonly used foundation in (informal) mathematics is *set theory*, in particular *ZFC*; a set of axioms in (usually) *first-order logic*. However, in *computer proof assistants* and similar systems, *type theories* like *higher-order logic* or the *calculus of (inductive) constructions* are more popular, because they lend themselves better to computer implementations.

In as far as possible, we prefer to remain "foundationally agnostic", or **foundation independent**: Every foundation has advantages and disadvantages, and which one is appropriate often depends on the particular setting one is working in.

Nevertheless, certain "meta-principles" have proven themselves to be extremely effective in representing and checking mathematical content in software, and while we do not fix a particular foundation or specific checking rules, we will make use of those principles in general. These include e.g. the *Curry-Howard Correspondance*, or *Judgments-as-Types paradigm*, and *Higher-Order Abstract Syntax*.

Full formal verification of document content is an extremely lofty goal, and hardly realistic if you're not willing to write your content in pretty specific ways, and informed by a decent amount of background knowledge in formal logic. Moreover, formally verifying content in sTeX is an ongoing research project, so we will not go into the specifics in detail here.

While full formal verification is out of reach for now, annotating adequate types can strike a useful balance between the effort required and the benefit of automated meaning checking afforded by them. In this sense sTeX is pragmatically similar to programming languages where adding types can raise the quality and correctness assurance in programs.

Keep in mind that getting `Invalid Unit` warnings does not impact at all what your document is going to look like – feel free to ignore them entirely.

Types are particularly useful for *variables*:

A **variable** represents a *generic* or *unspecified* object.

Variables can be declared using the `\vardef`-macro, whose syntax is analogous to `\symdef`.

Note that variables are local to the current TeX-group (e.g. environment).

Let's leave our equality-module aside for now and turn our attention to something simpler: natural numbers. Consider the following module:

Example 5

Input:

```
 \begin{smodule}{Nat}
   \symdef{Nat}[name=natural numbers]{\mathbb N}
   \begin{sparagraph}[style=symdoc]
     The \definame{Nat} $\defnotation{\Nat}$ are the numbers
     $0,1,2,...$
   \end{sparagraph}
   \symdef{plus}[name=addition,args=2]{#1 \mathbin{\maincomp{+}} #2}
   \begin{sparagraph}[style=symdoc]
     \Definame{addition} $\defnotation{\plus{a}{b}}$
     refers to the process of adding two \sn{Nat}.
   \end{sparagraph}
 \end{smodule}
```

Output:

> The **natural numbers** $\mathbb{N}$ are the numbers 0,1,2,...
> **Addition** $a+b$ refers to the process of adding two natural numbers.

(like \definame and \definiendum, the \defnotation macro is only allowed in documenting environments like sparagraph[style=symdoc] or sdefinition, and highlights the notation components marked with \comp or \maincomp the same way as \definame and \definiendum do.)

Note, that as the \Nat semantic macro does not take any arguments, we do not need to wrap the notation in a \comp or \maincomp.

Note also, that the \plus{a}{b} is again underlined in the IDE with an `Invalid Unit` warning.

The above fragment uses two variables a and b. In fact, MMT will consider them variables even though they are not marked up as such – but since they are not marked up, we are missing out on useful functionality.

Let's change that by adding two variable definitions[1]:

Example 6
Input:

```
 \begin{sparagraph}[style=symdoc]
   \vardef{va}[name=a]{a}\vardef{vb}[name=b]{b}
   \Definame{addition} $\defnotation{\plus{\va}{\vb}}$
   refers to the process of adding two \sn{Nat}.
 \end{sparagraph}
```

Output:

> **Addition** $a+b$ refers to the process of adding two natural numbers.

Okay, so now a and b are gray, but besides that, we haven't achieved much yet.

Let's change that by giving the variables the type $\mathbb{N}$:

---

[1] Technically, this is called a *variable reservation*, for those in the know.

Example 7

Input:

```
\begin{sparagraph}[style=symdoc]
  \vardef{va}[name=a,type=\Nat]{a}\vardef{vb}[name=b,type=\Nat]{b}
  \Definame{addition} $\defnotation{\plus{\va}{\vb}}$
  refers to the process of adding two \sn{Nat}.
\end{sparagraph}
```

Output:

Addition $a+b$ refers to the process of adding two natural numbers.

Now if we hover over the a and b (in the HTML), it will show us that their type is $\mathbb{N}$!

We can of course also assign types to symbols. In the IDE, find the symbol *"function space"* with semantic macro `\funspace` (in `[sTeX/MathBase/Functions]{mod?Function}`). The OMDoc preview window shows you how to use this symbol (Figure 19). This tells

▼ Symbol function space (`\funspace{a_1,...,a_n}{b}`)

Type	$(A : \mathrm{SET}, B : \mathrm{SET}) \rightarrow \mathrm{SET}$	
Notations	**id**	**notation**
	arrowtimes	$a_1 \times ... \times a_n \rightarrow b$
	arrowcurry	$a_1 \rightarrow ... \rightarrow a_n \rightarrow b$
	Arrowtimes	$a_1 \times ... \times a_n \Rightarrow b$
	Arrowcurry	$a_1 \Rightarrow ... \Rightarrow a_n \Rightarrow b$

Figure 19: Syntax Preview

us that if we write `\funspace{a_1,...,a_n}{b}` (depending on which notation we use), we will get $a_1 \times ... \times a_n \rightarrow b$.

We want addition to have type $\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$, hence we do:

```
\symdef{plus}[name=addition,args=2,
  type=\funspace{\Nat,\Nat}{\Nat}
]{#1 \mathbin{\maincomp{+}} #2}
```

> ⚠ So far (and when using the **use** button in the IDE), we have been using the `\usemodule` macro to import content. `\usemodule` is allowed anywhere and imports the referenced module content local to the current TeX group.
> Now that we use imported symbols in types (and since we are *in* a module), we need to make sure that the imported modules are also (transitively) *exported*, since our new symbols now *depend* on the imported module.

> For that we use the `\importmodule` macro within the module; i.e. the file should now look something like this:
> ```
> \begin{smodule}{Nat}
>   \importmodule[sTeX/MathBase/Functions]{mod?Function}
>   ...
> ```

Note that the HTML is aware of this now (after you save): *Clicking* on any occurence of addition now yields Figure 20.
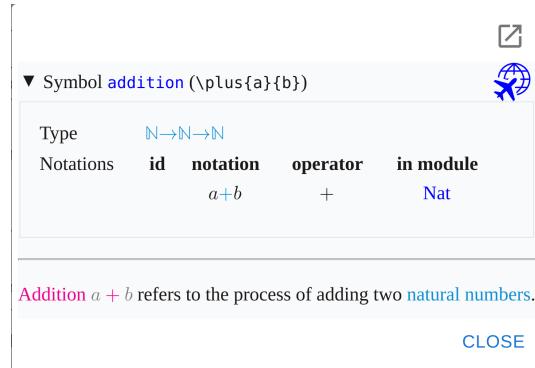
Figure 20: On-Click Popup in the HTML

However, the squiggly yellow `Invalid Unit` warnings are still there – that's because everything we did with types so far still depends on our natural numbers symbol, which does not have a type yet.

By virtue of using `[sTeX/MathBase/Functions]{mod?Function}`, we also imported `[sTeX/MathBase/Sets]{mod?Set}`, which gives us the *"collection"* symbol. Let's use this as a type for the natural numbers:

```
\symdef{Nat}[name=natural numbers,type=\collection]{\mathbb N}
```

Now if we save the file, all the squiggly lines are gone. Moreover, if you look at the OMDoc tab in the preview window, you can find Figure 21. The **Document Elements**

Figure 21: Inferred Type

block collects all semantically annotated expressions in a module or document; including variables and the `$\plus{\va}{\vb}$`. Here, it tells us that it has checked the expression $a + b$ (in the context of $a : \mathbb{N}$ and $b : \mathbb{N}$), and inferred that it has type $\mathbb{N}$.

Here's what just happened:

1. The Mmt system realized, that `$\plus{\va}{\vb}$` is the symbol "addition" applied to the two arguments a and b.

2. It knows, that "addition" has type $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$[2].

3. It knows, that this means that if the two arguments a and b both have type $\mathbb{N}$, then the full expression has type $\mathbb{N}$.

Here's something you can now try: If we *remove* the types from the variables a and b again, the warnings are *still* gone. We lose the type information on hover, but Mmt still doesn't complain, because it now realizes that since a and b have no explicit types given, it should infer them. And by the same chain of reasoning as above, it can infer that since they are being used as arguments for addition, they need to have type $\mathbb{N}$.

3.1.3   Flexary Macros and Argument Modes

Here is one thing you might wonder: Writing `$\plus{a}{b}$` is one thing, but what if we want to produce $a + b + c + d + e$? Do we really need to write `$\plus{a}{\plus{b}{\plus{c}{...}}}$`?

Of course not. We can declare the symbol such that the semantic macro `\plus` expects a (comma-separated) *sequence* of arguments instead of two "normal" arguments.

ːⓈTEX

The optional `args`-argument of `\symdecl` expects a string of characters indicating the semantic macro's **argument modes**. There are four such modes:

- **i** a **simple argument**,
- **a** a – *(left or right) associative* – **sequence argument**, represented as a single TEX-argument `{a,b,...}`,
- **b** A **binding argument** that expects a variable that is bound by the symbol in its application, and
- **B** A **binding sequence argument** of arbitrarily many bound variables by the symbol (`{x,y,z,...}`).

If `args` is given as a number n instead, the semantic macro takes n arguments of mode **i**.

Example 8

- For `\plus{a,b,c}` yielding $a + b + c$, we do `\symdecl{plus}[args=a]`,

- for `\inset{a,b,c}{A}` yielding $a, b, c \in A$, we do `\symdecl{inset}[args=ai]`,

- in `\add{i}{1}{n}{f(i)}` yielding $\sum_{i=1}^{n} f(i)$, the variable i is bound in the expression, we hence do `\symdecl{add}[args=biii]`,

---

[2]Do not worry that the IDE actually reports the type $\{a : \mathbb{N}, b : \mathbb{N}\}_I \mathbb{N}$, this is an artefact of the underlying type system with dependent types used by ςTEX; it just means $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$ in this special case, but would also allow a and b to appear in the range type in more complex situations; see **??** for details.

- in `\foral{x,y,z}{P(x,y,z)}` yielding $\forall x, y, z.\ P(x, y, z)$, the variables x, y, z are all bound by the $\forall$, we hence do `\symdecl{foral}[args=Bii]`.

So when we wrote `\symdecl{plus}[args=2]`, this was actually shorthand for `\symdecl{plus}[args=ii]`.

Let's revise our previous declaration and the syntax of the `\plus` macro:

```
\symdef{plus}[name=addition,args=a,
  type=\funspace{\Nat,\Nat}{\Nat}
]{#1 \mathbin{\maincomp{+}} #2}
\begin{sparagraph}[style=symdoc]
  \vardef{va}[name=a]{a}\vardef{vb}[name=b]{b}
  \Definame{addition} $\defnotation{\plus{\va,\vb}}$
  refers to the process of adding two \sn{Nat}.
\end{sparagraph}
```

Now we get new errors, that are easy to explain: Our notation `{#1 \mathbin{\maincomp{+}} #2}` refers to *two* arguments, but our semantic macro only takes *one* (albeit a sequence argument). We now need to let sTeX know what to do with the sequence argument in our notation. Using the `\argsep` macro, we can tell sTeX to insert the *separator* "+" between the individual elements of the argument sequence #1:

```
\symdef{plus}[name=addition,args=a,
  type=\funspace{\Nat,\Nat}{\Nat}
]{\argsep{#1}{\mathbin{\maincomp{+}}}}
```

Now we can finally write `$\plus{a,b,c,d,e}$` and get $a + b + c + d + e$ – hooray!

...expect that our squiggly yellow `Invalid Unit` warnings are back. That's because the type of addition still corresponds to a binary operation, rather than a unary function on sequences.

We *could* change the type of course, but we shouldn't *want* to or *have* to: platonically, addition is *still* a *binary function*; we just introduced the `a`-mode argument for *our* convenience as authors.

Instead, we can tell Mmt how to "resolve" the sequence argument into a nested application of addition. In the very common case we have here, where the symbol represents an *associative binary operator*, we can just add the argument `assoc=bin` to the `\symdecl` (or `\symdef`) macro:

```
\symdef{plus}[name=addition,args=a,assoc=bin,
  type=\funspace{\Nat,\Nat}{\Nat}
]{\argsep{#1}{\mathbin{\maincomp{+}}}}
```

and the warnings are gone again. Formally/internally, Mmt will now turn the term `addition(sequence(a,b,c))` into `addition(a,addition(b,c))`.

Exercise

Analogously to the above, implement a symbol "`multiplication`" with semantic macro `\mult`, that takes a single sequence argument and has a default notation such that `\mult{a,b,c}` produces $a \cdot b \cdot c$.

Solution: Can be found in `[sTeX/MathTutorial]mod/Nat.en.tex`

3.1.4 Precedences

If you have done the previous exercise, you now have semantic macros `\plus` and `\mult` at your disposal. We can of course nest them to produce e.g. $a + b \cdot c$ (with `$\plus{a,\mult{b,c}}$`). If we do `$\mult{a,\plus{b,c}}$` however, we get $a \cdot b + c$. Annoying – we now have to insert parentheses: `$\mult{a,(\plus{b,c})}$`... or do we?

We do *not*. Instead, we can assign *precedences* to notations to have sTeX insert parentheses automatically.

> sTeX
>
> `\notation` (and hence `\symdef`) take an optional argument `prec=<opprec>;<argprec1>x...x<argprec n>` consisting of an **operator precedence** `<opprec>` and for each argument `k` an **argument precedence** `<argprec k>`.
>
> All precedences are integers, e.g. 10 or -500. It is good practice to use precedences that leave enough room to smuggle values inbetween, so that we can fine-tune them later as more symbols may intervene.
>
> The precise numbers used for precedences are arbitrary – what matters is which precedence is higher than which other precedence when used together.
>
> By default, all precedences are 0, unless the symbol takes no arguments, in which case the operator precedence is `\neginfprec` (negative infinity).
>
> If we only provide a single number in `prec=`, this is taken as both the operator precedence and all argument precedences.

The *lower* a precedence, the *stronger* a notation binds its arguments. In our particular case, we want multiplication to bind stronger than addition, so we can (arbitrarily) assign them precedences e.g. 10 and 20:

```
\symdef{plus}[name=addition,args=a,assoc=bin,prec=20,
  type=\funspace{\Nat,\Nat}{\Nat}
]{\argsep{#1}{\mathbin{\maincomp{+}}}}
\symdef{mult}[name=multiplication,args=a,assoc=bin,prec=10,
  type=\funspace{\Nat,\Nat}{\Nat}
]{\argsep{#1}{\mathbin{\maincomp{\cdot}}}}
```

And now if we type `$\mult{a,\plus{b,c}}$`, sTeX will automatically insert parentheses and yield $a \cdot (b + c)$ – and conversely, if we do `$\plus{a,\mult{b,c}}$`, sTeX will *not* insert parentheses and yield $a + b \cdot c$.

3.1.5 Implicit Arguments

Let us turn our attention back to equality. Here's an almost philosophical question: *What is the type of "equality"?* Asking (the right kind of) mathematicians this question can cause fist fights to break out. As such, we will not give a definitive answer, *but* here is an informative approach that has proven to be quite effective in computational settings:

Equality is a *polymorphic binary relation* on an *implicit* collection A. And a *relation* is a function into a type of *propositions*.

We will see the advantage of this approach over time. For now, consider that given objects a and b, the expression "$a = b$" is either true or false[3], and "equal" takes two argu-

---

[3] Assuming classical logic – if you prefer to remain intuitionistic/constructive, note that sTeX, being foundation independent, does not enforce the law of excluded middle!

ments, so if we have a type of "truth values", it makes sense to model "equal" as a function taking two arguments and returning that type. So we do type=`\funspace`{.....?

Here's the idea with respect to *implicit arguments.* Let's first declare a new variable of type "collection":

```
\vardef{vA}[name=a,type=\collection]{A}
```

We now assign the type $A \times A \to$ `Prop` to equal:

```
\symdef{eq}[name=equal,args=2,eq,
  type=\funspace{\vA,\vA}{\prop}
]{#1 \mathrel{\maincomp{=}} #2}
```

(The symbol "proposition" with semantic macro `\prop` comes with sTEX directly; we say that it is part of the sTEX.)

Now our type has a free variable A. For MMT, this now means that equal actually takes *one more argument,* but one whose value is uniquely determined from the other arguments. Indeed, if you consider equal to take three arguments (the first one being some A of type collection), then the *next* two arguments *enforce* that the first argument has to be the type of the other two.

In other words: A is now an implicit argument that MMT is tasked with inferring whenever we use equal, and that we never explicitly provide in sTEX.

Indeed, if we use our module `Nat` from before, and apply `\eq` to a variable of type $\mathbb{N}$, MMT does not complain:

```
\usemodule{mod?Nat}
\vardef{vn}[name=n,type=\Nat]{n}
$\eq{\vn}{m}$
```

And if we inspect the OMDoc tab in the HTML preview, we can see exactly what MMT did (Figure 22). We can see

Figure 22: Implicit Arguments

1. (by the $\{\cdot\}_I \dots$) that MMT considers A an implicit argument in the type of equal,

41

2. that the *inferred* type of $n = m$ is `Prop`,

3. that MMT inferred the implicit argument of equal in $n = m$ to be $\mathbb{N}$ (by the $\underbrace{\ldots}_{\mathbb{N}}$),

   and

4. that it was enough to give `\vn` the explicit type $\mathbb{N}$ – MMT also inferred that hence m also has to have type $\mathbb{N}$!

3.1.6 Finishing Equality

You might wonder if – as with addition – we can make "equal" take a sequence argument as well. Naturally, we can:

```
1    \symdef{eq}[name=equal,args=a,eq,
2      type=\funspace{\vA,\vA}{\prop}
3    ]{\argsep{#1}{\mathrel{\maincomp=}}}
4    \notation{eq}[equiv]{\argsep{#1}{\mathrel{\maincomp\equiv}}}
```

and as before, we now get `Invalid Unit` warnings. Unlike before, however, we can not just fix this with adding `assoc=bin`. As mentioned, `bin` instructs MMT to "fold" the symbol over the arguments, so when doing `\eq{a,b,c}`, MMT would turn this into `equal(a,equal(b,c))`, i.e. the claim that *"a" is equal to "b = c"* – but that's not what $a = b = c$ means. What we mean by $a = b = c$ is really *"a = b and b = c"*.

For that, we can use `assoc=conj` – however, that requires that some symbol that can be used for *conjunction* (i.e. "and") is in the current scope.

If we search for `conjunction` in the IDE, we should find the module `[sTeX/Logic/General]{mod/syntax?Conjunction}`.

Using that, we can now write the following:

```
\usemodule{mod?Nat}
\usemodule[sTeX/Logic/General]{mod/syntax?Conjunction}
\vardef{vn}[name=n,type=\Nat]{n}
$\eq{\vn,m,p}$
```

Upon saving, MMT does not complain; and if we inspect the OMDOC tab in the HTML window again, we now notice that MMT correctly resolved this as in Figure 23.

3.1.7 Variable Sequences

There is a special kind of variable in sTeX for when we want to use *sequences* of variables.

We can use the `\varseq` macro to declare a new sequence variable; in the simplest case that looks something like the following:

```
\varseq{seqn}[name=n,type=\Nat]{1,\ellipses,k}{\maincomp{n}_{#1}}
```

We have just declared a new variable sequence of type $\mathbb{N}$, that ranges over indices $1, \ldots, k$, with notation n_i for some specific index i.

If we now do `\seqn{i}`, we get n_i, and if we do `\seqn!`, we get $n_1, \ldots, n_k$.

We can also do multi-dimensional sequences, e.g.

```
\varseq{seqm}[name=m,type=\Nat,args=2]
  {{1}{1},\ellipses,{\ell}{k}}
  {\maincomp{m}_{#1}^{#2}}
```

Figure 23: Conjunction of Equalities

Now `\seqm{i}{j}` produces m_i^j, and `\seqm!` produces $m_1^1, \ldots, m_\ell^k$.

Of course, we can manually change the way `\seqn!` is typeset by providing an explicit operator notation using `op=`; e.g. if we do

```
\varseq{seqn}[name=n,type=\Nat,op={(n_i)_{i=1}^k}]
  {1,\ellipses,k}{\maincomp{n}_{#1}}
```

then `\seqn!` produces $(n_i)_{i=1}^k$.

So far so nice, but sequence variables get especially useful in combination with sequence arguments: Consider for example the `\plus` semantic macro for addition. This expects one sequence argument, or alternatively, a *sequence variable*: `\plus{\seqn}` now produces $n_1 + \ldots + n_k$, and `\eq{\seqm}` now produces $m_1^1 = \ldots = m_\ell^k$.

TODO[4]

3.2 Statements

Now that we have equality, natural numbers, addition and multiplication at our disposal, let's implement some *statements*. Both addition and multiplication are, for example, *associative* and *commutative*.

We could state these properties directly for the two operations, but we can also first define *associativity* and *commutativity* in general, and then assert them specifically for addition and multiplication.

3.2.1 Definitions

Let's define what it means to be *associative*. This means, of course, declaring a new symbol. Note that we don't need a semantic macro for associativity, since there is no notation to attach to it. We will also for now ignore its type. Note however, that associativity is still a property of (binary) operations, so it still makes sense to have the symbol take an *argument*; namely the operation it applies to.

---

[4]TODO: seqmap

43

We will also finally provide an actual (more or less) formal *definition* for the symbol, so where we used the `sparagraph` environment with `style=symdoc` before, we will now use the `sdefinition` environment, which also gives us `\definame`, `\definiendum`, `\defnotation` and all that.

A first variant of a corresponding module could look like this:

Example 9
Input:

File [sTeX/MathTutorial]**props/Associative1.en.tex**

```
4  \begin{smodule}{Associative}
5    \importmodule{mod?Equality}
6
7    \symdecl*{associative}[args=1]
8    \begin{sdefinition}[for=associative]
9      \vardef{vA}[name=A,type=\collection]{A}
10     \vardef{vop}[name=op,type=\funspace{\vA,\vA}\vA,args=a,assoc=bin]
11       {\argsep{#1}{\mathbin{\maincomp{\circ}}}}
12     %
13     A binary operation $\fun{\vop!}{\vA,\vA}\vA$ is called
14     \definame{associative}, if
15     $\eq{
16       \vop{(\vop{a,b}),c},
17       \vop{a,(\vop{b,c})}
18     }$ for all $\inset{a,b,c}\vA$.
19   \end{sdefinition}
20 \end{smodule}
```

Output:

> **Definition 3.2.1.** A binary operation $\circ : A \times A \to A$ is called **associative**, if $(a \circ b) \circ c = a \circ (b \circ c)$ for all $a, b, c \in A$.

Note, that the semantic macros `\fun` and `\inset` come from `[sTeX/MathBase/Functions]mod?Function` and `[sTeX/MathBase/Sets]mod?Set`, respectively. Also note, that the variable declaration for `\vop` makes use of all the fun features we already discussed for addition.

> Note that the above is more than good enough, if you merely want to produce nice-looking, "wikified" HTML and PDF documents. The rest of this subsection will cover how to add more flexiformal semantics to the above.
>
> If this seems laborious and/or difficult, keep in mind that this is to some degree experimental still, and you are not forced to go overboard with semantic annotations!
>
> But if you aim to create a "library of symbols" for mathematical concepts, then all of the possibilities that we discuss here will add value for the community. Generally, the higher the ratio of readers to authors the more any investment in semantization will pay off.

Semantic Macros in Text Mode

The first thing we can do to further improve this document is marking up the "for all" in the definition – after all, there naturally is a symbol for the universal quantifier, which can be found in `[sTeX/Logic/General]mod/syntax?UniversalQuantifier` and has the semantic macro `\foral` (as to not conflict with the T_EX primitive macro `\forall`).

The naive approach would be to replace the "for all" by e.g. `\sr{foral}{for all}`. That would (correctly) associate and highlight the text fragment with the symbol "universal quantifier", *but* we are not just referencing the symbol here – we are actually using it, by *applying* it to the variables a, b, c and the expression $(a \circ b) \circ c = a \circ (b \circ c)$.

In *math mode*, we can just use the semantic macro `\foral` – that will take two arguments (of modes B and i) and produce the corresponding notation, so that

```
$\foral{\inset{a,b,c}{\vA}}{
  \eq{ \vop{(\vop{a,b}),c} , \vop{a,(\vop{b,c})} }
}$
```

will produce $\forall a, b, c \in A.(a \circ b) \circ c = a \circ (b \circ c)$.

In *text mode*, however, we don't have a specific notation – instead, the specific "notation" is whatever sentence we want to mark up semantically. In text mode, semantic macros therefore behave differently:

1. They take *precisely* one argument, regardless of how many arguments the macro would take in math mode or (equivalently) the **args** property of the symbol.

2. *Within* that argument, we can use `\comp` to highlight arbitrary text fragments, and

3. we can use the `\arg` macro to mark up the *actual* arguments that the symbol is supposed to be applied to.

   `\arg` takes as optional argument the index of the argument that is being marked up; if not they are used consecutively. The starred variant `\arg*` produces no output.

So we could now do

```
\foral{\comp{For all} $\arg{\inset{a,b,c}{\vA}}$, we have
  $\arg{
    \eq{ \vop{(\vop{a,b}),c} , \vop{a,(\vop{b,c})}}
  }$
}
```

which produces "For all $a, b, c \in A$, we have $(a \circ b) \circ c = a \circ (b \circ c)$".

In our case though, we want to "switch the arguments around" – first comes the equation, then the variables to be bound. Hence:

```
\foral{
  $\arg[2]{
    \eq{ \vop{(\vop{a,b}),c}, \vop{a,(\vop{b,c})} }
  }$
  \comp{for all}
  $\arg[1]{ \inset{a,b,c}{\vA} }$
}
```

which produces "$(a \circ b) \circ c = a \circ (b \circ c)$ for all $a, b, c \in A$".

Definientia

Now we have a fully semantically annotated expression in the definition for "associative". Can we let Mᴍᴛ know, that this expression really is *the* definition of the symbol?

Yes, we can. All we need to do is wrap the sentence in a `\definiens` macro (plural: *definientia*; like the word *"definiendum"* refers to "the term being defined", *"definiens"* refers to "the thing the term is being defined *as*").

The `\definiens` macro is only allowed within the `sdefinition` environment, and requires that the environment lists the symbol that gets the definiens attached explicitly in its `for=` argument. It is possible to attach definientia to multiple symbols within an `sdefinition` environment, in which case the symbol needs to be provided as an optional argument, e.g. we could do `\definiens[associative]{...}`. Since "associative" is the only symbol being defined in our definition, we can omit that argument.

Alternatively, as with types we can attach definientia to a `\symdecl` directly using the optional argument `def=...`.

At this point, you might justifiably wonder, why we even still need to declare associative with `\symdecl*` before we define it. And indeed, we don't – the `sdefinition` environment takes the same optional arguments as the `\symdecl` macro, and if we explicitly provide a `name=` (or a `macro=`), it will generate a symbol for us. We can hence get rid of the `\symdecl*` and instead do:

```
1  \begin{sdefinition}[name=associative,args=1]
2    ...
3  \end{sdefinition}
```

One more problem remains: We stated that associative is to take one argument – but we haven't told sTₑX what it is yet. In our case, the argument is represented by the variable `\vop`. In fact, chances are that arguments to symbols in types or definientia are almost always represented by some variable.

We can use one of two ways to a variable as being an argument:

1. If the variable (e.g. `\vop` with name `op`) was already declared prior to the `sdefinition` environment, we can use the `\varbind` macro in the environment; e.g. by adding `\varbind{op}`.

2. We can move (or copy) the `\vardef` for the variable into the environment and add `bind` to its optional arguments.

In total, our fully annotated definition now looks like this:

Example 10
Input:

File [sTeX/MathTutorial]**props/Associative.en.tex**

```
 8   \begin{sdefinition}[name=associative,args=1]
 9     \vardef{vA}[name=A,type=\collection]{A}
10     \vardef{vop}[name=op,type=\funspace{\vA,\vA}\vA,
11       args=a,assoc=bin,bind % <- argument for the symbol
12     ]{\argsep{#1}{\mathbin{\maincomp{\circ}}}}
13     \vardef{va}[name=a,type=\vA]{a}
14     \vardef{vb}[name=b,type=\vA]{b}
15     \vardef{vc}[name=c,type=\vA]{c}
16     %
17     A binary operation $\fun{\vop!}{\vA,\vA}\vA$ is called
18     \definame{associative}, if
19     \definiens{\foral{$\arg[2]{\eq{
20       \vop{(\vop{\va,\vb}),\vc},
21       \vop{\va,(\vop{\vb,\vc})}
22     }}$ \comp{for all} $\arg[1]{\inset{\va,\vb,\vc}\vA}$}}.
23   \end{sdefinition}
24 %
```

Output:

> **Definition 3.2.2.** A binary operation $\circ : A \times A \to A$ is called **associative**, if $(a \circ b) \circ c = a \circ (b \circ c)$ for all $a, b, c \in A$.

And indeed, if we look at the OMDOC tab of the HTML preview, we can see that not only does MMT attach the definiens to the symbol, it has also inferred the type of "associative" from the definiens (Figure 24).

▼ Symbol `associative`

Definiens $\{A : \text{SET}\}_I (\circ : A \to A \to A) \to \forall a : A, b : A, c : A. \left(\underbrace{(a \circ b) \circ c = a \circ (b \circ c)}_{A} \right)$

Type $\{A : \text{SET}\}_I (\circ : A \to A \to A) \to \text{Prop}$

Figure 24: Type Inferred from Definiens

Using Symbols Without Semantic Macros and Exporting Code in Modules

So now we don't have a semantic macro for "associative", but it *does* take an argument. How can we ever actually *use* the symbol now?

The answer is: with the `\symuse` macro. Like `\symref` and friends, `\symuse` takes a symbol name or the name of its semantic macro as argument, but behaves otherwise like using a semantic macro directly. So for, say, addition, `\symuse{addition}` and `\symuse{plus}` behave exactly like `\plus`.

In our case, this means we can do `\symuse{associative}`. "associative" does not have a notation, but in practice, we say something like *"+ is associative"* rather than using some specific mathematical notation for the same thing.

Combining this with what we just learned, we can now say that addition is associative by doing:

```
\symuse{associative}{$\arg{\plus!}$} \comp{is associative}}
```

In fact, we would do the exact same thing every time we want to say that *some* operator is associative, so it makes sense to introduce a macro for this. In fact, such a macro is easy to define using standard LATEX methods. This is where \STEXexport becomes very handy:

In a module, we can put arbitrary LATEX code in an \STEXexport, and this code will be executed every time the module is imported via \usemodule or \importmodule. This is especially useful for macro definitions, and this way modules can almost act like LATEX packages!

So we can define a new macro \isassociative that applies "associative" to an arbitrary operation and produces the semantically marked-up text "#1 is associative", and wrap that macro definition in an \STEXexport, and whenever we use the Associative module, we also get the \isassociative macro:

```
\STEXexport{
  \def\isassociative#1{
    \symuse{associative}{\arg{#1} ~is ~\comp{associative}}
  }
}
```

And now, we can do e.g. \isassociative{$\plus!$} to produce "$+$ is associative".

For technical reasons, \STEXexport processes its content in the expl3 category code scheme – what this means is that all spaces are ignored entirely, and the characters _ and : are valid characters in macro names.
In practice, this means you will have to use the ~ character for spaces, and if you want to use a subscript _, you should use the macro \c_math_subscript_token instead.

Exercise
Analogously to all the above, implement a module for *commutativity*; i.e the property of a binary operation that $a \circ b = b \circ a$ for all a, b. Make the module export a macro \iscommutative analogously to \isassociative.

Solution: Can be found in **[sTeX/MathTutorial]props/Commutative.en.tex**

TODO[5]

3.2.2 Assertions

Having defined associativity and commutativity, we can now assert that both properties hold for addition and multiplication.

For *assertions* (i.e. theorems, lemmata, axioms, claims,...), sTEX provides the sassertion environment.

In the simplest case, that can look like the following:

```
\begin{sassertion}
  \isassociative{\Sn{plus}}
\end{sassertion}
```

---

[5]TODO: intent?

which yields

> Addition is associative

Do we want this to be typeset as a **Theorem**? For that we just add a `[style=theorem]` to the `sassertion` environment, provided we have a customization for that – (see chapter 9). We can also load the stexthm package, which uses the amsthm package to provide common typesettings for the types: `theorem`, `observation`, `corollary`, `lemma`, `axiom` and `remark`.

So far, this is not too useful – after all, we could have just as well used e.g. the amsthm package and gone straight for the non-sTeX variant

```
\begin{theorem}
  \isassociative{\Sn{plus}}
\end{theorem}
```

But as with `sdefinition`, we can immediately add a corresponding symbol in the `sassertion` environment, and have it be documented directly by the environment:

```
\begin{sassertion}[style=theorem,name=addition is associative]
  \isassociative{\Sn{plus}}
\end{sassertion}
```

And now, if we do `\sn{addition is associative}`, we get addition is associative with a corresponding hover pop-up (in the HTML).

Of course, the usefulness of this grows with more elaborate assertions. For very short assertions such as the above, we might not even want to typeset them in such a space hungry manner.

For that purpose, we provide the `\inlineass` macro (and analogously: `\inlinedef` for `sdefinition`), which takes the same optional arguments as the environment. So we could also do:

```
\inlineass[name=addition is associative]{\isassociative{\Sn{plus}}}
```

So far, MmT is blissfully unaware of the semantic contents of our assertions. We can easily remedy that by wrapping the expression representing the assertion in a `\conclusion` macro, analogously to the definiens macro in `sdefinition`s:

```
\inlineass[name=addition is associative]{
  \conclusion{\isassociative{\Sn{plus}}}
}
```

We can now see the statement in the OMDoc tab of the HTML preview (Figure 25).

$$\triangleright\text{Assertion } \texttt{addition is associative} \vdash \text{ apply } \left(\text{apply } (\underbrace{\text{associative} \mathbb{N}}_{\mathbb{N}}) + \right)$$

Figure 25: Assertion Statement in OMDoc

Not exactly pretty – the OMDoc tab uses notations to render content, and we did not provide any for associative.

Notice the $\vdash$ symbol after the name of the assertion? As an aside for those who are curious:

The **judgments as types** paradigm represents the validity of proposition via a designated *type of proofs*: For any proposition P, we introduce a collection $\vdash P$ of *proofs* of P.

To say that the proposition *holds* is then equivalent to positing that *some* element $p :\vdash P$ exists – in which case *proofs* become typed objects in their own right.

Let's consider a more interesting statement now. How about the induction axiom?

```
\begin{sassertion}[style=axiom,name=induction axiom]
  Let $\varphi(n)$ a property on \sn{Nat}. If
  \begin{enumerate}
    \item $\varphi(0)$ and
    \item if $\varphi(m)$ holds for some $m$, then
    $\varphi(\plus{m,1})$ also holds,
  \end{enumerate}
  then $\varphi(n)$ holds for all $\inset{n}{\Nat}$.
\end{sassertion}
```

Axiom 3.2.3. *Let $\varphi(n)$ a property on natural numbers. If*

1. *$\varphi(0)$ and*

2. *if $\varphi(m)$ holds for some m, then $\varphi(m+1)$ also holds,*

then $\varphi(n)$ holds for all $n \in \mathbb{N}$.

Exercise

Annotate the above by:

1. Variables with appropriate notations for φ, m and n, and

2. marking up the second premise ("if $\varphi(m)$ holds for some...") in text mode as the formula $\forall m.\varphi(m) \Rightarrow \varphi(m+1)$ using the semantic macros `\foral` (which we saw earlier already) and `\imply` (implication) from `[sTeX/Logic/General]mod/syntax?Implication`. The text fragments that should be highlighted are "`if`" and "`then`".

3. marking up the conclusion ("$\varphi(n)$ holds for all $n \in \mathbb{N}$") in text mode as the formula $\forall n.\varphi(n)$. The text fragment that should be highlighted is "`for all`".

Solution: Can be found in `[sTeX/MathTutorial]mod/NatTheorems.en.tex`

So how can we teach MMT the semantics of this statement? Here's what we can do:

1. As with the simpler assertions (and hence the name), the *conclusion* of the assertion can be marked up with `\conclusion`.

2. As with `sdefinition`, we can mark variables as *bound* (using either `bind` in the `\vardef` or `\varbind`). *If* a symbol that can act as a universal quantifier is in scope, variables marked as bound are abstracted away using that symbol.

3. Similarly to `\conclusion`, *premises* can be marked up as such using the `\premise` macro. If a symbol is in scope that can act as an implication, that will be used to connect the premise(s) to the conclusion.

Hence, if we mark the variable φ as bound and use `\premise` and `\conclusion` (see `[sTeX/MathTutorial]mod/NatTheorems.en.tex`), we can inspect the OMDoc tab in the HTML preview again and see that MMT has now constructed the proposition (Figure 26).

▷ Assertion `induction axiom` ⊢ $\forall \varphi\colon \mathbb{N}{\to}\mathsf{Prop}.\varphi(0){\Rightarrow}\big(\,\forall m\colon \mathbb{N}.\varphi(m){\Rightarrow}\varphi(m{+}1)\,\big)\Rightarrow\big(\,\forall n\colon \mathbb{N}.\varphi(n)\,\big)$

Figure 26: The Induction Axiom in OMDoc

3.2.3 Proofs

> sTeX provides the `sproof` environment for marking up *proofs*.
> The markup mechanism for `sproof` is still highly experimental and likely subject to change in the near future. As such, we omit a closer explanation of its usage until the syntax and functionality have sufficiently stabilized.

3.3 Mathematical Structures

A common concept in mathematics is that of a mathematical structure – a *tuple* of interdependent components. For example: A *monoid* is a structure $\langle M, \circ, e \rangle$ such that certain axioms hold; where M is a set, $\circ$ is a binary operation, and $e \in M$.

From a representational perspective, this is particularly interesting: M, $\circ$ and e in the above are not symbols in the same way that the previous symbols we considered were – they don't represent definite objects. Instead, they are *components* of some other object, namely a monoid; where a *particular* monoid could either be a fixed object (such as $\langle \mathbb{Z}, +, 0 \rangle$) or an *indefinite* monoid; i.e. a variable. We call the components of a mathematical structure **fields**.

In this section, we will discuss how to declare and use mathematical structures in sTeX, build them up modularly, and connect them among each other to avoid duplication.

We will do so by considering *lattices* both algebraically and order-theoretically, and identify the two perspectives.

3.3.1 Declaring and Using Structures

The simplest kinds of structures are *magmas* and *(directed) graphs*, so we might as well start there:

Definition 3.3.1. A **magma** is a structure $\langle U, \circ \rangle$, where U is a collection and $\circ$ a binary operation $U \times U \to U$.

The obvious start is to create a new module `Magma`. Within this module, we import the `Functions` module so we can later assign a type to the operation. We can then use the `mathstructure` environment, that creates a new symbol "magma":

```
\begin{smodule}{Magma}
  \importmodule[sTeX/MathBase/Functions]{mod?Function}
  \begin{mathstructure}{magma}
    ...
  \end{mathstructure}
\end{smodule}
```

mathstructure behaves very similarly as smodule – within the environment, we can declare new symbols, notations and all that.

So within the mathstructure, we can add symbols for the two fields U and $\circ$:

```
\symdef{univ}[name=universe,type=\collection]{U}
\symdef{op}[name=operation,args=a,assoc=bin,
  type=\funspace{\univ,\univ}\univ
]{\argsep{#1}{\mathbin{\maincomp{\circ}}}}
```

Once we close the environment (with `\end{mathstructure}`), the symbols are "gone". However, we now have a new symbol "magma" with semantic macro `\magma`. It's usage is somewhat more complicated than "normal" semantic macros, but one thing we *can* do with it now is `$\magma!$`, which will produce $\langle U, \circ \rangle$.

Notably however, the `\magma` macro is already available *within* the mathstructure environment as well.

This allows us to provide an sdefinition using the semantic macros declared in the structure:

Example 11

Input:

```
                              File [sTeX/MathTutorial]algebra/Magma.en.tex

 7  \begin{mathstructure}{magma}
 8    \symdef{univ}[name=universe,type=\collection]{U}
 9    \symdef{op}[name=operation,args=a,assoc=bin,
10      type=\funspace{\univ,\univ}\univ]
11      {\argsep{#1}{\mathbin{\maincomp\circ}}}
12
13    \begin{sdefinition}[for={magma,univ,op}]
14      A \definame{magma} is a \sr{mathstruct}{structure} $\magma!$,
15      where $\univ$ is a \sn{collection} and $\op!$
16      a binary operation $\funspace{\univ,\univ}\univ$.
17    \end{sdefinition}
18  \end{mathstructure}
```

Output:

> **Definition 3.3.2.** A **magma** is a structure $\langle U, \circ \rangle$, where U is a collection and $\circ$ a binary operation $U \times U \to U$.

Instantiating Structures

More importantly however, we can now declare a variable magma, using the optional `return=` argument. For example, we can now do

```
\vardef{vM}[name=M,return=\magma]{M}
```

and we get the semantic macro `\vM` with which we can do the following:

Syntax	Result
`$\vM!$`	M
`$\vM{}$`	$\langle U_M, \circ_M \rangle$
`$\vM{univ}$`	U_M
`$\vM{op}!$`	$\circ_M$
`$\vM{op}{a,b,c}$`	$a \circ_M b \circ_M c$

In other words: Given a symbol or variable with semantic macro `\foo` and `return=\struct`, then `\foo{<fn>}` behaves like the semantic macro `\fn` *within* the `mathstructure` environment for `struct` – but instantiated for the specific instance `foo`.

By default, sTEX attaches the symbol's (or variable's) operator notation as a subscript suffix to the notation component marked with `\maincomp` – e.g., since the "`\circ`" in the notation for `op` is marked with `\maincomp`, doing `$\vM{op}{a,b}$` ultimately outputs `a \circ_{\vM!} b`. Hence, we get $a \circ_M b$.

We can change the way the `\maincomp` notation component is modified, by using the optional argument `copm=` in the semantic macro for the mathematical structure. For example, to not change it at all, we can do:

```
\vardef{vM}[name=M,return={\magma[comp=##1]}]{M}
```

...or to suffix it with a ', we can do

```
\vardef{vMp}[name=Mp,return={\magma[comp=##1']}]{M'}
```

This allows us to do things like:

```
Let $\vM!:=\vM{}$ and $\vMp!:=\vMp{}$ \sns{magma}. Then...
```

yielding

Let $M := \langle U, \circ \rangle$ and $M' := \langle U', \circ' \rangle$ magmas. Then...

We can also *assign* fields to (arbitrary) expressions, by doing `name=<tex>` in square brackets. For example we can do the following:

```
\vardef{vA}[type=\collection]{A}
\vardef{vM}[name=M,return={\magma[comp=##1][univ=\vA]}]{M}
\vardef{vMp}[name=Mp,return={\magma[comp={{##1}'}][univ=\vA]}]{M'}

Let $\vM!:=\vM{}$ and $\vMp!:=\vMp{}$ \sns{magma} on $\vA$....
```

Let $M := \langle A, \circ \rangle$ and $M' := \langle A, \circ' \rangle$ magmas.

Of course, we can also use `return=` with variable sequences – for example:

```
\varseq{vMs}[name=M,return={\magma[comp={##1}_{#1}]},op=(M_i)_1^n]
  {1,\ellipses,n}{\maincomp{M}_{#1}}
Let $\vMs! := \vMs{i}{}_1^n$ a sequence of \sns{magma}...
```

Let $(M_i)_1^n := \langle U_i, \circ_i \rangle_1^n$ a sequence of magmas...

Note that in the above, it seems that using `#1` in the `return` argument is allowed. Indeed, it is - the `return` statement takes the same arguments as the semantic macro itself does and is appropriately instantiated. Since the first (and only) argument to the

sequence `\vMs` is the index, when doing `\vMs{i}`... the `#1` in the `return`-statement will be replaced by `i`.

Also, note that if we want to produce M_i – i.e. the magma at index i in the sequence, we can do `\vMs{i}!`.

> ⚠ Think of the `!` as a "stop sign" - if the expression up to the `!` has an associated presentation, the `!` tells sTeX to "stop eating arguments" and present whatever it has until now.

3.3.2 Extending Structures and Axioms

It is extremely common to "build up" structures in a hierarchical manner by adding new fields or axioms: A *semigroup* is an associative magma. A *band* is an idempotent semigroup. A *monoid* is a semigroup with a unit. A *partial order* is an antisymmetric preorder.

We alluded to the fact earlier, that the `mathstructure` environment behaves like an `smodule` – that is literally true: Every `mathstructure foo` in a module `FooMod` is in fact also a module `?FooMod/foo-module`. We can therefore easily extend structures using `\importmodule{...?FooMod/foo-module}` – but extending structures is so common, and using `\importmodule` tiring, that there is a shortcut: the `extstructure` environment. It takes as second argument a comma-separated list of structure names. That allows us to easily define semigroups:

> **Example 12**
> Input:
>
> File [sTeX/MathTutorial]`algebra/Semigroup.en.tex`
>
> ```
>  8  \begin{extstructure}{semigroup}{magma}
>  9    \begin{sdefinition}
> 10      A \definame{semigroup} is a \sn{magma} $\semigroup!$,
> 11      where \inlineass[name=associative axiom]{
> 12        \conclusion{\isassociative{$\op!$}}.
> 13      }
> 14    \end{sdefinition}
> 15  \end{extstructure}
> ```
>
> Output:
>
> > **Definition 3.3.3.** A **semigroup** is a magma $\langle U, \circ \rangle$, where $\circ$ is associative.

Note our usage of `\inlineass` to generate a new symbol for the associative axiom.

If we look at the OMDoc tab in the HTML preview window, we can see the output in Figure 27.

So MMT has decided that our statement is an *axiom*.

Conservative Extensions

For structures, there is a *critical* distinction between *defined* and *undefined* symbols; and analogously between *theorems* and *axioms*.
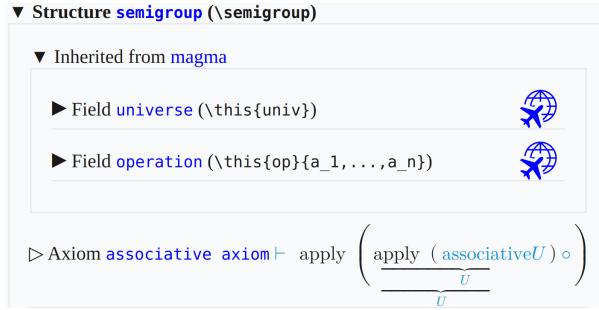
Figure 27: Axioms in OMDOC

Remember that structures are more like *templates* that are *instantiated* by particular objects. An *undefined* field in a structure, in that sense, is like an *obligation*: If something is supposed to be a semigroup, it *has to* have a universe, an operation and the operation needs to satisfy the associative axiom.

Defined fields on the other hand have a *definiens* on the basis of the remaining fields – they don't need to be explicitly provided for something to instantiate the structure; if all the *undefined* fields are provided, the *defined* ones we get *"for free"*.

The same holds for *theorems*: If a statement is *provable* from the axioms, then we don't need to explicitly prove it to hold for some particular instance – we have a proof already, provided the axioms hold.

> The relation between axioms and theorems is not just analogous to that between undefined and defined symbols: It is the very same. Remember the judgments as types paradigm?
>
> For a proposition P, an assertion in sTEX induces a symbol of type $\vdash P$. Without a proof, this symbol is *undefined* – and hence an *axiom*. A *proof* for P is a specific term of type $\vdash P$ – i.e. a potential *definiens*. To prove an assertion turns it into a *theorem*, which is to say that the symbol can be *defined*.

One consequence of this is: Extending a structure only by *defined* fields does not actually (conceptually) introduce a *new* structure – every instance of the old one *should* also be an instance of the new one. The new fields are basically just "syntactic sugar".

There is a name for extending a structure only by defined fields (or theorems): A *conservative extension*.

sTEX provides the `extstructure*` environment for that purpose. Unlike `extstructure`, it does *not* take a name (technically, sTEX generates one internally). Instead, conceptually `extstructure*` modifies the extended structure directly, rather than generating a new structure. The caveat however is, that every symbol introduced in an `extstructure*` **must** be defined.

Consider the following conservative extension:

Example 13
Input:

55

```
                    File [sTeX/MathTutorial]algebra/MagmaSquare.en.tex
7    \begin{extstructure*}{magma}
8      \begin{sdefinition}[macro=sq,args=1]
9        \notation{sq}[op=\cdot^2]{{#1}^{\comp 2}}
10       \vardef{va}[name=a,type=\univ,bind]{a}
11       Let $\inset{\va}{\univ}$. We define
12       $\defnotation{\sq{\va}} := \definiens{\op{\va,\va}}$.
13     \end{sdefinition}
14   \end{extstructure*}
```

Output:

Definition 3.3.4. Let $a \in U$. We define $a^2 := a \circ a$.

Via `\definiens`, the new symbol sq is now *defined* (note the `macro=` argument, taht generates a semantic macro as well). Whenever we import the containing module, we now have an additional field `sq` in (any extension of) magma – e.g., the following is now valid:

```
\usemodule[sTeX/MathTutorial]{algebra?MagmaSquare}
\vardef{vsg}[name=S,return=\semigroup]{S}
$\vsg{sq}{a}$
```

...producing a^2.

3.3.3   Nesting Structures and \this

A prehaps not too surprising, but a notable aspect of structures is that fields themselves can be instances. This is important for example for implementing *vector spaces*, but can also be used to bundle things that are not normally thought of as structures, such as objects with certain defining properties.

Take as an example, the notion of a (magma) homomorphism:

Definition 3.3.5. Let $M_1 = \langle U_1, \circ_1 \rangle$ and $M_2 = \langle U_2, \circ_2 \rangle$ magmas. A **magma homomorphism** is a function $F : U_1 \to U_2$ such that $F(a \circ_1 b) = F(a) \circ_2 F(b)$ for all $a, b \in U_1$.

So a homomorphism is a function with certain properties. And structures can be used to "bundle" the function itself with both the magmas on whose universes the function operates, as well as the *axiom* that *makes* it a homomorphism. After all, considered as a mere function, $F : U_1 \to U_2$ contains no information about the operation with respect to which it is homomorphic.

The first thing to note is that we can provide `mathstructure` with an optional argument for a *name* distict from the name of its semantic macro. We then add two fields that `return` magmas. So far, so unexciting:

```
\begin{mathstructure}{magmahom}[magma homomorphism]
   \symdef{dom}[name=domain,return={\magma[comp={##1}_1]}]{M_1}
   \symdef{cod}[name=codomain,return={\magma[comp={##1}_2]}]{M_2}
```

For the function itself, we know how to give it a maningful type, already:

```
\symdef{f}[type=\funspace{\dom{univ}}{\cod{univ}},args=1]{???}
```

...but what should its notation be? Ideally we would want it to just be the notation of whatever particular instance it is – in informal mathematics, we rarely distinguish notationally between a homomorphism and its underlying function (to the point where it's not clear, whether *informally* the distinction is even meaningful). Similarly, we rarely distinguish e.g. between a magma (or semigroup, monoid, group, ring, vector space,...) and its underlying universe.

This is where `\this` comes into play (pun intended). Within an `mathstructure` or `exstructure`, or in the context of a particular instance of one, `\this` represents "the" instance.

We can set it in the context of `mathstructure` as a further optional argument; e.g.

```
\begin{mathstructure}{magmahom}[magma homomorphism,this=F]
```

and then use `\this` in the notation for the function. We can further provide the homomorphism condition as an axiom using `\inlineass`:

Example 14

Input:

```
                           File [sTeX/MathTutorial]algebra/Homomorphism.en.tex
 9   \begin{mathstructure}{magmahom}[magma homomorphism,this=F]
10     \symdef{dom}[name=domain,return={\magma[comp={##1}_1]}]{M_1}
11     \symdef{cod}[name=codomain,return={\magma[comp={##1}_2]}]{M_2}
12     \symdef{f}[op=\this,args=1,
13       type=\funspace{\dom{univ}}{\cod{univ}}
14     ]{\this \dobrackets{#1}}
15
16     \begin{sdefinition}[for={magmahom,dom,cod,f}]
17       \vardef{va}[name=a,type=\dom{univ}]{a}
18       \vardef{vb}[name=b,type=\dom{univ}]{b}
19       Let $\dom!=\dom{}$ and $\cod!=\cod{}$ \sns{magma}.
20       A \definame{magmahom} is a function
21       $\fun{\f!}{\dom{univ}}{\cod{univ}}$ such that
22       \inlineass[name=homomorphism condition]{\conclusion{\foral{
23         $\arg[2]{\eq{
24           \f{\dom{op}{\va,\vb}}, \cod{op}{\f{\va},\f{\vb}}
25         }}$ \comp{for all} $\arg[1]{\inset{\va,\vb}{\dom{univ}}}$.
26       }}}
27     \end{sdefinition}
28   \end{mathstructure}
```

Output:

Definition 3.3.6. Let $M_1 = \langle U_1, \circ_1 \rangle$ and $M_2 = \langle U_2, \circ_2 \rangle$ magmas. A **magma homomorphism** is a function $F : U_1 \to U_2$ such that $F(a \circ_1 b) = F(a) \circ_2 F(b)$ for all $a, b \in U_1$.

Now if we instantiate our magma homomorphism:

```
\vardef{vh}[name=H,return={\magmahom[this=H]}]{H}
```

Here is a list of what we can do now:

Syntax	Result
`$\vh!$`	H
`$\vh{}$`	$\langle M_1, M_2, H \rangle$
`$\vh{f}!$`	H
`$\vh{f}{a}$`	$H(a)$
`$\vh{dom}!$`	M_1
`$\vh{cod}{}$`	$\langle U_2, \circ_2 \rangle$
`$\vh{cod}{univ}$`	U_2
`$\vh{dom}{op}!$`	$\circ_1$
`$\vh{cod}{op}{a,b,c}$`	$a \circ_2 b \circ_2 c$

Note how – as one would expect – we can treat `\vh{dom}` and `\vh{cod}` like any other instance of magma.

Note that some of the outputs in the above table are probably not quite what we want. Determining the precise typesetting of an expression involving *nested paths* of fields is difficult, to say the least (e.g., what exactly should `\this` refer to in a deeply nested sequence of fields?).

Using instances within structures is still very useful; at the very least when defining structures. When subsequently *using* structures, however, accessing fields of fields (of fields (of ...)) of an instance should be avoided.

Luckily, there is rarely a need for doing so – in practice, those fields we might want to access in such a way, we usually also want to provide specific notations and talk about independently of the "containing" instance, such that introducing a new variable (or symbol), and assigning the corresponding field to that variable, makes considerably more sense. And subsequently using the variable is easier than concatenating `{...}`, too.

3.4   Complex Inheritance and Theory Morphisms

We are starting to approach seriously experimental territory.

While the theory behind all the following is relatively well understood, and their implementation in Mmt is mature, the same can not be said out the implementation in sTeX.

There are still kinks to be ironed out, but feel free to experiment.

We now have all the tools available to progress towards something more interesting. Here is a list of documents with respective modules and symbols we will build on in the following:

[sTeX/MathTutorial]props/Idempotent.en.tex

Definition 3.4.1. Let $e \in A$ and $\circ : A \times A \to A$. e is called **idempotent** with respect to $\circ$, if $e \circ e = e$.

Definition 3.4.2. The operation $\circ : A \times A \to A$ is called **idempotent**, if every element $a \in A$ is idempotent with respect to $\circ$.

Definition 3.4.3. Let $\odot : B \times A \to A$ and $\oplus : A \times A \to A$. We say $\odot$ **distributes over** $\oplus$, if $b \odot (a_1 \oplus a_2) = (b \odot a_1) \oplus (b \odot a_2)$ for all $a_1, a_2 \in A$ and $b \in B$.

Definition 3.4.4. Let $\odot : A \times B \to A$ and $\oplus : A \times B \to B$. We say $\odot$ **absorbs** $\oplus$, if $a_1 \odot (a_1 \oplus b) = a_1$ for all $a_1 \in A$ and $b \in B$.

Definition 3.4.5. A **band** is an idempotent semigroup.

Definition 3.4.6. A **semilattice** is a commutative band.

Definition 3.4.7. A binary relation R on A is called **reflexive**, if $R(a, a)$ for all $a \in A$.

Definition 3.4.8. A binary relation R on A is called **symmetric**, if $R(a, b)$ implies $R(b, a)$ for all $a, b \in A$.

Definition 3.4.9. A binary relation R on A is called **transitive**, if $R(a, b)$ and $R(b, c)$ implies $R(a, c)$ for all $a, b, c \in A$.

Definition 3.4.10. A binary relation R on A is called **antisymmetric**, if $R(a, b)$ and $R(b, a)$ implies $a = b$ for all $a, b \in A$.

Definition 3.4.11. A **directed graph** is a structure $\langle U, R \rangle$, where U is a collection and R a binary relation on U.

Definition 3.4.12. An **(undirected) graph** is a directed graph $\langle U, R \rangle$, where R is symmetric.

```
[sTeX/MathTutorial]orders/Preorder.en.tex
```

Definition 3.4.13. A structure $\langle U, \leq \rangle$ is called a **preorder** (or **quasiorder**, or **preordered set**; in short **proset**), if $\leq$ is reflexive and transitive.

```
[sTeX/MathTutorial]orders/Poset.en.tex
```

Definition 3.4.14. A preorder $\langle U, \leq \rangle$ is called a **partial order** (or **poset**), if $\leq$ is antisymmetric.

```
[sTeX/MathTutorial]orders/InfSup.en.tex
```

Definition 3.4.15. Let $\langle U, \leq \rangle$ a poset. An element $a \in U$ is called an **infimum** or **greatest lower bound** of x_1 and x_2, if $a \leq x_1$, $a \leq x_2$, and for any x with $x \leq x_1$ and $x \leq x_2$, we have $x \leq a$.

Definition 3.4.16. Let $\langle U, \leq \rangle$ a poset. An element $a \in U$ is called a **supremum** or **least upper bound** of x_1 and x_2, if $x_1 \leq a$, $x_2 \leq a$, and for any x with $x_1 \leq x$ and $x_2 \leq x$, we have $a \leq x$.

Note that infima and suprema are more generally defined on *sets* of elements. Doing so in sTeX is significantly more complicated *for now*, and will require some amount of research to make convenient – especially if we want to subsequently define *operators* on pairs of elements, as below. We therefore opt for the simpler version where it is defined as binary from the get go.

```
[sTeX/MathTutorial]orders/MeetJoinSemilattice.en.tex
```

Definition 3.4.17. A poset $\langle U, \leq \rangle$ is called a **meet semilattice** if for every two elements a, b the infimum $a \wedge b$ exists.

Definition 3.4.18. A poset $\langle U, \leq \rangle$ is called a **join semilattice** if for every two elements a, b the supremum $a \vee b$ exists.

Definition 3.4.19. An **(order) semilattice** is a meet and join semilattice.

> **Exercise**
> Try to implement all of the above yourself!

3.4.1 Glueing Structures Together

We now want to progress towards lattices, i.e. the following:

Definition 3.4.20. A lattice is a structure $\langle U, \wedge, \vee \rangle$ such that $\langle U, \wedge \rangle$ and $\langle U, \vee \rangle$

are semilattices, and ∨ absorbs ∧ and vice versa; i.e. $a \vee (a \wedge b) = a$ and $a \wedge (a \vee b) = a$. The operations ∧ and ∨ are called **meet** and **join**, respectively.

So we make a new module, open an `extstructure` environment and... realize two problems:

1. We can't just extend semilattice: We need *two* copies of semilattice that share a universe, and importing semilattice twice is of course redundant.

2. We also want to *rename* the operations of the two semilattices to be subsequently called `join` and `meet`.

What we need is a way to *inherit* from semilattice while a) *modifying* the symbols therein, and b) not be idempotent – i.e. two imports from the same structure or module should not be identified. We can do that with the `\copymod` macro, which takes three arguments:

1. A *name* for the copy,

2. the structure or module to copy, and

3. a comma-separated list of renamings and redefinitions of the symbol. ⟨*symbol*⟩=⟨*def*⟩ redefines ⟨*symbol*⟩, ⟨*symbol*⟩@⟨*newname*⟩ renames it, ⟨*symbol*⟩=⟨*def*⟩@⟨*newname*⟩ (or ⟨*symbol*⟩@⟨*newname*⟩=⟨*def*⟩) does both.

In our case, we want two copies of semilattice, which we will call `meetsl` and `joinsl`. In the first copy, we only want to rename `op` to `meet`. In the second, we want to rename `op` to `join`, and *also* redefine the universe to be the one from `meetsl`:

```
\copymod{meetsl}{semilattice}{
  op @ meet
}
\copymod{joinsl}{semilattice}{
  univ = \univ,
  op @ join
}
```

You might have already noticed some problem with that – which of the two universes does `\univ` refer to now? (They are *defined* as equal, but LaTeX does not know that!) Or which of the two `commutative axiom`s does "`commutative axiom`" refer to now? Everything is ambiguous now!

Not really - if you have wondered why the `\copymod` takes a *name* as argument: The name is prefixed to every symbol name. Hence, the `universe` in `joinsl` is now called `joinsl/universe`, and the one in `meetsl` is called `meetsl/universe`. Furthermore, `\copymod` by default generates no semantic macros for any of the imported symbols – except for those renamed with @. In fact, what the @ syntax actually does, is to generated a semantic macro by that name. If we want to change the *name* (that is shown when using `\symname` et al), we add that new name in square brackets. Hence, what we really want to do is:

```
\copymod{meetsl}{semilattice}{
  univ @ univ,
  op @ [meet]meet
}
\copymod{joinsl}{semilattice}{
  univ = \univ,
  op @ [join]join
}
```

This now gives us two copies of semilattice, generates semantic macros `\univ` for `meetsl/universe`, `\meet` for `meetsl/op` and `\join` for `joinsl/op`, and renames `meetsl/op` to `meet` and `joinsl/op` to `join`.

That allows us to then add the absorption axioms, an `sdefinition` for lattice and subsequently `$\lattice!$` produces $\langle U, \wedge, \vee \rangle$, with all axioms inherited (see `[sTeX/MathTutorial]algebra/Lattice.en.tex`).

3.4.2 Realizations

A very common situation we find in connection with mathematical structures is that "every *this* is a *that*" (or the conrete case "*this* is a *that*").

With what we did so far, we are in this situation regarding the algebraic definition of semilattices and the order-theoretic one (exemplary meet semilattice).

In Mmt parlance, this corresponds to a total (implicit) theory morphism from "that" to "this".

In sTeX words, we want to inherit from "that" by assigning all the symbols in "that" to concrete terms. In our case:

[sTeX/MathTutorial]algebra/SemiLatticeOrder.en.tex

Definition 3.4.21. Let $\langle U, \circ \rangle$ a semilattice. We let $a \leq b$ iff $a \circ b = a$.

Theorem 3.4.22. $\langle U, \leq \rangle$ *is a meet semilattice.*

Proof: We need to prove the following

reflexivity $a \leq a$: We need to show $a \circ a = a$. Follows from the idempotent axiom.

antisymmetry $a \leq b$ and $b \leq a$ implies $a = b$: Assume $a \circ b = a$ and $b \circ a = b = a \circ b$ (by the commutative axiom). Hence, $a = b$

transitivity If $a \leq b$ and $b \leq c$, then $a \leq c$. : Assume $a \circ b = a$ and $b \circ c = b$ Then $a \circ c = (a \circ b) \circ c = a \circ (b \circ c) = a \circ b = a$. Hence, $a \leq c$.

$a \circ b$ is the infimum of $\{a, b\}$: By definition (and the commutative axiom), $a \circ b \leq a$ and $a \circ b \leq b$. We need to show, that if $x \leq a$ and $x \leq b$, then $x \leq a \circ b$. Assume $x \circ a = x$ and $x \circ b = x$. Then $x \circ (a \circ b) = (x \circ a) \circ b = x \circ b = x$. Hence $x \leq a \circ b$

So to be precise, we want to provide *definientia* for all undefined symbols in meet semilattice (i.e. the relation and meet) and *proofs* for all *axioms* (reflexive axiom, antisymmetric axiom, transitive axiom, and infimum axiom), and by so obtain the fact that every semilattice is a meet semilattice.

For that purpose, we have the `\realize` macro. It behaves like `\copymod`, but does not take a name, and additionally requires that all undefined fields get assigned. So we could do the following:

Example 15
Input:

```
                    File [sTeX/MathTutorial]algebra/SemiLatticeOrder1.en.tex
 8    \begin{extstructure*}{semilattice}
 9      \realize{meetsl}{
10         univ = \univ,
11         meet = \op!,
12         rel @ [order]order = \map{a,b}{\eq{\op{a,b},a}},
13         reflexive axiom = trivial,
14         transitive axiom = trivial,
15         antisymmetric axiom = trivial,
16         infimum axiom = trivial
17      }
18    \end{extstructure*}
19
20    \vardef{mysl}[return=\semilattice]{S}
21    $\mysl{order}{a,b} \qquad \mysl{}[univ,op,order]$
```

Output:

$$a \leq_S b \qquad \langle U_S, \circ_S, \leq_S \rangle$$

As we can see, we can now access the field `order`, which is renamed from `relation` in meet semilattice and also has the desired definiens in Mmt. But of course this approach is very "declarative": We do all the assigning in one macro, rather than narratively as what they *should* be: definitions and proofs.

If we want to achieve the more narrative version at the beginning of the subsection, we can use the `realization` environment instead. It behaves like the `\realize` macro, but allows us to do the assignments and renamings individuall somewhere in the body of the environment, interleaved with arbitrary text. Additionally, within the environment, all sTeX features that introduce *definientia* (like the `\definiens` macro) induce assignments instead.

To declaratively rename or assign fields, we can then use the `\assign` and `\renamedecl` macros instead. That allows us to do the following instead:

```
\begin{realization}{meetsl}
  \assign{univ}{\univ}
  \assign{meet}{\op!}
  \renamedecl{rel}[order]{order}
  ...
```

...and then use text to do the remaining assignments. For example, we can use the `sdefinition` environment to assign `rel` to the desired definiens:

```
\usestructure{meetsl}
\begin{sdefinition}[for=order]
  \varbind{va,vb}
  Let $\semilattice![univ,op]$ a \sn{semilattice}.
  We let $\rel{\va,\vb}$
  iff $\definiens{\eq{\op{\va,\vb},\va}}$.
\end{sdefinition}
```

And now sTeX will use the `\definiens` to assign $a, b \mapsto a \circ b = a$ to the relation of meet semilattice.

Analogously, we can use the `sproof` and `subproof` environments to produce "definientia" (i.e. proofs) for the axioms (see `[sTeX/MathTutorial]algebra/SemiLatticeOrder.en.tex`)

Chapter 4

Extensions for Education

The last two chapters have shown generic markup and semantization facilities in sTeX. As said before, investments in semantic markup pay off, iff the impact of a document is high, e.g. if there are many more readers than authors or if the semantic services afforded by the semantic markup can help reduce the help readers need to understand the material.

Educational documents constitute one category of high-impact documents which are supported by the sTeX ecosystem, we will cover them here.

4.1 Slides and Course Notes

TODO[6]

4.2 Problems and Exercises

TODO[7]

4.3 Exams

TODO[8]

---

[6]TODO: notesslides.sty
[7]TODO: problem.sty
[8]TODO: hwexam.sty

Part II
User Manual

Chapter 5

Basics

5.1 Package and Class Options

- `debug=`⟨*prefixes*⟩: (see Developer Manual)

- `lang=`⟨*languages*⟩: If set, sTeX will load the `babel` package with the provided languages. Supported languages (currently) are:

`ar`	arabic
`bg`	bulgarian
`de`	german (with option `ngerman`)
`en`	english
`fi`	finnish
`fr`	french
`ro`	romanian
`ru`	russian
`tr`	turkish (with option `shorthands=:!`)

- `mathhub=`⟨*path*⟩: Uses the provided file path as `MathHub` directory (see section 5.2).

- `usesms`/`writesms`: If `writesms` is set, content loaded from external math archives (i.e. modules) is persisted in the file `\jobname.sms`.

  If `usesms` is set, the content of the `.sms`-file is loaded, obviating the need to reprocess the original files.

  The options are not mutually exclusive, but care should be taken if dependencies have changed between builds.

  This offers two advantages:

  1. If a document has many (transitive) dependencies, `usesms` should significantly speed up the build process, and

  2. setting `usesms` allows for distributing the `.sms`-file to make the document *standalone*, allowing for compilation without needing imported/used modules to be present.

The options `debug`, `mathhub`, `usesms` and `writesms` can also be set by the environment variables `STEX_DEBUG`, `MATHHUB`, `STEX_USESMS` and `STEX_WRITESMS`. In fact, the `MATHHUB` environment variable is the recommended way to set the `MathHub` directory. This is the only option where the *package option* overrides the environment variable.

The environment variables for `USE`/`WRITESMS` are particularly useful, in that they allow for convenient compilation workflows. For example, the `Build PDF/XHTML/OMDoc`-button in the IDE does the following:

```
STEX_WRITESMS=true pdflatex <job>.tex
[bibtex|biber] <job>
STEX_USESMS=true pdflatex <job>.tex
STEX_USESMS=true pdflatex <job>.tex
```

Guaranteeing (in the first run) that all dependencies are loaded from their respective sources and persisted, and in the two subsequent runs read from the generated `.sms` file, (likely) speeding ub the subsequent runs significantly.

5.2 Math Archives and the `MathHub` Directory

sTeX uses math archives to organize document content modularly, without a user having to specify absolute paths, which would differ between users and machines.

All sTeX archives need to exist in the local `MathHub`-directory. sTeX knows where this folder is via one of five 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. If that too fails, sTeX will look for a file `~/.stex/mathhub.path`. If this file exists, sTeX will assume that it contains the path to the local `MathHub`-directory. This method is recommended on systems where it is difficult to set environment variables, and is used by the IDE setup.

5. Finally, if all else fails, sTeX considers `~/MathHub` to be the `MathHub` directory.

The sTeX IDE allows you to directly download math archives from `gl.mathhub.info` – currently available archives there are:

- `sTeX/*` – a group of semi-experimental documents showcasing sTeX3.3 features,

- `smglom/*` – a vast collection of multilingual modules of concepts in mathematics and computer science. The SMGloM predates sTeX3 and is thus largely "underannotated" with respect to (formal) semantics,

- `MiKoMH/*` – a vast collection of lecture slides and notes in computer science for courses held by Michael Kohlhase. They largely make use of SMGloM modules.

5.2.1 The Structure of Math Archives

An archive `group/name` is stored in the directory `<MathHub>/group/name`; e.g. assuming your local `MathHub`-directory is set as `/user/foo/MathHub`, then in order for the `sTeX/Documentation` archive to be found by the sTeX system, it needs to be in `/user/foo/MathHub/sTeX/Documentation`.

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 discussed in section 5.3.

5.2.2 MANIFEST.MF-Files

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

```
id: sTeX/Documentation
ns: http://mathhub.info/sTeX/Documentation
narration-base: http://mathhub.info/sTeX/Documentation
format: stex
title: The sTeX Documentation
teaser: The full Documentation for the sTeX system
url-base: https://stexmmt.mathhub.info/:sTeX
dependencies:sTeX/ComputerScience/Software,sTeX/MathTutorial
ignore: */code/*|*/tikz/*|*/tutorial/solution/*
```

Many of these are in fact ignored by sTeX, but some are important:

id: The name of the archive, including its group (e.g. `sTeX/Document`). This is used by the MMT system in favor of the directory, but TeX's limited access to the file system enforces the directory structure.

source-base or

ns: The namespace from which all symbol and module MMT-URIs in this archive are formed.

narration-base: The namespace from which all document MMT-URI in this repository are formed. It can safely match the `ns`-field.

url-base: A URL that is formed as a basis for *external references*. and hyperlinks. An MMT (or comparable system) instance should run there and host (sTeX-generated) HTML.

dependencies: All archives that this archive depends on. sTeX ignores this field, but MMT can pick up on them to resolve dependencies, e.g. when downloading archives in the IDE, which will also download all dependencies.

ignore: A regular expression of `.tex` files in the `source` directory that should be ignored; e.g. they will not be compiled when building a whole directory or archive in the IDE.

5.3 The `lib`-Directory

A math archive `group/archive` may have a `lib` directory primarily intended for preamble code, packages, `.bib` files, etc., the files in which can be referenced in various ways.

Additionally, a *group* of archives `group` may have an additional archive `group/meta-inf`. If this `meta-inf` archive has a `/lib`-subdirectory, they too will be considered by the following.

\libinput   `\libinput {some/file}` searches for a file `some/file[.tex]` in

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

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

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

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

`\libinput[some/archive]{some/file}` will do the same, but starting in the `lib` directory of the math archive `some/archive`.

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

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

\addmhbibresource   `\addmhbibresource [some/archive]{some/file}` searches for a file like `some/file.bib` in `some/archive`'s `lib` directory and calls `\addbibresource` to the result.

\libusetikzlibrary   `\libusetikzlibrary` behaves like `\libusepackage` but looks specifically for tikz libraries and calls `\usetikzlibrary` on the results.

throws an error if not *exactly one* candidate for for the library is found.

A good practice is to have individual sTeX fragments follow basically this document frame:

```
\documentclass{stex}
\libinput{preamble}
\setsectionlevel{<your preference>}
\begin{document}
    \IfInputref{}{
      ...
      \maketitle
      \ifstexhtml \else \tableofcontents \fi
    }
    ...
    \IfInputref{}{\libinput{postamble}}
\end{document}
```

Then the `preamble.tex` files can take care of loading the generally required packages, setting presentation customizations etc. (per archive or archive group or both), and a `postamble.tex` can e.g. print the bibliography, index etc.

`\libusepackage` is particularly useful in such a `preamble.tex` when we want to use custom packages that are not part of a TeX distribution, or on CTAN. In this case we commit the respective packages in one of the `lib` folders and use `\libusepackage` to load them.

5.4   Basic Macros

`\sTeX`
`\stex`
The `\sTeX` macro produces this sTeX logo. It is provided by the stex-logo package, included with the stex package.

`\ifstexhtml`
The TeX conditional `\ifstexhtml` is *true* if the current compilation generates HTML, and *false* otherwise (i.e. generates PDF).

`\STEXinvisible`

`\STEXinvisible{⟨code⟩}`

Processes ⟨code⟩, but does not generate any output. In the HTML, ⟨code⟩ is exported with `display:none`.

Can be used to declare formal content and preserve its semantics in HTML without generating output.

Chapter 6

Document Features

6.1 Document Fragments

sfragment (*env.*)   To make reusability of document fragments more feasible, sTEX provides the **sfragment** environment. `\begin{sfragment}[id=⟨id⟩,short=⟨short title⟩]{section title}` calls `\part, \chapter, \section, \subsection, \subsubsection, \paragraph` or `\subparagraph` with argument `{section title}` depending on the current *section level* and availability, and increases the level accordingly.

The ⟨*id*⟩ can be used for cross-document references (see section 6.3).

blindfragment (*env.*)   In the case where we want to increase the section level *without* producing a corresponding section header, the **blindfragment** environment can be used. This allows e.g. typesetting `\section`s before the first `\chapter`.

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

\setsectionlevel   The `\setsectionlevel` macro sets the current section level to that provided as argument. This is particularly useful in the preamble of a document, as to be ignored in e.g. `\inputref` and make sure that sectioning proceeds as desired; e.g. `\setsectionlevel{section}` make sure that the first **sfragment** will be typeset as a `\section` (rather than e.g. a `\part`).

\currentsectionlevel
\Currentsectionlevel   The `\currentsectionlevel` macro produces the literal string corresponding to the current section level – e.g. within a chapter (but outside of a section), `\currentsectionlevel` produces "chapter".

The `\Currentsectionlevel` macro does the same, but capitalizes the first letter; e.g. in the above situation, `\Currentsectionlevel` produces "Chapter".

6.2 Using and Referencing Document Fragments

`\inputref`

`\inputref [`⟨*archive*⟩`]{`⟨*file*⟩`}`

Inputs the file ⟨*file*⟩ in ⟨*archive*⟩'s `source` directory. If [⟨*archive*⟩] is empty, the current archive's `source` directory is used. If there is no current archive, ⟨*file*⟩ is resolved relative to the current file.

The file's content is processed within a TeX group when using `pdflatex`. When converting to HTML however, the file is not processed *at all*, and instead, a reference to the file is inserted, that can be replaced by the HTML generated by the referenced file by e.g. the MMT system.

This is the recommended method to assemble documents from individual `.tex` files.

`\mhinput`

Like `\inputref`, but actually calls `\input` in both PDF and HTML mode. Useful for small fragments or those without modules, but generally `\inputref` should be preferred.

`\ifinputref`
`\IfInputref`

`\ifinputref` is a TeX conditional for whether the current file is currently processed via `\inputref`.

`\IfInputref {`⟨*true code*⟩`}{`⟨*false code*⟩`}` behaves like `\ifinputref`⟨*true code*⟩`\else`⟨*false code*⟩`\fi` when using `pdflatex`; in HTML mode however, *both* arguments are processed and marked-up accordingly, so a hosting server (like MMT) can dynamically decide which parts to show or omit.

`\mhgraphics`
`\cmhgraphics`

If the `graphicx` package is loaded, `\mhgraphics` takes the same arguments as `\includegraphics`, with the additional optional key `archive`. It then resolves the file path in `\mhgraphics[archive=some/archive]{some/image}` relative to the `source`-folder of the `some/archive` archive. If no `archive` is provided, the file path `some/image` is resolved relative to the current archive (if existent).

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

`\lstinputmhlisting`
`\clstinputmhlisting`

Like `\mhgraphics`, but for `\lstinputlisting` instead of `\includegraphics`. Only defined if the `listings` package is loaded.

6.3 Cross-Document References

If we take features like `\inputref` and `\mhinput` (and the `sfragment` environment) seriously and build large documents modularly from individually compiling documents for sections, chapters and so on, cross-referencing becomes an interesting problem.

Say, we have a document `main.tex`, which `\inputref`s a section `section1.tex`, which references a definition with label `some_definition` in `section2.tex` (subsequently also `\inputref`ed in `main.tex`). Then the numbering of the definition will depend on the *document context* in which the document fragment `section2.tex` occurs - in `section2.tex` itself (as a standalone document), it might be *Definition 1*, in `main.tex`

it might be *Definition 3.1*, and in `section1.tex`, the definition *does not even occur*, so it needs to be referenced by some other text.

What we would want in that instance is an equivalent of `\autoref`, that takes the document context into account to yield something like *Definition 1*, *Definition 3.1* or "*Definition 1 in the section on Foo*" respectively.

For that to work, we need to supply (up to) three pieces of information:

- The *label* of the reference target (e.g. `some_definition`),

- (optionally) the *file*/document containing the reference target (e.g. `section2`). This is not strictly necessary if the reference target occurs in the *same* document, but if not, we need to know where to find the label,

- (optionally) the document context, in which we want to refer to the reference target (e.g. `main`).

Additionally, the document in which we want to reference a label needs a title for external references.

`\sref`

`\sref` `[archive=`⟨*archive1*⟩`,file=`⟨*file1*⟩`]`
`{`⟨*label*⟩`}[archive=`⟨*archive2*⟩`,file=`⟨*file2*⟩`,title=`⟨*title*⟩`]`

This macro references ⟨*label*⟩ (declared in ⟨*file1*⟩ in math archive ⟨*archive1*⟩). If the object (section, figure, etc.) with that label occurs (eventually) in the same document, `\sref` will ignore the second set of optional arguments and simply defer to `\autoref` if that command exists, or `\ref` if the hyperref package is not included.

If the referenced object does *not* occur in the current document however, `\sref` will refer to it by the object's name as it occurs in the file ⟨*file2*⟩ in archive ⟨*archive2*⟩, followed by the title.

In HTML mode, the reference additionally links to the HTML of the `file1`.[9]

This works by storing labels during compilation in a file ⟨*jobname*⟩`.sref`, analogous to e.g. the `.toc`. Note that this consequently requires both `file1.tex` and `file2.tex` to have been compiled previously, to generate the `.sref` file.

For example, doing
`\sref[file=tutorial/full.en]{sec:basics}[file=tutorial.en,title=the \stex Tutorial]`
in this very document fragment (`[sTeX/Documentation]macros/sref.en.tex`) will yield Part I (The Basics) in the sTEX Tutorial if compiled itself, or if compiled as part of the sTEX manual, and will yield the `\autoref` link chapter 2 in the documentation (which includes the tutorial).

`\srefsetin`

`\srefsetin` `[`⟨*archive2*⟩`]{`⟨*file2*⟩`}{`⟨*title*⟩`}`

Sets a default value for the optional arguments ⟨*archive2*⟩, ⟨*file2*⟩ and ⟨*title*⟩ of `\sref`. If the second set of optional arguments in `\sref` are omitted, these default values are used. Particularly useful to set in a preamble.

`\sreflabel`

`\sreflabel` `{`⟨*label*⟩`}` sets a label analogous to `\label{`⟨*label*⟩`}`, but for use in `\sref`.

Note that for every sTEX macro or environment that takes an optional `id=`⟨*id*⟩ argument, the ⟨*id*⟩ (if non-empty) generates an `\sreflabel` automatically.

For example, `\begin{sfragment}[id=foo]{Foo}` is equivalent to
`\begin{sfragment}{Foo}\sreflabel{foo}`.

\extref

```
\extref [archive=⟨archive1⟩,file=⟨file1⟩]
{⟨label⟩}{archive=⟨archive2⟩,file=⟨file2⟩,title=⟨title⟩}
```

Like **\sref**, but with the third argument mandatory, **\extref** will *always* produce the output as if ⟨*label*⟩ would *not* occur in the current document.

Chapter 7

Modules and Symbols

7.1 Modules

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

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

smodule (*env.*)   A new module is declared using the basic syntax

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

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

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

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

style (⟨*string*⟩∗) for use in customizations, see chapter 9

id (⟨*string*⟩) for cross-referencing, see `\sreflabel`.

ns (⟨*URI*⟩) the namespace to use. *Should not be used, unless you know precisely what you're doing.* If not explicitly set, is computed from the containing file and archive's namespace.

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

sig (⟨*language*⟩) see below.

\STEXexport   `\STEXexport`{⟨*code*⟩} executes ⟨*code*⟩ immediately and every time the current module is being used.

> For technical reasons, `\STEXexport` processes its content in the `expl3` category code scheme – what this means is that all spaces are ignored entirely, and the characters _ and : are valid characters in macro names.
>
> In practice, this means you will have to use the ~ character for spaces, and if you want to use a subscript _, you should use the macro `\c_math_subscript_token` instead.
>
> Also, note that no *global* macro definitions should happen in `\STEXexport`; this can lead to unexpected behaviour if the containing module has been used previously in the current document.

7.1.1 Signature Modules, Languages, and Multilinguality

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

For example, we can have a file `Foo.en.tex`, that declares and documents a module Foo (using `\begin{smodule}{Foo}`). If we put a file `Foo.de.tex` next to it, we can do `\begin{smodule}[sig=en]{Foo}` to have all the content in the module Foo (as declared in `Foo.en.tex`) available and translate its document content to german.

The MMT backend, when serving sTeX content as HTML, will always attempt to find documentation in the language corresponding to the context; e.g. a user's preference.

7.2 Symbol Declarations

`\symdecl`

`\symdecl` `{⟨mname⟩}[⟨options⟩]`

The `\symdecl` macro is the simplest way to introduce a new symbol. If ⟨*options*⟩ contains `name=⟨name⟩`, then ⟨*name*⟩ is the *name* of the symbol; otherwise, ⟨*mname*⟩ is used for the *name*. Additionally, a semantic macro `\mname` is generated.

The starred variant `\symdecl*` does not generate a semantic macro, in which case the `name`-option is superfluous.

> ⟵M⟶ `\symdecl` introduces a new MMT/OMDOC constant in the current module (i.e. MMT/OMDOC theory). Correspondingly, they get assigned the MMT-URI ⟿T⟿ `<module-URI>?<constant-name>`.

`\symdecl` takes the following optional arguments:

name see above,

args the arity of the symbol and its semantic macro; may be a number 0...9 or a string consisting of the characters `i`, `a`, `b` and `B` of length ≤ 9,

type the symbol's type,

def the symbol's definiens,

return the symbol's *return code* (see below), most commonly the semantic macro of a mathematical structure,

assoc how to resolve arguments of mode a or B; may be `pre`, `bin`, `binl`, `binr` or `conj`,

reorder how to reorder the arguments in OMDOC (*advanced*),

role symbols with certain roles are treated in particular ways in MMT/OMDOC (*advanced*),

argtypes TODO[10].

\textsymdecl

`\textsymdecl{⟨mname⟩}[⟨options⟩]{⟨code⟩}`

Like `\symdecl`, but requires that the symbol has arity 0 (hence `\textsymdecl` does not take the `args`-option), and generates a semantic macro that takes no arguments in either text or math mode, and produces marked-up ⟨*code*⟩ as output.

Additionally, a macro \⟨*mname*⟩`name` is generated that produces ⟨*code*⟩ without any semantic markup.

\symdef

`\symdef{⟨mname⟩}[⟨options⟩]{⟨notation⟩}`

Combines the functionalities and optional arguments of `\symdecl` and `\notation` in one.

7.2.1 Returns

Assume we have a symbol `foo` with semantic macro `\foo`, (examplary) taking two arguments, and `return=`⟨*code*⟩. If we do `\foo{a}{b}!`, the return code is simply ignored. If we do `\foo{a}{b}` *without* the `!`, here is what happens:

1. STEX will replace `#1` and `#2` in ⟨*code*⟩ by `a` and `b`, yielding ⟨*retcode*⟩.

2. STEX will insert ⟨*retcode*⟩`{\foo{a}{b}!}` in the input token stream.

This means that ⟨*code*⟩ should contain at most ⟨*arity of foo*⟩ argument markers, and eat precisely one argument appended to ⟨*code*⟩.

When in doubt, we recommend only using semantic macros for mathematical structures (with only optional arguments) and `\apply` (with only optional arguments) in `return`.

7.3 Referencing Symbols

\symref
\sr

`\symref{⟨symbol⟩}{⟨text⟩}`

The `\symref` macro (and its short version `\sr`) is the most general variant to mark-up arbitrary LATEX code ⟨*text*⟩ with the symbol.

---

[10]TODO: experimental

This is as good a place as any other to explain how sTeX resolves a string `symbol` to an actual symbol.

If `\symbol` is a semantic macro, then sTeX has no trouble resolving `symbolname` to the full MMT-URI of the symbol that is being invoked.

However, especially in `\symname` (or if a symbol was introduced using `\symdecl*` without generating a semantic macro), we might prefer to use the *name* of a symbol directly for readability – e.g. we would want to write `A \symname{natural number} is...` rather than `A \symname{Nat} is...`. sTeX attempts to handle this case thusly:

If `symbol` does *not* correspond to a semantic macro `\symbol` and does *not* contain a `?`, then sTeX checks all symbols currently in scope until it finds one, whose name is `symbol`. If `symbol` is of the form `pre?name`, sTeX first looks through all modules currently in scope, whose full MMT-URI ends with `pre`, and then looks for a symbol with name `name` in those. This allows for disambiguating more precisely, e.g. by saying `\symname{Integers?addition}` or `\symname{RealNumbers?addition}` in the case where several `addition`s are in scope.

`\symref{⟨symbol⟩}{⟨text⟩}` in MMT/OMDoc generates the term `<OMS name="{⟨symbol URI⟩}"/>`.

`\symname`	`\symname[pre=⟨pre⟩,post=⟨post⟩]{⟨symbol⟩}`
`\sn`	
`\Symname`	If the symbol referenced by ⟨*symbol*⟩ has name `name`, this is a shortcut for
`\Sn`	`\symref{⟨symbol⟩}{⟨pre⟩name⟨post⟩}`.
`\sns`	For example, given a symbol `agroup` with name `abelian group`, we can do
`\Sns`	`\symname[pre=Non-,post=s]{agroup}` to produce Non-abelian groups.

`\sn` is a shorter variant for `\symname`; `\Symname` and `\Sn` additionally capitalize the first letter. `\sns` and `\Sns` are short for `\sn [post=s]` and `\Sn [post=s]`, respectively.

`\srefsym`	`\srefsym{⟨symbol⟩}{⟨text⟩}`
`\srefsymuri`	

turns ⟨*text*⟩ into a link to

- The documentation of ⟨*symbol*⟩, if it occurs in the same document, or

- the symbol's documentation online, based on the containing math archive's `url-base`.

`\srefsymuri` does the same, but expects a symbol's full MMT-URI as first argument. This is particularly useful for e.g. customizing highlighting (see chapter 9).

`\symuse` `\symuse{⟨symbol⟩}` behaves exactly like a semantic macro for ⟨*symbol*⟩.

7.4 Notations and Semantic Macros

`\notation`{⟨*symbol*⟩}[⟨*options*⟩]{⟨*code*⟩}

introduces a new notation for the referenced symbol.

The starred variant `\notation*` sets this notation as the (new) default notation.
The optional arguments are:

- prec=⟨*opprec*⟩;⟨*argprec 1*⟩x...x⟨*argprec n*⟩: An operator precedence and one argument precedence for each argument of the semantic macro. If no argument precedences are given, all argument precedences are equal to the operator precedence. By default, all precedences are 0, unless the symbol takes no argument, in which case the operator precedence is `\neginfprec` (negative infinity).

  prec=nobrackets is an abbreviation for prec=`\neginfprec`;`\infprec` x...x`\infprec`.

- op=⟨*code*⟩: An operator notation. If none is given, the notation component marked with `\maincomp` is used. If no `\maincomp` occurs in the notation, the default operator notation is `\symname`{⟨*symbol*⟩}.

- variant=⟨*id*⟩: An id for this notation. The key variant= can be omitted; i.e. `\notation[foo]` is equivalent to `\notation[variant=foo]`.

`\comp` is used to mark notation components in a `\notation` to be highlighted. Additionally, each notation can use `\maincomp` at most once to mark the *primary* notation component.

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

Note that it is required that

1. the argument markers #n never occur inside a `\comp`, and
2. no semantic macros may ever occur inside a notation.

Both criteria are not just required for technical reasons, but conceptionally meaningful:
The underlying principle is that the arguments to a semantic macro represent *arguments to the mathematical operation* represented by a symbol. For example, a semantic macro application `\plus{a}{b}` would represent *the actual addition of (mathematical objects) a and b*. It should therefore be impossible for *a* or *b* to be part of a notation component of `\plus`.
Similarly, a semantic macro can not conceptually be part of the notation of `\plus`,

since a symbol represents a *distinct (mathematical) concept* with *its own semantics*, and notations are syntactic representations of the very symbol to which the notation belongs.

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

`\setnotation` The first notation provided will stay the default notation unless explicitly changed:

`\setnotation{⟨symbol⟩}{⟨id⟩}` sets the default notation of ⟨symbol⟩ to that with id ⟨id⟩.

7.4.1 Precedences and Bracketing

`\infprec`   `\infprec` and `\neginfprec` represent *infinitely large* and *infinitely small* precedences,
`\neginfprec`   respectively.

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

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

Example 16

Consider semantic macros `\plus` and `\mult` taking two arguments, with notations $a + b$ and $a \cdot b$ respectively, and precedences 100 for `\plus` and 50 for `\mult`.

Consider `$\plus{a,\mult{b,\plus{c,d}}}$` (i.e. $a + b \cdot (c + d)$). Then:

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

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

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

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

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

80

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

7. Next, sTeX encounters the inner `\plus{c,...}` whose notation has $p_{op} = 100$. We compare to the current downward precedence p_d set by `\mult`, arriving at $p_{op} = 100 > 50 = p_d$ – which finally prompts sTeX to insert parentheses, and we proceed as before.

\dobracket   `\dobrackets{`⟨*code*⟩`}` wraps parentheses around `{`⟨*code*⟩`}`.

\withbrackets   `\withbrackets{`⟨*left*⟩`}{`⟨*right*⟩`}{`⟨*code*⟩`}` uses the opening and closing parentheses ⟨*left*⟩ and ⟨*right*⟩ for the next pair of parentheses automatically inserted in `{`⟨*code*⟩`}`.

7.4.2   Notations for Argument Sequences

The following macros can be used in notations that take mode `a` or `B` arguments:

\argsep   `\argsep{`⟨*parameter token*⟩`}{`⟨*separator*⟩`}`

takes the elements of the argument sequence in position ⟨*parameter token*⟩ and separates them by ⟨*separator*⟩.

Note that the first argument *must* be a parameter token of the form `#k`, and the argument at position `k` of the notation has to have argument mode `a` or `B`.

\argmap   `\argmap{`⟨*parameter token*⟩`}{`⟨*code*⟩`}{`⟨*separator*⟩`}`

takes the elements of the argument sequence in position ⟨*parameter token*⟩, applies the code `{`⟨*code*⟩`}` to each of them (which therefore should use `##1`) and separates them by ⟨*separator*⟩.

For example, the notation `{\argmap{#1}{X^{##1}}{++}}` applied to the argument `{a,b,c}` produces $X^a + + X^b + + X^c$.

\argarraymap   TODO[11]

7.4.3   Semantic Macros

Assume we have a semantic macro `\smacro` taking (exemplary) two arguments. The precise behaviour of `\smacro` depends on whether we are in text or math mode.

Math Mode   `\smacro!` produces the default operator notation of its symbol. Without `!`, `\smacro` expects at least two arguments, and `\smacro{a}{b}!` produces the default notation of its symbol.

If the symbol has a `return` code, then `\smacro{a}{b}` continues with executing the `return` code. Otherwise, `\smacro{a}{b}` also simply produces the default operator notation.

The starred variants `\smacro*` and `\smacro!*` behave as in *text mode.*

Text Mode `\smacro!{⟨arg⟩}` marks up ⟨*arg*⟩ similarly to how `\symref{smacro}{⟨arg⟩}` would.

Without the `!`, `\smacro` still only takes a single argument, but it is expected, that within ⟨*arg*⟩, the arguments for the symbol are explicitly marked up. The `\comp` macro is allowed in ⟨*arg*⟩ to determine the components of ⟨*arg*⟩ to be highlighted.

`\arg` The `\arg` macro can be used to explicitly mark the arguments of a semantic macro in text mode.

By default, they are numbered consecutively; e.g. `\smacro{...\arg{a}...\arg{b}}` determines `a` and `b` to be the (first and second) arguments.

The starred variant `\arg*` allows for marking up the arguments, but does not produce any output. This can be used to provide arguments that are not mentioned in the text we want to mark up, because they are implicitly obvious or mentioned elsewhere.

If we want to change the order of the arguments, we can provide the precise argument number as an optional argument; e.g. `\smacro{...\arg[2]{a}...\arg[1]{b}}` determines `b` to be the first and `a` to be the second argument.

An argument number may be used repeatedly, if the corresponding argument mode is `a` or `B`.

> ←M→
> —M→
> ~T↝
> Applications of semantic macros with arguments are translated to MMT/OMDOC as `OMA`-terms with head `<OMS name="⟨symbol⟩"/>`, or `<OMBIND name="⟨symbol⟩"/>`, depending on the absence or presence of argument mode `b` or `B` arguments. Semantic macros with no arguments or invoked with `!` correspond to `OMS` directly.

7.5   Simple Inheritance

There are three macros that allow for opening a module, making its contents available for use:

`\usemodule` `\usemodule{⟨module⟩}` is allowed anywhere and makes the module's contents available up to the current TEX group. This is the right macro to use outside of modules, or when none of its contents use any of the used module's symbols directly (e.g. in types or definientia).

`\requiremodule` `\requiremodule{⟨module⟩}` is only allowed in modules and makes the required module's contents available within the current module. The imported symbols can be safely used in types and definientia, but not in the `return` code of symbols, and the imported content is not exported further – i.e. using the current module does not also open the required module.

`\importmodule` `\importmodule{⟨module⟩}` is only allowed in modules and makes the required module's contents available within the current module. The imported symbols can be safely used anywhere, and the imported content exported to any modules subsequently importing the current one.

↩M→ In MMT, every *document* and every module induces an MMT theory. \usemodule
—M→ induces and MMT include in the document theory, \importmodule and
⤳T↝ \requiremodule both induce an include in the module's theory.

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

Ideally, sTeX would allow for arbitrary MMT-URIs for modules, with no forced
relationships between the *logical* namespace of a module and the *physical* location
of the file declaring it – like MMT in fact allows for.
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 accord-
ingly. Largely, users need not concern themselves with namespaces at all, but for
completenesses sake, we describe how they are constructed:

- If \begin{smodule}{Foo} occurs in a file /path/to/file/Foo[.⟨*lang*⟩].tex
  which does not belong to an math 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 math archives, the namespace corresponds to the file
URI with the filename dropped iff it is equal to the module name, and ignoring
the (optional) language suffix.
If the current file is in an archive, the procedure is the same except that the initial
segment of the file path up to the archive's source directory is replaced by the
archive's namespace URI.

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

- \importmodule{Foo} outside of an archive refers to module Foo in the cur-
  rent namespace. Consequently, Foo must have been declared earlier in the
  same file 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 de-
  clared earlier in the same file, 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 directory.

- 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 directory, 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).

7.6   Variables and Sequences

`\vardef`

`\vardef{⟨mname⟩}[⟨options⟩]{⟨notation⟩}`

Takes the same arguments as `\symdef`, but produces a variable rather than a symbol. Variables definitions are always local to the current TeX group and are allowed anywhere (i.e. outside of modules).

⟨*options*⟩ may include the additional keyword `bind`, in which case the variable will be appropriately abstracted away in statements (see also `\varbind`).

Unlike `\symdef`, there is no starred variant `\vardef*` – variables always generate a semantic macro.

The semantic macro for a variable behaves analogously to that of a symbol.

↪M→
—M→   Variables induces the same MMT/OMDOC terms as symbols do, except for the
∼T↝   head of the term being `<OMV name="..."/>` instead of `<OMS/>`.

`\varnotation`

`\varnotation{⟨variable⟩}[⟨options⟩]{⟨notation⟩}`

Takes the exact same argumens as `\notation`, but attaches an additional notation to the *variable* ⟨*variable*⟩ rather than a symbol.

`\svar`

`\svar[⟨name⟩]{⟨text⟩}`

Semantically marks up ⟨*text*⟩ as representing a variable ⟨*name*⟩. The variable does not need to have been defined prior. If no ⟨*name*⟩ is givenm ⟨*text*⟩ will be used as the name.

This is useful in situations like "throwaway expressions" or remarks; e.g.
`$\plus{\svar{n},\svar{m}}$` means...

`\varseq`

`\varseq{⟨mname⟩}[⟨options⟩]{⟨range⟩}{⟨notation⟩}`

Declares a new variable sequence. The ⟨*options*⟩ are the same as for `\vardef`. If not provided, `args=1` by default (a 0-ary sequence would just be a normal variable).

A type (given as `type=`) is interpreted to be the type of every element a_i of the sequence $a_1, ..., a_n$ (not of the sequence itself). If the type is itself a sequence $A_1, ..., A_n$, the assumption is that its range is the same as the one of the new sequence, and the type of every a_i in the sequence is A_i.

⟨*range*⟩ needs to be a comma-separated sequence of either `args` many arguments, or `\ellipses`.

The resulting semantic macro is allowed anywhere sTeX expects an argument mode `a` or `B` argument.

\ellipses Represents ellipses in a range; produces \ellipses in math mode.

\seqmap \seqmap{⟨*code*⟩}{⟨*sequence*⟩}

Maps the function ⟨*code*⟩ (containing #1) over every element of the ⟨*sequence*⟩.
Is allowed anywhere sTEX expects an argument mode a or B argument.

7.7 Structures

Mathematical structure bundle interdependent symbols together.

mathstructure (*env.*) \begin{mathstructure}{⟨*mname*⟩}[⟨*name*⟩,this=⟨*code*⟩] opens a new mathematical structure with name ⟨*mname*⟩ (if provided) or ⟨*mname*⟩ (otherwise), and semantic macro \mname. It subsequently behaves like the smodule environment.

\this The optional this=⟨*code*⟩ option allows for setting the typesetting of the \this macro within a mathstructure. In particular, \this can be used in notations for symbols declared in the structure. \this can be though of as representing "*the*" (current) instance of this structure.

extstructure (*env.*) \begin{extstructure}{⟨*mname*⟩}[⟨*name*⟩,this=⟨*code*⟩]{⟨*structs*⟩} opens a new mathematical structure extending the structures given in ⟨*structs*⟩ (a comma-separated list of names).

extstructure* (*env.*) \begin{extstructure*}{⟨*struct*⟩} opens a new mathematical structure *conservatively* extending the (single) structure ⟨*struct*⟩. *Conservative* meaning: Every symbol newly introduced in this structure needs to have a *definiens*. The new symbols are attached as fields directly to ⟨*struct*⟩.

\usestructure The \usestructure macro behaves like \usemodule for mathematical structures, making the symbols available to use directly.

> mathstructure make use of the *Theories-as-Types* paradigm (see
> [**MueRabKoh:tat18**]):
> \begin{mathstructure}{⟨*name*⟩} creates a nested theory with name
> ⟨*name*⟩-module. The constant ⟨*name*⟩ is defined as a *dependent record type*
> *with manifest fields*, the fields of which are generated from (and correspond to)
> the constants in ⟨*name*⟩-module.

7.7.1 Semantic Macros for Structures

Assume we have a mathematical structure with semantic macro **\struct**:

Example 17

```
\begin{mathstructure}{struct}
\symdef{fielda}{a}
\symdef{fieldb}[args=2]{#1 \maincomp{b} #2}
\symdef{fieldc}[args=2,def=\sn{fieldb}]{#1 \maincomp{c} #2}
```

```
    \inlineass[name=axiom1]{\conclusion{some axiom}}
  \end{mathstructure}
  \notation{struct}{StRuCt}
```

- If **\struct** has no notations, then **$\struct!$** produces $\langle a, b, c \rangle$. Otherwise, it produces the notation, i.e. *StRuCt*. In both cases, **$\struct{}{}$** produces $\langle a, b, c \rangle$ (for reasons that will become clearer in a moment).

- **$\struct{}{}$** (or **$\struct!$** in the case where no notations are around) can be modified in the following ways:

  - **$\struct{}{}[⟨fieldname⟩,...]$** lets you pick, which precise fields to show, so e.g. **$\struct{}{}[fielda,fieldb]$** produces $\langle a, b \rangle$. By default, **$\struct{}{}$** shows exactly the fields that have semantic macros (which are also used to access the fields in **$\struct{...}{fieldname}$**).

  - **$\struct{}{}[⟨id⟩]$** lets you pick the notation of the "mathematical structure" symbol to use to typeset the structure; e.g. **$\struct{}{}[parens]$** yields (a, b, c), and (combining both) **$\struct{}{}[fielda,fieldb][angle]$** yields $\langle a, b \rangle$.

- The two arguments in **$\struct{first}{second}$** represent 1. the term that is to be treated as an instance of **\struct**, and 2. the precise field to invoke. If the first is empty, then there is no instance. If the second is empty, sTeX will present all of them (that are not assertions). Hence, **$\struct{S}{}$** yields $\langle a_S, b_S, c_S \rangle$, **$\struct{S}{fielda}$** produces a_S, **$\struct{}{fielda}$** produces a, and **$\struct{S}{fieldb}{x}{y}$** produces $x b_S y$.

- For the sake of completion, **$\struct{first}!$** simply produces the given argument; e.g. **$\struct{S}!$** simply produces S.

More precisely: **$\struct{⟨code⟩}$** acts like a "type coercion" of ⟨code⟩ to be an instance of **\struct**.

Of course, it is (occasionally, but) rarely useful to use the semantic macro **\struct** by giving it two arguments *manually*; but this is what sTeX does when using **\struct** in the `return` of a symbol (or variable).

Continuing:

- **$\struct[comp=⟨code⟩]{...}{...}$** applies ⟨code⟩ (using #1) to every occurence of **\maincomp** in the notations of the fields, as a replacement for the default modification {#1}_{\this}. For example, **$\struct[comp={#1}^{Foo}]{S}{}$** produces $\langle a^{Foo}, b^{Foo}, c^{Foo} \rangle$, and **$\struct[comp={#1}^{\this}]{S}{fieldb}{x}{y}$** yields $x b^S y$.

- **$\struct[this=⟨code⟩]{...}{...}$** modifies the way **\this** is being typeset; i.e. the presentation of the first {...} argument. For example **$\struct[this=T]{S}{}$** produces $\langle a, b, c \rangle$ – since **\this** is not being used in the notations of the fields. Note that the `this=` and `comp=` variants are (as of yet) mutually incompatible.

- Finally, **$\struct[⟨fieldname⟩=⟨code⟩]{...}{...}$** *assigns* the field ⟨fieldname⟩ to the term ⟨code⟩. ⟨code⟩ is subsequently used when using {...}{}, but not in fields directly. For example, **$\struct[fielda=A]{S}{}$** produces $\langle A!, b_S, c_S \rangle$, but **$\struct[fielda=A]{S}{fielda}$** produces a_S.

Note the insertion of `!` behind the `A` – this is to make sure that assignments to semantic macros that takes arguments don't accidentally eat more than they should.

Also note that multiple assignments can be done in the same pair of `[]`, or chained – i.e. both `$\struct[fielda=A,fieldb=B]...$` and `$\struct[fielda=A][fieldb=B]...$` are valid and equivalent. `$\struct[fielda=A,fielda=B]...$` however is not – every field may be assigned at most once.

Chapter 8

Statements

sTeX provides four environments to semantically annotate various kinds of statements:

sdefinition (*env.*)   The **sdefinition** environment represents (primarly mathematical) *definitions*; in particular for symbols. The contents of the environment will be used as *documentation* for any symbol that either orrurs as a `\definiendum` (or `\definame`) within the **sdefinition**, or that is listed in the optional `for=` argument of the environment.

If a `\definiens` occurs, this will be used by Mmt as the formal definiens for the respective symbol.

sassertion (*env.*)   The **sassertion** environment represents *assertions*, i.e. propositions such as *theorems*, *lemmata*, *axioms*, etc. If a `\conclusion` occurs within the **sassertion**, its argument will be used as the formal *statement* of the assertion.

sexample (*env.*)   The **sexample** environment represents examples (or counterexamples).

sparagraph (*env.*)   The **sparagraph** environment represents all other kinds of (logical) paragraphs, such as remarks, comments, transitions bwetween topics, recaps, reminders, etc.

All of these take the same arguments:

- `for=`⟨*csl*⟩: a comma-separated list of symbols.

- The same optional arguments as `\symdecl`, with `macro=` replacing the name of the semantic macro. All of them are only relevant, if either `name=` or `macro=` are provided.

  As with `\symdecl`, if no `name` is given, but `macro` is, then the same name is used for both the semantic macro and the symbol itself.

  If `name` is given but `macro` isn't, no semantic macro is generated. Subsequently, the newly generated symbol is added the `for`-list.

- `style`: see chapter 9.

- `title`: a title to use in various styles (see chapter 9).

- `id`: a label to use for `\sref`.

\inlineass
\inlinedef
\inlineex

The macros `\inlineass`, `\inlinedef` and `\inlineex` behave like the `sassertion`, `sdefinition` and `sexample` environments respectively, but take the text to annotate as an argument, rather than as the body of an environment, and do not break paragraphs.

The same macros available in the environments are also available in the argument of the `\inline*` macros.

\varbind

`\varbind{⟨cls⟩}`

retroactively attaches the `bind` option to every variable provided (as a comma-separated list).

8.1 More on Definitions

In `sdefinition` (and `sparagraph` with `style=symdoc`), the following additional macros are available:

\definiendum
\definame
\Definame
\defnotation

The `\definiendum` macro behaves largely like `\symref`, but it uses a dedicated highlighting for *definienda* and adds the referenced symbol to the `for=` list of the environment.

`\definame` is to `\definiendum` as `\symname` is to `\symref`. Analogously, `\Definame` behaves like `\Symname`.

`\defnotation` can be used in math mode to apply the `\definiendum` highlighting to notations.

\definiens

The `\definiens` macro can be used to semantically annotate the *definiens* in a `sdefinition`.

If the `sdefinition` environment has several elements in its `for` list, an optional argument `\definiens[⟨symbol⟩]{...}` can be used to tell sTEX which symbol's definiens this is. By default, the *first* symbol in the `for` list is used.

> Here is how MMT will treat the fragment marked up with `\definiens`:
> Firstly, it will attempt to translate its contents into an MMT/OMDoc term. This succeeds easily if `\definiens` is some semantic macro (applied to arguments).
> Secondly, it will check for all variables currently in scope, that were defined with the optional argument `bind`. It then will check, whether a symbol is in scope, that has `role=lambda`. If so, it will use that `lambda` symbol to bind all these variables (in the order in which they were defined) in the term. If no `lambda` symbol is found, it will use the bind symbol that ships with sTEX.
> The final term will be attached as definiens to the corresponding MMT constant, if it was declared in the same module as the `\definiens` occurrence.

8.2 More on Assertions

\premise
\conclusion

The **\conclusion** macro can be used to mark up the actual statement within an sassertion. The **\premise** macro can be used to additionally mark up *premises*.

If the sassertion environment has several elements in its `for` list, an optional argument **\conclusion[**⟨*symbol*⟩**]{...}** can be used to tell STEX which symbol's statement this is. By default, the *first* symbol in the `for` list is used.

> Here is how MMT will treat the fragments marked up with **\conclusion** and **\premise**:
>
> Firstly, it will attempt to translate the contents of **\conclusion** into an MMT/OM-Doc term c. This succeeds easily if the **\conclusion** is some semantic macro (applied to arguments).
>
> Secondly, it will collect all fragments marked up with **\premise** and do the same to them $(p_1, ..., p_n)$.
>
> It will then check, whether a symbol is in scope, that has `role=implication`. If so, it will use that `implication` symbol to attach all the premises to the conclusion, resulting in $t := \mathtt{imply}(p_1, ..., p_n, c)$.
>
> Next, it will check for all variables currently in scope, that were defined with the optional argument `bind`. It then will check, whether a symbol is in scope, that has `role=forall`. If so, it will use that `forall` symbol to bind all these variables (in the order in which they were defined) in the term t.
>
> Finally, it will check, whether a symbol is in scope, that has `role=judgment`. If so, it will set $t := \mathtt{judgment}(t)$.
>
> If no `forall` symbol is found, it will first apply the `judgment` symbol (if existent) and then use the bind symbol that ships with STEX to bind the variables.
>
> The final term will be attached as type to the corresponding MMT constant, if it was declared in the same module as the **\definiens** occurrence.

`sproof` (*env.*)   TODO[12]

---

[12]TODO: proofs

Chapter 9

Customizing Typesetting

There are two kinds of typesetting that can be customized in sTeX: symbol references (notation components, `\symref`, variables, etc.) are highlighted using a small set of macros that can be simply redefined by authors.

Other macros and environments usually have more complicated "typesetting rules" associated with them – often in the form of other already existing environments that should be used.

Lastly, in HTML we can provide custom CSS rules in math archives that determine the styling of certain environments, so that the actual presentation depends on the document in which the fragments are included (e.g. via `\inputref`), rather than the file the fragment is implemented in.

It is generally recommended to implement these customizations in a preamble in the `lib` directory (see section 5.3)

9.1   Highlighting Symbol References

`\symrefemph`
`\symrefemph@uri`

`\symrefemph` governs how references via `\symref` (or `\symname`, or theirt short variants) are highlighted;

Doing `\symref{`⟨*symbol*⟩`}{`⟨*text*⟩`}` ultimately expands to `\symrefemph@uri{`⟨*text*⟩`}{`⟨*symbol URI*⟩`}`, the default implementation of which is just `\symrefemph{`⟨*text*⟩`}`. The default implementation of `\symrefemph`, in turn, is just `\emph{`⟨*text*⟩`}`.

If you only want to change e.g. the color of `\symref`s, you only need to redefine `\symrefemph`, e.g. using

```
\renewcommand\symrefemph[1]{\textcolor{red}{#1}}
```

the `@uri` variant is useful if you want to link somewhere, or show the URI in a tooltip. The stex-highlighting package does both, using:

```
\usepackage{pdfcomment}
\protected\def\symrefemph@uri#1#2{%
  \pdftooltip{%
    \srefsymuri{#2}{\symrefemph{#1}}%
  }{%
    URI:~\detokenize{#2}%
  }%
```

```
  }
\def\symrefemph#1{%
  \ifcsname textcolor\endcsname
    \textcolor{teal}{#1}%
  \else#1\fi
}
```

\compemph
\compemph@uri

Like `\symrefemph` and `\symrefemph@uri`, but governs the highlighting of components (marked with `\comp` or `\maincomp`) in notations.

\defemph
\defemph@uri

Like `\symrefemph` and `\symrefemph@uri`, but governs the highlighting of definienda marked with `\definiendum` (or `\definame`).

\varemph
\varemph@uri

Like `\compemph` and `\compemph@uri`, but governs the highlighting of components (marked with `\comp` or `\maincomp`) in the notations of variables.

The second argument to `\varemph@uri` is the *name* of the variable.

9.2 Styling Environments and Macros

A variety of environments and macros provided by sTEX are *stylable* using the macros `\stexstyle⟨name⟩[⟨style⟩]{...}`. These stylable environments and macros bind various of their parameters to macros `\this⟨parameter⟩`, which can then be used in the styles.

For example, if we have a `definition` environment that we would want to use to style our sdefinitions, we can do (in the simplest case)

```
\stexstyledefinition{\begin{definition}}{\end{definition}}
```

This tells sTEX to insert `\begin{definition}` at the beginning of every sdefinition environment, and `\end{definition}` at the end.

If we have a environments `theorem` and `lemma`, we probably want the sassertion environment to use those for theorems and lemma. We can achieve that by doing

```
\stexstyleassertion[theorem]{\begin{theorem}}{\end{theorem}}
\stexstyleassertion[lemma]{\begin{lemma}}{\end{lemma}}
```

Now if we do `\begin{sassertion}[style=theorem]`, it will wrap the environment with `\begin{theorem}...\end{theorem}`.

Of course, many such statements might have a title, as e.g. in

---

Theorem 9.2.1 (Gödel's First Incompleteness Theorem)**.** ...

In sassertion, we can provide that title as optional argument using `title=...`. Before calling the styling provided, sassertion will store that title in the macro `\thistitle`, which we can use in the styling. For example, we might prefer to pass it on to the `theorem` environment:

```
\stexstyleassertion[theorem]{\ifx\thistitle\@empty
  \begin{theorem}\else\begin{theorem}[\thistitle]\fi}
  {\end{theorem}}
```

TODO[13]

Additionally, we can style certain macros, if we want them to produce output. For example, we might (for debuggin or documentation purposes) \symdecl to give a short summary of the symbol.

We can achieve that by doing, for example:

```
\stexstylesymdecl[debug]{
  Symbol \thisdeclname~(with arity \thisargs) of type $\thistype$.
}
```

in which case

```
\symdecl{foo}[args=2,type=\mathbb{N},style=debug]
```

will yield

> Symbol foo (with arity ii) of type ℕ.

TODO[14]

9.3 Custom CSS for Environments

Chapter 10

Additional Packages

10.1 NotesSlides Manual

10.1.1 Introduction

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

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

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

10.1.2 Package Options

The notesslides class takes a variety of class options:

slides
notes
The options `slides` and `notes` switch between slides mode and notes mode (see subsection 10.1.3).

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

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

10.1.3 Notes and Slides

frame *(env.)* Slides are represented with the `frame` environment just like in the `beamer` class, see [**Tantau:ugbc**] for details.

note *(env.)* The notesslides class adds the `note` environment for encapsulating the course note fragments.

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

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

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

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

> ⚠ We need to give the title frame the `noframenumbering` option so that the frame numbering is kept in sync between the slides and the course notes.
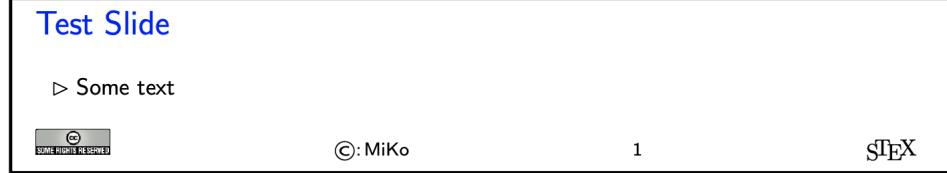
> ⚠ The `beamer` class recommends not to use the `allowframebreaks` option on frames (even though it is very convenient). This holds even more in the `notesslides` case: At least in conjunction with `\newpage`, frame numbering behaves funnily (we have tried to fix this, but who knows).

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

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

10.1.4 Customizing Header and Footer Lines

The notesslides package and class comes with a simple default theme named sTeX that provided by the beamterthemesTeX. It is assumed as the default theme for sTeX-based notes and slides. The result in `notes` mode (which is like the `slides` version except that the slide hight is variable) is

The footer line can be customized. In particular the logos.

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

The default footer line of the notesslides package mentions copyright and licensing. In notesslides `\source` stores the author's name as the copyright holder. By default it is the author's name as defined in the `\author` macro in the preamble. `\setsource{⟨name⟩}` can change the writer's name.

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

10.1.5 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 sTeX notes.

\frameimage
\mhframeimage

In this case we can use `\frameimage[`⟨*opt*⟩`]{`⟨*path*⟩`}`, where ⟨*opt*⟩ are the options of `\includegraphics` from the `graphicx` package [**CarRah:tpp99**] and ⟨*path*⟩ is the file path (extension can be left off like in `\includegraphics`). We have added the `label` key that allows to give a frame label that can be referenced like a regular `beamer` frame.

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

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

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

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

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

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

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

\textwarning

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

10.1.6 Ending Documents Prematurely

\prematurestop
\afterprematurestop

For prematurely stopping the formatting of a document, sTeX provides the `\prematurestop` macro. It can be used everywhere in a document and ignores all input after that – backing out of the `sfragment` environments 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 [**lmhtools:github:on**].

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

10.1.7 Global Document Variables

To make document fragments more reusable, we sometimes want to make the content depend on the context. We use **document variables** for that.

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

10.1.8 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:

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

It generates a paragraph that references the excursion whose source is in the file `../fragments/founif.en.tex` and automatically books the file for the \printexcursions command that is used here to put it into the appendix. We will look at the mechanics now.

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

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

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

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

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

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

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

Local Variables: mode: latex TeX-master: `../stex-manual` End:

10.2 Problem Manual

10.2.1 Introduction

The `problem` package supplies an infrastructure that allows specify problem. Problems are text fragments that come with auxiliary functions: hints, notes, and solutions[15]. 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.

10.2.2 Problems and Solutions

`solutions` The `problem` package takes the options `solutions` (should solutions be output?), `notes`
`notes` (should the problem notes be presented?), `hints` (do we give the hints?), `gnotes` (do we
`hints` show grading notes?), `pts` (do we display the points awarded for solving the problem?),
`gnotes` `min` (do we display the estimated minutes for problem soling). If theses are specified, then
`pts` the corresponding auxiliary parts of the problems are output, otherwise, they remain
`min` invisible.
`boxed` The `boxed` option specifies that problems should be formatted in framed boxes so
`test` that they are more visible in the text. Finally, the `test` option signifies that we are in a test situation, so this option does not show the solutions (of course), but leaves space for the students to solve them.

`problem` (*env.*) The main environment provided by the `problem`package is (surprise surprise) the `problem` environment. It is used to mark up problems and exercises. The environment takes an optional KeyVal argument with the keys `id` as an identifier that can be reference later, `pts` for the points to be gained from this exercise in homework or quiz situations, `min` for the estimated minutes needed to solve the problem, and finally `title` for an informative title of the problem.

> **Example 18**
> Input:

---

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

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

Output:

> **Exercise (Fitting Elefants)**
> How many Elefants can you fit into a Volkswagen beetle?
> ___
> *Solution for=elefants:* Four, two in the front seats, and two in the back.

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

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

\startsolutions Sometimes we would like to locally override the `solutions` option we have given to
\stopsolutions 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.

\ifsolutions 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.

10.2.3 Markup for Added-Value Services

The `problem` package is all about specifying the meaning of the various moving parts of practice/exam problems. The motivation for the additional markup is that we can base added-value services from these, for instance auto-grading and immediate feedback.

The simplest example of this are multiple-choice problems, where the `problem` package allows to annotate answer options with the intended values and possibly feedback that can be delivered to the users in an interactive setting. In this section we will give some infrastructure for these, we expect that this will grow over time.

Multiple Choice Blocks

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

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

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

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

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

What we see when this is formatted to PDF depends on the context. In solutions mode (we start the solutions in the code fragment below) we get

Example 19
Input:

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

Output:

Exercise (Functions)
What is the keyword to introduce a function definition in python?
☐ def – Correct

☐ function – Wrong
that is for C and C++

☐ fun – Wrong
that is for Standard ML

☐ public static void – Noooooooooo
that is for Java

In "exam mode" where disable solutions (here via **\stopsolutions**)

> **Example 20**
> Input:
>
> ```
>  1  \stopsolutions
>  2 \begin{sproblem}[title=Functions,name=functions1]
>  3   What is the keyword to introduce a function definition in python?
>  4   \begin{mcb}
>  5     \mcc[T]{def}
>  6     \mcc[F,feedback=that is for C and C++]{function}
>  7     \mcc[F,feedback=that is for Standard ML]{fun}
>  8     \mcc[F,Ftext=Noooooooooo,feedback=that is for Java]{public static void}
>  9   \end{mcb}
> 10 \end{sproblem}
> ```
>
> Output:
>
> > **Exercise (Functions)**
> > What is the keyword to introduce a function definition in python?
> > ☐ def
> >
> > ☐ function
> >
> > ☐ fun
> >
> > ☐ public static void

we get the questions without solutions (that is what the students see during the exam/quiz).

Filling-In Concrete Solutions

The next simplest situation, where we can implement auto-grading is the case where we have fill-in-the-blanks

`\fillinsol` The `\fillinsol` macro takes a single argument, which contains a concrete solution (i.e. a number, a string, . . .), which generates a fill-in-box in test mode:

> **Example 21**
> Input:
>
> ```
> 1     \stopsolutions
> 2     \begin{sproblem}[id=elefants.fillin,title=Fitting Elefants]
> 3     How many Elefants can you fit into a Volkswagen beetle? \fillinsol{4}
> 4 \end{sproblem}
> ```
>
> Output:
>
> > **Exercise (Fitting Elefants)**
> >
> > How many Elefants can you fit into a Volkswagen beetle?

and the actual solution in solutions mode:

> **Example 22**
> Input:
>
> ```
> 1     \begin{sproblem}[id=elefants.fillin,title=Fitting Elefants]
> 2     How many Elefants can you fit into a Volkswagen beetle? \fillinsol{4}
> 3   \end{sproblem}
> ```
>
> Output:
>
> > **Exercise (Fitting Elefants)**
> >
> > How many Elefants can you fit into a Volkswagen beetle?

If we do not want to leak information about the solution by the size of the blank we can also give `\fillinsol` an optional argument with a size: `\fillinsol[3cm]{12}` makes a box three cm wide.

Obviously, the required argument of `\fillinsol` can be used for auto-grading. For concrete data like numbers, this is immediate, for more complex data like strings "soft comparisons" might be in order. [16]

---

[16]For the moment we only assume a single concrete value as correct. In the future we will almost certainly want to extend the functionality to multiple answer classes that allow different feedback like im MCQ. This still needs a bit of design. Also we want to make the formatting of the answer in solutions/test mode configurable.

10.2.4 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.

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

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

10.2.5 Testing and Spacing

The `problem` package is often used by the `hwexam` package, which is used to create homework assignments and exams. Both of these have a "test mode" (invoked by the package option `test`), where certain information –master solutions or feedback – is not shown in the presentation.

\testspace \testspace takes an argument that expands to a dimension, and leaves verti-
\testsmallspace cal space accordingly. Specific instances exist: \testsmallspace, \testsmallspace,
\testsmallspace \testsmallspace give small (1cm), medium (2cm), and big (3cm) vertical space.
\testsmallspace \testnewpage makes a new page in `test` mode, and \testemptypage generates an
\testnewpage empty page with the cautionary message that this page was intentionally left empty.
\testemptypage Local Variables: mode: latex TeX-master: `../stex-manual` End:

LocalWords: solutions,notes,hints,gnotes,pts,min,boxed,test gnotes elefants,pts gnote LocalWords: 2,title exnote hint,exnote,gnote ifsolutions mcb keyvals Ttext Ftext Local-Words: Functions,name F,feedback Nooooooooo,feedback 2,title includeproblem Local-Words: elefants.fillin,title

10.3 HWExam Manual

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

10.3.2 Package Options

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

solutions
notes
hints
gnotes
pts
min

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).

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.

</div>

10.3.3 Assignments

assignment (*env.*) 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).

10.3.4 Including Assignments

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

10.3.5 Typesetting Exams

testheading (*env.*) The \testheading takes an optional keyword argument where the keys duration speci-
duration fies a string that specifies the duration of the test, min specifies the equivalent in number
min of minutes, and reqpts the points that are required for a perfect grade.

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

Will result in

Name: Matriculation Number:

320101 General Computer Science (Fall 2010)

2023-03-21

You have one hour (sharp) for the test;
Write the solutions to the sheet.
The estimated time for solving this exam is 10 minutes, leaving you
50 minutes for revising your exam.
You can reach 10 points if you solve all problems. You will only need
27 points for a perfect score, i.e. -17 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.	4.1	1.1	1.2	2.1	2.2	2.3	2.4	2.5	Su
total	0	0	0	10	0	0	0	0	10
reached									

To be used for grading, do not write here

good luck

Local Variables: mode: latex TeX-master: `../stex-manual` End:
LocalWords: hwexam solutions,notes,hints,gnotes,pts,min gnotes testemptypage re-
qpts LocalWords: inputassignment reqpts hour,min 60,reqpts

10.4   Tikzinput Manual

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

---

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

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

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

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

`\tikzinput`
`\ctikzinput`

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

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

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

`\mhtikzinput`
`\cmhtikzinput`

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

`\libusetikzlibrary`

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

Local Variables: mode: latex TeX-master: `../stex-manual` End:

Part III
Documentation

Chapter 11

sTeX Developer Manual

> ⚠ Keeping the implementation properly up-to-date is pretty much incompatible with the kinds of workflows systemically enforced in academia. Any of the following may be out of date.

⋆ indicates fully expandable functions, which can be used within an `x`-type argument (in plain TeX terms, inside an `\edef`), as well as within an `f`-type argument. ☆ indicates restricted expandable functions, which can be used within an `x`-type argument but cannot be fully expanded within an `f`-type argument. *TF* indicates conditional (`if`) functions whose variants with `T`, `F` and `TF` argument specifiers expect different "true"/"false" branches.

\stex_debug:nn

`\stex_debug:nn` {⟨*prefix*⟩} {⟨*msg*⟩}

Logs the debug message {⟨*msg*⟩} under the prefix {⟨*prefix*⟩}. A message is shown if its prefix is in a list of prefixes given either via the package option `debug=`⟨*prefixes*⟩ or the environment variable `STEX_DEBUG=`⟨*prefixes*⟩, where the latter overrides the former.

_stex_do_deprecation:n

TODO `\_stex_do_deprecation:n`

11.1 Documents

\l_stex_docheader_sect   integer register keeping track of the current sectioning level:

> 0   part
>
> 1   chapter
>
> 2   section
>
> 3   subsection
>
> 4   subsubsection
>
> 5   paragraph
>
> > 5   subparagraph

\setsectionlevel sets \l_stex_docheader_sect to the corresponding integer value.

\stex_ref_new_doc_target:n   internal variant of \sreflabel. If the argument is empty, the label is determined to be REF⟨*counter*⟩ and ⟨*counter*⟩ is increased.

\stex_ref_new_symbol:n   registers a new link target for the symbol with the given full uri (as string), using the url-base-field of the current archive's manifest file to link the symbol to ⟨*url-base*⟩/symbol?⟨*uri*⟩.

\stex_ref_new_sym_target:n   sets a new label for the symbol with the given full uri (as string). If called in sms-mode, defers to \stex_ref_new_symbol:n, if not already registered. Otherwise, sets a \label for the symbol.

\stex_ref_new_sym_target:nn

\stex_ref_new_sym_target:nn {⟨*symbol*⟩}{⟨*target*⟩}

redirects links for ⟨*symbol*⟩ to the one for the symbol ⟨*target*⟩. Useful for e.g. symbols elaborated from structural features. Note that this acts as a *default*, in that previous or subsequent calls of \stex_ref_new_sym_target:n{⟨*symbol*⟩} are prioritized.

Requires that either \stex_ref_new_sym_target:n{⟨*target*⟩} or \stex_ref_new_sym_target:nn{⟨*target*⟩}{⟨*other-target*⟩} have been called previously.

11.2 Modules

The contents of a module with full URI ⟨*uri*⟩ are represented as four macros:

- \c_stex_module_⟨*uri*⟩_code: code to be executed every time the module is *activated*; e.g. the contents of \STEXexport, defines semantic macros and macros for notations, activates dependency modules, etc.

- `\c_stex_module_`⟨*uri*⟩`_morphisms_prop`: property list containing all module dependencies of this module (see `\stex_module_add_morphism:nnn`).

- `\c_stex_module_`⟨*uri*⟩`symbols_prop`: property list containing all declarations in this module (see `\stex_module_add_symbol:nnnnnnN`).

- `\c_stex_module_`⟨*uri*⟩`_notations_prop`: property list containing all notations introduced in this module (see `\stex_add_module_notation:nnnnn`).

`\l_stex_current_module_str` contains the full URI of the current module.

`\l_stex_all_modules_seq` contains the full URIs of all modules currently in scope.

`\stex_module_setup:n`

`\stex_module_setup:n` {⟨*name*⟩}

Computes the full URI of a new module with name ⟨*name*⟩ in the current namespace, initializes the four macros above and sets `\l_stex_current_module_str` accordingly. Also takes care of correct naming for nested modules, activates the meta theory and loads the signature module if `sig=` was provided (according to `\l_stex_key_sig_str`).

`\stex_close_module:` closes the current module; checks whether we are currently in sms mode, and if so, calls `\stex_persist:n` to write the module contents to the sms-file.

`\stex_every_module:n`

`\stex_every_module:n` {⟨*code*⟩}

executes ⟨*code*⟩ every time a new module is opened.

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

tests whether we are currently in a module.

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

tests whether a module with the given full URI exists, in the sense of *has been parsed at some point in the current document*.

`\stex_activate_module:n`
`\stex_activate_module:`(o|x)

activates the module with the given full URI *if and only if* it has not already been activated in the current scope.

`\stex_execute_in_module:n`
`\stex_execute_in_module:x`

executes the provided code, adds it to the current module activation code, and makes sure the macros defined in it are valid in the current module TeX group level.

This macro is a combination of the following two macros:

`\stex_do_up_to_module:n`
`\stex_do_up_to_module:x` executes the provided code such that all definitions in it are valid in the current module regardless of TeX group level (using `\stex_metagroup_do_in:nn`).

`\stex_module_add_code:n`
`\stex_module_add_code:x` adds the provided code to the module's `\c_stex_module_`⟨*uri*⟩`_code`-macro.

11.3   Symbols

A symbol in STeX is represented as a tuple of several components:

`\stex_module_add_symbol:nnnnnnnN`

#1 : ⟨*id*⟩: An *identifier* (possibly empty) that determines its semantic macro name, or e.g. in `mathstructure` its accessor-identifier (if empty, the name is used for that),

#2 : ⟨*name*⟩ a unique *name*, which in combination with the containing module determines its URI as ⟨*module-URI*⟩`?`⟨*name*⟩,

#3 : ⟨*arity*⟩ a numeric string in the range `0..9`,

#4 : ⟨*args*⟩ a list of argument specifiers of the form ⟨*i*⟩⟨*mode*⟩`{`⟨*argname*⟩`}` (the length of ⟨*args*⟩ must be 3·⟨*arity*⟩),

#5 : ⟨*definiens*⟩ (or empty),

#6 : ⟨*type*⟩ (or empty),

#7 : ⟨*return code*⟩ (or empty), and

#8 : ⟨`\invokation_macro`⟩.

the arguments are stored in the property list `\c_stex_module_`⟨`\l_stex_current_-module`⟩`_symbols_prop` under key ⟨*name*⟩.

If the identifier ⟨*id*⟩ is non-empty, the macro `\id` is defined as `{\_stex_invoke_-symbol:nnnnnnnN}` with the arguments described there.

TeXhackers note: `\invokation_macro` must be `\protected`.

`\stex_iterate_symbols:n`
`\stex_iterate_break:`
`\stex_iterate_break:n`

`\stex_iterate_symbols:n` `{`⟨*code*⟩`}`

iterates over all symbols currently in scope and calls ⟨*code*⟩ for all of them with arguments `{`⟨*module*⟩`}{`⟨*name*⟩`}{`⟨*arity*⟩`}`⟨*args*⟩`{`⟨*definiens*⟩`}` `{`⟨*type*⟩`}{`⟨*return code*⟩`}{`⟨`\invokation_-macro`⟩`}`.

Iteration can be stopped prematurely with `\stex_iterate_break:` and can stop and return code with `\stex_iterate_break:n`.

`\stex_iterate_symbols:nn`

`\stex_iterate_symbols:nn` `{`⟨*csl*⟩`}{`⟨*code*⟩`}`

iterates over all symbols in the provided ⟨*csl*⟩ of full module URIs. ⟨*code*⟩ receives the same arguments as `\stex_iterate_symbols:n`, but iteration can not be stopped early.

```
\stex_get_symbol:n
\l_stex_get_symbol_mod_str
\l_stex_get_symbol_name_str
\l_stex_get_symbol_arity_int
\l_stex_get_symbol_args_tl
\l_stex_get_symbol_def_tl
\l_stex_get_symbol_type_tl
\l_stex_get_symbol_return_tl
\l_stex_get_symbol_invoke_cs
```

\stex_get_symbol:n attemps to find a symbol with the given name or id that is currently in scope. A name may be prefixed with a module name/path separated by ?.

Throws an error if no such symbol is found; otherwise, sets the listed \l_stex_get_symbol_...- macros to the components of the found symbol.

```
\stex_if_check_terms_p:  ⋆
\stex_if_check_terms: TF ⋆
```

whether to typeset declaration components (notations, types, definientia etc.) in a throw-away box. Is true iff the backend is pdflatex and either the STEX_CHECKTERMS environment variable or checkterms package option is set.

```
\stex_check_term:n
```

typesets the argument in a throwaway box in math mode iff \stex_if_check_terms: is true.

Is deactivated in sms-mode.

```
\stex_symdecl_do:
\l_stex_key_name_str
\l_stex_key_args_str
\l_stex_key_argnames_clist
\l_stex_assoc_args_count
\l_stex_argnames_seq
```

\stex_symdecl_do: processes the shared (mandatory and optional) arguments of e.g. \symdecl, \symdef, \vardef etc.

Requires the following macros to be set

- \l_stex_key_name_str: the name of the symbol,

- \l_stex_key_args_str: the args-string (e.g. 3 or ai)

- \l_stex_key_argnames_clist: a list of *names* for the arguments, the length of which should be ≤ the arity of the symbol

and will generate the following macros:

- \l_stex_get_symbol_arity_int: the arity of the symbol,

- \l_stex_key_args_str: the args string as a definite sequence of argument-mode characters, whose length is the arity of the symbol; e.g. 3 is turned into iii,

- \l_stex_assoc_args_count: the number of sequence arguments (i.e. a or B mode),

- \l_stex_argnames_seq: the full sequence of argument names; those not provided by \l_stex_key_argnames_clist are set to be $j, where j is the index of the argument,

- \l_stex_get_symbol_args_tl: a token list of triples jm{⟨*argname*⟩}, where j is the index and m the respective argument mode character (i.e. i, a, b or B).

_stex_symdecl_check_terms:

calls `\stex_check_term:n` for the type and definiens stored in `\l_stex_key_type_tl` and `\l_stex_key_def_tl`

\stex_symdecl_top:n

`\stex_symdecl_top:n` {⟨*maybename*⟩}

checks whether `\l_stex_key_name_str` is empty, and if so, sets it to be ⟨*maybename*⟩. Then calls `\stex_symdecl_do:` and `\_stex_symdecl_check_terms:`, writes the components to the HTML (if applicable) and adds the symbol to the current module with invokation macro `\stex_invoke_symbol:` and id/macroname `\l_stex_macroname_str`.

Variables work very similar to symbols, except that their declarations are local to the current TeX-group rather than the current module, and they are not exported in modules.

\l_stex_variables_prop

stores all variables currently in scope as a property list with key ⟨*name*⟩ and value {⟨*id*⟩}{⟨*name*⟩}{⟨*arity*⟩}⟨*args*⟩{⟨*definiens*⟩} {⟨*type*⟩}{⟨*return code*⟩}{⟨`\invokation_macro`⟩}.

The invokation macro for "normal" variables declared with `\vardef` is `\stex_-invoke_symbol:`.

_stex_vardecl_notation_macro:

generates the notation macro for a variable, based on the values of the `\l_stex_key`-macros und `\l_stex_notation_macrocode_cs`.

\stex_get_var:n like `\stex_get_symbol:n`, but attempts to retrieve a variables and throws an error if none is found.

\stex_get_symbol_or_var:n like `\stex_get_symbol:n`, but first attempts to find a *variable*, and if none is found, defers to `\stex_get_symbol:n`.

11.4 Notations

\stex_module_add_notation:nnnnn
\stex_module_add_notation:eoexo

`\stex_module_add_notation:nnnnn`
{⟨*symboluri*⟩}{⟨*id*⟩}{⟨*arity*⟩}⟨*code*⟩⟨*op code*⟩

stores the arguments in the property list `\c_stex_module_`⟨`\l_stex_current_module`⟩`_notations_prop` under key ⟨*symboluri*⟩!⟨*id*⟩ and calls `\stex_set_notation_macro:nnnnn`.

`\stex_set_notation_macro:nnnnn`
`\stex_set_notation_macro:eoexo`

`\stex_set_notation_macro:nnnnn`
{⟨*symboluri*⟩}{⟨*id*⟩}{⟨*arity*⟩}⟨*code*⟩⟨*op code*⟩

Declares the following macros:

An active notation for a symbol with uri ⟨*symboluri*⟩ and id ⟨*id*⟩ is represented as a macro `\l_stex_notation_`⟨*symboluri*⟩`_`⟨*id*⟩`_cs`, that takes ⟨*arity*⟩ many argument and expands to ⟨*code*⟩, and (if nonempty) an *operator* notation as a macro `\l_stex_-notation_`⟨*symboluri*⟩`_op_`⟨*id*⟩`_cs` that expands to ⟨*op code*⟩.

The default notation is represented as the *empty* id. If the correpoding macros `\l_-stex_notation_`⟨*symboluri*⟩`__cs`, and `\l_stex_notation_`⟨*symboluri*⟩`_op__cs` do not yet exist, they are now defined as ⟨*id*⟩.

`\stex_iterate_notations:nn`

`\stex_iterate_notations:nn` {⟨*csl*⟩}{⟨*code*⟩}

iterates over all notations in the provided ⟨*csl*⟩ of full module URIs and calls ⟨*code*⟩ on each of them with arguments {⟨*symboluri*⟩}{⟨*id*⟩}{⟨*arity*⟩}⟨*code*⟩⟨*op code*⟩.

`\stex_notation_parse_and_then:nw`
`\l_stex_key_prec_str`
`\l_stex_key_variant_str`
`\l_stex_notation_macrocode_cs`

`\stex_notation_parse_and_then:nw` {⟨*code*⟩}⟨*notations*⟩

parses a notation (which may consist of multiple braced components, depending on the argument modes) and subsequently calls ⟨*code*⟩.

Requires the following macros to be set:

- `\l_stex_get_symbol_arity_int`,

- `\l_stex_get_symbol_args_tl`,

- `\l_stex_key_prec_str`: The precedence string as provided by a user in the optional `precs`-argument,

- `\l_stex_key_variant_str`: the id of the notation variant.

Stores the final notation in the macro `\l_stex_notation_macrocode_cs` taking `\l_stex_get_symbol_arity_int` many arguments.

Some thing to consider when we generate a notation macro:

- The notation macro generated by the `\notation`-command should contain the variant identifier and the precedences, as these are intrinsic parts of the notation.

- It should *not* contain the symbol itself however, so that notations can be copied between symbols.

- Notations as provided by users will largely adhere to the standard LaTeX category code scheme, and

- notations need to be "persistable" in `.deps`-files.

We therefore want to augment the simple code provided by a user by various "annotations" that contain the relevant information (such as precedences) and to mark the

argument positions for semantic extractions, but we should adhere to the default LaTeX category code scheme in doing so.

```
\STEXInternalTermMathArgiii
\STEXInternalTermMathAssocArgiiiii
\STEXInternalAssocArgMarkerI
\STEXInternalAssocArgMarkerII
\STEXInternalTermMathOMSOrOMViii
\STEXInternalTermMathOMAiii
\STEXInternalTermMathOMBiii
\STEXInternalSymbolAfterInvokationTL
```

In OPENMATH/OMDOC, there are (for our purposes) three kinds of expressions that an application of a semantic macro – and hence a notation macro – can represent, each of which corresponds to a macro taking care of the semantic annotations:

- `OMS/OMV`: a simple symbol (arity 0) (`\STEXInternalTermMathOMSOrOMViii`)

- `OMA`: an application of a symbol to argument (arity > 0, `\STEXInternalTermMathOMAiii`)

- `OMB`: a binding application of a symbol that binds/declares one or more variables (argument string contains `b` or `B`, `\STEXInternalTermMathOMBiii`).

The arguments are marked with `\STEXInternalTermMathArgiii` (`i` or `b`) or `\STEXInternalTermMathAssocArgiiiii` (`a` or `B`). Finally, the notation is closed with `\STEXInternalSymbolAfterInvokationTL`.

How this works is best explained by example.

Example 23

Assume we have a symbol and notation:

```
1 \symdecl{somesymbol}[args=iai]
2 \notation{somesymbol}[prec=10;20x30x40,variant=foo]
3 {First: #1; Second: #2; Third: #3; End}
4 {(#1 -- ##1 split ##2 -- #3)}
```

Since the symbol corresponds to an `OMA`, the whole notation is wrapped in `\STEXInternalTermMathOMAiii`, taking as arguments the variant identifier (`foo`), the operator precedence (`10`) and the body of the notation.

The second argument in the notation, being associative, is wrapped in a `\STEXInternalTermMathAssocArgiiiii`, taking as arguments the argument number (`2`), the precedence (`30`), the TeX parameter token (`#2`) the notation body (`(#1 -- ##1 split ##2 -- #3)`), and finally the argument mode (`a`). Additionally, the markers `##1` and `##2` are replaced by `\STEXInternalAssocArgMarkerI` and `\STEXInternalAssocArgMarkerII`, respectively.

Subsequently, the non-sequence parameter tokens are wrapped in `\STEXInternalTermMathArgiii` with arguments `mj` (where `m` is the mode und `j` the index), the precedence (`20` or `40` respectively), and the parameter token.

Finally, a `\STEXInternalSymbolAfterInvokationTL` is inserted.

The final expansion of `\l_stex_notation_macrocode_cs` is thus:

```
1 \STEXInternalTermMathOMAiii{foo}{10}{
2   First: \STEXInternalTermMathArgiii{i1}{20}{#1};
3   Second: \STEXInternalTermMathAssocArgiiiii{2}{30}{#2}{
4     (\STEXInternalTermMathArgiii{i1}{20}{##1} --
```

```
 5        \STEXInternalAssocArgMarkerI split
 6        \STEXInternalAssocArgMarkerII --
 7        \STEXInternalTermMathArgiii{i3}{40}{##3})
 8      }{a};
 9      Third: \STEXInternalTermMathArgiii {i3}{40}{#3}; End
10     }\STEXInternalSymbolAfterInvokationTL
```

\stex_notation_top:nnw

\stex_notation_top:nnw {⟨*symboluri*⟩}{⟨*code*⟩}

calls \stex_notation_parse_and_then:nw{⟨*code*⟩} and, adds the notation for the symbol with URI ⟨*symboluri*⟩ to the current module and exports it to the HTML (if applicable).

11.5   Structural Features

\stex_structural_feature_module:nn
\stex_structural_feature_module_end:
\g_stex_last_feature_str

\stex_structural_feature_module:nn {⟨*name*⟩}{⟨*typeid*⟩}

opens a new module-like structural feature of type ⟨*typeid*⟩ with name ⟨*name*⟩, which needs to be closed with \stex_structural_feature_module_end:.

Its body behaves like a nested module with name ⟨*modulename*⟩/⟨*name*⟩, the full URI of which is stored in \g_stex_last_feature_str for subsequent elaboration.

`\stex_structural_feature_morphism:nnnnn`
`\stex_structural_feature_morphism_end:`
`\l_stex_current_domain_str`
`\l_stex_feature_name_str`
`\l_stex_morphism_symbols_prop`
`\l_stex_morphism_renames_prop`
`\l_stex_morphism_morphisms_seq`

`\stex_structural_feature_morphism:nnnnn`
`{⟨moprhismname⟩}{⟨typeid⟩}{⟨archive⟩}{⟨domain⟩}{⟨annotations⟩}`

opens a new morphism-like structural feature of type ⟨*typeid*⟩ with name ⟨*morphismname*⟩ and the module `[⟨`*archive*`⟩]{⟨`*domain*`⟩}` as domain, which needs to be closed with `\stex_-structural_feature_morphism_end:`.

Deactivates `\symdecl`, `\textsymdecl`, `\symdef`, `\notation` and `smodule`, and activates `\assign`, `\assignMorphism` and `\renamedecl`.

Defines the following macros:

- `\l_stex_feature_name_str` `={⟨`*name*`⟩}`.

- `\l_stex_current_domain_str` `=` the full uri of ⟨*domain*⟩.

- `\l_stex_morphism_symbols_prop`: This property list is initialized as follows: For every symbol transitively included in ⟨*domain*⟩ with data ⟨*module*⟩, ⟨*name*⟩, ⟨*id*⟩, ⟨*arity*⟩, ⟨*args*⟩, ⟨*definiens*⟩, ⟨*type*⟩, and ⟨*return code*⟩, the property list contains an entry with key `[⟨`*module*`⟩]/[⟨`*name*`⟩]` and value `{⟨`*id*`⟩}{⟨`*arity*`⟩}{⟨`*args*`⟩}{⟨`*definiens*`⟩}` `{⟨`*type*`⟩}{⟨`*return code*`⟩}`.

- `\l_stex_morphism_renames_prop`: An initially empty property list.

- `\l_stex_morphism_morphisms_seq`: TODO

At `\stex_structural_feature_morphism_end:`, the elaboration is computed from the above data thusly:

For every entry in `\l_stex_morphism_symbols_prop`, a new symbol is created with the values ⟨*arity*⟩, ⟨*args*⟩, ⟨*definiens*⟩, ⟨*type*⟩ and ⟨*return code*⟩ from that property list, and either:

- if `\l_stex_morphism_renames_prop` does *not* contain an entry with key ⟨*module*⟩?⟨*name*⟩, then the elaborated name is ⟨*morphismname*⟩/⟨*name*⟩ and its ⟨*id*⟩ is empty (no semantic macro is generated), or

- if `\l_stex_morphism_renames_prop` contains an entry with key ⟨*module*⟩?⟨*name*⟩, then its value needs to be of the form `{⟨`*id*`⟩}{⟨`*name*`⟩}`, which are used for the elaborated symbol.

All notations of the symbols transitively included in the domain are copied over to their elaborations.

11.6 Imports and Morphisms

`\stex_module_add_morphism:nnnn`
`\stex_module_add_morphism:(nonn|ooox)`

---

adds a new module morphism (i.e. inheritance, possibly with modification) to the current module

`#1` : The name of the morphism (may be empty for e.g. `\importmodule`, but may be named for e.g. `copymodule`),

`#2` : the URI of the module being "imported",

`#3` : the "type" of the morphism (e.g. `import` or `copymodule`),

`#4` : a list of assignments as pairs `{`⟨*source*⟩`}{`⟨*target*⟩`}` that signify that the symbol with full URI ⟨*source*⟩ is being mapped or elaborated to the new symbol with name ⟨*target*⟩ in the current module. May be empty for e.g. `\importmodule`.

The provided arguments are stored in `\c_stex_module_`⟨*uri*⟩`_morphisms_prop` with key `#1` (if non-empty) or `[#2]` (if `#1` is empty) and value `{#1}{#2}{#3}{#4}`

Import-like statements in SᴛEX are usually given as pairs `[`⟨*archive*⟩`]{`⟨*path*⟩`?`⟨*module*⟩`}`, which be relative to the current archive and/or file path. For persistence and sms-mode, these pairs first need to be resolved into an *absolute* specification:

`\stex_import_module_uri:nn`
`\l_stex_import_archive_str`
`\l_stex_import_name_str`
`\l_stex_import_uri_str`
`\l_stex_import_path_str`

`\stex_import_module_uri:nn` `{`⟨*archive*⟩`}{`⟨*pathstring*⟩`}`

(where ⟨*archive*⟩ may be empty) resolves the given arguments into:

- `\l_stex_import_archive_str` the given archive id (in which case `\stex_require_-archive:n` is called), or the id of the current archive (if existent and ⟨*archive*⟩ empty), or empty,

- `\l_stex_import_uri_str` if an archive id is given, or we currently are in an archive, its corresponding namespace; otherwise `{file:}`,

- `\l_stex_import_path_str` the path of the URI relative to the given (or current) archive, or (if not existent) the absolute path of ⟨*pathstring*⟩ (without a module name) resolved relative to the current file's parent directory, and

- `\l_stex_import_name_str` the name of the module.

If ⟨*pathstring*⟩ does not contain the character `?`, the whole pathstring is assumed to be the name of the module and the path is empty (or the current file's parent directory, depending on the above).

`\stex_import_require_module:nnn`

---

takes as arguments values `\l_stex_import_archive_str`, `\l_stex_import_path_str` and `\l_stex_import_name_str` as computed by `\stex_import_module_uri:nn` and (optionally loads and) activates the corresponding module (or throws an error if it does not exist).

Most complex morphisms (`copymodule` et al) are implemented as structural features using `\stex_structural_feature_morphism:nnnn`.

```
\stex_get_in_morphism:n
\l_stex_get_symbol_macro_str
```

finds a symbol with the provided name or id in the domain of the current morphism. Sets the same macros as `\stex_get_symbol:n`, and additionaly `\l_stex_get_symbol_-macro_str` to the ⟨*id*⟩ of the symbol. Throws an error if no such symbol is found.

11.7 Expressions and Semantic Macros

```
\_stex_invoke_symbol:nnnnnnnN        \l_stex_current_symbol_str
\l_stex_current_arity_str
\l_stex_current_args_tl
\l_stex_current_return_tl
\l_stex_current_type_tl
```

```
\_stex_invoke_symbol:nnnnnnnN
{⟨module⟩}{⟨name⟩}{⟨arity⟩}{⟨args⟩}{⟨definiens⟩}
{⟨type⟩}
{⟨return code⟩}
{⟨\invokation_macro⟩}
```

is how a semantic macro is/should be defined. `\_stex_invoke_symbol:nnnnnnnN` first checks whether semantic macros are currently allowed, and throws an error if not. Otherwise, it sets the `\comp`-controlled highlighting to `\compemph@uri`, initializes `\STEXInternalSymbolAfterInvokationTL`, defines the following macros for all of its arguments, and subsequently calls the `\invokation_macro`:

- `\l_stex_current_symbol_str` ={⟨*module*⟩?⟨*name*⟩}

- `\l_stex_current_arity_str` ={⟨*arity*⟩}

- `\l_stex_current_args_tl` ={⟨*args*⟩}

- `\l_stex_current_type_tl` ={⟨*type*⟩}

- `\l_stex_current_return_tl` ={⟨*return code*⟩}

The simplest example for an `\invokation_macro` is `\stex_invoke_symbol:`.

```
\_stex_invoke_variable:nnnnnnN
```

analogous to `\_stex_invoke_symbol:nnnnnnnN`, but for variables; sets the `\comp`-controlled highlighting to `\varemph@uri`.

```
\stex_invoke_symbol:
```
branches based on the mode and following characters:

- If math, check next character:

  ! operator; defer to operator notation

  else defer to notation and check the value of `\l_stex_current_return_tl`=⟨*return*⟩.

  - ∗ If ⟨*return*⟩ is empty, call the notation macro,
  - ∗ otherwise, call ⟨*return*⟩{`\stex_invoke_symbol!`}.

- If text:

`\stex_invoke_sequence_range:`	
`\stex_invoke_sequence_in:`	

TODO

11.8 Optional (Key-Value) Argument Handling

LaTeX3 is surprisingly weak when it comes to handling optional (key-val) arguments in such a manner that *only* the freshly set macros are defined, and to modularly build up sets of argument keys. The following macros attempt to fix that:

`\stex_keys_define:nnnn`	`\stex_keys_define:nnnn` {⟨*id*⟩}{⟨*setup code*⟩}
`\stex_keys_set:nn`	{⟨*keyval setup code*⟩}{⟨*parents*⟩}

Defines a set of keys and their allowed values with identifier `stex/`⟨*id*⟩, that inherits from the sets with identifiers in ⟨*parents*⟩.

   `\stex_keys_set:nn` {⟨*id*⟩}{⟨*CSL*⟩} first executes ⟨*setup code*⟩ (e.g. to empty the macros holding the values) and then sets the keys in set ⟨*id*⟩ with the values provided in ⟨*CSL*⟩.

`\_stex_do_id:`   should be called whenever a macro or environment has a label id, i.e. calls `\stex_keys_-set:nn{id}{...}`, after the title has been typeset. Sets a `\label` by calling `\stex_-ref_new_doc_target:n{`⟨*id*⟩`}`.

> **Example 24**
>
>   If we define a set of keys with:
>
> ```
> 1 \stex_keys_define:nnnn{archive file}{
> 2   \str_clear:N \l_stex_key_archive_str
> 3   \str_clear:N \l_stex_key_file_str
> 4 }{
> 5   archive .str_set_x:N = \l_stex_key_archive_str ,
> 6   file .str_set_x:N = \l_stex_key_file_str
> 7 }{id}
> ```
>
> then calling `\stex_keys_set:nn{archive file}{id=foo,file=bar}` sets `\l_stex_-key_file_str={bar}`, assures that `\l_stex_key_archive_str` is empty, and executes the code associated with **id**, i.e. it sets `\l_stex_key_id_str={foo}`.

11.9 Stylable Commands and Environments

\stex_new_stylable_cmd:nnnn

\stex_new_stylable_cmd:nnnn {⟨*name*⟩}{⟨*arity*⟩}{⟨*code*⟩}
{⟨*default*⟩}

Creates a new macro \⟨*name*⟩ with expansion ⟨*code*⟩ taking ⟨*arity*⟩ many arguments, that is customizable in presentation by a user by calling \stexstyle⟨*name*⟩. On calling \stex_style_apply: executes the presentation code provided by a user.

⟨*code*⟩ should:

- Call \stex_keys_set:nn{style}{...} (or a keyset inheriting from style),

- set macros with prefix \this... that a user might want to use for presentation (e.g. \thistitle),

- call \stex_style_apply: at some point.

\stex_new_stylable_env:nnnnnnn

\stex_new_stylable_env:nnnnnnn {⟨*name*⟩}{⟨*arity*⟩}{⟨*begincode*⟩}
{⟨*endcode*⟩}{⟨*default begin*⟩}{⟨*default end*⟩}{⟨*prefix*⟩}

Like \stex_new_stylable_cmd:nnnn, but defines a new environment {⟨*prefix*⟩⟨*name*⟩} stylable via \stexstyle⟨*name*⟩. Should call \stex_style_apply: twice; once in the ⟨*begincode*⟩ and once in ⟨*endcode*⟩.

\stex_style_apply: Sets \thisstyle to be the head of the CSL \l_stex_key_style_clist and checks whether a style for the current stylable macro/environment has been defined; if not, executes the code for the default style.

> **Example 25**
> \importmodule is defined something like the following:
>
> ```
> 1 \stex_new_stylable_cmd:nnnn{importmodule}{O{} m}{
> 2   ...
> 3   \def\thismoduleuri{...}
> 4   \def\thismodulename{...}
> 5   \stex_style_apply:
> 6   ...
> 7 }{}
> ```
>
> A user can then customize the output generated by \importmodule (by default none) via \stexstyleimportmodule{...\thismodulename...}.

> **Example 26**
> smodule does something like
>
> ```
> 1 \stex_new_stylable_env:nnnnnnn{module}{O{} m}{
> 2   ...
> 3   \def\thismoduleuri{...}
> 4   \def\thismodulename{...}
> 5   \stex_style_apply:
> 6   ...
> 7 }{
> 8   ...
> ```

```
 9    \stex_style_apply:
10    ...
11 }{}{}{s}
```

which defines the environment name to be `smodule` and generates `\stexstylemodule`.

11.10   Math Archives

Math archives are represented as LaTeX3 property lists, the keys/values of which correspond to the entries in the archive's manifest file. The most important fields are

- `id`,

- `narr` the document namespace,

- `ns` the content namespace, and

- `docurl` the document URL base.

`\stex_resolve_path_pair:Nnn`
`\stex_resolve_path_pair:Nxx`

`\stex_resolve_path_pair:Nnn` {⟨\target⟩}{⟨*archive-id*⟩}{⟨*pathstring*⟩}

computes the absolute file path of ⟨*pathstring*⟩ relative to the **source**-folder of ⟨*archive-id*⟩ (if non-empty), or the current archive (if existent) or the parent working directory (otherwise), and stores the result in `\target`.

`\l_stex_current_archive_prop`

`\l_stex_current_archive_prop` always points to the current math archive or is `\undefined`, if the current file is not part of a math archive.

`\c_stex_main_archive_prop` `\c_stex_main_archive_prop` represents the math archive in which the main file resides (if existent).

`\stex_require_archive:n`
`\stex_require_archive:o`

`\stex_require_archive:n` {⟨*id*⟩}

looks for a math archive ⟨*id*⟩ in the MathHub directory, parses its manifest file, creates the corresponding property list `\c_stex_mathhub_`⟨*id*⟩`_manifest_prop`, and throws a fatal error if the archive is not found.

   If the archive has been found and parsed before, does nothing, so it is cheap and safe to call repeatedly for the same id.

`\stex_set_current_archive:n`

`\stex_set_current_archive:n` {⟨*id*⟩}

Calls `\stex_require_archive:n`{⟨*id*⟩} and sets `\l_stex_current_archive_prop`.

`\stex_in_archive:nn`

`\stex_in_archive:nn` {⟨*opt-id*⟩}{⟨*code*⟩}

Executes ⟨*code*⟩ in the context of math archive ⟨*opt-id*⟩ (using `\stex_require_-archive:n`), i.e. iff ⟨*opt-id*⟩ is non-empty, changes the current archive to the one with id ⟨*opt-id*⟩, call ⟨*code*⟩ with ⟨*opt-id*⟩ as argument (in `#1`) and changes it back afterwards.

   If ⟨*opt-id*⟩ is empty, ⟨*code*⟩ is called with the id of the *current* math archive as `#1`, or with `#1` empty if there is no current math archive.

11.11   SMS-Mode

sTeX has to extract formal content (i.e. modules and their symbols) from LaTeX-files, that may otherwise contain arbitrary code, including macros that may not be defined unless the file is fully processed by TeX. Those modules and symbols also may depend on other modules that have not yet been loaded. The naive way to achieve this, which would be to just suppress output (e.g. by storing it in a box register) and then `\input` the required file, does not work thanks to TeX's limited *file stack*, which would overflow quickly for modules that have a deeply nested list of dependencies.

To solve those problems, sTeX reads dependencies in what we call *sms mode*, which can be summarized thusly:

- In a first pass, we parse the file token by token, ignoring everything other than a select list of macros and environments that introduce dependencies (such as `\importmodule` and `\begin{`smodule`}[sig=...]`). Instead of loading those, we remember them for later.

- After the file as been fully parsed thusly, the dependencies found are loaded, again in sms-mode.

- In a second pass, we parse the file *again* in the same way, but this time execute all macros that are explicitly allowed in sms mode, such as `\importmodule`, `\symdecl`, `\notation`, `\symdef`, etc.

- all this parsing happens additionally in a `\setbox\throwaway\vbox{...}`-block to suppress any accidental output.

This means that TeX's input stack never grows by more than +1, but still behaves *as if* the dependencies were loaded recursively, at the detriment of being somewhat slow.

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

tests for whether we are currently in sms-mode.

`\stex_file_in_smsmode:nn`
`\stex_file_in_smsmode:on`

`\stex_file_in_smsmode:nn` {⟨*filestring*⟩}{⟨*setup-code*⟩}

sets up sms-mode and internal grouping, calls ⟨*setup-code*⟩ and subsequently processes the file ⟨*filestring*⟩ in sms-mode as described above.

11.11.1   Second Pass

`\stex_sms_allow:N`

`\stex_sms_allow:N` {⟨\macro⟩}

registers the `\macro`-command to be allowed in sms mode.

This only works, if `\macro` takes no arguments and/or does not touch the subsequent tokens.

For macros taking arguments, we can use

124

\stex_sms_allow_escape:N

`\stex_sms_allow_escape:N` `{⟨\macro⟩}`

registers the `\macro`-command to be allowed in sms mode.

If `\macro` is subsequently encountered in sms-mode, parsing is halted and `\macro` can process arguments as desired. It then needs to continue parsing manually though, by calling `\stex_smsmode_do:` as (usually) its last token.

\stex_sms_allow_env:n

`\stex_sms_allow_env:n` `{⟨envname⟩}`

registers the environment ⟨*envname*⟩ to be allowed in sms mode.

As with `\stex_sms_allow_escape:N`, the `\begin{⟨envname⟩}` is escaped, hence the begin-code of the environment needs to call `\stex_smsmode_do:`. Since `\end{⟨envname⟩}` never takes arguments, it does not need to be escaped.

\stex_smsmode_do:

continues with sms-mode-style parsing. Does nothing if not in sms-mode, and is therefore safe to be called "just in case".

11.11.2 First Pass

\stex_sms_allow_import:Nn
\stex_sms_allow_import_env:nn

behave like `\stex_sms_allow_escape:N` and `\stex_sms_allow_env:n` respectively, but the macro or environment provided is now allowed in the *first* pass of sms-mode.

This macro should process arguments, add content to `\g_stex_sms_import_code`, and finally call `\stex_smsmode_do:`.

The code provided in the *second* argument is called before the first pass of sms-mode, as to set up functionality for these macros. For example, `\importmodule` provides code that redefines `\importmodule` to store the identified dependency in `\g_stex_sms_-import_code` instead of activating it directly.

\g_stex_sms_import_code

is built up in the first pass of sms mode and called subsequently; before the second pass.

Code in this token list should load and activate dependencies found in the first pass.

11.12 Strings, File Paths, URIs

\stex_str_if_starts_with_p:nn ⋆
\stex_str_if_starts_with:nn_TF_ ⋆

`\stex_str_if_starts_with:nn` `{⟨first⟩}{⟨second⟩}`

Checks whether the string ⟨*first*⟩ starts with the string ⟨*second*⟩ (i.e. ⟨*second*⟩ is a prefix of ⟨*first*⟩).

\stex_str_if_ends_with_p:nn ⋆
\stex_str_if_ends_with:nn_TF_ ⋆

`\stex_str_if_ends_with:nn` `{⟨first⟩}{⟨second⟩}`

Checks whether the string ⟨*first*⟩ ends with the string ⟨*second*⟩ (i.e. ⟨*second*⟩ is a suffix of ⟨*first*⟩).

11.12.1 File Paths

File paths are represented as LaTeX3 sequences. The following methods make sure to

- canonicalize paths, i.e. resolve `..` and `.` segments,

- set all category codes to 12 (other), and

- tranform windows file paths containing `\` uniformly to `/`.

`\stex_file_resolve:Nn`
`\stex_file_resolve:(No|Nx)`

`\stex_file_resolve:Nn` {⟨\macro⟩}{⟨*string*⟩}

resolves and canonicalizes the file path string ⟨*string*⟩ and stores the result in `\macro`.

`\stex_file_set:Nn`
`\stex_file_set:(No|Nx)`

`\stex_file_set:Nn` {⟨\macro⟩}{⟨*string*⟩}

represents an already canonicalized file path string as a LaTeX3 sequence and stores it in `\macro`.

`\stex_if_file_absolute_p:N` ⋆
`\stex_if_file_absolute:NTF` ⋆

`\stex_if_file_absolute:N` tests whether the given file path (represented as a canonicalized LaTeX3 sequence) is an absolute file path.

`\stex_file_use:N` ⋆ `\stex_file_use:N` expands to a string representation of the given file path.

`\stex_if_file_starts_with:NNTF`

`\stex_if_file_starts_with:NN` {⟨\first⟩}{⟨\second⟩}

tests whether the file path `\first` is a child of `\second`. *(Not expandable)*

`\stex_file_split_off_ext:NN`

`\stex_file_split_off_ext:NN` {⟨\target⟩}{⟨\source⟩}

splits off the file extension of `\source` and stores the resulting file path in `\target`

`\stex_file_split_off_lang:NN`

`\stex_file_split_off_lang:NN` {⟨\target⟩}{⟨\source⟩}

checks whether the file path `\source` ends with a language abbreviation (e.g. `.en`), if so removes it, and stores the result in `\target`.

The following are primarily used in file hooks, but might occasionally be useful to call manually:

`\stex_filestack_push:n` pushes the given file to the file stack, recomputing `\g_stex_current_file`, the current language, document URI and namespace.

`\stex_filestack_pop:` pops the current top entry of the file stack. If the file stack is empty, resets to `\c_stex_-main_file`.

File Path Constants and Variables

`\c_stex_pwd_file`
`\c_stex_main_file` store the parent working directory and the absolute path of the main file being processed (with guessed file extension `.tex`).

`\c_stex_home_file` stores the user's home directory.

`\c_stex_mathhub_file` stores the user's MathHub directory; its string representation is stored in `\mathhub`.

`\g_stex_current_file` always points to the *current* file.

11.12.2   URIs

Mᴍᴛ URIs are represented as token lists of the form
    `{\__stex_path_auth:n{`⟨*authority*⟩`} \__stex_path_path:n{`⟨*path*⟩`} \__stex_path_-module:n{`⟨*modulename*⟩`} \__stex_path_name:n{`⟨*declname*⟩`}}`,
    all of which may be empty. Largely, URIs are used as strings only, but the above representation is used in `\stex_uri_resolve:Nn` to canonicalize URIs when they are computed the first time.

`\stex_map_uri:Nnnnn`
`\stex_map_uri:Nnnnn {`⟨`\uri`⟩`}{`⟨*authority code*⟩`}`
`{`⟨*path code*⟩`}{`⟨*modulename code*⟩`}{`⟨*declname code*⟩`}`

executes the provided ⟨*code*⟩s with the components of the `\uri` as arguments.

`\stex_uri_resolve:Nn`
`\stex_uri_resolve:(No|Nx)` behaves analogously to `\stex_file_resolve:Nn`.

`\stex_uri_set:Nn`
`\stex_uri_set:(No|Nx)` behaves analogously to `\stex_file_set:Nn`.

`\stex_uri_use:N` ⋆ behaves analogously to `\stex_file_use:N`.

A common usage of URIs is computing the namespace of content elements (modules or documents) from the namespace of a math archive and some relative file path within that archive.

`\stex_uri_from_repo_file:NNNn`
`\stex_uri_from_repo_file:NNNn {`⟨`\target`⟩`}{`⟨`\repo_prop`⟩`}{`⟨`\filepath`⟩`}`
`{`⟨`ns_field`⟩`}`

computes the namespace URI from the property list `\repo_prop` of some math archive, the file path `\filepath` and the archive field `{`⟨*ns_field*⟩`}` (**narr** or **ns**), and stores the result in `\target`.

`\stex_uri_from_repo_file_nolang:NNNn`

behaves like `\stex_uri_from_repo_file:NNNn`, but makes sure to split off language abbreviations from the file name (e.g. `.en`).

`\stex_uri_from_current_file:Nn`
`\stex_uri_from_current_file_nolang:Nn`

Special cases for `\stex_uri_from_repo_file[_nolang]:NNNn`, for `\repo_prop`=`\l_-stex_current_archive_prop` and `\filepath`=`\g_stex_current_file`.

`\stex_set_document_uri:` sets the current value of `\l_stex_current_doc_uri` based on the current file and archive.

`\stex_set_current_namespace:`

sets the current value of `\l_stex_current_ns_uri` based on the current file and archive.

`\stex_uri_add_module:NNn`
`\stex_uri_add_module:NNo`
`\stex_uri_add_module:NNn` {⟨\target⟩}{⟨\uri⟩}
{⟨*name*⟩}

Checks that URI `\uri` has no module name, adds the provided ⟨*name*⟩ and stores the result in `\target`.

URI Constants and Variables

`\l_stex_current_doc_uri` always points to the current document URI.

`\l_stex_current_ns_uri` always points to the current content namespace.

11.13 Language Handling

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

Property lists converting babel languages to/from their abreviations; e.g.

- `\prop_item:Nn \c_stex_languages_prop` {de} yields ngerman, and

- `\c_stex_language_abbrevs_prop` {ngerman} yields de.

`\l_stex_current_language_str`

always stores the current language.

`\stex_set_language:n`
`\stex_set_language:(x|o)`

`\stex_set_language:n {⟨abbrev⟩}`

Sets `\l_stex_current_language_str`, and, if the babel package is loaded, calls `\selectlanguage` on the language corresponding to `{⟨abbrev⟩}`.

Note that the package option `lang=` automatically loads the babel package.

If `⟨abbrev⟩=tr`, additionally call `\selectlanguage` with the option `shorthands=:!`.

Throws `error/unknownlanguage` if no language with abbreviation `{⟨abbrev⟩}` is known.

`\stex_language_from_file:` infers the current language from file ending (e.g. `.en.tex`) and sets it appropriately.

Is called in a file hook, i.e. always switches language when inputting a file `.<lang>.<ext>`.

11.14   Inserting Annotations

sTeX can be used to produce either HTML or PDF. In HTML-mode, multiple macros exist to insert annotations. The same macros are also valid in PDF mode but implemented as null operations.

`\stex_suppress_html:n`

`\stex_suppress_html:n {⟨code⟩}`

turns annotations off temporarily in ⟨code⟩ (e.g. as to not generate additional annotations for elaborated declarations, or in sms-mode).

For that to work, code that inserts annotations should use

`\stex_if_do_html_p:` ⋆
`\stex_if_do_html:TF` ⋆
tests whether to generate HTML annotations.

`\stex@backend` should be set by a backend engine, such that a file `stex-backend-\stex@backend.cfg` exists.

11.14.1   Backend macros

Such a backend config file should provide the following:

`\stex_if_html_backend_p:` ⋆
`\stex_if_html_backend:TF` ⋆
can be used to determine whether the backend produces HTML (e.g. R$_{U}$sTeX or LaTeXML) or not (e.g. `pdflatex`).

`\ifstexhtml` is set accordingly.

`\stex_annotate:nnn`

`\stex_annotate:nnn {⟨attr⟩}{⟨value⟩}{⟨code⟩}`

In HTML mode, annotates the output of ⟨code⟩ with the XML-attribute ⟨attr⟩="⟨value⟩". In PDF mode, just calls ⟨code⟩.

`\stex_annotate_invisible:nnn` `\stex_annotate_invisible:n`	`\stex_annotate_invisible:n {⟨code⟩}`

Should annotate ⟨*code*⟩ with `shtml:visible="false"` `style="display:none;"`. In PDF mode, does nothing.

`\stex_annotate_invisible:nnn` combines `\stex_annotate_invisible:n` and `\stex_-annotate:nnn`.

stex_annotate_env (*env.*)	`\begin{stex_annotate_env}{⟨attr⟩}{⟨value⟩}` ⟨*code*⟩ `\end{stex_annotate_env}`

should behave like `\stex_annotate:nnn{⟨attr⟩}{⟨value⟩}{⟨code⟩}`

`\stex_mathml_intent:nn` `\stex_mathml_arg:nn`	MathML Intent (TODO)

11.15 Persisting Content from Math Archives in sms-Files

`\stex_persist:n` `\stex_persist:e`	`\stex_persist:n {⟨code⟩}`

writes ⟨*code*⟩ to the `\jobname.sms`-file, if `writesms` is active.

TEXhackers note: ⟨*code*⟩ is being read with `expl3` category codes (except for spaces having catcode 10), but not pretokenized; i.e. ⟨*code*⟩ can safely change the current catcode scheme.

`\c_stex_persist_mode_bool` `\c_stex_persist_write_mode_bool`

whether `usesms` or `writesms` are active.

11.16 Utility Methods

`\stex_macro_body:N` ⋆	expands to the *expansion* of the provided macro, including parameter tokens, with the original category codes intact; e.g. if `\def\foo#1{First #1}`, then `\stex_macro_body:N` `\foo` expands to `First #1`.

`\stex_macro_definition:N` ⋆	expands to the token list *defining* the provided macro, including parameters, command attributes (i.e. `\long`, `\protected`), with the original category codes intact; e.g. if `\protected\def\foo#1{First #1}`, then `\stex_macro_definition:N` `\foo` expands to `\protected\def\foo#1{First #1}`.

TEXhackers note: Does not work with "higher" parameter tokens, i.e. `##1`, `####1` etc.

\stex_deactivate_macro:Nn
\stex_reactivate_macro:N

`\stex_deactivate_macro:Nn` {⟨`\macro`⟩} {⟨`msg`⟩}

Makes `\macro` throw an error message `error/deactivated-macro`{⟨*msg*⟩}, notifying an author that the macro is only allowed in certain environments.

`\stex_reactivate_macro:N` restores the functionality of the macro.

\stex_kpsewhich:Nn

`\stex_kpsewhich:Nn` {⟨`\macro`⟩} {⟨`args`⟩}

Calls "`kpsewhich` ⟨*args*⟩" and stores the result in `\macro`,

TₑXhackers note: Does not require `shell-escape`

\stex_get_env:Nn

`\stex_get_env:Nn` {⟨`\macro`⟩} {⟨`envvar`⟩}

Stores the value of the environment variable ⟨*envvar*⟩ in `\macro`.

\stex_fatal_error:n
\stex_fatal_error:nnn
\stex_fatal_error:nxx

Mimic the `\msg_error:`-macros, but make sure that TₑX stops processing.

TₑXhackers note: Calls `\input{non-existent file}`.

\stex_ignore_spaces_and_pars:

As the name suggests, ignores all subsequent spaces and `\pars` until the first non-expandable macro is encountered.

Useful for e.g. ending `\symdecl` and related macros with, so that formatting sources with empty lines does not cause paragraph breaks.

11.16.1 Group-like Behaviours

\stex_pseudogroup_with:nn

`\stex_pseudogroup_with:nn` {⟨`macros`⟩}{⟨`code`⟩}

Calls ⟨*code*⟩ and subsequently restores the values of the ⟨*macros*⟩ given.

TₑXhackers note: Does *not* work recursively!

\stex_pseudogroup:nn

`\stex_pseudogroup:nn` {⟨`code1`⟩}{⟨`code2`⟩}

Expands ⟨*code2*⟩, and inserts the result after ⟨*code1*⟩. Works recursively and allows for restoring the values of macros in combination with `\stex_pseudogroup_restore:N`, but *only for macros that take no arguments*:

\stex_pseudogroup_restore:N ⋆

`\stex_pseudogroup_restore:N` {⟨`\macro`⟩}

Example 27

```
1 \stex_pseudogroup:nn{
2   something changing \foo
3   something changing \num
4 }{
5   \stex_pseudogroup_restore:N\foo
6   \int_set:Nn \num {\int_use:N \num}
7 }
```

restores the values of macro `\foo` and register `\num` after calling the first block.

    Commands like `\symdecl` and `\importmodule` that generate (semantic) macros should be local *to the current module*, e.g. `smodule`. For that purpose, we open a new "metagroup" with some identifier (e.g. `\l_stex_current_module_str`) and then execute the relevant code *in the metagroup with that identifier*:

`\stex_metagroup_new:n`
`\stex_metagroup_new:o`

`\stex_metagroup_new:n {⟨id⟩}`

Opens a new metagroup at the current TEX group level with identifier ⟨*id*⟩.

`\stex_metagroup_do_in:nn`
`\stex_metagroup_do_in:nx`

`\stex_metagroup_do_in:nn {⟨id⟩}{⟨code⟩}`

Executes ⟨*code*⟩ and adds its content to `\aftergroup` up until the TEX group level of the metagroup with identifier ⟨*id*⟩.

Chapter 12

Additional Packages

12.1 NotesSlides Documentation

TODO

12.2 Problem Documentation

TODO

12.3 HWExam Documentation

TODO

12.4 Tikzinput Documentation

TODO

Part IV

Implementation

Chapter 13

The sTeX Implementation

13.1 Setting up

Setup code for the document class

```
1 ⟨∗cls⟩
2 %%%%%%%%%%%%   stex.dtx   %%%%%%%%%%%%
3
4 \RequirePackage{expl3,l3keys2e}
5 \ProvidesExplClass{stex}{2023/03/19}{3.3.0}{sTeX document class}
6
7 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
8 \ProcessOptions
9
10 \RequirePackage{stex}
11
12 \LoadClass{article}
13 ⟨/cls⟩
```

Setup code for the package

```
14 ⟨∗package⟩
15 \RequirePackage{expl3,l3keys2e,ltxcmds}
16 \ProvidesExplPackage{stex}{2023/03/19}{3.3.0}{sTeX package}
17 \RequirePackage{stex-logo} % externalized for backwards-compatibility reasons
18 \RequirePackage{standalone}
19
20 \message{^^J*~This~is~sTeX~version~3.3.0~*^^J}
```

Package options:

```
21 \keys_define:nn { stex / package } {
22   debug      .str_set_x:N  = \c_stex_debug_clist ,
23   lang       .clist_set:N  = \c_stex_languages_clist ,
24   mathhub    .tl_set_x:N   = \mathhub ,
25   usesms     .bool_set:N   = \c_stex_persist_mode_bool ,
26   writesms   .bool_set:N   = \c_stex_persist_write_mode_bool ,
27   checkterms .bool_set:N   = \c_stex_check_terms_bool ,
28   image      .bool_set:N   = \c_tikzinput_image_bool,
29   unknown    .code:n       = {}
30 }
```

```
31 \exp_args:NNo \clist_set:Nn \c_stex_debug_clist \c_stex_debug_clist
32 \ProcessKeysOptions { stex / package }
```

Error messages:

```
33 \input{stex-en.ldf}
```

13.2   Utilities

```
34 \cs_set_eq:NN \stex_undefined: \undefined
```

13.2.1   Calling kpsewhich and Environment Variables

```
35 ⟨@@=stex_envs⟩
```

\stex_kpsewhich:Nn

```
36 \cs_new_protected:Nn \stex_kpsewhich:Nn {\group_begin:
37   \catcode`\ =12
38   \sys_get_shell:nnN { kpsewhich ~ #2 } { } \l_tmpa_tl
39   \tl_gset_eq:NN \l_tmpa_tl \l_tmpa_tl
40   \group_end:
41   \exp_args:NNo\str_set:Nn #1 {\l_tmpa_tl}
42   \tl_trim_spaces:N #1
43 }
```

(*End definition for* \stex_kpsewhich:Nn. *This function is documented on page 131.*)

\stex_get_env:Nn

```
44 \sys_if_platform_windows:TF{
45   \cs_new_protected:Nn \stex_get_env:Nn {\group_begin:
46     \escapechar=-1\catcode`\\=12
47     \exp_args:NNe \stex_kpsewhich:Nn #1 {-expand-var~\c_percent_str#2\c_percent_str}
48     \exp_args:NNx\use:nn\group_end:{
49       \str_set:Nn \exp_not:N #1 { #1 }
50     }
51   }
52 }{
53   \cs_new_protected:Nn \stex_get_env:Nn {
54     \stex_kpsewhich:Nn #1 {-var-value~#2}
55   }
56 }
```

(*End definition for* \stex_get_env:Nn. *This function is documented on page 131.*)

13.2.2   Logging

```
57 ⟨@@=stex_debug⟩
```

\stex_debug:nn

```
58 \cs_new_protected:Nn \stex_debug:nn {
59   \exp_args:NNo \clist_if_in:NnTF \c_stex_debug_clist { \tl_to_str:n{all} }{
60     \__stex_debug_:nn{#1}{#2}
61   }{
62     \exp_args:NNo \clist_if_in:NnT \c_stex_debug_clist { \tl_to_str:n{#1} }{
63       \__stex_debug_:nn{#1}{#2}
64     }
```

```
65      }
66    }
67
68    \cs_new_protected:Nn \__stex_debug_:nn {
69      \msg_set:nnn{stex}{debug / #1}{
70        \\Debug~#1:~#2\\
71      }
72      \msg_none:nn{stex}{debug / #1}
73    }
```

(*End definition for* \stex_debug:nn. *This function is documented on page 109.*)

\stex_fatal_error:n
\stex_fatal_error:nnn
\stex_fatal_error:nxx

To avoid dead locks etc., we throw errors and make tex stop entirely:

```
74    \cs_new_protected:Nn \stex_fatal_error:n {
75      \msg_error:nn{stex}{#1}\input{Fatal~Error!!}
76    }
77    \cs_new_protected:Nn \stex_fatal_error:nnn {
78      \msg_error:nnn{stex}{#1}{#2}{#3}\input{Fatal~Error!!}
79    }
80    \cs_generate_variant:Nn \stex_fatal_error:nnn {nxx}
```

(*End definition for* \stex_fatal_error:n *and* \stex_fatal_error:nnn. *These functions are documented on page 131.*)

We check an environment variable for debugging and set things up:

```
81    \stex_get_env:Nn\__stex_debug_env_str{STEX_DEBUG}
82    \str_if_empty:NTF\__stex_debug_env_str {
83      \clist_set_eq:NN \l__stex_debug_cl \c_stex_debug_clist
84    }{
85      \clist_set:No \l__stex_debug_cl {\__stex_debug_env_str}
86    }
87    \clist_clear:N \c_stex_debug_clist
88    \clist_map_inline:Nn \l__stex_debug_cl {
89      \exp_args:NNo \clist_put_right:Nn \c_stex_debug_clist
90      { \tl_to_str:n{#1} }
91    }
92
93    \exp_args:NNo \clist_if_in:NnTF \c_stex_debug_clist {\tl_to_str:n{all}} {
94      \msg_redirect_module:nnn{ stex }{ none }{ term }
95      \stex_debug:nn{all}{Logging~everything!}
96    }{
97      \clist_map_inline:Nn \c_stex_debug_clist {
98        \msg_redirect_name:nnn{ stex }{ debug / #1 }{ term }
99        \stex_debug:nn{#1}{Logging~#1}
100     }
101   }
```

13.2.3 Languages

```
102   ⟨@@=stex_lang⟩
```

\c_stex_languages_prop
\c_stex_language_abbrevs_prop

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

```
103   \exp_args:NNx \prop_const_from_keyval:Nn \c_stex_languages_prop { \tl_to_str:n {
104     en = english ,
105     de = ngerman ,
```

```
106    ar = arabic ,
107    bg = bulgarian ,
108    ru = russian ,
109    fi = finnish ,
110    ro = romanian ,
111    tr = turkish ,
112    fr = french
113 }}
114
115 \exp_args:NNx \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop { \tl_to_str:n {
116    english   = en ,
117    ngerman   = de ,
118    arabic    = ar ,
119    bulgarian = bg ,
120    russian   = ru ,
121    finnish   = fi ,
122    romanian  = ro ,
123    turkish   = tr ,
124    french    = fr
125 }}
126 % todo: chinese simplified (zhs)
127 %       chinese traditional (zht)
```

(*End definition for* `\c_stex_languages_prop` *and* `\c_stex_language_abbrevs_prop`. *These variables are documented on page* *128.*)

\l_stex_current_language_str

```
128 \str_new:N \l_stex_current_language_str
```

(*End definition for* `\l_stex_current_language_str`. *This variable is documented on page* *128.*)

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

\stex_set_language:n
\stex_set_language:x
\stex_set_language:o

```
129 \cs_new_protected:Nn \stex_set_language:n {
130    \str_set:Nn \l_stex_current_language_str { #1 }
131    \prop_if_in:NnTF \c_stex_languages_prop {#1} {
132      \tl_set_rescan:Nnx \l__stex_lang_lang_str {}{\prop_item:Nn \c_stex_languages_prop {#1}}
133      \cs_if_eq:NNTF\@onlypreamble\@notprerr{
134        \ltx@ifpackageloaded{babel}{
135          \exp_args:No\selectlanguage\l__stex_lang_lang_str
136        }{}
137      }{
138        \ltx@ifpackageloaded{babel}{}{
139          \str_if_eq:nnTF {#1} {tr} {
140            \RequirePackage[turkish,shorthands=:!]{babel}
141          }{
142            \RequirePackage[\l__stex_lang_lang_str]{babel}
143          }
144        }
145      }
146    }{
147      \msg_error:nnx{stex}{error/unknownlanguage}{#1}
148    }
149 }
150 \cs_generate_variant:Nn \stex_set_language:n {x,o}
```

138

(End definition for `\stex_set_language:n`*. This function is documented on page 129.)*

`\stex_language_from_file:`

```
151 \cs_new_protected:Nn \stex_language_from_file: {
152   \seq_get_right:NN \g_stex_current_file \l_tmpa_str
153   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
154   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % = ".tex/.dtx/.sty"
155   \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
156     \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
157       \exp_args:No \str_if_eq:nnF \l_tmpa_str {ltx} {
158         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
159       }
160     }
161   }
162   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
163   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
164     \seq_pop_right:NN \l_tmpa_seq \l__stex_lang_str
165     \str_if_eq:NNF \l__stex_lang_str \l_stex_current_language_str {
166       \exp_args:NNo \prop_if_in:NnT \c_stex_languages_prop \l__stex_lang_str {
167         \stex_set_language:o \l__stex_lang_str
168       }
169     }
170     \stex_debug:nn{lang} {Language~\l_stex_current_language_str~
171       inferred~from~file~name}
172   }
173 }
```

(End definition for `\stex_language_from_file:`*. This function is documented on page 129.)*

Loading babel:

```
174 \clist_if_empty:NF \c_stex_languages_clist {
175   \bool_set_false:N \l__stex_lang_turkish_bool
176   \seq_clear:N \l_tmpa_seq
177   \clist_map_inline:Nn \c_stex_languages_clist {
178     \str_set:Nx \l_tmpa_str {#1}
179     \str_if_eq:nnT {#1}{tr}{
180       \bool_set_true:N \l__stex_lang_turkish_bool
181     }
182     \prop_get:NoNTF \c_stex_languages_prop \l_tmpa_str \l_tmpa_str {
183       \tl_set_rescan:Nno \l_tmpa_str {} \l_tmpa_str
184       \seq_put_right:No \l_tmpa_seq \l_tmpa_str
185     } {
186       \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
187     }
188   }
189   \stex_debug:nn{lang} {Languages:~\seq_use:Nn \l_tmpa_seq {,~} }
190   \bool_if:NTF \l__stex_lang_turkish_bool {
191     \exp_args:NNe \use:nn \RequirePackage
192       {[main=\seq_use:Nn \l_tmpa_seq, ,shorthands=:!]}{babel}
193   }{
194     \exp_args:NNe \use:nn \RequirePackage
195       {[main=\seq_use:Nn \l_tmpa_seq, ]}{babel}
196   }
197 }
```

13.2.4 Group-like Behaviours

```
198 ⟨@@=stex_groups⟩
```

\stex_pseudogroup:nn
\stex_pseudogroup_restore:N

```
199 \cs_new_protected:Npn \stex_pseudogroup:nn {
200     \exp_args:Nne \use:nn
201 }
202 \cs_new:Nn \stex_pseudogroup_restore:N {
203     \tl_if_exist:NTF #1 {
204         \tl_set:Nn \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
205     }{
206         \cs_undefine:N \exp_not:N #1
207     }
208 }
```

(*End definition for* \stex_pseudogroup:nn *and* \stex_pseudogroup_restore:N. *These functions are documented on page 131.*)

\stex_pseudogroup_with:nn

```
209 \cs_new_protected:Nn \stex_pseudogroup_with:nn {
210     \tl_map_inline:nn{#1}{
211         \cs_set_eq:cN{__stex_groups_\tl_to_str:n{##1}}##1
212     }
213     #2
214     \tl_map_inline:nn{#1}{
215         \cs_set_eq:Nc##1{__stex_groups_\tl_to_str:n{##1}}
216         \cs_undefine:c{__stex_groups_\tl_to_str:n{##1}}
217     }
218 }
```

(*End definition for* \stex_pseudogroup_with:nn. *This function is documented on page 131.*)

\stex_metagroup_new:n
\stex_metagroup_new:o

List of all currently existing metagroup identifiers

```
219 \seq_new:N \l__stex_groups_ids_seq
```

start a new metagroup at the current group level with id #1

```
220 \cs_new_protected:Nn \stex_metagroup_new:n {
221     \str_set:cx{l__stex_groups_#1_int} {\int_use:N\currentgrouplevel}
222     \seq_put_right:Nn \l__stex_groups_ids_seq {#1}
223 }
224 \cs_generate_variant:Nn \stex_metagroup_new:n {o}
```

(*End definition for* \stex_metagroup_new:n. *This function is documented on page 132.*)

\stex_metagroup_do_in:nn
\stex_metagroup_do_in:nx

```
225 \prg_new_conditional:Nnn \__stex_groups_exists:n {TF} {
226     \str_if_exist:cTF{l__stex_groups_#1_int}
227         \prg_return_true: \prg_return_false:
228 }
229
230 \cs_new_protected:Nn \stex_metagroup_do_in:nn {
231     \__stex_groups_exists:nTF{#1}{
232         \__stex_groups_do_in:nn{#1}{#2}
233     }{
234         \msg_error:nnn{stex}{error/metagroup/missing}{#1}
```

```
235        }
236    }
237    \cs_generate_variant:Nn \stex_metagroup_do_in:nn {nx}
238
239    \cs_new_protected:Nn \__stex_groups_do_in:nn {
240      \exp_args:Nnx\stex_debug:nn{metagroup}{adding~to~\detokenize{#1}:^^J\tl_to_str:n{#2}}
241      \tl_set:Nn\__stex_groups_tmp{#2}
242      \exp_args:Nx \int_compare:nNnF {\use:c{l__stex_groups_#1_int}}
243          = \currentgrouplevel {
244          \tl_if_exist:cTF{g__stex_groups_#1_\the\currentgrouplevel _content}{
245            \exp_args:Nno \tl_gput_right:cn{g__stex_groups_#1_\the\currentgrouplevel _content}
246          }{
247            \exp_args:Nno \tl_gset:cn{g__stex_groups_#1_\the\currentgrouplevel _content}
248          }\__stex_groups_tmp
249          \bool_if_exist:cF {l__stex_groups_\the\currentgrouplevel _bool} {
250              \group_insert_after:N \__stex_groups_do:
251              \bool_set_true:c {l__stex_groups_\the\currentgrouplevel _bool}
252          }
253      }
254      \__stex_groups_tmp
255    }
256
257    \cs_new_protected:Nn \__stex_groups_do: {
258      \seq_map_inline:Nn \l__stex_groups_ids_seq {
259        \tl_if_exist:cT{g__stex_groups_##1_\int_eval:n{\currentgrouplevel+1}_content}{
260          \exp_args:NNno \exp_args:Nno \__stex_groups_do_in:nn{##1}{
261            \csname g__stex_groups_##1_\int_eval:n{\currentgrouplevel+1}_content\endcsname
262          }
263          \cs_undefine:c{g__stex_groups_##1_\int_eval:n{\currentgrouplevel+1}_content}
264        }
265        \bool_if_exist:cF {l__stex_groups_\int_eval:n\currentgrouplevel _bool} {
266          \group_insert_after:N \__stex_groups_do:
267          \bool_set_true:c {l__stex_groups_\int_eval:n\currentgrouplevel _bool}
268        }
269      }
270    }
```

(*End definition for* `\stex_metagroup_do_in:nn`*. This function is documented on page* *132.*)

13.2.5   HTML Annotations

```
271    ⟨@@=stex_annotate⟩
```

`\stex_if_do_html:`*TF*   Whether to (locally) produce HTML output

```
272    \bool_new:N \_stex_html_do_output_bool
273    \bool_set_true:N \_stex_html_do_output_bool
274
275    \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
276      \bool_if:nTF \_stex_html_do_output_bool
277        \prg_return_true: \prg_return_false:
278    }
```

(*End definition for* `\stex_if_do_html:TF`*. This function is documented on page* *129.*)

\stex_suppress_html:n Temporarily disable HTML output

```
279 \cs_new_protected:Nn \stex_suppress_html:n {
280   \stex_pseudogroup:nn{
281     \bool_set_false:N \_stex_html_do_output_bool
282     #1
283   }{
284     \stex_if_do_html:T {
285       \bool_set_true:N \_stex_html_do_output_bool
286     }
287   }
288 }
```

(*End definition for* \stex_suppress_html:n. *This function is documented on page 129.*)

We determine the backend:

\stex_if_html_backend_p:
\stex_if_html_backend:*TF*
\ifstexhtml
\stex@backend

```
289 \ifcsname if@rustex\endcsname\else
290   \expandafter\newif\csname if@rustex\endcsname
291   \@rustexfalse
292 \fi
293
294 \stex_get_env:Nn\__stex_annotate_env_str{STEX_FORCE_PDF}
295 \exp_args:No \str_if_eq:nnTF \__stex_annotate_env_str {true} {
296   \def\stex@backend{pdflatex}
297 }{
298   \tl_if_exist:NF\stex@backend{
299     \if@rustex
300       \def\stex@backend{rustex}
301     \else
302       \cs_if_exist:NTF\HCode{
303         \def\stex@backend{tex4ht}
304       }{
305         \def\stex@backend{pdflatex}
306       }
307     \fi
308   }
309 }
310
311 \input{stex-backend-\stex@backend.cfg}
312
313 \newif\ifstexhtml
314 \stex_if_html_backend:TF {
315   \stexhtmltrue
316   \bool_set_true:N \_stex_html_do_output_bool
317 }{
318   \stexhtmlfalse
319   \bool_set_false:N \_stex_html_do_output_bool
320 }
```

(*End definition for* \stex_if_html_backend:TF, \ifstexhtml, *and* \stex@backend. *These functions are documented on page 129.*)

_stex_annotate_force_break:n

```
321 \stex_if_html_backend:TF {
322   \cs_new_protected:Nn \_stex_annotate_force_break:n {
```

```
323      \stex_annotate_invisible:n{~}
324      #1
325      \stex_annotate_invisible:n{~}
326    }
327  }{
328    \cs_new_protected:Nn \_stex_annotate_force_break:n { #1 }
329  }
```

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

\mmlintent
\mmlarg
```
330  \stex_if_html_backend:TF {
331    \cs_new_protected:Npn \mmlintent #1 #2 {
332      \stex_annotate:nn{mml:intent={#1}}{#2}
333    }
334    \cs_new_protected:Npn \mmlarg #1 #2 {
335      \stex_annotate:nn{mml:arg={#1}}{#2}
336    }
337  }{
338    \cs_new_protected:Npn \mmlintent #1 #2 { #2 }
339    \cs_new_protected:Npn \mmlarg #1 #2 { #2 }
340  }
```

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

13.2.6   Auxiliary Methods

```
341  ⟨@@=stex_aux⟩
```

\stex_deactivate_macro:Nn
\stex_reactivate_macro:N
```
342  \cs_new_protected:Nn \stex_deactivate_macro:Nn {
343    \tl_set_eq:cN{\tl_to_str:n{#1}~-~orig}#1
344    \tl_set:Nn#1{
345      \msg_error:nnnn{stex}{error/deactivated-macro}{\detokenize{#1}}{#2}
346    }
347  }
348  \cs_new_protected:Nn \stex_reactivate_macro:N {
349    \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1}~-~orig\endcsname
350  }
```

(*End definition for* \stex_deactivate_macro:Nn *and* \stex_reactivate_macro:N. *These functions are documented on page* *131*.)

\stex_ignore_spaces_and_pars:
```
351  \protected\def\stex_ignore_spaces_and_pars:{
352    \begingroup\catcode13=10\relax
353    \@ifnextchar\par{
354      \endgroup\expandafter\stex_ignore_spaces_and_pars:\@gobble
355    }{
356      \endgroup
357    }
358  }
```

(*End definition for* \stex_ignore_spaces_and_pars:. *This function is documented on page* *131*.)

```
359 \cs_new_nopar:Nn \stex_keys_define:nnnn {
360   \tl_gset:cn {__stex_aux_keys_#1_pre_tl}{#2}
361   \tl_gset:cn {__stex_aux_keys_#1_def_tl}{#3}
362   \tl_if_empty:nF{#4}{
363     \clist_map_inline:nn{#4}{
364       \tl_set_eq:Nc \l__stex_aux_tl {__stex_aux_keys_##1_pre_tl}
365       \tl_gput_left:co{__stex_aux_keys_#1_pre_tl} \l__stex_aux_tl
366       \tl_set_eq:Nc \l__stex_aux_tl {__stex_aux_keys_##1_def_tl}
367       \tl_gput_left:cn{__stex_aux_keys_#1_def_tl} ,
368       \tl_gput_left:co{__stex_aux_keys_#1_def_tl} \l__stex_aux_tl
369     }
370   }
371   \tl_set_eq:Nc \l__stex_aux_tl {__stex_aux_keys_#1_def_tl}
372   \exp_args:Nno \keys_define:nn {stex / #1} {\l__stex_aux_tl}
373 }
```

(*End definition for* `\stex_keys_define:nnnn`*. This function is documented on page 121.*)

```
374 \cs_new_nopar:Nn \stex_keys_set:nn {
375   \use:c{__stex_aux_keys_#1_pre_tl}
376   \keys_set:nn {stex / #1} { #2 }
377 }
```

(*End definition for* `\stex_keys_set:nn`*. This function is documented on page 121.*)

Some ubiquitous key sets:

```
378 \stex_keys_define:nnnn{archive file}{
379   \str_clear:N \l_stex_key_archive_str
380   \str_clear:N \l_stex_key_file_str
381 }{
382   archive .str_set_x:N = \l_stex_key_archive_str ,
383   file    .str_set_x:N = \l_stex_key_file_str
384 }{}
385
386 \stex_keys_define:nnnn{id}{
387   \str_clear:N \l_stex_key_id_str
388 }{
389   id .str_set_x:N = \l_stex_key_id_str
390 }{}
391
392 \stex_keys_define:nnnn{title}{
393   \tl_clear:N \l_stex_key_title_tl
394 }{
395   title .tl_set:N = \l_stex_key_title_tl
396 }{}
397
398 \stex_keys_define:nnnn{style}{
399   \clist_clear:N \l_stex_key_style_clist
400 }{
401   style .clist_set:N = \l_stex_key_style_clist
402 }{}
403
404 \stex_keys_define:nnnn{deprecate}{
```

144

```
405    \str_clear:N \l_stex_key_deprecate_str
406  }{
407    deprecate      .str_set_x:N  = \l_stex_key_deprecate_str
408  }{}
409
410  \stex_keys_define:nnnn{uses}{}{
411    uses .code:n = {
412      \clist_map_inline:nn{#1}{
413        \stex_str_if_starts_with:nnTF{##1}[{
414          \__stex_aux_split_at_bracket:w ##1 \_stex_end:
415        }{
416          \usemodule{##1}
417        }
418      }
419    }
420  }{}
421
422  \cs_new_protected:Npn \__stex_aux_split_at_bracket:w [ #1 ] #2 \_stex_end: {
423    \usemodule[#1]{#2}
424  }
```

_stex_do_deprecation:n

```
425  \cs_new:Nn \_stex_do_deprecation:n {
426    \str_if_empty:NF \l_stex_key_deprecate_str {
427      \msg_warning:nnxx{stex}{warning/deprecated}{#1}{\l_stex_key_deprecate_str}
428    }
429  }
```

(*End definition for* _stex_do_deprecation:n. *This function is documented on page 109.*)

_stex_do_id:

```
430  \cs_new_protected:Nn \_stex_do_id: {
431    \stex_if_smsmode:F {
432      \str_if_empty:NF \l_stex_key_id_str {
433        \exp_args:No \stex_ref_new_doc_target:n \l_stex_key_id_str
434      }
435    }
436  }
```

(*End definition for* _stex_do_id:. *This function is documented on page 121.*)

\stex_new_stylable_env:nnnnnnn
\stex_new_stylable_cmd:nnnn
\stex_style_apply:

```
437  \cs_new_protected:Nn \stex_new_stylable_cmd:nnnn {
438    \exp_after:wN \newcommand \cs:w stexstyle#1 \cs_end:[2][]{
439      \__stex_aux_patch:nnn{#1}{##1}{##2}
440    }
441    \exp_after:wN \NewDocumentCommand\cs:w #1\cs_end:{#2}{
442      \cs_set:Npn \stex_style_apply: {
443        \__stex_aux_apply_patch:n{#1}
444      }
445      #3
446    }
447    \tl_set:cn {__stex_aux_style_#1:} { #4 }
448  }
```

145

```
449
450 \cs_new_protected:Nn \__stex_aux_apply_patch:n {
451   \clist_if_empty:NTF \l_stex_key_style_clist {
452     \tl_clear:N \thisstyle
453     \use:c{__stex_aux_style_#1:}
454   }{
455     \clist_get:NN \l_stex_key_style_clist \thisstyle
456     \tl_if_exist:cTF{__stex_aux_style_#1_\thisstyle :}{
457       \use:c{__stex_aux_style_#1_\thisstyle :}
458     }{
459       \use:c{__stex_aux_style_#1:}
460     }
461   }
462 }
463
464 \cs_new_protected:Nn \__stex_aux_patch:nnn {
465   \str_if_empty:nTF {#2}{
466     \tl_set:cn{__stex_aux_style_#1:}{#3}
467   }{
468     \tl_set:cn{__stex_aux_style_#1_#2:}{#3}
469   }
470 }
471
472 \cs_new_protected:Nn \stex_new_stylable_env:nnnnnnn {
473   \exp_after:wN \newcommand \cs:w stexstyle#1 \cs_end:[3][]{
474     \__stex_aux_patch:nnnn{#1}{##1}{##2}{##3}
475   }
476   \NewDocumentEnvironment{#7#1}{#2}{
477     \cs_set:Npn \stex_style_apply: {
478       \__stex_aux_apply_patch_begin:n{#1}
479     }
480     #3
481   }{
482     \cs_set:Npn \stex_style_apply: {
483       \__stex_aux_apply_patch_end:n{#1}
484     }
485     #4
486   }
487   \tl_set:cn {__stex_aux_style_#1_start:} { #5 }
488   \tl_set:cn {__stex_aux_style_#1_end:} { #6 }
489 }
490
491 \cs_new_protected:Nn \__stex_aux_patch:nnnn {
492   \str_if_empty:nTF {#2}{
493     \tl_set:cn{__stex_aux_style_#1_start:}{#3}
494     \tl_set:cn{__stex_aux_style_#1_end:}{#4}
495   }{
496     \tl_set:cn{__stex_aux_style_#1_#2_start:}{#3}
497     \tl_set:cn{__stex_aux_style_#1_#2_end:}{#4}
498   }
499 }
500
501 \cs_new_protected:Nn \__stex_aux_apply_patch_begin:n {
502   \clist_if_empty:NTF \l_stex_key_style_clist {
```

```
503    \tl_clear:N \thisstyle
504    \use:c{__stex_aux_style_#1_start:}
505  }{
506    \clist_get:NN \l_stex_key_style_clist \thisstyle
507    \stex_debug:nn{styling}{dominant~style:~\thisstyle}
508    \tl_if_exist:cTF{__stex_aux_style_#1_\thisstyle _start:}{
509      \use:c{__stex_aux_style_#1_\thisstyle _start:}
510    }{
511      \use:c{__stex_aux_style_#1_start:}
512    }
513  }
514 }
515
516 \cs_new_protected:Nn \__stex_aux_apply_patch_end:n {
517   \tl_if_empty:NTF \thisstyle {
518     \use:c{__stex_aux_style_#1_end:}
519   }{
520     \tl_if_exist:cTF{__stex_aux_style_#1_\thisstyle _end:}{
521       \use:c{__stex_aux_style_#1_\thisstyle _end:}
522     }{
523       \use:c{__stex_aux_style_#1_end:}
524     }
525   }
526 }
```

(*End definition for* \stex_new_stylable_env:nnnnnnn, \stex_new_stylable_cmd:nnnn, *and* \stex_-
style_apply:. *These functions are documented on page* *122.*)

\stex_str_if_ends_with_p:nn
\stex_str_if_ends_with:nn*TF*

```
527 \prg_new_conditional:Nnn \stex_str_if_ends_with:nn {p,T,F,TF} {
528   \exp_args:Ne \str_if_eq:nnTF {
529     \str_range:nnn{#1}{- \str_count:n{#2}}{-1}
530   }{#2}\prg_return_true: \prg_return_false:
531 }
```

(*End definition for* \stex_str_if_ends_with:nnTF. *This function is documented on page* *125.*)

\stex_str_if_starts_with_p:nn
\stex_str_if_starts_with:nn*TF*

```
532 \prg_new_conditional:Nnn \stex_str_if_starts_with:nn {p,T,F,TF} {
533   \exp_args:Ne \str_if_eq:nnTF {
534     \str_range:nnn{#1}{1}{\str_count:n{#2}}
535   }{#2}\prg_return_true: \prg_return_false:
536 }
```

(*End definition for* \stex_str_if_starts_with:nnTF. *This function is documented on page* *125.*)

\stex_macro_body:N

```
537 \cs_new:Npn \__stex_aux_start:#1\__stex_aux_end: {\exp_not:n{#1}}
538 \cs_new_protected:Nn \__stex_aux_end: {}
539 \cs_new:Nn \stex_macro_body:N {
540   \exp_args:Nne\use:nn{\exp_after:wN \__stex_aux_start: #1}{
541     \__stex_aux_args:e {\cs_parameter_spec:N #1}\__stex_aux_end:
542   }
543 }
544
```

```
545 \cs_new:Nn \__stex_aux_args:n {
546   \tl_if_empty:nF{#1}{
547       {##\exp_args:Ne \tl_head:n {\tl_tail:n {#1}}}
548       \__stex_aux_args:e {\exp_args:Ne\tl_tail:n{\tl_tail:n{#1}}}
549   }
550 }
551 \cs_generate_variant:Nn \__stex_aux_args:n {e}
```

(*End definition for* `\stex_macro_body:N`. *This function is documented on page* *130.*)

`\stex_macro_definition:N`

```
552 \cs_new:Nn \stex_macro_definition:N {
553   \__stex_aux_prefix:e {\cs_prefix_spec:N #1}
554   \def\exp_not:N #1
555   \__stex_aux_params:e {\cs_parameter_spec:N #1}
556   {
557     \stex_macro_body:N #1
558   }
559 }
560
561 \cs_new:Nn \__stex_aux_prefix:n {
562   \tl_if_empty:nF{#1}{
563     \str_if_eq:eeTF {
564       \tl_range:nnn{#1}{1}{10}~
565     }{\tl_to_str:n{\protected}}{
566       \protected
567       \__stex_aux_prefix_long:e {
568         \str_range:nnn{#1}{11}{-1}
569       }
570     }{
571       \__stex_aux_prefix_long:n {#1}
572     }
573   }
574 }
575 \cs_generate_variant:Nn \__stex_aux_prefix:n {e}
576
577 \cs_new:Nn \__stex_aux_prefix_long:n {
578   \tl_if_empty:nF{#1}{
579     \str_if_eq:eeT {
580       \tl_range:nnn{#1}{1}{10}~
581     }{\tl_to_str:n{\long}}{\long}
582   }
583 }
584 \cs_generate_variant:Nn \__stex_aux_prefix_long:n {e}
585
586 \cs_new:Nn \__stex_aux_params:n {
587   \tl_if_empty:nF{#1}{
588     \exp_args:NNe \str_if_eq:VnTF \c_hash_str {\tl_head:n{#1}}{
589       ####
590     }{
591       \tl_head:n{#1}
592     }
593     \__stex_aux_params:e {\tl_tail:n{#1}}
594   }
```

148

```
595 }
596 \cs_generate_variant:Nn \__stex_aux_params:n {e}
```

(*End definition for* \stex_macro_definition:N. *This function is documented on page* *130.*)

13.2.7 Persistence

```
597 ⟨@@=stex_persist⟩
```

We check the environment variables:

```
598 \stex_get_env:Nn\__stex_persist_env_str{STEX_USESMS}
599 \str_if_empty:NF\__stex_persist_env_str{
600   \exp_args:No \str_if_eq:nnF \__stex_persist_env_str{false}{
601     \bool_set_true:N \c_stex_persist_mode_bool
602   }
603 }
604 \stex_get_env:Nn\__stex_persist_env_str{STEX_WRITESMS}
605 \str_if_empty:NF\__stex_persist_env_str{
606   \exp_args:No \str_if_eq:nnF \__stex_persist_env_str{false}{
607     \bool_set_true:N \c_stex_persist_write_mode_bool
608   }
609 }
```

\stex_persist:n
\stex_persist:e

```
610 \iow_new:N \c__stex_persist_sms_iow
611
612 \bool_if:NTF \c_stex_persist_write_mode_bool {
613   \stex_if_html_backend:TF{
614     \cs_new:Npn \stex_persist:n #1 {}
615     \cs_new:Npn \stex_persist:e #1 {}
616   }{
617     \cs_new_protected:Nn \stex_persist:n {
618       \iow_now:Nn \c__stex_persist_sms_iow {#1}
619     }
620     \cs_generate_variant:Nn \stex_persist:n {e}
621   }
622 }{
623   \cs_new:Npn \stex_persist:n #1 {}
624   \cs_new:Npn \stex_persist:e #1 {}
625 }
```

(*End definition for* \stex_persist:n. *This function is documented on page* *130.*)

Is called at the end of the `.sty`-file:

```
626
627 \cs_new_protected:Nn \__stex_persist_load_file:n {
628   \file_if_exist:nT{#1}{
629     \group_begin:
630     \cs_set:Npn \stex_persist:n ##1 {}
631     \cs_set:Npn \stex_persist:e ##1 {}
632     \stex_debug:nn{persist}{restoring~from~sms~file}
633     \catcode`\ =10\relax
634     \cs:w @ @ input \cs_end:#1\relax
635     \group_end:
636   }
637 }
```

149

```
638
639  \cs_new_protected:Nn \__stex_persist_write_only: {
640    \iow_open:Nn \c__stex_persist_sms_iow {\jobname.sms}
641    \AtEndDocument{ \iow_close:N \c__stex_persist_sms_iow }
642  }
643
644  \cs_new_protected:Nn \__stex_persist_read_and_write: {
645    \file_if_exist:nTF{\jobname.sms}{
646      \ior_open:Nn \g_tmpa_ior {\jobname.sms}
647      \iow_open:Nn \g_tmpa_iow {\jobname.sms2}
648      \ior_str_map_inline:Nn \g_tmpa_ior {
649        \iow_now:Nn \g_tmpa_iow {##1}
650      }
651      \iow_close:N \g_tmpa_iow
652      \ior_close:N \g_tmpa_ior
653      \__stex_persist_write_only:
654      \ior_open:Nn \g_tmpa_ior {\jobname.sms2}
655      \ior_str_map_inline:Nn \g_tmpa_ior {
656        \iow_now:Nn \c__stex_persist_sms_iow {##1}
657      }
658      \ior_close:N \g_tmpa_ior
659      \__stex_persist_load_file:n{\jobname.sms2}
660    }\__stex_persist_write_only:
661  }
662
663  \cs_new_protected:Nn \_stex_persist_read_now: {
664    \bool_if:NTF \c_stex_persist_mode_bool {
665      \bool_if:NTF \c_stex_persist_write_mode_bool
666        \__stex_persist_read_and_write:
667        {
668          \__stex_persist_load_file:n{\jobname.sms}
669        }
670    }{
671      \bool_if:NT \c_stex_persist_write_mode_bool \__stex_persist_write_only:
672    }
673  }
```

13.2.8   Files, Paths and URIs

```
674  ⟨@@=stex_path⟩
```

\stex_file_set:Nn
\stex_file_set:No
\stex_file_set:Nx

```
675  \cs_new_protected:Nn \stex_file_set:Nn {
676    \str_if_empty:nTF {#2} { \seq_clear:N #1 }{
677      \exp_args:NNno \seq_set_split:Nnn #1 / { \tl_to_str:n{#2} }
678    }
679  }
680  \cs_generate_variant:Nn \stex_file_set:Nn {No, Nx}
```

(*End definition for* \stex_file_set:Nn. *This function is documented on page 126.*)

\stex_file_resolve:Nn
\stex_file_resolve:No
\stex_file_resolve:Nx

```
681  \sys_if_platform_windows:TF{
682    \cs_new_protected:Npn \__stex_path_win_take:w #1#2#3 \__stex_path_: {
```

```
683       \str_set:Nn \l__stex_path_win_drive {#1#2}
684       \str_set:Nn \l__stex_path_str{#3}
685     }
686   \cs_new_protected:Nn \stex_file_resolve:Nn {
687     \str_set:Nn \l__stex_path_str {#2}
688     \str_clear:N \l__stex_path_win_drive
689     \exp_args:NNo \str_replace_all:Nnn \l__stex_path_str \c_backslash_str /
690     \exp_args:Nx \str_if_eq:nnT {\str_item:Nn \l__stex_path_str 2} : {
691       \exp_after:wN \__stex_path_win_take:w \l__stex_path_str \__stex_path_:
692     }
693     \stex_file_set:No #1 \l__stex_path_str
694     \__stex_path_canonicalize:N #1
695     \str_if_empty:NF \l__stex_path_win_drive {
696       \seq_pop_left:NN #1 \l__stex_path_str
697       \seq_put_left:No #1 \l__stex_path_win_drive
698     }
699     %\stex_debug:nn{files}{Set~\tl_to_str:n{#1}~to~\stex_file_use:N #1}
700   }
701 }{
702   \cs_new_protected:Nn \stex_file_resolve:Nn {
703     \str_set:Nn \l__stex_path_str {#2}
704     \stex_file_set:No #1 \l__stex_path_str
705     \__stex_path_canonicalize:N #1
706   % \stex_debug:nn{files}{Set~\tl_to_str:n{#1}~to~\stex_file_use:N #1}
707   }
708 }
709 \cs_generate_variant:Nn \stex_file_resolve:Nn {No, Nx}
710
711 \cs_new_protected:Nn \__stex_path_canonicalize:N {
712   \seq_if_empty:NF #1 {
713     \seq_pop:NN #1 \l__stex_path_str
714     \seq_clear:N \l__stex_path_seq
715     \str_if_empty:NTF \l__stex_path_str {
716       \seq_map_function:NN #1 \__stex_path_dodots:n
717       \seq_put_left:Nn \l__stex_path_seq {}
718     }{
719       \seq_push:No #1 \l__stex_path_str
720       \seq_map_function:NN #1 \__stex_path_dodots:n
721     }
722     \seq_set_eq:NN #1 \l__stex_path_seq
723   }
724 }
725
726 \cs_new_protected:Nn \__stex_path_dodots:n {
727   \str_if_empty:nF{#1}{
728     \str_if_eq:nnF {#1} {.} {
729       \str_if_eq:nnTF {#1} {..} {
730         \seq_if_empty:NF \l__stex_path_seq {
731           \seq_pop_right:NN \l__stex_path_seq \l__stex_path_str
732         }
733       }{
734         \seq_put_right:Nn \l__stex_path_seq {#1}
735       }
736     }
```

151

```
737    }
738 }
```

(End definition for `\stex_file_resolve:Nn`*. This function is documented on page 126.)*

`\stex_if_file_absolute_p:N`
`\stex_if_file_absolute:N`*TF*

```
739
740 \sys_if_platform_windows:TF {
741    \prg_new_conditional:Nnn \stex_if_file_absolute:N {p, T, F, TF} {
742       \seq_if_empty:NTF #1 \prg_return_false: {
743          \tl_set:Nx \l__stex_path_maybewin_str {\seq_item:Nn #1 1}
744          \exp_args:No \tl_if_empty:nTF \l__stex_path_maybewin_str \prg_return_true: {
745             \exp_args:Nx \str_if_eq:nnTF {\str_item:Nn \l__stex_path_maybewin_str 2} :
746                \prg_return_true: \prg_return_false:
747          }
748       }
749    }
750 }{
751    \prg_new_conditional:Nnn \stex_if_file_absolute:N {p, T, F, TF} {
752       \seq_if_empty:NTF #1 \prg_return_false: {
753          \exp_args:Nx \tl_if_empty:nTF {\seq_item:Nn #1 1}
754             \prg_return_true: \prg_return_false:
755       }
756    }
757 }
```

(End definition for `\stex_if_file_absolute:NTF`*. This function is documented on page 126.)*

`\stex_file_use:N`

```
758 \cs_new:Nn \stex_file_use:N {
759    \seq_use:Nn #1 /
760 }
```

(End definition for `\stex_file_use:N`*. This function is documented on page 126.)*

`\stex_if_file_starts_with:NN`*TF*

```
761 \prg_new_protected_conditional:Nnn \stex_if_file_starts_with:NN {T,F,TF} {
762    \seq_set_eq:NN \l__stex_path_a_seq #1
763    \seq_set_eq:NN \l__stex_path_b_seq #2
764    \tl_clear:N \l__stex_path_return_tl
765    \bool_while_do:nn{
766       \bool_not_p:n{
767          \bool_lazy_any_p:n{
768             {\seq_if_empty_p:N \l__stex_path_a_seq}
769             {\seq_if_empty_p:N \l__stex_path_b_seq}
770             {\bool_not_p:n{\tl_if_empty_p:N \l__stex_path_return_tl}}}
771       }
772    }
773    }{
774       \seq_pop_left:NN \l__stex_path_a_seq \l__stex_path_a_tl
775       \seq_pop_left:NN \l__stex_path_b_seq \l__stex_path_b_tl
776       \str_if_eq:NNF \l__stex_path_a_tl \l__stex_path_b_tl {
777          \tl_set:Nn \l__stex_path_return_tl {\prg_return_false:}
778       }
779    }
```

```
780    \tl_if_empty:NTF \l__stex_path_return_tl {
781      \seq_if_empty:NTF \l__stex_path_b_seq \prg_return_true: \prg_return_false:
782    } \l__stex_path_return_tl
783  }
```

(*End definition for* `\stex_if_file_starts_with:NNTF`. *This function is documented on page* *126.*)

`\stex_file_split_off_ext:NN`
`\stex_file_split_off_lang:NN`

```
784  \cs_new_protected:Nn \stex_file_split_off_ext:NN {
785    \seq_set_eq:NN #1 #2
786    \seq_pop_right:NN #1 \l__stex_path_str
787    \seq_set_split:NnV \l__stex_path_seq . \l__stex_path_str
788    \seq_pop_right:NN \l__stex_path_seq \l__stex_path_str
789    \seq_put_right:Nx #1 {\seq_use:Nn \l__stex_path_seq .}
790  }
791  \cs_new_protected:Nn \stex_file_split_off_lang:NN {
792    \seq_set_eq:NN #1 #2
793    \seq_pop_right:NN #1 \l__stex_path_str
794    \seq_set_split:NnV \l__stex_path_seq . \l__stex_path_str
795    \seq_pop_right:NN \l__stex_path_seq \l__stex_path_str
796
797    \seq_pop_right:NN \l__stex_path_seq \l__stex_path_str
798    \exp_args:NNo \prop_if_in:NnF \c_stex_languages_prop \l__stex_path_str {
799      \seq_put_right:No \l__stex_path_seq \l__stex_path_str
800    }
801
802    \seq_put_right:Nx #1 {\seq_use:Nn \l__stex_path_seq .}
803  }
```

(*End definition for* `\stex_file_split_off_ext:NN` *and* `\stex_file_split_off_lang:NN`. *These functions are documented on page* *126.*)

URIs:

`\stex_map_uri:Nnnnn`

```
804  \cs_set_protected:Nn \__stex_path_auth:n {
805    \msg_error:nnx{stex}{error/misused-uri}{\tl_to_str:n{#1}}
806  }
807  \cs_set_eq:NN \__stex_path_path:n \__stex_path_auth:n
808  \cs_set_eq:NN \__stex_path_module:n \__stex_path_auth:n
809  \cs_set_eq:NN \__stex_path_name:n \__stex_path_auth:n
810
811  \cs_set_protected:Nn \stex_map_uri:Nnnnn{
812    \stex_pseudogroup_with:nn{\__stex_path_auth:n\__stex_path_path:n\__stex_path_module:n\__st
813      \cs_set:Npn \__stex_path_auth:n ##1 {#2}
814      \cs_set:Npn \__stex_path_path:n ##1 {#3}
815      \cs_set:Npn \__stex_path_module:n ##1 {#4}
816      \cs_set:Npn \__stex_path_name:n ##1 {#5}
817      #1
818    }
819  }
```

(*End definition for* `\stex_map_uri:Nnnnn`. *This function is documented on page* *127.*)

153

\stex_uri_set:No
\stex_uri_set:Nx

```
820 \str_set:Nx\__stex_path_colonslash{\c_colon_str/}
821 \cs_new_protected:Nn \__stex_path_uri_set:NnN {
822   \str_if_empty:nTF {#2} {
823     \msg_error:nnxx{stex}{error/invalid-uri}{\tl_to_str:n{#2}}{empty}
824   }{
825     \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn #1 \__stex_path_colonslash { \tl_to_str
826     \seq_pop_left:NN #1 \l__stex_path_auth_str
827     \seq_if_empty:NTF #1 {
828       \msg_error:nnxx{stex}{error/invalid-uri}{\tl_to_str:n{#2}}{missing~authority}
829     }{
830       \exp_args:NNnx \seq_set_split:Nnn #1 ? {\exp_args:NNo \seq_use:Nn #1 \__stex_path_colo
831       \seq_pop_left:NN #1 \l__stex_path_path
832       #3 \l__stex_path_path \l__stex_path_path
833       \seq_if_empty:NTF #1 {
834         \exp_args:NNo \__stex_path_uri_set:Nnxnn #1 \l__stex_path_auth_str
835           {\stex_file_use:N \l__stex_path_path} {} {}
836       }{
837         \seq_pop_left:NN #1 \l__stex_path_mod
838         \seq_if_empty:NTF #1 {
839           \exp_args:NNo \__stex_path_uri_set:Nnxon #1 \l__stex_path_auth_str
840             {\stex_file_use:N \l__stex_path_path} \l__stex_path_mod {}
841         }{
842           \seq_pop_left:NN #1 \l__stex_path_name
843           \seq_if_empty:NTF #1 {
844             \exp_args:NNo \__stex_path_uri_set:Nnxon #2 \l__stex_path_auth_str
845               {\stex_file_use:N \l__stex_path_path} \l__stex_path_mod \l__stex_path_name
846           }{
847             \msg_error:nnxx{stex}{error/invalid-uri}{\tl_to_str:n{#2}}{too~many~?s}
848           }
849         }
850       }
851     }
852   }
853   \stex_debug:nn{uris}{Set~\tl_to_str:n{#1}~to~\stex_uri_use:N #1}
854 }
855
856 \cs_new_protected:Nn \__stex_path_uri_set:Nnnnn{
857   \tl_set:Nn #1 {
858     \__stex_path_auth:n{ #2 }
859     \__stex_path_path:n{ #3 }
860     \__stex_path_module:n{ #4 }
861     \__stex_path_name:n{ #5 }
862   }
863 }
864 \cs_generate_variant:Nn\__stex_path_uri_set:Nnnnn {Nnxnn,Nnxon,Nnxoo}
865
866 \cs_new_protected:Nn \stex_uri_set:Nn {
867   \__stex_path_uri_set:NnN #1 {#2} \stex_file_set:No
868 }
869 \cs_generate_variant:Nn \stex_uri_set:Nn {No, Nx}
```

(*End definition for* \stex_uri_set:Nn*. This function is documented on page* *127.*)

\stex_uri_resolve:Nn
\stex_uri_resolve:No
\stex_uri_resolve:Nx

```
870 \cs_new_protected:Nn \stex_uri_resolve:Nn {
871   \__stex_path_uri_set:NnN #1 {#2} \stex_file_resolve:No
872 }
873 \cs_generate_variant:Nn \stex_uri_resolve:Nn {No, Nx}
```

(*End definition for* \stex_uri_resolve:Nn. *This function is documented on page 127.*)

\stex_uri_use:N

```
874 \cs_new:Npn \__stex_path_uri_use:w \__stex_path_auth:n #1 \__stex_path_path:n #2 \__stex_pat
875   #1\c_colon_str/ # 2 \tl_if_empty:nF { #3 }{ ? #3
876     \tl_if_empty:nF { #4 }{ ? #4 } }
877 }
878 \cs_new:Nn \stex_uri_use:N {
879   \exp_args:Ne \cs_if_eq:NNTF { \tl_head:N #1 } \__stex_path_auth:n {
880     \exp_after:wN \__stex_path_uri_use:w #1
881   }{
882     \msg_error:nnnn{stex}{error/invalid-uri}{#1}{Not~a~URI}
883   }
884 }
```

(*End definition for* \stex_uri_use:N. *This function is documented on page 127.*)

\stex_uri_from_repo_file:NNNn
\stex_uri_from_repo_file_nolang:NNNn

```
885 \cs_new_protected:Npn \stex_uri_from_repo_file_nolang:NNNn {
886   \__stex_path_from_repo_file:NNNNn \stex_file_split_off_lang:NN
887 }
888 \cs_new_protected:Npn \stex_uri_from_repo_file:NNNn {
889   \__stex_path_from_repo_file:NNNNn \stex_file_split_off_ext:NN
890 }
891
892 \cs_new_protected:Nn \__stex_path_from_repo_file:NNNNn {
893   #1 \l__stex_path_file #4
894   \prop_if_exist:NTF #3 {
895     \str_clear:N \l__stex_path_uri
896     \prop_get:NnNF #3 {#5} \l__stex_path_uri {
897       \prop_get:NnNF #3 {ns} \l__stex_path_uri {
898         \__stex_path_uri_set:Nnxnn #2 {file}
899           {\stex_file_use:N \l__stex_path_file} {} {}
900       }
901     }
902     \str_if_empty:NF \l__stex_path_uri {\__stex_path_relativize:N #2}
903   }{
904     \exp_args:NNx \__stex_path_uri_set:Nnxnn #2 {\tl_to_str:n{file}}
905       {\stex_file_use:N \l__stex_path_file} {} {}
906   }
907 }
908
909 \cs_new_protected:Nn \__stex_path_relativize:N {
910   \seq_set_eq:NN \l__stex_path_seq \l__stex_path_file
911   \seq_map_inline:Nn \c_stex_mathhub_file { % mathhub path
912     \seq_pop_left:NN \l__stex_path_seq \l__stex_path_tl
913   }
914   \stex_file_set:Nx \l__stex_path_path {\prop_item:Nn \l_stex_current_archive_prop {id} }
```

155

```
915    \seq_map_inline:Nn \l__stex_path_path { % id
916      \seq_pop_left:NN \l__stex_path_seq \l__stex_path_tl
917    }
918    \seq_pop_left:NN \l__stex_path_seq \l__stex_path_tl % source
919
920    \stex_uri_set:Nx #1 { \l__stex_path_uri / \stex_file_use:N \l__stex_path_seq }
921 }
```

(*End definition for* \stex_uri_from_repo_file:NNNn *and* \stex_uri_from_repo_file_nolang:NNNn*. These functions are documented on page* *127.*)

\stex_uri_from_current_file:Nn
\stex_uri_from_current_file_nolang:Nn

```
922 \cs_new_protected:Nn \stex_uri_from_current_file:Nn {
923    \stex_debug:nn{docuri}{Here:~\stex_file_use:N \g_stex_current_file}
924    \stex_uri_from_repo_file:NNNn #1 \l_stex_current_archive_prop
925      \g_stex_current_file {#2}
926    \stex_debug:nn{docuri}{resolved:~\stex_uri_use:N #1}
927 }
928 \cs_new_protected:Nn \stex_uri_from_current_file_nolang:Nn {
929    \stex_uri_from_repo_file_nolang:NNNn #1 \l_stex_current_archive_prop
930      \g_stex_current_file {#2}
931 }
```

(*End definition for* \stex_uri_from_current_file:Nn *and* \stex_uri_from_current_file_nolang:Nn*. These functions are documented on page* *128.*)

\stex_uri_add_module:NNn
\stex_uri_add_module:NNo

```
932 \cs_new_protected:Nn \stex_uri_add_module:NNn {
933    \exp_args:Ne \cs_if_eq:NNTF { \tl_head:N #2 } \__stex_path_auth:n {
934      \stex_pseudogroup_with:nn
935        {\__stex_path_auth:n\__stex_path_path:n\__stex_path_module:n\__stex_path_name:n}
936        {
937          \cs_set:Npn \__stex_path_module:n ##1 {
938            \tl_if_empty:nTF{##1}{
939              \exp_not:N \__stex_path_module:n {#3}
940            }{
941              \msg_error:nnn{stex}{error/invalid-dpath}{#2}
942            }
943          }
944          \cs_set:Npn \__stex_path_name:n ##1 {
945            \tl_if_empty:nTF{##1}{
946              \exp_not:N \__stex_path_name:n {}
947            }{
948              \msg_error:nnn{stex}{error/invalid-dpath}{#2}
949            }
950          }
951          \tl_set:Nx #1 {#2}
952        }
953    }{
954      \msg_error:nnnn{stex}{error/invalid-uri}{#2}{Not~a~URI}
955    }
956 }
957 \cs_generate_variant:Nn \stex_uri_add_module:NNn {NNo}
```

(*End definition for* \stex_uri_add_module:NNn*. This function is documented on page* *128.*)

958 `\tl_new:N \l_stex_current_doc_uri`

(*End definition for* `\l_stex_current_doc_uri`*. This variable is documented on page 128.*)

959 `\cs_new_protected:Nn \stex_set_document_uri: {`
960 `  \stex_uri_from_current_file:Nn \l_stex_current_doc_uri {narr}`
961 `  %\stex_debug:nn{sref}{Document~URI:~\stex_uri_use:N \l_stex_current_doc_uri}`
962 `}`

(*End definition for* `\stex_set_document_uri:`*. This function is documented on page 128.*)

963 `\cs_new_protected:Nn \stex_set_current_namespace: {`
964 `  \stex_uri_from_current_file_nolang:Nn \l_stex_current_ns_uri {source-base}`
965 `  %\stex_debug:nn{modules}{Namespace~URI:~\stex_uri_use:N \l_stex_current_ns_uri}`
966 `}`

(*End definition for* `\stex_set_current_namespace:`*. This function is documented on page 128.*)

We determine the PWD

967 `\sys_if_platform_windows:TF{`
968 `  \stex_get_env:Nn\l__stex_path_str{CD}`
969 `}{`
970 `  \stex_get_env:Nn\l__stex_path_str{PWD}`
971 `}`
972 `\stex_file_resolve:No \c_stex_pwd_file \l__stex_path_str`
973 `\seq_set_eq:NN \c_stex_main_file \c_stex_pwd_file`
974 `\seq_put_right:Nx \c_stex_main_file {\jobname\tl_to_str:n{.tex}}`
975
976 `\stex_debug:nn {files} {PWD:~\stex_file_use:N \c_stex_pwd_file}`

(*End definition for* `\c_stex_pwd_file` *and* `\c_stex_main_file`*. These variables are documented on page 127.*)

13.2.9 File Hooks

keeps track of file changes:

977 `\seq_gclear_new:N\g__stex_path_stack`
978 `\seq_gclear_new:N\g_stex_current_file`

979 `\cs_new_protected:Nn \stex_filestack_push:n {`
980 `  \stex_str_if_ends_with:nnTF {#1}{.tex}{`
981 `    \stex_file_resolve:No \g_stex_current_file {#1}`
982 `  }{`
983 `    \stex_file_resolve:No \g_stex_current_file {#1.tex}`
984 `  }`
985 `  \stex_if_file_absolute:NF \g_stex_current_file {`
986 `    \stex_file_resolve:Nx \g_stex_current_file {`
987 `      \stex_file_use:N \c_stex_pwd_file / \stex_file_use:N \g_stex_current_file`
988 `    }`

157

```
989     }
990     \seq_gset_eq:NN \g_stex_current_file \g_stex_current_file
991     \exp_args:NNx \seq_gpush:Nn \g__stex_path_stack {\stex_file_use:N \g_stex_current_file}
992     \_stex_every_file:
993   }
994
995   \cs_new_protected:Nn \_stex_every_file: {
996     \stex_set_document_uri:
997     \stex_language_from_file:
998     \stex_set_current_namespace:
999   }
1000  %\AtBeginDocument{\_stex_every_file:}
```

(*End definition for* \stex_filestack_push:n. *This function is documented on page 126.*)

\stex_filestack_pop:

```
1001  \cs_new_protected:Nn \stex_filestack_pop: {
1002    \seq_if_empty:NF \g__stex_path_stack {
1003      \seq_gpop:NN \g__stex_path_stack \l__stex_path_str
1004    }
1005    \seq_if_empty:NTF \g__stex_path_stack {
1006      \seq_gset_eq:NN \g_stex_current_file \c_stex_main_file
1007    }{
1008      \seq_get:NN \g__stex_path_stack \l__stex_path_str
1009      \exp_args:NNo \stex_file_set:Nn \g_stex_current_file \l__stex_path_str
1010      \seq_gset_eq:NN \g_stex_current_file \g_stex_current_file
1011    }
1012    \_stex_every_file:
1013  }
```

(*End definition for* \stex_filestack_pop:. *This function is documented on page 126.*)

Hooks for the current file:

```
1014  \cs_new_protected:Nn \stex_input_with_hooks:n {
1015    \tl_set:Nn \l__stex_path_do_hooks_pre_tl {
1016      \tl_gset:Nn \l__stex_path_do_hooks_pre_tl {}
1017      \stex_debug:nn{HERE}{Hook~for~#1^^J\meaning\CurrentFilePath^^J\CurrentFile}
1018      \tl_if_empty:NTF\CurrentFilePath{
1019        \exp_args:No \stex_filestack_push:n {\CurrentFile}
1020      }{
1021        \exp_args:Ne \stex_filestack_push:n { \CurrentFilePath / \CurrentFile }
1022      }
1023    }
1024    \input{#1}
1025    \stex_debug:nn{HERE}{Hook~end~for~#1}
1026    \stex_filestack_pop:
1027  }
1028  \tl_new:N \l__stex_path_do_hooks_pre_tl {}
1029
1030  \AddToHook{file/before}{
1031    \l__stex_path_do_hooks_pre_tl
1032  }
1033  %\AddToHook{file/after}{ \stex_filestack_pop: }
```

13.3 Math Archives

1034 ⟨@@=stex_mathhub⟩

\mathhub
\c_stex_home_file
\c_stex_mathhub_file

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

```
1035
1036 \sys_if_platform_windows:TF{
1037   \stex_get_env:Nn \l__stex_mathhub_str {homedrive\c_percent_str\c_percent_str homepath}
1038 }{
1039   \stex_get_env:Nn \l__stex_mathhub_str {HOME}
1040 }
1041 \stex_file_resolve:No \c_stex_home_file \l__stex_mathhub_str
1042
1043 \str_if_empty:NTF\mathhub{
1044   \stex_get_env:Nn \l__stex_mathhub_str {MATHHUB}
1045   \str_if_empty:NTF \l__stex_mathhub_str {
1046     \ior_open:NnTF \g_tmpa_ior{\stex_file_use:N \c_stex_home_file/.stex/mathhub.path}{
1047       \group_begin:
1048         \escapechar=-1\catcode`\\=12
1049         \ior_str_get:NN \g_tmpa_ior \l__stex_mathhub_str
1050         \str_gset_eq:NN \l__stex_mathhub_str \l__stex_mathhub_str
1051       \group_end:
1052       \ior_close:N \g_tmpa_ior
1053       \stex_debug:nn{mathhub}{MathHub~directory~determined~from~home~directory}
1054     }{
1055       \str_clear:N \l__stex_mathhub_str
1056     }
1057   }{
1058     \stex_debug:nn{mathhub}{MathHub~directory~determined~from~environment~variable}
1059   }
1060 }{
1061   \str_set_eq:NN \l__stex_mathhub_str \mathhub
1062 }
1063
1064 \str_if_empty:NTF \l__stex_mathhub_str {
1065   \msg_warning:nn{stex}{warning/nomathhub}
1066   \exp_args:NNe \stex_file_set:Nn \c_stex_mathhub_file {\stex_file_use:N \c_stex_home_file \
1067   \seq_clear:N \c_stex_mathhub_file
1068 }{
1069   \stex_file_resolve:No \c_stex_mathhub_file \l__stex_mathhub_str
1070   \stex_if_file_absolute:NF \c_stex_mathhub_file {
1071     \exp_args:NNe \stex_file_resolve:Nn \c_stex_mathhub_file {
1072       \stex_file_use:N \c_stex_main_file / .. / \l__stex_mathhub_str
1073     }
1074   }
1075 }
1076
1077 \exp_args:NNe \str_set:Nn \mathhub {\stex_file_use:N \c_stex_mathhub_file}
1078 \stex_debug:nn{mathhub}{MATHHUB:~\mathhub}
```

(*End definition for* \mathhub, \c_stex_home_file, *and* \c_stex_mathhub_file. *These variables are documented on page* **??**.)

\l_stex_current_archive
\stex_set_current_archive:n

```
1079 \cs_new_protected:Nn \stex_set_current_archive:n {
1080   \stex_require_archive:n { #1 }
1081   \stex_debug:nn{mathhub}{switching~to~archive~#1}
1082   \prop_set_eq:Nc \l_stex_current_archive_prop {
1083     c_stex_mathhub_#1_manifest_prop
1084   }
1085 }
```

(*End definition for* `\l_stex_current_archive` *and* `\stex_set_current_archive:n`. *These variables are documented on page* **??**.)

\stex_in_archive:nn

```
1086 \cs_new_protected:Nn \stex_in_archive:nn {
1087   \cs_if_exist:NF \l__stex_mathhub_cs {
1088     \cs_set:Npn \l__stex_mathhub_cs ##1 {}
1089   }
1090   \stex_pseudogroup:nn{
1091     \cs_set:Npn \l__stex_mathhub_cs ##1 {#2}
1092     \tl_if_empty:nTF{#1}{
1093       \prop_if_exist:NTF \l_stex_current_archive_prop {
1094         \exp_args:Ne \l__stex_mathhub_cs {\prop_item:Nn \l_stex_current_archive_prop { id }
1095       }{
1096         \l__stex_mathhub_cs {}
1097       }
1098     }{
1099       \stex_set_current_archive:n{#1}
1100       \l__stex_mathhub_cs {#1}
1101     }
1102   }{
1103     \stex_pseudogroup_restore:N \l_stex_current_archive_prop
1104     \cs_set:Npn \exp_not:N \l__stex_mathhub_cs ##1 {
1105       \exp_args:No \exp_not:n {\l__stex_mathhub_cs {##1}}
1106     }
1107   }
1108 }
```

(*End definition for* `\stex_in_archive:nn`. *This function is documented on page 123.*)

\stex_require_archive:n
\stex_require_archive:o

```
1109 \cs_new_protected:Nn \stex_require_archive:n {
1110   \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
1111     \seq_if_empty:NTF \c_stex_mathhub_file {
1112       \stex_fatal_error:n{warning/nomathhub}
1113     }{
1114       \stex_debug:nn{mathhub}{Opening~archive:~#1}
1115       \__stex_mathhub_do_manifest:n { #1 }
1116     }
1117   }
1118 }
1119 \cs_generate_variant:Nn \stex_require_archive:n {o}
```

(*End definition for* `\stex_require_archive:n`. *This function is documented on page 123.*)

Code for finding and parsing manifest files:

```
1120 \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
```

```
1121    \exp_args:Ne \__stex_mathhub_find_manifest:n {\stex_file_use:N \c_stex_mathhub_file / #1}
1122    \str_if_empty:NT \l__stex_mathhub_manifest_str {
1123      \stex_fatal_error:nxx{error/noarchive}
1124        {#1}{\stex_file_use:N \c_stex_mathhub_file}
1125    }
1126    \__stex_mathhub_parse_manifest:n {#1}
1127 }
1128
1129 \cs_new_protected:Nn \__stex_mathhub_find_manifest:n {
1130    \str_clear:N \l__stex_mathhub_manifest_str
1131    \seq_set_split:Nnn \l__stex_mathhub_seq / {#1}
1132    \bool_set_true:N \l__stex_mathhub_bool
1133    \bool_while_do:Nn \l__stex_mathhub_bool {
1134      \tl_if_eq:NNTF \l__stex_mathhub_seq \c_stex_mathhub_file {
1135        \bool_set_false:N \l__stex_mathhub_bool
1136      }{
1137        \__stex_mathhub_check_manifest:
1138        \bool_if:NT \l__stex_mathhub_bool {
1139          \seq_pop_right:NN \l__stex_mathhub_seq \l__stex_mathhub_tl
1140        }
1141      }
1142    }
1143 }
1144 \cs_generate_variant:Nn \__stex_mathhub_find_manifest:n {x}
1145
1146 \cs_new_protected:Nn \__stex_mathhub_check_manifest: {
1147    \__stex_mathhub_check_manifest:n {MANIFEST.MF}
1148    \bool_if:NT \l__stex_mathhub_bool {
1149      \__stex_mathhub_check_manifest:n {META-INF/MANIFEST.MF}
1150      \bool_if:NT \l__stex_mathhub_bool {
1151        \__stex_mathhub_check_manifest:n {meta-inf/MANIFEST.MF}
1152      }
1153    }
1154 }
1155
1156 \cs_new_protected:Nn \__stex_mathhub_check_manifest:n {
1157    \stex_debug:nn{mathhub}{Checking~\stex_file_use:N \l__stex_mathhub_seq / #1}
1158    \file_if_exist:nT {\stex_file_use:N \l__stex_mathhub_seq / #1} {
1159      \bool_set_false:N \l__stex_mathhub_bool
1160      \str_set:Nx \l__stex_mathhub_manifest_str {\stex_file_use:N \l__stex_mathhub_seq / #1}
1161    }
1162 }
1163
1164 \ior_new:N \c__stex_mathhub_manifest_ior
1165 \cs_new_protected:Nn \__stex_mathhub_parse_manifest:n {
1166    \ior_open:Nn \c__stex_mathhub_manifest_ior \l__stex_mathhub_manifest_str
1167    \prop_clear:N \l__stex_mathhub_prop
1168    \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
1169      \exp_args:NNNo \exp_args:NNNx
1170        \seq_set_split:Nnn \l__stex_mathhub_seq \c_colon_str {\tl_to_str:n{##1}}
1171      \seq_pop_left:NNT \l__stex_mathhub_seq \l__stex_mathhub_key {
1172        \exp_args:NNo \str_set:Nn \l__stex_mathhub_key \l__stex_mathhub_key
1173        \str_set:Nx \l__stex_mathhub_val {\seq_use:Nn \l__stex_mathhub_seq :}
1174        \str_case:Nn \l__stex_mathhub_key {
```

```
1175        {id}              {\prop_put:Nno \l__stex_mathhub_prop { id }     \l__stex_mathhub_
1176        {narration-base} {\prop_put:Nno \l__stex_mathhub_prop { narr }   \l__stex_mathhub_
1177        {url-base}       {\prop_put:Nno \l__stex_mathhub_prop { docurl } \l__stex_mathhub_
1178        {source-base}    {\prop_put:Nno \l__stex_mathhub_prop { ns }     \l__stex_mathhub_
1179        {ns}             {\prop_put:Nno \l__stex_mathhub_prop { ns }     \l__stex_mathhub_
1180      }
1181    }
1182  }
1183  \ior_close:N \c__stex_mathhub_manifest_ior
1184  \prop_gset_eq:cN { c_stex_mathhub_#1_manifest_prop } \l__stex_mathhub_prop
1185  \stex_debug:nn{mathhub}{Result:~\prop_to_keyval:N \l__stex_mathhub_prop}
1186  \stex_persist:e {
1187    \prop_gset_from_keyval:cn {c_stex_mathhub_#1_manifest_prop}{
1188      \prop_to_keyval:N \l__stex_mathhub_prop
1189    }
1190  }
1191 }
```

Current MathHub archive

```
1192 \cs_new_protected:Nn \_stex_main_archive: {
1193   \stex_if_file_starts_with:NNTF \c_stex_pwd_file \c_stex_mathhub_file {
1194     \__stex_mathhub_find_manifest:x { \stex_file_use:N \c_stex_pwd_file }
1195     \str_if_empty:NTF \l__stex_mathhub_manifest_str {
1196       \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~archive}
1197     }{
1198       \__stex_mathhub_parse_manifest:n { main }
1199       \prop_set_eq:NN \c_stex_main_archive_prop \c_stex_mathhub_main_manifest_prop
1200       \cs_undefine:N \c_stex_mathhub_main_manifest_prop
1201       \prop_get:NnN \c_stex_main_archive_prop {id}
1202         \l__stex_mathhub_str
1203       \prop_set_eq:cN { c_stex_mathhub_\l__stex_mathhub_str _manifest_prop }
1204         \c_stex_main_archive_prop
1205       \exp_args:No \stex_set_current_archive:n { \l__stex_mathhub_str }
1206       \stex_debug:nn{mathhub}{Current~archive:~
1207         \prop_item:Nn \l_stex_current_archive_prop {id}
1208       }
1209       \bool_if:NT \c_stex_persist_write_mode_bool {
1210         \tl_put_right:Nx \_stex_persist_read_now: {
1211           \stex_persist:n {
1212             \prop_gset_from_keyval:cn {c_stex_mathhub_\l__stex_mathhub_str _manifest_prop}{
1213               \prop_to_keyval:N \c_stex_main_archive_prop
1214             }
1215             \prop_gset_eq:Nc \exp_not:N \l_stex_current_archive_prop {
1216               c_stex_mathhub_\l__stex_mathhub_str _manifest_prop
1217             }
1218             \prop_gset_eq:Nc \exp_not:N \c_stex_main_archive_prop {
1219               c_stex_mathhub_\l__stex_mathhub_str _manifest_prop
1220             }
1221           }
1222         }
1223       }
1224     }
1225   }{
```

162

```
1226         \stex_debug:nn{mathhub}{Not~currently~in~the~MathHub~directory}
1227     }
1228 }
1229
1230 %\bool_if:NF \c_stex_persist_mode_bool {
1231     \_stex_main_archive:
1232 %}
```

(*End definition for* `\c_stex_main_archive_prop` *and* `\l_stex_current_archive_prop`. *These variables are documented on page* *123.*)

13.4 Documents

13.4.1 Title

Stores the title, if it exists:

```
1233 ⟨@@=stex_doc⟩
1234 \tl_new:N \g__stex_doc_title_tl
```

\stexdoctitle Initital definition, will be changed at begin document:

```
1235 \cs_new_protected:Npn \stexdoctitle #1 {
1236     \tl_gset:Nn \g__stex_doc_title_tl { #1 }
1237     \global\def\stexdoctitle##1{}
1238 }
```

At begin document, we switch to:

```
1239 \cs_new_protected:Nn \__stex_doc_set_title:n {
1240     \stex_if_smsmode:F{
1241         \global\def\stexdoctitle##1{}
1242         \stex_debug:nn{title}{Setting~title~to:\tl_to_str:n{#1}}
1243         \tl_gset:Nn \g__stex_doc_title_tl { #1 }
1244         \__stex_doc_title_html:
1245     }
1246 }
```

Hooks, changes and HTML:

```
1247 \cs_new_protected:Nn \__stex_doc_title_html: {
1248     \stex_if_do_html:T{\stex_if_html_backend:T{
1249         \stex_annotate_invisible:nn{shtml:doctitle={}}{ \hbox{\g__stex_doc_title_tl} }
1250     }}
1251 }
1252
1253 \AtBeginDocument {
1254     \tl_if_empty:NTF \g__stex_doc_title_tl {
1255         \cs_set_eq:NN \stexdoctitle \__stex_doc_set_title:n
1256     }{
1257         \stex_debug:nn{title}{Setting~title~to:\exp_args:No\tl_to_str:n\g__stex_doc_title_tl}
1258         \global\def\stexdoctitle#1{}
1259         \__stex_doc_title_html:
1260     }
1261
1262     \cs_set_eq:NN \__stex_doc_maketitle: \maketitle
1263     \global\protected\def\maketitle{
1264         \tl_if_empty:NF \@title {
```

```
1265        \exp_args:No \stexdoctitle \@title
1266      }
1267      \__stex_doc_maketitle:
1268    }
1269 }
```

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

13.4.2 Sectioning

```
1270 \int_new:N \l_stex_docheader_sect
1271
1272 \tl_set:Nn \stex_current_section_level {document}
1273
1274 \cs_set_protected:Npn \currentsectionlevel {
1275   \stex_if_do_html:TF{
1276     \stex_annotate:nn{shtml:currentsectionlevel={},shtml:capitalize=false}{}
1277   }{
1278     \stex_current_section_level
1279   }
1280   \tl_if_exist:NT\xspace\xspace
1281 }
1282
1283 \cs_set_protected:Npn \Currentsectionlevel {
1284   \stex_if_do_html:TF{
1285     \stex_annotate:nn{shtml:currentsectionlevel={},shtml:capitalize=true}{}
1286   }{
1287     \exp_args:No \_stex_capitalize:n \stex_current_section_level
1288   }
1289   \tl_if_exist:NT\xspace\xspace
1290 }
1291
1292 \stex_if_html_backend:TF {
1293   \cs_new_protected:Nn \_sfragment_do_level:nn {
1294     \stexdoctitle{#2}
1295     \par
1296     \begin{stex_annotate_env}{shtml:section={\int_use:N \l_stex_docheader_sect}}
1297       \noindent\stex_annotate:nn{shtml:sectiontitle={}}{
1298         \_stex_annotate_force_break:n{#2}
1299       }\par
1300   }
1301   \cs_new_protected:Nn \_sfragment_end: {
1302     \end{stex_annotate_env}
1303   }
1304 }{
1305   \cs_new_protected:Nn \_sfragment_do_level:nn {
1306     \stexdoctitle{#2}
1307     \tl_if_empty:NTF \l_stex_key_short_tl {
1308       \use:c{#1}
1309     }{
1310       \exp_args:Nnx \use:nn{\use:c{#1}}{[\exp_args:No \exp_not:n \l_stex_key_short_tl]}
1311     }{#2}
1312     \int_incr:N \l_stex_docheader_sect
1313     \tl_set:Nn \stex_current_section_level{#1}
```

```
1314    }
1315    \cs_new_protected:Nn \_sfragment_end: {}
1316  }
1317
1318
1319  \cs_new_protected:Npn \__stex_doc_do_section:n {
1320    \int_case:nnF \l_stex_docheader_sect {
1321      {0}{\cs_if_exist:NTF \thepart {\_sfragment_do_level:nn{part}}{
1322        \int_incr:N \l_stex_docheader_sect
1323        \__stex_doc_do_section:n
1324      }}
1325      {1}{\cs_if_exist:NTF \thechapter {\_sfragment_do_level:nn{chapter}}{
1326        \int_incr:N \l_stex_docheader_sect
1327        \__stex_doc_do_section:n
1328      }}
1329      {2}{\_sfragment_do_level:nn{section}}
1330      {3}{\_sfragment_do_level:nn{subsection}}
1331      {4}{\_sfragment_do_level:nn{subsubsection}}
1332      {5}{\_sfragment_do_level:nn{paragraph}}
1333    }{\_sfragment_do_level:nn{subparagraph}}
1334  }
1335
1336  \stex_keys_define:nnnn{ sfragment }{
1337    \tl_clear:N \l_stex_key_short_tl
1338  }{
1339    short    .tl_set:N   = \l_stex_key_short_tl
1340  }{id}
1341
1342  \NewDocumentEnvironment{sfragment}{ O{} m}{
1343    \stex_keys_set:nn{sfragment}{#1}
1344    \__stex_doc_do_section:n{#2}
1345    \_stex_do_id:
1346  }{
1347    \_sfragment_end:
1348  }
1349
1350  %\int_incr:N \l_stex_docheader_sect
1351  \NewDocumentEnvironment{blindfragment}{}{
1352    \__stex_doc_skip_section:
1353  }{
1354    \stex_if_html_backend:T{
1355      \stex_annotate_invisible:n{~}
1356      \end{stex_annotate_env}
1357    }
1358  }
1359
1360
1361  \cs_new_protected:Nn \__stex_doc_skip_section_i: {
1362    \int_case:nn \l_stex_docheader_sect {
1363      {0}{\cs_if_exist:NF \thepart {
1364        \int_incr:N \l_stex_docheader_sect \__stex_doc_skip_section_i:
1365      }}
1366      {1}{\cs_if_exist:NF \thechapter {
1367        \int_incr:N \l_stex_docheader_sect \__stex_doc_skip_section_i:
```

```
1368        }}
1369      }
1370      \int_incr:N \l_stex_docheader_sect
1371    }
1372
1373    \stex_if_html_backend:TF {
1374      \cs_new_protected:Nn \__stex_doc_skip_section: {
1375        \__stex_doc_skip_section_i:
1376        \begin{stex_annotate_env}{shtml:skipsection={\int_use:N \l_stex_docheader_sect}}
1377        \stex_annotate_invisible:n{~}
1378      }
1379    }{
1380      \cs_set_eq:NN \__stex_doc_skip_section: \__stex_doc_skip_section_i:
1381    }
1382
1383
1384    \cs_new_protected:Nn \__stex_doc_skip_fragment:n {
1385      \stepcounter{#1}
1386    }
1387
1388    \cs_new_protected:Npn \skipfragment {
1389      \int_case:nnF \l_stex_docheader_sect {
1390        {0}{\cs_if_exist:NTF \thepart {\__stex_doc_skip_fragment:n{part}}{
1391          \int_incr:N \l_stex_docheader_sect
1392          \skipfragment
1393        }}
1394        {1}{\cs_if_exist:NTF \thechapter {\__stex_doc_skip_fragment:n{chapter}}{
1395          \int_incr:N \l_stex_docheader_sect
1396          \skipfragment
1397        }}
1398        {2}{\__stex_doc_skip_fragment:n{section}}
1399        {3}{\__stex_doc_skip_fragment:n{subsection}}
1400        {4}{\__stex_doc_skip_fragment:n{subsubsection}}
1401        {5}{\__stex_doc_skip_fragment:n{paragraph}}
1402      }{\__stex_doc_skip_fragment:n{subparagraph}}
1403    }
```

\setsectionlevel

```
1404    \cs_new_protected:Npn \setsectionlevel #1 {
1405      \str_case:nnF{#1}{
1406        {part}{\int_set:Nn \l_stex_docheader_sect 0}
1407        {chapter}{\int_set:Nn \l_stex_docheader_sect 1}
1408        {section}{\int_set:Nn \l_stex_docheader_sect 2}
1409        {subsection}{\int_set:Nn \l_stex_docheader_sect 3}
1410        {subsubsection}{\int_set:Nn \l_stex_docheader_sect 4}
1411        {paragraph}{\int_set:Nn \l_stex_docheader_sect 5}
1412      }{
1413        \int_set:Nn \l_stex_docheader_sect 6
1414      }
1415      \cs_if_eq:NNTF\@onlypreamble\@notprerr{
1416        \stex_annotate_invisible:nn{shtml:sectionlevel={\int_use:N\l_stex_docheader_sect}}{}
1417      }{}
1418    }
1419
```

```
1420  \stex_if_html_backend:T{
1421    \cs_new_protected:Nn \__stex_doc_check_topsect: {
1422      \int_case:nnF \l_stex_docheader_sect {
1423        {0}{\cs_if_exist:NTF \thepart {
1424          \stex_annotate_invisible:nn{shtml:sectionlevel=0}{}
1425        }{
1426          \int_incr:N \l_stex_docheader_sect
1427          \__stex_doc_check_topsect:
1428        }}
1429        {1}{\cs_if_exist:NTF \thechapter {
1430          \stex_annotate_invisible:nn{shtml:sectionlevel=1}{}
1431        }{
1432          \int_incr:N \l_stex_docheader_sect
1433          \__stex_doc_check_topsect:
1434        }}
1435      }{
1436        \stex_annotate_invisible:nn{shtml:sectionlevel={\int_use:N\l_stex_docheader_sect}}{}
1437      }
1438    }
1439    \AtBeginDocument{\__stex_doc_check_topsect:}
1440  }
1441
1442  \AtBeginDocument{
1443    \cs_if_exist:NTF\frontmatter{
1444      \let\__stex_doc_orig_frontmatter\frontmatter
1445      \let\frontmatter\relax
1446    }{
1447      \tl_set:Nn\__stex_doc_orig_frontmatter{
1448        \clearpage
1449        %\@mainmatterfalse
1450        \pagenumbering{roman}
1451      }
1452    }
1453    \cs_if_exist:NTF\backmatter{
1454      \let\__stex_doc_orig_backmatter\backmatter
1455      \let\backmatter\relax
1456    }{
1457      \tl_set:Nn\__stex_doc_orig_backmatter{
1458        \clearpage
1459        %\@mainmatterfalse
1460        \pagenumbering{roman}
1461      }
1462    }
1463    \newenvironment{frontmatter}{
1464      \__stex_doc_orig_frontmatter
1465    }{
1466      \cs_if_exist:NTF\mainmatter{
1467        \mainmatter
1468      }{
1469        \clearpage
1470        %\@mainmattertrue
1471        \pagenumbering{arabic}
1472      }
1473    }
```

167

```
1474    \newenvironment{backmatter}{
1475      \__stex_doc_orig_backmatter
1476    }{
1477      \cs_if_exist:NTF\mainmatter{
1478        \mainmatter
1479      }{
1480        \clearpage
1481        %\@mainmattertrue
1482        \pagenumbering{arabic}
1483      }
1484    }
1485  }
```

(*End definition for* \setsectionlevel. *This function is documented on page* *71*.)

13.4.3 References

```
1486  ⟨@@=stex_refs⟩
```

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

```
1487  \iow_new:N \c__stex_refs_iow
1488  \AtBeginDocument{\iow_open:Nn \c__stex_refs_iow {\jobname.sref}}
1489  \AtEndDocument{\iow_close:N \c__stex_refs_iow}
```

The following macros are written to the .aux-file, and hence use LATEX2e character code scheme:

```
1490  \cs_new_protected:Npn \STEXInternalSrefRestoreTarget #1#2#3#4#5 {}
1491
1492  \cs_new_protected:Npn \STEXInternalSetSrefSymURL #1 #2 {
1493    \str_gset:cn{g_stex_sref_sym_\tl_to_str:n{#1}_target}{#2}
1494  }
1495
```

\stex_ref_new_doc_target:n
\sreflabel

```
1496  \seq_new:N \g__stex_refs_files_seq
1497  \int_new:N \l__stex_refs_unnamed_counter_int
1498
1499  \cs_new_protected:Nn \_stex_ref_new_id:n {
1500    \str_if_empty:nTF {#1}{
1501      \int_gincr:N \l__stex_refs_unnamed_counter_int
1502      \str_set:Nx \l__stex_refs_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
1503    }{
1504      \str_set:Nn \l__stex_refs_str {#1}
1505    }
1506    \str_set:Nx \l_stex_ref_url_str {\stex_uri_use:N \l_stex_current_doc_uri ? \l__stex_refs_s
1507  }
1508
1509  \cs_new_protected:Nn \stex_ref_new_doc_target:n {
1510    \_stex_ref_new_id:n{#1}
1511    %\stex_uri_add_module:NNo \l__stex_refs_uri \l_stex_current_doc_uri \l__stex_refs_str
1512    %\stex_debug:nn{sref}{New~document~target:~\stex_uri_use:N \l__stex_refs_uri}
1513    \__stex_refs_add_doc_ref:xo {\stex_uri_use:N \l_stex_current_doc_uri} \l__stex_refs_str
1514    \stex_if_smsmode:F {
1515      \iow_now:Nx \c__stex_refs_iow {
```

```
1516        \STEXInternalSrefRestoreTarget
1517        {\stex_uri_use:N \l_stex_current_doc_uri}
1518        {\l__stex_refs_str}
1519        {\@currentcounter}
1520        {\@currentlabel}
1521        {
1522          \tl_if_exist:NT\@currentlabelname{
1523            \exp_args:No\exp_not:n\@currentlabelname
1524          }
1525        }
1526      }
1527      \exp_args:Nx \label {sref@\l_stex_ref_url_str}
1528      \stex_if_do_html:T {
1529        \pdfdest name "sref@\l_stex_ref_url_str" xyz\relax
1530      }
1531    }
1532 }
1533 \NewDocumentCommand \sreflabel {m} {\stex_ref_new_doc_target:n {#1}}
1534
1535 \cs_new_protected:Nn \__stex_refs_add_doc_ref:nn {
1536   \seq_if_in:NnTF \g__stex_refs_files_seq {#1} {
1537     \seq_if_in:cnF {g__stex_refs_#1_seq}{#2}{
1538       \seq_gput_left:cn{g__stex_refs_#1_seq}{#2}
1539     }
1540   }{
1541     \seq_gput_right:Nn \g__stex_refs_files_seq {#1}
1542     \seq_new:c{g__stex_refs_#1_seq}
1543     \seq_gput_left:cn{g__stex_refs_#1_seq}{#2}
1544   }
1545 }
1546 \cs_generate_variant:Nn \__stex_refs_add_doc_ref:nn {xo,xx}
```

(*End definition for* `\stex_ref_new_doc_target:n` *and* `\sreflabel`*. These functions are documented on page 110.*)

\sref   Optional arguments:
\extref
```
1547 \stex_keys_define:nnnn{sref / 1}{}{
1548   % TODO get rid of this
1549   fallback  .code:n = {},
1550   pre       .code:n = {},
1551   post      .code:n = {}
1552 }{archive file}
1553 \stex_keys_define:nnnn{sref / 2}{}{}{archive file, title}
1554
1555 \str_new:N \l__stex_refs_default_archive_str
1556 \str_new:N \l__stex_refs_default_file_str
1557 \tl_new:N \l__stex_refs_default_title_tl
1558
1559 \cs_set_protected:Nn \__stex_refs_set_keys_b:n {
1560   \tl_if_empty:nTF{#1}{
1561     \str_set_eq:NN \l_stex_key_archive_str \l__stex_refs_default_archive_str
1562     \str_set_eq:NN \l_stex_key_file_str \l__stex_refs_default_file_str
1563     \tl_set_eq:NN \l_stex_key_title_tl \l__stex_refs_default_title_tl
1564   }{
```

169

```
1565      \stex_keys_set:nn{ sref / 2 }{ #1 }
1566    }
1567  }
1568
1569  \newcommand\srefsetin[3][]{
1570    \str_set:Nx \l__stex_refs_default_archive_str {#1}
1571    \str_set:Nx \l__stex_refs_default_file_str {#2}
1572    \tl_set:Nn \l__stex_refs_default_title_tl {#3}
1573  }
1574
```

Auxiliary methods:

```
1575  \cs_new_protected:Nn \__stex_refs_find_uri:n {
1576    \str_clear:N \l__stex_refs_uri_str
1577    \stex_debug:nn{sref}{
1578      File:~\l_stex_key_file_str^^J
1579      Repo:\l_stex_key_archive_str
1580    }
1581    \str_if_empty:NTF \l_stex_key_file_str {
1582      \stex_debug:nn{sref}{Empty.~Checking~current~file~for~#1}
1583      \seq_if_exist:cT{g__stex_refs_\stex_uri_use:N \l_stex_current_doc_uri _seq}{
1584        \exp_args:Nnx \__stex_refs_find_uri_in_file:nnn{#1}
1585          {\stex_uri_use:N \l_stex_current_doc_uri}\seq_map_break:
1586      }
1587      \str_if_empty:NT \l__stex_refs_uri_str {
1588        \seq_map_inline:Nn \g__stex_refs_files_seq {
1589          \__stex_refs_find_uri_in_file:nnn{#1}{##1}{\seq_map_break:n{\seq_map_break:}}
1590        }
1591      }
1592    }{
1593      \str_if_empty:NTF \l_stex_key_archive_str {
1594        \prop_if_exist:NTF \l_stex_current_archive_prop {
1595          \__stex_refs_find_uri_in_prop_file:N \l_stex_current_archive_prop
1596        }{
1597          \stex_file_resolve:Nx \l__stex_refs_file
1598            { \stex_file_use:N \g_stex_current_file / .. / \l_stex_key_file_str }
1599          \str_set:Nx \l__stex_refs_uri_str { file:/ \stex_file_use:N \l__stex_refs_file }
1600        }
1601      }{
1602        \stex_require_archive:o \l_stex_key_archive_str
1603        \prop_set_eq:Nc \l__stex_refs_prop { c_stex_mathhub_\l_stex_key_archive_str _manifest_
1604        \__stex_refs_find_uri_in_prop_file:N \l__stex_refs_prop
1605      }
1606    }
1607  }
1608
1609  \cs_new_protected:Nn \__stex_refs_find_uri_in_prop_file:N {
1610    \str_set:Nx \l__stex_refs_uri_str {
1611      \stex_file_use:N \c_stex_mathhub_file /
1612      \prop_item:Nn #1 {id} /
1613        source / \l_stex_key_file_str .sref
1614    }
1615    \stex_file_resolve:No \l__stex_refs_file \l__stex_refs_uri_str
1616    \stex_uri_from_repo_file:NNNn \l__stex_refs_uri #1
```

```
1617      \l__stex_refs_file {narr}
1618    \str_set:Nx \l__stex_refs_uri_str {\stex_uri_use:N \l__stex_refs_uri}
1619  }
1620
1621  \cs_new_protected:Nn \__stex_refs_find_uri_in_file:nnn {
1622    \stex_debug:nn{sref}{Checking~file~#2}
1623    \seq_map_inline:cn{g__stex_refs_#2_seq}{
1624      \str_if_eq:nnT{#1}{##1}{
1625        \str_set:Nx \l__stex_refs_uri_str {\stex_uri_use:N \l_stex_current_doc_uri}
1626        \stex_debug:nn{sref}{Found.}
1627        #3
1628      }
1629    }
1630  }
```

Doing the actual referencing:

```
1631  \cs_new_protected:Nn \__stex_refs_do_autoref:n {
1632    \cs_if_exist:cTF{autoref}{
1633      \exp_args:Nx\autoref{sref@#1}
1634    }{
1635      \exp_args:Nx\ref{sref@#1}
1636    }
1637  }
1638
1639  \cs_new_protected:Nn \__stex_refs_do_sref:nn {
1640    \str_if_empty:NTF \l__stex_refs_uri_str {
1641      \str_if_empty:NTF \l_stex_key_file_str {
1642        \stex_debug:nn{sref}{autoref~on~\stex_uri_use:N \l_stex_current_doc_uri?#1}
1643        \exp_args:Ne \__stex_refs_do_autoref:n{\stex_uri_use:N \l_stex_current_doc_uri ? #1}
1644      }{
1645        \stex_debug:nn{sref}{srefin~on~#1}
1646        \__stex_refs_set_keys_b:n{ #2 }
1647        \__stex_refs_do_sref_in:n{#1}
1648      }
1649    }{
1650      \exp_args:NNo \seq_if_in:NnTF \g__stex_refs_files_seq \l__stex_refs_uri_str {
1651        \stex_debug:nn{sref}{Using~ref~file~\l__stex_refs_uri_str}
1652        \exp_args:Nnx \seq_if_in:cnTF{g_stex_ref_\l__stex_refs_uri_str _seq}{\detokenize{#1}}{
1653          \stex_debug:nn{sref}{Reference~found~in~ref~files;~autoref~on~\l__stex_refs_uri_str?
1654          \__stex_refs_do_autoref:n{\l__stex_refs_uri_str?#1}
1655        }{
1656          \str_if_empty:NTF \l_stex_key_file_str {
1657            \stex_debug:nn{sref}{in~empty;~autoref~on~\l__stex_refs_uri_str?#1}
1658            \__stex_refs_do_autoref:n{\l__stex_refs_uri_str?#1}
1659          }{
1660            \stex_debug:nn{sref}{in~non-empty;~srefin~on~\l__stex_refs_uri_str?#1}
1661            \__stex_refs_set_keys_b:n{ #2 }
1662            \__stex_refs_do_sref_in:n{#1}
1663          }
1664        }
1665      }{
1666        \stex_debug:nn{sref}{No~ref~file~found~for~\l__stex_refs_uri_str}
1667        \str_if_empty:NTF \l_stex_key_file_str {
1668          \stex_debug:nn{sref}{in~empty;~autoref~on~\l__stex_refs_uri_str?#1}
```

```
1669          \__stex_refs_do_autoref:n{\l__stex_refs_uri_str?#1}
1670       }{
1671          \stex_debug:nn{sref}{in~non-empty;~srefin~on~\l__stex_refs_uri_str?#1}
1672          \__stex_refs_set_keys_b:n{ #2 }
1673          \__stex_refs_do_sref_in:n{#1}
1674       }
1675    }
1676  }
1677 }
1678
1679 \cs_new_protected:Nn \__stex_refs_do_sref_in:n {
1680   \stex_debug:nn{sref}{In: \l_stex_key_file_str^^JRepo:\l_stex_key_archive_str}
1681   \stex_debug:nn{sref}{URI: \l__stex_refs_uri_str?#1}
1682   \tl_if_exist:cTF{r@sref@\l__stex_refs_uri_str?#1}{
1683     \__stex_refs_do_autoref:n{\l__stex_refs_uri_str?#1}
1684   }{
1685     %\msg_warning:nnn{stex}{warning/smsmissing}{<filename>}
1686     \group_begin:\catcode13=9\relax\catcode10=9\relax
1687       \str_if_empty:NTF \l_stex_key_archive_str {
1688         \prop_if_exist:NTF \l_stex_current_archive_prop {
1689           \str_set:Nx \l__stex_refs_file_str {
1690             \stex_file_use:N \c_stex_mathhub_file /
1691             \prop_item:Nn \l_stex_current_archive_prop { id }
1692             / source / \l_stex_key_file_str .sref
1693           }
1694         }{
1695           \str_set:Nx \l__stex_refs_file_str {
1696             \stex_file_use:N \g_stex_current_file / .. / \l_stex_key_file_str . sref
1697           }
1698         }
1699       }{
1700         \str_set:Nx \l__stex_refs_file_str {
1701           \stex_file_use:N \c_stex_mathhub_file / \l_stex_key_archive_str
1702           / source / \l_stex_key_file_str . sref
1703         }
1704       }
1705       \stex_file_resolve:No \l__stex_refs_file \l__stex_refs_file_str
1706       \str_set:Nx \l__stex_refs_file_str {\stex_file_use:N \l__stex_refs_file }
1707       \stex_debug:nn{sref}{File: \l__stex_refs_file_str }
1708       \exp_args:NNNx \exp_args:No \str_if_eq:nnTF \l__stex_refs_file_str {\stex_file_use:N\c
1709         \__stex_refs_do_autoref:n{
1710           \str_if_empty:NF\l__stex_refs_uri_str{\l__stex_refs_uri_str?}#1
1711         }
1712       }{
1713         \exp_args:No \IfFileExists \l__stex_refs_file_str {
1714           \tl_clear:N \l__stex_refs_return_tl
1715           \str_set:Nn \l__stex_refs_id_str {#1}
1716           \let\STEXInternalSrefRestoreTarget\__stex_refs_restore_target:nnnnn
1717           \use:c{@ @ input}{\l__stex_refs_file_str}
1718           \exp_args:No \tl_if_empty:nTF \l__stex_refs_return_tl {
1719             \exp_args:Nnno \msg_warning:nnnn{stex}{warning/smslabelmissing}\l__stex_refs_fil
1720             \__stex_refs_do_autoref:n{
1721               \str_if_empty:NF\l__stex_refs_uri_str{\l__stex_refs_uri_str?}#1
1722             }
```

```
1723        }{
1724          \l__stex_refs_return_tl
1725        }
1726      }{
1727        \exp_args:Nnno \msg_warning:nnn{stex}{warning/smsmissing}\l__stex_refs_file_str
1728        \__stex_refs_do_autoref:n{
1729          \str_if_empty:NF\l__stex_refs_uri_str{\l__stex_refs_uri_str?}#1
1730        }
1731      }
1732    }
1733    \group_end:
1734  }
1735 }
1736
1737 \cs_new_protected:Nn \__stex_refs_do_return:nnnn {
1738   \tl_set:Nn \l__stex_refs_return_tl {
1739     \stex_annotate:nn{shtml:sref={#4},shtml:srefin={\l__stex_refs_file_str}}{
1740       \use:c{#3autorefname}~#1\tl_if_empty:nF{#2}{~(#2)}
1741       \tl_if_empty:NF\l_stex_key_title_tl{
1742         {}~in~\l_stex_key_title_tl
1743       }
1744     }
1745   }
1746 }
1747
1748 \cs_new_protected:Nn \__stex_refs_restore_target:nnnnn {
1749   \str_if_empty:NTF \l__stex_refs_uri_str {
1750     \exp_args:No \str_if_eq:nnT \l__stex_refs_id_str {#2}{
1751       \__stex_refs_do_return:nnnn{#4}{#5}{#3}{#1?#2}
1752     }
1753   }{
1754     \stex_debug:nn{sref}{\l__stex_refs_uri_str{}~ == ~ #1 ~ ?}
1755     \exp_args:No \str_if_eq:nnT \l__stex_refs_uri_str {#1}{
1756       \stex_debug:nn{sref}{\l__stex_refs_id_str~ == ~ #2 ~ ?}
1757       \exp_args:No \str_if_eq:nnT \l__stex_refs_id_str {#2}{
1758         \stex_debug:nn{sref}{success!}
1759         \__stex_refs_do_return:nnnn{#4}{#5}{#3}{#1?#2}
1760         \endinput
1761       }
1762     }
1763   }
1764 }
```

The actual macros:

```
1765 \NewDocumentCommand \sref { O{} m O{}}{
1766   \stex_keys_set:nn { sref / 1 }{ #1 }
1767   \__stex_refs_find_uri:n { #2 }
1768   \__stex_refs_do_sref:nn{#2}{#3}
1769 }
1770 \NewDocumentCommand \extref { O{} m m}{
1771   \stex_keys_set:nn { sref / 1 }{ #1 }
1772   \__stex_refs_find_uri:n { #2 }
1773   \__stex_refs_set_keys_b:n{ #3 }
1774   \str_if_empty:NT \l_stex_key_file_str {
1775     \msg_error:nn{stex}{error/extrefmissing}
```

```
1776    }
1777    \__stex_refs_do_sref_in:n{#2}
1778 }
```

(*End definition for* `\sref` *and* `\extref`. *These functions are documented on page 73.*)

`\stex_ref_new_sym_target:n`

```
1779 \cs_new_protected:Nn \stex_ref_new_symbol:n {
1780   \cs_if_exist:cF{r@sref@sym@\tl_to_str:n{#1}}{
1781     \__stex_refs_new_symbol:n{#1}
1782   }
1783 }
1784
1785 \cs_new_protected:Nn \_stex_sref_do_aux:n {
1786   #1 \iow_now:Nn \@auxout {#1}
1787 }
1788
1789 \cs_new_protected:Nn \__stex_refs_new_symbol:n {
1790   \prop_if_exist:NTF \l_stex_current_archive_prop {
1791     \prop_get:NnNTF \l_stex_current_archive_prop {docurl} \l__stex_refs_str {
1792       \exp_args:Ne \_stex_sref_do_aux:n {
1793         \STEXInternalSetSrefSymURL{#1}{\l__stex_refs_str / symbol? #1}
1794       }
1795     }{
1796       \_stex_sref_do_aux:n { \STEXInternalSetSrefSymURL{#1}{} }
1797     }
1798   }{
1799     \_stex_sref_do_aux:n { \STEXInternalSetSrefSymURL{#1}{} }
1800   }
1801 }
1802
1803 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
1804   \exp_args:Ne \label{\tl_to_str:n{sref@sym@ #1}}
1805 }
1806
1807 \cs_new_protected:Nn \stex_ref_new_sym_target:nn {
1808   \str_if_eq:nnTF{#1}{#2}{
1809     \stex_ref_new_sym_target:n{#1}
1810   }{
1811     \str_gset:cn{g_stex_sref_sym_ #1 _label}{#2}
1812   }
1813 }
```

(*End definition for* `\stex_ref_new_sym_target:n`. *This function is documented on page 110.*)

`\srefsym`

```
1814 \NewDocumentCommand \srefsym { m m}{
1815   \stex_get_symbol:n { #1 }
1816   \exp_args:Ne
1817   \__stex_refs_sym_aux:nn{\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
1818 }
1819
1820 \cs_new_protected:Npn \__stex_refs_do_internal_link:nn #1 {
1821   \cs_if_exist:NTF \hyperref {
1822     \hyperref[{#1}]
```

174

```
1823    }\use:n
1824 }
1825
1826 \cs_new_protected:Npn \__stex_refs_do_url_link:nn {
1827    \cs_if_exist:NTF \href \href \use_ii:nn
1828 }
1829
1830 \cs_new_protected:Npn \__stex_refs_sym_aux:nn #1 {
1831    \cs_if_exist:cTF{r@sref@sym@\tl_to_str:n{#1}}{
1832       \exp_args:Ne \__stex_refs_do_internal_link:nn{\tl_to_str:n{sref@sym@#1}}
1833    }{
1834       \str_if_exist:cTF{g_stex_sref_sym_#1_label}{
1835          \exp_args:Ne \__stex_refs_sym_aux:nn{\use:c{g_stex_sref_sym_#1_label}}
1836       }{
1837          \str_if_empty:cTF{g_stex_sref_sym_#1_target}{
1838             \exp_args:Ne \__stex_refs_do_internal_link:nn{\tl_to_str:n{sref@sym@#1}}
1839          }{
1840             \exp_args:Ne \__stex_refs_do_url_link:nn{\use:c{g_stex_sref_sym_#1_target}}
1841          }
1842       }
1843    }
1844 }
```

(*End definition for* \srefsym. *This function is documented on page* *78.*)

\srefsymuri

```
1845 \cs_new_protected:Npn \srefsymuri #1 {
1846    \__stex_refs_sym_aux:nn{#1}
1847 }
```

(*End definition for* \srefsymuri. *This function is documented on page* *78.*)

13.4.4 Inputs

```
1848 ⟨@@=stex_inputs⟩
```

\stex_resolve_path_pair:Nnn
\stex_resolve_path_pair:Nxx

```
1849 \cs_new_protected:Nn \stex_resolve_path_pair:Nnn {
1850    \stex_debug:nn{resolving~path}{#3~in~[#2]}
1851    \str_if_empty:nTF{#2}{
1852       \prop_if_exist:NTF \l_stex_current_archive_prop {
1853          \str_set:Nx #1 {\stex_file_use:N \c_stex_mathhub_file /
1854          \prop_item:Nn \l_stex_current_archive_prop { id } / source /
1855          #3}
1856          \stex_debug:nn{resolving~path}{In~current~archive~
1857             \prop_item:Nn \l_stex_current_archive_prop { id }
1858             ;~result:~#1}
1859       }{
1860          \str_set:Nx #1 {\stex_file_use:N \c_stex_pwd_file / .. / #3 }
1861          \stex_debug:nn{resolving~path}{No~current~archive;~result:~#1}
1862       }
1863    }{
1864       \stex_require_archive:n { #2 }
1865       \str_set:Nx #1 {\stex_file_use:N \c_stex_mathhub_file /
1866       \prop_item:cn {c_stex_mathhub_#2_manifest_prop} { id } / source /
```

```
1867        #3}
1868        \stex_debug:nn{resolving~path}{result:~#1}
1869      }
1870 }
1871 \cs_generate_variant:Nn \stex_resolve_path_pair:Nnn {Nxx}
```

(*End definition for* \stex_resolve_path_pair:Nnn. *This function is documented on page* *123*.)

\inputref
\mhinput
\ifinputref

```
1872 \newif \ifinputref \inputreffalse
1873
1874 \cs_new_protected:Nn \__stex_inputs_mhinput:nn {
1875   \stex_in_archive:nn {#1} {
1876     \ifinputref
1877       \stex_input_with_hooks:n{ \stex_file_use:N \c_stex_mathhub_file / ##1 / source / #2 }
1878     \else
1879       \inputreftrue
1880       \stex_input_with_hooks:n{ \stex_file_use:N \c_stex_mathhub_file / ##1 / source / #2
1881       \inputreffalse
1882     \fi
1883   }
1884 }
1885
1886 \NewDocumentCommand \mhinput { O{} m}{
1887   \exp_args:NNx\exp_args:Nnx\__stex_inputs_mhinput:nn{ \tl_to_str:n{#1} }{ \tl_to_str:n{#2}
1888 }
1889
1890 \cs_new_protected:Nn \__stex_inputs_inputref_html:nn {
1891   \str_clear:N \l_tmpa_str
1892   \prop_get:NnNF \l_stex_current_archive_prop { narr } \l_tmpa_str {
1893     \prop_get:NnNF \l_stex_current_archive_prop { ns } \l_tmpa_str {}
1894   }
1895   \tl_if_empty:nTF{ #1 }{
1896     \IfFileExists{#2}{
1897       \ifvmode\noindent\fi\stex_annotate_invisible:nn{shtml:inputref={
1898         \l_tmpa_str / #2
1899       }}{}
1900     }{
1901       \stex_input_with_hooks:n{#2}
1902     }
1903   }{
1904     \IfFileExists{ \stex_file_use:N \c_stex_mathhub_file / #1 / source / #2 }{
1905       \par\stex_annotate_invisible:nn{shtml:inputref={
1906         \l_tmpa_str / #2
1907       }}{~}
1908     }{
1909       \input{ \stex_file_use:N \c_stex_mathhub_file / #1 / source / #2 }
1910     }
1911   }
1912 }
1913
1914 \cs_new_protected:Nn \__stex_inputs_inputref_pdf:nn {
1915   \begingroup
1916     \inputreftrue
```

```
1917    \tl_if_empty:nTF{ #1 }{
1918        \stex_input_with_hooks:n{#2}
1919    }{
1920        \stex_input_with_hooks:n{ \stex_file_use:N \c_stex_mathhub_file / #1 / source / #2 }
1921    }
1922    \endgroup
1923 }
1924
1925 \cs_new_protected:Nn \__stex_inputs_inputref:nn {
1926    \stex_in_archive:nn {#1} {
1927        \stex_if_html_backend:TF
1928            \__stex_inputs_inputref_html:nn
1929            \__stex_inputs_inputref_pdf:nn
1930        {##1}{#2}
1931    }
1932 }
1933
1934 \NewDocumentCommand \inputref { O{} m }{
1935    \exp_args:NNx \exp_args:Nnx \__stex_inputs_inputref:nn{ \tl_to_str:n{#1} }{ \tl_to_str:n{#
1936 }
```

(*End definition for* \inputref, \mhinput, *and* \ifinputref. *These functions are documented on page* *72.*)

\addmhbibresource

```
1937 \cs_new_protected:Nn \__stex_inputs_bibresource:n {
1938    \__stex_inputs_up_archive:nn{#1}{bib}
1939    \seq_if_empty:NTF \l__stex_inputs_libinput_files_seq {
1940        \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\addmhbibresource}{#1.bib}
1941    }{
1942        \seq_map_inline:Nn \l__stex_inputs_libinput_files_seq {
1943            \addbibresource{ ##1 }
1944        }
1945    }
1946 }
1947 \newcommand\addmhbibresource[2][]{
1948    \tl_if_empty:nTF{#1}{
1949        \__stex_inputs_bibresource:n{#2}
1950    }{
1951        \stex_in_archive:nn{#1}{\__stex_inputs_bibresource:n{#2}}
1952    }
1953 }
```

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

\IfInputref

```
1954 \stex_if_html_backend:TF{
1955    \newcommand \IfInputref[2]{
1956        \stex_annotate:nn{shtml:ifinputref=true}{#1}
1957        \stex_annotate:nn{shtml:ifinputref=false}{#2}
1958    }
1959 }{
1960    \newcommand \IfInputref[2]{
1961        \ifinputref #1 \else #2 \fi
1962    }
```

```
1963 }
```
(End definition for `\IfInputref`*. This function is documented on page 72.)*

<code>\libinput</code>
```
1964 \cs_new_protected:Nn \__stex_inputs_up_archive:nn {
1965   \prop_if_exist:NF \l_stex_current_archive_prop {
1966     \msg_error:nnn{stex}{error/notinarchive}\libinput
1967   }
1968   \prop_get:NnNF \l_stex_current_archive_prop {id} \l__stex_inputs_id_str {
1969     \msg_error:nnn{stex}{error/notinarchive}\libinput
1970   }
1971   \seq_clear:N \l__stex_inputs_libinput_files_seq
1972   \seq_set_eq:NN \l__stex_inputs_path_seq \c_stex_mathhub_file
1973   \seq_set_split:NnV \l__stex_inputs_id_seq / \l__stex_inputs_id_str
1974
1975   \bool_while_do:nn { ! \seq_if_empty_p:N \l__stex_inputs_id_seq }{
1976     \str_set:Nx \l__stex_inputs_path_str {\stex_file_use:N \l__stex_inputs_path_seq / meta-i
1977     \IfFileExists{ \l__stex_inputs_path_str }{
1978       \seq_put_right:No \l__stex_inputs_libinput_files_seq \l__stex_inputs_path_str
1979     }{}
1980     \seq_pop_left:NN \l__stex_inputs_id_seq \l__stex_inputs_path_str
1981     \seq_put_right:No \l__stex_inputs_path_seq \l__stex_inputs_path_str
1982   }
1983
1984   \str_set:Nx \l__stex_inputs_path_str {\stex_file_use:N \l__stex_inputs_path_seq / lib / #1
1985   \IfFileExists{ \l__stex_inputs_path_str }{
1986     \seq_put_right:No \l__stex_inputs_libinput_files_seq \l__stex_inputs_path_str
1987   }{}
1988 }
1989
1990 \cs_new_protected:Nn \__stex_inputs_libinput:n {
1991   \__stex_inputs_up_archive:nn{#1}{tex}
1992   \seq_if_empty:NTF \l__stex_inputs_libinput_files_seq {
1993     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
1994   }{
1995     \seq_map_inline:Nn \l__stex_inputs_libinput_files_seq {
1996       \input{ ##1 }
1997     }
1998   }
1999 }
2000
2001 \newcommand \libinput [2][] {
2002   \tl_if_empty:nTF{#1}{
2003     \__stex_inputs_libinput:n{#2}
2004   }{
2005     \stex_in_archive:nn{#1}{\__stex_inputs_libinput:n{#2}}
2006   }
2007 }
```
(End definition for `\libinput`*. This function is documented on page 69.)*

<code>\libusepackage</code>
```
2008 \newcommand\libusepackage[2][]{
2009   \__stex_inputs_up_archive:nn{#2}{sty}
```

```
2010    \int_compare:nNnTF {\seq_count:N \l__stex_inputs_libinput_files_seq} = 1 {
2011      \str_set:Nx \l__stex_inputs_tmp_str {\seq_item:Nn \l__stex_inputs_libinput_files_seq 1}
2012      \exp_args:Nne \use:n {\usepackage[#1]} {
2013        \str_range:Nnn\l__stex_inputs_tmp_str 1 {-5}
2014      }
2015    }{
2016      \stex_fatal_error:nnn{error/nofile}{\libusepackage}{#1.sty}
2017    }
2018  }
```

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

```
2019  \str_new:N \l__stex_inputs_gin_repo_str
2020  \ltx@ifpackageloaded{graphicx}{\use:n}{\AtEndOfPackageFile{graphicx}}{
2021    \define@key{Gin}{archive}{
2022      \tl_set:Nx\Gin@mhrepos{\stex_file_use:N \c_stex_mathhub_file / #1 / source /}
2023    }
2024    \providecommand\mhgraphics[2][]{
2025      \tl_set:Nx\Gin@mhrepos{
2026        \stex_file_use:N \c_stex_mathhub_file / \prop_item:Nn \l_stex_current_archive_prop {id
2027      }
2028      \setkeys{Gin}{#1}
2029      \includegraphics[#1]{ \Gin@mhrepos #2 }
2030    }
2031    \providecommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
2032  }
2033
2034  \ltx@ifpackageloaded{listings}{\use:n}{\AtEndOfPackageFile{listings}}{
2035    \define@key{lst}{archive}{
2036      \def\lst@mhrepos{\stex_file_use:N \c_stex_mathhub_file / #1 / source /}
2037    }
2038    \newcommand\lstinputmhlisting[2][]{%
2039      \def\lst@mhrepos{
2040        \stex_file_use:N \c_stex_mathhub_file / \prop_item:Nn \l_stex_current_archive_prop {id
2041      }
2042      \setkeys{lst}{#1}%
2043      \lstinputlisting[#1]{\lst@mhrepos #2}}
2044    \newcommand\clstinputmhlisting[2][]{\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
2045  }
2046
2047  \ltx@ifpackageloaded{tikzinput}{\use:n}{\AtEndOfPackageFile{tikzinput}}{
2048    \define@key{Gin}{archive}{
2049      \str_set:Nn \l__stex_inputs_gin_repo_str {#1}
2050      \def\Gin@mhrepos{\stex_file_use:N \c_stex_mathhub_file / #1 / source /}
2051    }
2052    \newcommand\mhtikzinput[2][]{%
2053      \str_clear:N \l__stex_inputs_gin_repo_str
2054      \def\Gin@mhrepos{
2055        \stex_file_use:N \c_stex_mathhub_file / \prop_item:Nn \l_stex_current_archive_prop {id
2056      }
2057      \setkeys{Gin}{#1}%
2058      \exp_args:No \stex_in_archive:nn \l__stex_inputs_gin_repo_str {
2059        \tikzinput[#1]{\Gin@mhrepos #2}
```

```
2060        }
2061      }
2062    \newcommand\cmhtikzinput[2][]{\begin{center}\mhtikzinput[#1]{#2}\end{center}}
2063  }
```

(*End definition for* \mhgraphics *and others. These functions are documented on page* *72.*)

\libusetikzlibrary

```
2064  \cs_new_protected:Nn \__stex_inputs_usetikzlibrary:nn {
2065    \pgfkeys@spdef\pgf@temp{#1}
2066    \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
2067    \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgfut
2068    \expandafter\edef\csname tikz@library@#1@atcode\endcsname{\the\catcode`\@}
2069    \expandafter\edef\csname tikz@library@#1@barcode\endcsname{\the\catcode`\|}
2070    \expandafter\edef\csname tikz@library@#1@dollarcode\endcsname{\the\catcode`\$}
2071    \catcode`\@=11
2072    \catcode`\|=12
2073    \catcode`\$=3
2074    \pgfutil@InputIfFileExists{#2}{}{}
2075    \catcode`\@=\csname tikz@library@#1@atcode\endcsname
2076    \catcode`\|=\csname tikz@library@#1@barcode\endcsname
2077    \catcode`\$=\csname tikz@library@#1@dollarcode\endcsname
2078  }
2079
2080  \cs_new_protected:Npn \libusetikzlibrary #1 {
2081    \cs_if_exist:NF \usetikzlibrary {
2082      \msg_error:nnx{stex}{error/notikz}{\tl_to_str:n{\libusetikzlibrary}}
2083    }
2084    \__stex_inputs_up_archive:nn{tikzlibrary#1}{code.tex}
2085    \int_compare:nNnTF {\seq_count:N \l__stex_inputs_libinput_files_seq} = 1 {
2086      \exp_args:Nne \__stex_inputs_usetikzlibrary:nn{#1}{ \seq_item:Nn \l__stex_inputs_libinpu
2087    }{
2088      \stex_fatal_error:nnn{error/nofile}{\libusetikzlibrary}{tikzlibrary#1.code.tex}
2089    }
2090  }
```

(*End definition for* \libusetikzlibrary. *This function is documented on page* *107.*)

13.5   SMS Mode

```
2091  ⟨@@=stex_smsmode⟩
```

Macros and environments allowed in sms mode:

```
2092  \tl_new:N \g__stex_smsmode_allowed_tl
2093  \tl_new:N \g__stex_smsmode_allowed_escape_tl
2094  \seq_new:N \g__stex_smsmode_allowedenvs_seq
```

\stex_sms_allow:N
\stex_sms_allow_escape:N
\stex_sms_allow_env:n

```
2095  \cs_new_protected:Nn \stex_sms_allow:N {
2096    \tl_gput_right:Nn \g__stex_smsmode_allowed_tl {#1}
2097  }
2098
2099  \cs_new_protected:Nn \stex_sms_allow_escape:N {
2100    \tl_gput_right:Nn \g__stex_smsmode_allowed_escape_tl {#1}
2101  }
```

180

```
2102
2103 \cs_new_protected:Nn \stex_sms_allow_env:n {
2104    \exp_args:NNx \seq_gput_right:Nn \g__stex_smsmode_allowedenvs_seq {\tl_to_str:n{#1}}
2105 }
```

(*End definition for* `\stex_sms_allow:N`*,* `\stex_sms_allow_escape:N`*, and* `\stex_sms_allow_env:n`*. These functions are documented on page* *124.*)

Some initial allowed macros:

```
2106 \stex_sms_allow:N \makeatletter
2107 \stex_sms_allow:N \makeatother
2108 \stex_sms_allow:N \ExplSyntaxOn
2109 \stex_sms_allow:N \ExplSyntaxOff
2110 \stex_sms_allow:N \rustexBREAK
```

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

```
2111 \bool_new:N \g__stex_smsmode_bool
2112 \bool_set_false:N \g__stex_smsmode_bool
2113 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
2114    \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
2115 }
```

(*End definition for* `\stex_if_smsmode:TF`*. This function is documented on page* *124.*)

`\stex_sms_allow_import:Nn`
`\stex_sms_allow_import_env:nn`

```
2116 \tl_new:N \g__stex_smsmode_allowed_import_tl
2117 \seq_new:N \g__stex_smsmode_allowed_import_env_seq
2118 \cs_new_protected:Nn \stex_sms_allow_import:Nn {
2119    \tl_gput_right:Nn \g__stex_smsmode_allowed_import_tl {#1}
2120    \tl_gset:cn{\tl_to_str:n{#1}~-~smsmode} {#2}
2121 }
2122 \cs_new_protected:Nn \stex_sms_allow_import_env:nn {
2123    \exp_args:NNx \seq_gput_right:Nn \g__stex_smsmode_allowed_import_env_seq {\tl_to_str:n{#1
2124    \tl_gset:cn{\tl_to_str:n{#1}~-~env~-~smsmode} {#2}
2125 }
2126
2127 \tl_new:N \g_stex_sms_import_code
```

(*End definition for* `\stex_sms_allow_import:Nn` *and* `\stex_sms_allow_import_env:nn`*. These functions are documented on page* *125.*)

`\stex_file_in_smsmode:nn`
`\stex_file_in_smsmode:on`

```
2128 \cs_new_protected:Nn \__stex_smsmode_in_smsmode:n { \stex_suppress_html:n {
2129    \vbox_set:Nn \l_tmpa_box {
2130       \bool_set_true:N \g__stex_smsmode_bool
2131       \bool_set_false:N \_stex_html_do_output_bool
2132       #1
2133    }
2134    %\box_clear:N \l_tmpa_box
2135 } }
2136
2137 \quark_new:N \q__stex_smsmode_break
2138
2139 \cs_new_protected:Nn \__stex_smsmode_start_smsmode:n {
2140    \everyeof{\q__stex_smsmode_break\exp_not:N}
2141    \let\stex_smsmode_do:\__stex_smsmode_smsmode_do:
```

```
2142    \exp_after:wN \exp_after:wN \exp_after:wN
2143    \stex_smsmode_do:
2144    \cs:w @ @ input\cs_end: "#1" \relax
2145 }
2146
2147 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
2148    \seq_gclear:N \l__stex_smsmode_importmodules_seq
2149    \seq_gclear:N \l__stex_smsmode_sigmodules_seq
2150    \tl_clear:N \g_stex_sms_import_code
2151    \group_begin:
2152      \let \l_stex_metatheory_uri \c_stex_default_metatheory
2153      \cs_set:Npn \stex_check_term:n ##1 {}
2154      \seq_clear:N \l_stex_all_modules_seq
2155      \str_clear:N \l_stex_current_module_str
2156      #2
2157      \stex_filestack_push:n{#1}
2158      \__stex_smsmode_in_smsmode:n {
2159        \let \__stex_smsmode_do_aux_curr:N \__stex_smsmode_do_aux_imports:N
2160        \tl_map_inline:Nn \g__stex_smsmode_allowed_import_tl {
2161          \use:c{\tl_to_str:n{##1}~-~smsmode}
2162        }
2163        \seq_map_inline:Nn \g__stex_smsmode_allowed_import_env_seq {
2164          \use:c{\tl_to_str:n{##1}~-~env~-~smsmode}
2165        }
2166        \__stex_smsmode_start_smsmode:n{#1}
2167      }
2168      \__stex_smsmode_in_smsmode:n \g_stex_sms_import_code
2169      \__stex_smsmode_in_smsmode:n {
2170        \let \__stex_smsmode_do_aux_curr:N \__stex_smsmode_do_aux_normal:N
2171        \__stex_smsmode_start_smsmode:n{#1}
2172      }
2173      \stex_filestack_pop:
2174    \group_end:
2175 }
2176 \cs_generate_variant:Nn \stex_file_in_smsmode:nn {on}
```

(*End definition for* `\stex_file_in_smsmode:nn`. *This function is documented on page 124.*)

`\stex_smsmode_do:`

```
2177 \cs_new_protected:Nn \__stex_smsmode_smsmode_do: {
2178 %\stex_if_smsmode:T {
2179    \__stex_smsmode_do:w
2180  %}
2181 }
2182 \let\stex_smsmode_do:\relax
2183
2184
2185 \cs_new:Nn \__stex_smsmode_check_cs:NNn {
2186    \exp_after:wN\if\exp_after:wN\relax\exp_not:N#3
2187    \exp_after:wN#1\exp_after:wN#3\else
2188    \exp_after:wN#2\fi
2189 }
2190
2191 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
```

```
2192    \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 }}{
2193      \__stex_smsmode_check_cs:NNn \__stex_smsmode_do_aux:N \__stex_smsmode_do:w { #1 }
2194    }{
2195      \__stex_smsmode_do:w
2196    }
2197 }

2198
2199 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
2200   \cs_if_eq:NNF #1 \q__stex_smsmode_break {
2201     \__stex_smsmode_do_aux_curr:N #1
2202   }
2203 }

2204
2205 \cs_new_protected:Nn \__stex_smsmode_do_aux_imports:N {
2206 % \stex_debug:nn{sms}{Checking~\tl_to_str:n{#1}~in~import}
2207   \tl_if_in:NnTF \g__stex_smsmode_allowed_import_tl {#1} {
2208     \stex_debug:nn{sms}{Executing~\tl_to_str:n{#1}~in~import}
2209     #1
2210   }{
2211     \cs_if_eq:NNTF \begin #1 {
2212       \__stex_smsmode_check_begin:Nn \g__stex_smsmode_allowed_import_env_seq
2213     }{
2214       \cs_if_eq:NNTF \end #1 {
2215         \__stex_smsmode_check_end:Nn \g__stex_smsmode_allowed_import_env_seq
2216       }{
2217         \__stex_smsmode_do:w
2218       }
2219     }
2220   }
2221 }

2222
2223 \cs_new_protected:Nn \__stex_smsmode_do_aux_normal:N {
2224 % \stex_debug:nn{sms}{Checking~\tl_to_str:n{#1}~in~sms~mode}
2225   \tl_if_in:NnTF \g__stex_smsmode_allowed_tl {#1} {
2226     \stex_debug:nn{sms}{Executing~\tl_to_str:n{#1}}
2227     #1\__stex_smsmode_do:w
2228   }{
2229     \tl_if_in:NnTF \g__stex_smsmode_allowed_escape_tl {#1} {
2230       \stex_debug:nn{sms}{Executing~escaped~\tl_to_str:n{#1}}
2231       #1
2232     }{
2233       \cs_if_eq:NNTF \begin #1 {
2234         \__stex_smsmode_check_begin:Nn \g__stex_smsmode_allowedenvs_seq
2235       }{
2236         \cs_if_eq:NNTF \end #1 {
2237           \__stex_smsmode_check_end:Nn \g__stex_smsmode_allowedenvs_seq
2238         }{
2239           \__stex_smsmode_do:w
2240         }
2241       }
2242     }
2243   }
2244 }

2245
```

```
2246 \cs_new_protected:Nn \__stex_smsmode_check_begin:Nn {
2247 % \stex_debug:nn{sms}{Checking~environment~#2}
2248   \seq_if_in:NxTF #1 { \tl_to_str:n{#2} }{
2249     \stex_debug:nn{sms}{Environment~#2}
2250     \begin{#2}
2251   }{
2252     \__stex_smsmode_do:w
2253   }
2254 }
2255 \cs_new_protected:Nn \__stex_smsmode_check_end:Nn {
2256 %  \stex_debug:nn{sms}{Checking~end~environment~#2}
2257   \seq_if_in:NxTF #1 { \tl_to_str:n{#2} }{
2258     \stex_debug:nn{sms}{End~Environment~#2}
2259     \end{#2}\__stex_smsmode_do:w
2260   }{
2261     %\str_if_eq:nnTF{#2}{document} \endinput
2262     \__stex_smsmode_do:w
2263   }
2264 }
```

(*End definition for* \stex_smsmode_do:. *This function is documented on page 125.*)

13.6 Modules

13.6.1 The smodule-environment

```
2265 ⟨@@=stex_modules⟩
```

\l_stex_current_module_str   The current module:

```
2266 \str_new:N \l_stex_current_module_str
```

(*End definition for* \l_stex_current_module_str. *This variable is documented on page 111.*)

\l_stex_all_modules_seq   Stores all modules currently in scope

```
2267 \seq_new:N \l_stex_all_modules_seq
```

(*End definition for* \l_stex_all_modules_seq. *This variable is documented on page 111.*)

\stex_every_module:n

```
2268 ⟨@@=stex_module_setup⟩
2269 \tl_clear:N \g_stex_every_module_tl {
2270 }
2271 \cs_new_protected:Nn \stex_every_module:n {
2272   \tl_gput_right:Nn \g_stex_every_module_tl { #1 }
2273 }
```

(*End definition for* \stex_every_module:n. *This function is documented on page 111.*)

\stex_module_setup:n   Sets up a new module:

```
2274 \cs_new_protected:Npn \stex_module_setup:n {
2275   \stex_if_in_module:TF \__stex_module_setup_setup_nested:n \__stex_module_setup_setup_top:n
2276 }
2277
2278 \cs_new_protected:Nn \__stex_module_setup_setup_top:n {
2279   \__stex_module_setup_get_uri_str:n{#1}
```

```
2280    \stex_debug:nn{module}{Module~URI:~\l__stex_module_setup_ns_str?#1}
2281    \str_if_empty:NTF \l_stex_key_sig_str
2282    \_stex_module_setup_top_nosig:n \__stex_module_setup_setup_top_sig:n {\l__stex_module_setu
2283    \stex_metagroup_new:o \l_stex_current_module_str
2284    \g_stex_every_module_tl
2285    \stex_execute_in_module:x {
2286       \_stex_do_deprecation:n{#1}
2287    }
2288    \__stex_module_setup_load_meta:
2289 }
2290
2291 \cs_new_protected:Nn \_stex_module_setup_top_nosig:n {
2292    \stex_if_module_exists:nTF{#1}{
2293       \stex_debug:nn{modules}{(already exists)}
2294    }{
2295       \tl_gclear:c{c_stex_module_ #1 _code}
2296       \prop_gclear:c{c_stex_module_ #1 _morphisms_prop }
2297       \prop_gclear:c{c_stex_module_ #1 _symbols_prop }
2298       \prop_gclear:c{c_stex_module_ #1 _notations_prop }
2299    }
2300    \str_set:Nx \l_stex_current_module_str {#1}
2301    \seq_put_right:No \l_stex_all_modules_seq \l_stex_current_module_str
2302 }
2303
2304 \cs_new_protected:Nn \__stex_module_setup_setup_top_sig:n {
2305    \stex_if_module_exists:nTF{#1}{
2306       \stex_debug:nn{modules}{(already exists)}
2307    }{
2308       \stex_debug:nn{modules}{(needs loading)}
2309       \__stex_module_setup_load_sig:
2310    }
2311    %\stex_if_smsmode:F { % WHY?
2312       \stex_activate_module:x {
2313          #1
2314       }
2315    %}
2316    \str_set:Nx\l_stex_current_module_str{#1}
2317 }
2318
2319 \cs_new_protected:Nn \__stex_module_setup_load_sig: {
2320    \stex_file_split_off_ext:NN \l__stex_module_setup_sigfile \g_stex_current_file
2321    \stex_file_split_off_lang:NN \l__stex_module_setup_sigfile \l__stex_module_setup_sigfile
2322    \exp_args:Ne \stex_file_in_smsmode:nn {
2323       \stex_file_use:N \l__stex_module_setup_sigfile . \l_stex_key_sig_str . tex
2324    }{}
2325 }
2326
2327 \cs_new_protected:Nn \__stex_module_setup_setup_nested:n {
2328    \exp_after:wN
2329       \__stex_module_setup_split_module:n \l_stex_current_module_str \__stex_module_setup_end:
2330    \stex_debug:nn{module}{Nested~Module~URI:~\l_stex_current_module_str}
2331    \seq_put_right:No \l_stex_all_modules_seq \l_stex_current_module_str
2332    \stex_metagroup_new:o \l_stex_current_module_str
2333 }
```

185

```
2334
2335
2336 \cs_new_protected:Nn \__stex_module_setup_get_uri_str:n {
2337   \str_clear:N \l__stex_module_setup_ns_str
2338   \stex_map_uri:Nnnnn \l_stex_current_ns_uri {
2339     \str_set:Nx \l__stex_module_setup_ns_str{##1\c_colon_str/}
2340   }{
2341     \seq_set_split:Nnn \l__stex_module_setup_seq / {##1}
2342     \seq_pop_right:NN \l__stex_module_setup_seq \l__stex_module_setup_seg
2343     \exp_args:No \str_if_eq:nnF \l__stex_module_setup_seg {#1} {
2344       \seq_put_right:No \l__stex_module_setup_seq \l__stex_module_setup_seg
2345     }
2346     \tl_put_right:Nx \l__stex_module_setup_ns_str {\seq_use:Nn \l__stex_module_setup_seq /}
2347   }{}{}
2348 }
2349
2350 \cs_new_protected:Npn \__stex_module_setup_split_module:n #1?#2 \__stex_module_setup_end: #3
2351   \_stex_module_setup_top_nosig:n { #1 ? #2 / #3}
2352 }
2353
2354 \bool_new:N \l_stex_in_meta_bool
2355 \bool_set_false:N \l_stex_in_meta_bool
2356
2357 \cs_new_protected:Nn \__stex_module_setup_load_meta: {
2358   \tl_if_empty:NF \l_stex_metatheory_uri {
2359     \stex_execute_in_module:x{
2360       \stex_pseudogroup_with:nn{\l_stex_in_meta_bool}{
2361         \stex_activate_module:n {\stex_uri_use:N \l_stex_metatheory_uri }
2362       }
2363     }
2364   }
2365 }
2366
2367 ⟨@@=stex_modules⟩
```

(*End definition for* `\stex_module_setup:n`. *This function is documented on page 111.*)

`\stex_close_module:`

```
2368 \cs_new:Nn \stex_close_module: {
2369   \bool_if:NT \c_stex_persist_write_mode_bool \__stex_modules_persist_module:
2370   \stex_debug:nn{module}{
2371     Closing~module~\l_stex_current_module_str^^J
2372     Code:~\expandafter\meaning\csname c_stex_module_\l_stex_current_module_str _code\endcsname
2373     Imports:\exp_after:wN \prop_to_keyval:N \cs:w c_stex_module_\l_stex_current_module_str _
2374     Declarations:\exp_after:wN \prop_to_keyval:N \cs:w c_stex_module_\l_stex_current_module_
2375     Notations:\exp_after:wN \prop_to_keyval:N \cs:w c_stex_module_\l_stex_current_module_str
2376   }
2377 }
2378
2379 \cs_new_protected:Nn \__stex_modules_persist_module: {
2380   \stex_persist:e {
2381     \__stex_modules_restore_module:nnnn {\l_stex_current_module_str}{
2382       \exp_after:wN \prop_to_keyval:N \cs:w
2383         c_stex_module_\l_stex_current_module_str _morphisms_prop
```

```
2384        \cs_end:
2385      }{
2386        \exp_after:wN \prop_to_keyval:N \cs:w
2387          c_stex_module_\l_stex_current_module_str _symbols_prop
2388        \cs_end:
2389      }{
2390        \exp_after:wN \exp_after:wN \exp_after:wN \exp_not:n
2391        \exp_after:wN \exp_after:wN \exp_after:wN
2392        { \cs:w c_stex_module_\l_stex_current_module_str _code \cs_end: }
2393      }{}
2394      \prop_map_function:cN{c_stex_module_\l_stex_current_module_str _notations_prop}
2395        \__stex_modules_persist_nots_i:nn
2396      \exp_not:N \STEXRestoreNotsEnd {}
2397    }
2398 }
2399
2400 \cs_new_protected:Nn \__stex_modules_restore_module:nnnn {
2401    \prop_gset_from_keyval:cn{c_stex_module_\tl_to_str:n{#1}_morphisms_prop}{#2}
2402    \cs_set:Npn \__stex_modules_tl {#3}
2403    \exp_args:Nno \prop_gset_from_keyval:cn{c_stex_module_\tl_to_str:n{#1}_symbols_prop}\__ste
2404    \prop_map_inline:cn{c_stex_module_\tl_to_str:n{#1}_symbols_prop}{
2405      \stex_ref_new_symbol:n{#1?##1}
2406    }
2407    \cs_gset:cpn{c_stex_module_\tl_to_str:n{#1}_code}{#4}
2408    \prop_gclear:c{c_stex_module_\tl_to_str:n{#1} _notations_prop}
2409    \str_set:Nn \l__stex_modules_restore_mod_str {#1}
2410    \group_begin:
2411      \catcode`_=8\relax
2412      \catcode`:=12\relax
2413      \__stex_modules_restore_nots:n
2414 }
2415
2416 \cs_new:Nn \__stex_modules_persist_nots_i:nn {
2417    \exp_not:n{#2}
2418 }
2419
2420 \quark_new:N \STEXRestoreNotsEnd
2421
2422 \cs_new_protected:Nn \__stex_modules_restore_nots:n {
2423    \__stex_modules_restore_nots_i:n
2424 }
2425
2426 \cs_new_protected:Nn \__stex_modules_restore_nots_i:n {
2427    \tl_if_eq:nnTF{#1}{\STEXRestoreNotsEnd}{
2428      \group_end:
2429    }{
2430      \__stex_modules_restore_nots_ii:nnnnn {#1}
2431    }
2432 }
2433
2434 \cs_new_protected:Nn \__stex_modules_restore_nots_ii:nnnnn {
2435    \cs_set:Npn \l__stex_modules_tl {{#4}{#5}}
2436    \exp_args:NNe\use:nn\prop_gput:cnn{
2437      {c_stex_module_\l__stex_modules_restore_mod_str _notations_prop}
```

187

```
2438        {\tl_to_str:n{#1!#2}}{
2439          {\tl_to_str:n{#1}}{\tl_to_str:n{#2}}{#3}
2440          \exp_args:No \exp_not:n \l__stex_modules_tl
2441        }
2442      }
2443    \__stex_modules_restore_nots_i:n
2444  }
```

(*End definition for* `\stex_close_module:`*. This function is documented on page* *111.*)

`\l_stex_metatheory_uri`

```
2445  \tl_new:N \l_stex_metatheory_uri
```

(*End definition for* `\l_stex_metatheory_uri`*. This variable is documented on page* **??**.)

`\setmetatheory`

```
2446  \cs_new_protected:Nn \__stex_modules_set_metatheory:nn {
2447    \group_begin:
2448      \stex_debug:nn{metatheory}{Setting~metatheory~[#1]#2}
2449      \stex_import_module_uri:nn { #1 } { #2 }
2450      \stex_debug:nn{metatheory}{Here:^^J
2451        \l_stex_import_archive_str^^J
2452        \l_stex_import_path_str^^J
2453        \l_stex_import_name_str^^J
2454      }
2455      \stex_import_require_module:ooo
2456        \l_stex_import_archive_str
2457        \l_stex_import_path_str
2458        \l_stex_import_name_str
2459      \stex_debug:nn{metatheory}{Found:~\l_stex_import_ns_str}
2460      \exp_args:Nne \use:nn {
2461        \group_end: \stex_uri_resolve:Nn \l_stex_metatheory_uri
2462      }{{\l_stex_import_ns_str}}
2463  }
2464
2465  \NewDocumentCommand \setmetatheory {O{} m}{
2466    \__stex_modules_set_metatheory:nn { #1 }{ #2 }
2467    \stex_smsmode_do:
2468  }
2469  \stex_sms_allow_escape:N \setmetatheory
```

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

Keys and key handling:

```
2470  \stex_keys_define:nnnn{smodule}{
2471    \str_clear:N \l_stex_key_sig_str
2472  }{
2473    meta            .code:n        = {
2474      \str_if_empty:nTF {#1}{
2475        \tl_clear:N \l_stex_metatheory_uri
2476      }{
2477        \stex_uri_resolve:Nx \l_stex_metatheory_uri { #1 }
2478      }
2479    },
2480    ns              .code:n        = {
```

188

```
2481        \stex_uri_resolve:Nx \l_stex_current_ns_uri { #1 }
2482     } ,
2483     lang           .code:n        = {
2484        \stex_set_language:n { #1 }
2485     } ,
2486     sig            .str_set_x:N  = \l_stex_key_sig_str ,
2487     creators       .code:n        = {} , % todo ?
2488     contributors   .code:n        = {} , % todo ?
2489     srccite        .code:n        = {}   % todo ?
2490  }{id, title, style, deprecate}
```

smodule *(env.)*

```
2491  \stex_new_stylable_env:nnnnnnn {module} {O{} m}{
2492     \stex_keys_set:nn { smodule }{ #1 }
2493     \tl_set_eq:NN \thistitle \l_stex_key_title_tl
2494     \tl_if_empty:NF \thistitle {
2495        \exp_args:No \stexdoctitle \thistitle
2496     }
2497     \exp_args:Nx \stex_module_setup:n { \tl_to_str:n{ #2 } }
2498
2499     \stex_if_do_html:T {
2500        \exp_args:Nne \begin{stex_annotate_env} {
2501          shtml:theory={\l_stex_current_module_str},
2502          shtml:language={ \l_stex_current_language_str},
2503          shtml:signature={\l_stex_key_sig_str}
2504          \tl_if_empty:NF \l_stex_metatheory_uri {,
2505             shtml:metatheory={\stex_uri_use:N \l_stex_metatheory_uri}
2506          }
2507        }
2508        \stex_annotate_invisible:n{}
2509     }
2510     \stex_if_smsmode:F {
2511        \str_set_eq:NN \thismoduleuri \l_stex_current_module_str
2512        \tl_set:Nn \thismodulename {#2}
2513        \stex_style_apply:
2514     }
2515     \stex_smsmode_do:
2516  }{
2517     \stex_close_module:
2518     \stex_if_smsmode:F \stex_style_apply:
2519     \stex_if_do_html:T{ \end{stex_annotate_env} }
2520  }{}{}{s}
2521
2522  \stex_sms_allow_env:n{smodule}
```

\stex_if_in_module_p:
\stex_if_in_module:*TF*   Are we currently in a module?

```
2523  \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
2524     \str_if_empty:NTF \l_stex_current_module_str
2525        \prg_return_false: \prg_return_true:
2526  }
```

(End definition for \stex_if_in_module:TF. *This function is documented on page 111.)*

\stex_if_module_exists_p:n   Does a module with this URI exist?
\stex_if_module_exists:n*TF*

189

```
2527 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
2528   \tl_if_exist:cTF { c_stex_module_#1_code }
2529     \prg_return_true: \prg_return_false:
2530 }
```

(*End definition for* `\stex_if_module_exists:nTF`*. This function is documented on page 111.*)

`\stex_do_up_to_module:n`
`\stex_do_up_to_module:x`

Execute code in the current module (i.e. as if the `\begin{⟨smodule⟩}` was the current tex group)

```
2531 \cs_new_protected:Nn \stex_do_up_to_module:n {
2532   \exp_args:No \stex_metagroup_do_in:nn \l_stex_current_module_str {#1}
2533 }
2534 \cs_generate_variant:Nn \stex_do_up_to_module:n {x}
```

(*End definition for* `\stex_do_up_to_module:n`*. This function is documented on page 112.*)

`\stex_module_add_code:n`
`\stex_module_add_code:x`

```
2535 \cs_new_protected:Nn \stex_module_add_code:n {
2536   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
2537 }
2538 \cs_generate_variant:Nn \stex_module_add_code:n {x}
```

(*End definition for* `\stex_module_add_code:n`*. This function is documented on page 112.*)

`\stex_execute_in_module:n`
`\stex_execute_in_module:x`

```
2539 \cs_new_protected:Nn \stex_execute_in_module:n { \stex_if_in_module:TF {
2540   \stex_module_add_code:n { #1 }
2541   \stex_do_up_to_module:n { #1 }
2542 }{ #1 }}
2543 \cs_generate_variant:Nn \stex_execute_in_module:n {x}
```

(*End definition for* `\stex_execute_in_module:n`*. This function is documented on page 111.*)

`\STEXexport`

```
2544 \NewDocumentCommand \STEXexport {} {
2545   \ExplSyntaxOn
2546   \__stex_modules_export:n
2547 }
2548 \cs_new_protected:Nn \__stex_modules_export:n {
2549   \stex_ignore_spaces_and_pars:#1\ExplSyntaxOff
2550   \stex_module_add_code:n { \stex_ignore_spaces_and_pars:#1}
2551   \stex_smsmode_do:
2552 }
```

Only allowed in modules, and allowed (escaped) in sms mode:

```
2553 \stex_deactivate_macro:Nn \STEXexport {module~environments}
2554 \stex_sms_allow_escape:N \STEXexport
2555 \stex_every_module:n {\stex_reactivate_macro:N \STEXexport}
```

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

```
2556 \cs_new_protected:Nn \stex_module_add_morphism:nnnn {
2557   \exp_args:Nne \prop_gput:cnn{c_stex_module_\l_stex_current_module_str _morphisms_prop}{
2558     \tl_if_empty:nTF{#1}{[#2]}{#1}
2559   }{{#1}{#2}{#3}{#4}}
2560 }
2561 \cs_generate_variant:Nn \stex_module_add_morphism:nnnn {nonn,ooox}
```

(*End definition for* \stex_module_add_morphism:nnnn. *This function is documented on page 119.*)

#1 : {⟨*Macro name*⟩}
#2 : {⟨*Name*⟩}
#3 : {⟨*arity*⟩}
#4 : {({⟨*Arg num*⟩}{⟨*Arg str*⟩})*}
#5 : Definiens
#6 : type
#7 : Return
#8 : Command

```
2562 \cs_new_protected:Nn \stex_module_add_symbol:nnnnnnnN {
2563   \stex_debug:nn{declaration}{New~declaration:~\l_stex_current_module_str?#2^^J
2564     Macro:#1^^JArity:#3~(#4)^^J
2565     Def:~\tl_to_str:n{#5}^^J
2566     Type:~\tl_to_str:n{#6}^^J
2567     Returns:~\tl_to_str:n{#7}
2568   }
2569   %\prop_gput:cnx{c_stex_module_\l_stex_current_module_str _symbols_prop}
2570   %{#2}{\exp_not:n{{#1}{#2}{#3}{#4}{#5}{#6}}{#7}\exp_not:n{#8}}
2571   \prop_gput:cnn{c_stex_module_\l_stex_current_module_str _symbols_prop}
2572   {#2}{{#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}}
2573   \tl_if_empty:nF{#1}{
2574     \stex_execute_in_module:n {
2575       \__stex_modules_activate_sym:n {#2}
2576     }
2577   }
2578 }
2579
2580 \cs_new_protected:Nn \__stex_modules_activate_sym:n {
2581   \prop_map_inline:cn{c_stex_module_\l_stex_current_module_str _symbols_prop}{
2582     \str_if_eq:nnT{#1}{##1}{
2583       \__stex_modules_activate_i:nnnnnnnn ##2
2584     }
2585   }
2586 }
2587 \cs_new_protected:Nn \__stex_modules_activate_i:nnnnnnnn {
2588   \stex_debug:nn{activating}{#1:\l_stex_current_module_str^^J
2589     \tl_to_str:n{{#2}{#3}{#4}{#5}{#6}{#7}#8}
2590   }
2591   \cs_set:cpx{#1} {
2592     \_stex_invoke_symbol:nnnnnnnN
2593     {\l_stex_current_module_str}
2594     \exp_not:n{{#2}{#3}{#4}{#5}{#6}{#7}{#8}}
2595   }
2596   \stex_debug:nn{activating}{done
```

191

```
2597        \prop_map_break:
2598    }
```

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

#1 : URI
#2 : variant
#3 : arity
#4 : macro body
#5 : op

```
2599  \cs_new_protected:Nn \stex_module_add_notation:nnnnn {
2600    \stex_debug:nn{notations}{Adding~notation:^^J
2601      #1~\c_hash_str#2~#3^^J
2602      to~\l_stex_current_module_str
2603    }
2604    \prop_gput:cnn{c_stex_module_\l_stex_current_module_str _notations_prop}
2605    {#1!#2}{{#1}{#2}{#3}{#4}{#5}}
2606    \stex_execute_in_module:n {
2607      \__stex_modules_activate_not:nn{#1}{#2}
2608    }
2609  }
2610  \cs_generate_variant:Nn \stex_module_add_notation:nnnnn {eoeoo}
2611
2612
2613  \cs_new_protected:Nn \__stex_modules_activate_not:nn {
2614    \prop_map_inline:cn{c_stex_module_\l_stex_current_module_str _notations_prop}{
2615      \str_if_eq:nnT{#1!#2}{##1}{
2616        \prop_map_break:n{\stex_set_notation_macro:nnnnn ##2 }
2617      }
2618    }
2619  }
```

(*End definition for* \stex_module_add_notation:nnnnn *and* \stex_set_notation_macro:nnnnn. *These functions are documented on page* *114*.)

```
2620  \cs_new_protected:Nn \stex_set_notation_macro:nnnnn {
2621    \tl_set:cn {l_stex_notation_#1_#2_cs}{#4}
2622    \cs_if_exist:cF{l_stex_notation_#1__cs}{
2623      \tl_set:cn {l_stex_notation_#1__cs}{#4}
2624    }
2625    \tl_if_empty:nF{#5}{
2626      \tl_set:cn{l_stex_notation_#1_op_#2_cs}{#5}
2627      \cs_if_exist:cF{l_stex_notation_#1_op__cs}{
2628        \cs_set_eq:cc {l_stex_notation_#1_op__cs}{l_stex_notation_#1_op_#2_cs}
2629      }
2630    }
2631  }
2632  \cs_generate_variant:Nn \stex_set_notation_macro:nnnnn {eoexo}
```

(*End definition for* \stex_set_notation_macro:nnnnn. *This function is documented on page* *115*.)

```
2633  \cs_new_protected:Nn \stex_activate_module:n {
```

```
2634    \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
2635      \stex_debug:nn{modules}{Activating~module~#1^^J\expandafter\meaning\csname c_stex_module
2636      \seq_put_right:Nn \l_stex_all_modules_seq { #1 }
2637      \stex_pseudogroup:nn{
2638        \str_set:Nn \l_stex_current_module_str {#1}
2639        \use:c{ c_stex_module_#1_code }
2640      }{
2641        \stex_pseudogroup_restore:N \l_stex_current_module_str
2642      }
2643    }
2644  }
2645  \cs_generate_variant:Nn \stex_activate_module:n {o,x}
```

(*End definition for* `\stex_activate_module:n`*. This function is documented on page 111.*)

Iterating:

```
2646  ⟨@@=stex_iterate⟩
```

`\stex_iterate_symbols:n`

```
2647  \cs_new_protected:Nn \stex_iterate_symbols:n {
2648    \stex_pseudogroup_with:nn{\__stex_iterate_sym_cs:nnnnnnnnN\stex_iterate_break:\stex_iterat
2649      \cs_set:Npn \__stex_iterate_sym_cs:nnnnnnnnN
2650      ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 ##9 { #1 }
2651      \cs_set:Npn \stex_iterate_break: {
2652        \prop_map_break:n{\seq_map_break:}
2653      }
2654      \cs_set:Npn \stex_iterate_break:n ##1 {
2655        \prop_map_break:n{\seq_map_break:n{##1}}
2656      }
2657      \seq_map_inline:Nn \l_stex_all_modules_seq {
2658        \prop_map_inline:cn{c_stex_module_##1_symbols_prop}{
2659          \__stex_iterate_sym_cs:nnnnnnnnN {##1} ####2
2660        }
2661      }
2662    }
2663  }
```

(*End definition for* `\stex_iterate_symbols:n`*. This function is documented on page 112.*)

`\stex_iterate_symbols:nn`

```
2664  \cs_new_protected:Nn \stex_iterate_symbols:nn {
2665    \seq_clear:N \l__stex_iterate_mods_seq
2666    \stex_pseudogroup_with:nn{\__stex_iterate_sym_cs:nnnnnnnnN}{
2667      \cs_set:Npn \__stex_iterate_sym_cs:nnnnnnnnN
2668      ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 ##9 { #2 }
2669      \clist_map_function:nN {#1} \__stex_iterate_it_decl_i:n
2670    }
2671  }
2672
2673  \cs_new_protected:Nn \__stex_iterate_it_decl_i:n {
2674    \seq_if_in:NnF \l__stex_iterate_mods_seq {#1} {
2675      \seq_put_left:Nn \l__stex_iterate_mods_seq {#1}
2676      \prop_map_inline:cn{c_stex_module_#1_morphisms_prop}{
2677        \__stex_iterate_it_decl_check:nnnn ##2
2678      }
```

```
2679        \prop_map_inline:cn{c_stex_module_#1_symbols_prop}{
2680            \__stex_iterate_sym_cs:nnnnnnnnN {#1} ##2
2681        }
2682    }
2683 }
2684 \cs_new_protected:Nn \__stex_iterate_it_decl_check:nnnn {
2685    \tl_if_empty:nT{#1}{
2686        \__stex_iterate_it_decl_i:n {#2}
2687    }
2688 }
```

(*End definition for* \stex_iterate_symbols:nn. *This function is documented on page 112.*)

```
2689 \cs_new_protected:Nn \stex_iterate_notations:nn {
2690    \seq_clear:N \l__stex_iterate_mods_seq
2691    \stex_pseudogroup_with:nn{\__stex_iterate_not_cs:nnnnn}{
2692        \cs_set:Npn \__stex_iterate_not_cs:nnnnn
2693        ##1 ##2 ##3 ##4 ##5 { #2 }
2694        \clist_map_function:nN {#1} \__stex_iterate_it_not_i:n
2695    }
2696 }
2697
2698 \cs_new_protected:Nn \__stex_iterate_it_not_i:n {
2699    \seq_if_in:NnF \l__stex_iterate_mods_seq {#1} {
2700        \seq_put_left:Nn \l__stex_iterate_mods_seq {#1}
2701        \prop_map_inline:cn{c_stex_module_#1_notations_prop}{
2702            \__stex_iterate_not_cs:nnnnn ##2
2703        }
2704        \prop_map_inline:cn{c_stex_module_#1_morphisms_prop}{
2705            \__stex_iterate_it_not_check:nnnn ##2
2706        }
2707    }
2708 }
2709 \cs_new_protected:Nn \__stex_iterate_it_not_check:nnnn {
2710    \tl_if_empty:nT{#1}{
2711        \__stex_iterate_it_not_i:n {#2}
2712    }
2713 }
```

(*End definition for* \stex_iterate_notations:nn. *This function is documented on page 115.*)

```
2714 \cs_new_protected:Nn \stex_iterate_morphisms:nn {
2715    \seq_clear:N \l__stex_iterate_mods_seq
2716    \bool_set_true:N \l__stex_iterate_continue_bool
2717    \cs_set:Npn \__stex_iterate_morphism_cs:nnnn ##1 ##2 ##3 ##4 ##5 {
2718        #2
2719        \bool_if:NT \l__stex_iterate_continue_bool {
2720            \str_if_eq:nnTF{##1}{[##2]}{
2721                \tl_put_right:Nn \l__stex_iterate_todo_tl {
2722                    \__stex_iterate_iterate_morphism:nn{##5}{##2}
2723                }
2724            }{
2725                \tl_put_right:Nn \l__stex_iterate_todo_tl {
```

194

```
2726            \__stex_iterate_iterate_morphism:nn{##5 / ##1}{##2}
2727          }
2728        }
2729      }
2730    }
2731    \cs_set:Npn \stex_iterate_break:n ##1 {
2732      \bool_set_false:N \l__stex_iterate_continue_bool
2733      \prop_map_break:n{##1}
2734    }
2735    \__stex_iterate_iterate_morphism:nn{}{#1}
2736  }
2737
2738  \cs_new_protected:Nn \__stex_iterate_iterate_morphism:nn {
2739    \tl_clear:N \l__stex_iterate_todo_tl
2740    \seq_if_in:NnF \l__stex_iterate_mods_seq {#1 #2}{
2741      \seq_put_right:Nn \l__stex_iterate_mods_seq {#1 #2}
2742      \prop_map_inline:cn{c_stex_module_#2_morphisms_prop}{
2743        \__stex_iterate_morphism_cs:nnnn ##2 {#1}
2744        % TODO
2745        % ##1: name or [mpath]
2746        % ##2 = {####1}{####2}{####3}{####4}
2747        % ####1 = name
2748        % ####2 = mpath
2749        % ####3 = type
2750        % ####4 = {origname}{newname}*
2751      }
2752      \bool_if:NT \l__stex_iterate_continue_bool \l__stex_iterate_todo_tl
2753    }
2754  }
```

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

13.6.2 Structural Features

```
2755  ⟨@@=stex_features⟩
```

```
2756  \cs_new_protected:Nn \stex_structural_feature_module:nn {
2757    \stex_if_do_html:TF {
2758      \exp_args:Nne \begin{stex_annotate_env} {
2759        shtml:feature-#2={
2760          \l_stex_current_module_str/#1}
2761        \str_if_empty:NF \l_stex_macroname_str {,
2762          shtml:macroname={\l_stex_macroname_str}
2763        }
2764      }
2765      \stex_annotate_invisible:n{}
2766    }\group_begin:
2767    \stex_module_setup:n {#1-module}
2768  }
2769
2770  \cs_new_protected:Nn \stex_structural_feature_module_end: {
2771    \tl_gset_eq:NN \g_stex_last_feature_str \l_stex_current_module_str
2772    \stex_close_module:
2773    \stex_if_do_html:TF{
```

```
2774        \end{stex_annotate_env}
2775    }\group_end:
2776 }
```

(*End definition for* \stex_structural_feature_module:nn *and* \stex_structural_feature_module_-
end:. *These functions are documented on page* *117.*)

\stex_structural_feature_morphism:nnn
\stex_structural_feature_morphism_end:

```
2777 \bool_new:N \l__stex_features_implicit_bool
2778 \cs_new_protected:Nn \stex_structural_feature_morphism:nnnnn {
2779    \str_clear:N \l_stex_current_domain_str
2780    \tl_if_empty:nT{#3}{
2781      \_stex_get_mathstructure:n{#4}
2782      \str_set_eq:NN \l_stex_current_domain_str \l_stex_get_structure_module_str
2783    }
2784    \str_if_empty:NT \l_stex_current_domain_str {
2785      \stex_import_module_uri:nn { #3 }{ #4 }
2786      \group_begin:
2787      \stex_import_require_module:ooo
2788        \l_stex_import_archive_str
2789        \l_stex_import_path_str
2790        \l_stex_import_name_str
2791      \exp_args:Nnx \use:nn \group_end: {
2792        \str_set:Nn \exp_not:N\l_stex_current_domain_str {\l_stex_import_ns_str}
2793      }
2794    }
2795    \tl_if_empty:nTF{#1}{
2796      \bool_set_true:N \l__stex_features_implicit_bool
2797      \str_set:Nx \l_tmpa_str {[\l_stex_current_domain_str]}
2798    }{
2799      \bool_set_false:N \l__stex_features_implicit_bool
2800      \str_set:Nn \l_tmpa_str {#1}
2801    }
2802
2803    \stex_if_do_html:TF {
2804      \begin{stex_annotate_env} {
2805        shtml:feature-#2={\l_stex_current_module_str?\l_tmpa_str},
2806        shtml:domain={\l_stex_current_domain_str}
2807        #5
2808      }
2809      \stex_annotate_invisible:n{}
2810    }\group_begin:
2811    \str_set:Nn \l__stex_features_feature_str {#2}
2812    \str_set_eq:NN \l_stex_feature_name_str \l_tmpa_str
2813    \__stex_features_setup:
2814    \__stex_features_reactivate:
2815    %^^A\stex_activate_module:o \l_stex_current_domain_str
2816    \exp_args:Ne \stex_metagroup_new:n {\l_stex_current_module_str / \l_stex_feature_name_str}
2817 }
2818
2819 \cs_new_protected:Nn \__stex_features_do_for_list: {
2820    \seq_clear:N \l_stex_fors_seq
2821    \clist_map_inline:Nn \l_stex_key_for_clist {
2822      \exp_args:Ne\stex_get_in_morphism:n{\tl_to_str:n{##1}}
```

196

```
2823    \seq_put_right:Nx \l_stex_fors_seq
2824      {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
2825  }
2826 }
2827
2828 \cs_new_protected:Nn \__stex_features_add_definiens:nn {
2829   \__stex_features_set_definiens_macros: #1\__stex_features_break:
2830   \_stex_assign_do:n{#2}
2831   #2
2832 }
2833 \cs_new_protected:Npn \__stex_features_set_definiens_macros: #1?#2?#3\__stex_features_break:
2834   \str_set:Nn \l_stex_get_symbol_mod_str {#1?#2}
2835   \str_set:Nn \l_stex_get_symbol_name_str {#3}
2836   \exp_args:Nne\use:nn{\__stex_features_set_definiens_macros_i:nnnnnnn}{
2837     \prop_item:Nn \l_stex_morphism_symbols_prop {[#1?#2]/[#3]}
2838   }
2839 }
2840 \cs_new_protected:Nn \__stex_features_set_definiens_macros_i:nnnnnnn {
2841   \tl_set:Nn \l_stex_get_symbol_def_tl{#4}
2842 }
2843
2844 \cs_new_protected:Nn \stex_structural_feature_morphism_end: {
2845   \str_gset_eq:NN \l_stex_feature_name_str \l_stex_feature_name_str
2846   \str_gset_eq:NN \l_stex_current_domain_str \l_stex_current_domain_str
2847   \seq_gset_eq:NN \l_stex_morphism_symbols_prop \l_stex_morphism_symbols_prop
2848   \seq_gset_eq:NN \l_stex_morphism_renames_prop \l_stex_morphism_renames_prop
2849   \seq_gset_eq:NN \l_stex_morphism_morphisms_seq \l_stex_morphism_morphisms_seq
2850   \__stex_features_do_elaboration:
2851   \stex_if_do_html:TF{
2852     \end{stex_annotate_env}
2853   }\group_end:
2854 }
2855
2856 \cs_new_protected:Nn \__stex_features_setup: {
2857   \prop_clear:N \l_stex_morphism_symbols_prop
2858   \prop_clear:N \l_stex_morphism_renames_prop
2859   \seq_clear:N \l_stex_morphism_morphisms_seq
2860   \__stex_features_do_decls:
2861   \exp_args:No \__stex_features_do_morphisms:n \l_stex_current_domain_str
2862 }
2863
2864 \cs_new_protected:Nn \__stex_features_rename_all: {
2865
2866 }
2867
2868 \cs_new:Npn \__stex_features_clean:nnw [#1] / #2 \_stex_end: {
2869   [#1]/[#2]
2870 }
2871
2872 \cs_new_protected:Nn \__stex_features_do_decls: {
2873   \exp_args:No \stex_iterate_symbols:nn \l_stex_current_domain_str {
2874     \stex_str_if_starts_with:nnTF{##3}[{
2875       \exp_args:NNe \prop_put:Nnn \l_stex_morphism_symbols_prop {
2876         \__stex_features_clean:nnw ##3 \_stex_end:
```

197

```
2877          }
2878       }{
2879       \prop_put:Nnn \l_stex_morphism_symbols_prop
2880         {[##1]/[##3]}
2881       }{
2882         {##2}{##4}{##5}{##6}{##7}{##8}##9
2883       }
2884     }
2885 }
2886
2887 \cs_new_protected:Nn \stex_structural_feature_morphism_check_total: {
2888   \prop_map_inline:Nn \l_stex_morphism_symbols_prop {
2889     \__stex_features_total_check: ##1 ##2
2890   }
2891 }
2892
2893 \cs_new_protected:Npn \__stex_features_total_check: [#1]/[#2] #3 #4 #5 #6 #7 #8 #9 {
2894   \tl_if_empty:nT{#6}{
2895     \msg_error:nnxx{stex}{error/needsdefiniens}{#1?#2}{total~morphism}
2896   }
2897 }
2898
2899 \cs_new:Npn \__stex_features_split_qm:w #1 ? #2 ? #3 { #3 }
2900 \cs_new_protected:Nn \__stex_features_do_elaboration: {
2901   \stex_debug:nn{morphisms}{
2902     Elaborating:^^J\prop_to_keyval:N \l_stex_morphism_symbols_prop
2903     ^^J
2904     Renamings:^^J
2905     \prop_to_keyval:N \l_stex_morphism_renames_prop
2906   }
2907   \prop_map_inline:Nn \l_stex_morphism_symbols_prop {
2908     \__stex_features_elab_check: ##1 ##2
2909   }
2910   \exp_args:No\stex_iterate_notations:nn\l_stex_current_domain_str{
2911     \prop_get:NnNTF \l_stex_morphism_renames_prop {##1}\l__stex_features_tmp {
2912       \exp_args:Ne \stex_module_add_notation:nnnnn
2913       {\l_stex_current_module_str ? \exp_after:wN \use_ii:nn \l__stex_features_tmp}
2914     }{
2915       \exp_args:Ne \stex_module_add_notation:nnnnn
2916       {\l_stex_current_module_str ? \l_stex_feature_name_str
2917       / \__stex_features_split_qm:w ##1}
2918     }{##2}{##3}{##4}{##5}
2919   }
2920   \stex_module_add_morphism:ooox
2921     \l_stex_feature_name_str
2922     \l_stex_current_domain_str
2923     \l__stex_features_feature_str
2924     {\prop_map_function:NN \l_stex_morphism_renames_prop \__stex_features_rename:nn}
2925 }
2926
2927 \cs_new:Nn \__stex_features_rename:nn{
2928   {#1}{\use_ii:nn#2}
2929 }
2930
```

198

```
2931  \cs_new_protected:Npn \__stex_features_elab_check: [#1]/[#2] #3 {
2932    \prop_get:NnNTF \l_stex_morphism_renames_prop {#1?#2} \l__stex_features_tmp {
2933      \stex_debug:nn{morphisms}{Generating~\l__stex_features_tmp}
2934      \exp_after:wN \stex_module_add_symbol:nnnnnnN \l__stex_features_tmp
2935    }{
2936      \bool_if:NTF \l__stex_features_implicit_bool {
2937        \stex_debug:nn{morphisms}{Generating~#3:~\l_stex_feature_name_str / #2}
2938        \exp_args:Nno \stex_module_add_symbol:nnnnnnN {#3}{\l_stex_feature_name_str / #2}
2939      }{
2940        \stex_debug:nn{morphisms}{Generating~\l_stex_feature_name_str / #2}
2941        \exp_args:Nno \stex_module_add_symbol:nnnnnnN {}{\l_stex_feature_name_str / #2}
2942      }
2943    }
2944  }
2945
2946  \cs_new_protected:Nn \__stex_features_do_morphisms:n {
2947      \prop_map_inline:cn {c_stex_module_#1_morphisms_prop}{
2948        \__stex_features_do_morph:nnnn ##2
2949      }
2950  }
2951
2952  \cs_new_protected:Nn \__stex_features_do_morph:nnnn {
2953      \tl_if_empty:nF{#3}{
2954        \seq_put_right:Nn \l_stex_morphism_morphisms_seq {{#1}{#2}{#3}}
2955      }
2956      \__stex_features_do_morphisms:n{#2}
2957  }
2958
2959  \cs_new_protected:Npn \__stex_features_reactivate: {
2960    \stex_deactivate_macro:Nn \symdecl {module~environments}
2961    \stex_deactivate_macro:Nn \textsymdecl {module~environments}
2962    \stex_deactivate_macro:Nn \symdef {module~environments}
2963    \stex_deactivate_macro:Nn \notation {module~environments}
2964    \stex_deactivate_macro:Nn \importmodule {module~environments}
2965    \stex_deactivate_macro:Nn \requiremodule {module~environments}
2966    \stex_deactivate_macro:Nn \smodule {outside~of~morphisms}
2967    \stex_reactivate_macro:N \assign
2968    \stex_reactivate_macro:N \assignMorphism
2969    \stex_reactivate_macro:N \renamedecl
2970    \cs_set_eq:NN \_stex_do_for_list: \__stex_features_do_for_list:
2971    \cs_set_eq:NN \_stex_add_definiens:nn \__stex_features_add_definiens:nn
2972  }
```

(*End definition for* \stex_structural_feature_morphism:nnn *and* \stex_structural_feature_morphism_-
end:. *These functions are documented on page* **??**.)

\stex_get_in_morphism:n

```
2973  \cs_new_protected:Nn \stex_get_in_morphism:n {
2974    \str_clear:N \l_stex_get_symbol_name_str
2975    \prop_map_inline:Nn \l_stex_morphism_symbols_prop {
2976      \exp_args:Nx\__stex_features_get_check:nnnn{\tl_to_str:n{#1}}##1##2
2977    }
2978    \str_if_empty:NT \l_stex_get_symbol_name_str {
2979      \prop_map_inline:Nn \l_stex_morphism_renames_prop {
```

```
2980        \__stex_features_renamed_check:nnnnn{#1}##1=##2
2981      }
2982      \str_if_empty:NT \l_stex_get_symbol_name_str {
2983        \msg_error:nnxx{stex}{error/unknownsymbolin}{#1}{
2984          morphism~\l_stex_feature_name_str
2985        }
2986      }
2987    }
2988  }

2990  \cs_new_protected:Npn \__stex_features_renamed_check:nnnnn #1#2?#3?#4=#5#6 {
2991    \str_if_eq:nnTF{#1}{#5}{
2992      \exp_args:Nnx \use:nn{\__stex_features_check_break:nnnnnnnnn{#2?#3}{#4}}{
2993        \prop_item:Nn \l_stex_morphism_symbols_prop {[#2?#3]/[#4]}
2994      }
2995    }{
2996      \str_if_eq:nnT{#1}{#6}{
2997        \exp_args:Nnx \use:nn{\__stex_features_check_break:nnnnnnnnn{#2?#3}{#4}}{
2998          \prop_item:Nn \l_stex_morphism_symbols_prop {[#2?#3]/[#4]}
2999        }
3000      }
3001    }
3002  }

3004  \cs_new_protected:Npn \__stex_features_get_check:nnnn #1[#2]/[#3]#4 {
3005    \str_if_eq:nnTF{#1}{#3}{
3006      \__stex_features_check_break:nnnnnnnnn{#2}{#3}{#4}
3007    }{
3008      \str_if_eq:nnTF{#1}{#4}{
3009        \__stex_features_check_break:nnnnnnnnn{#2}{#3}{#4}
3010      }{
3011        \use_none:nnnnnn
3012      }
3013    }
3014  }

3016  \cs_new_protected:Nn \__stex_features_check_break:nnnnnnnnn {
3017    \prop_map_break:n{
3018      \str_set:Nn \l_stex_get_symbol_mod_str{#1}
3019      \str_set:Nn \l_stex_get_symbol_name_str{#2}
3020      \str_set:Nn \l_stex_get_symbol_macro_str{#3}
3021      \int_set:Nn \l_stex_get_symbol_arity_int {#4}
3022      \tl_set:Nn \l_stex_get_symbol_args_tl {#5}
3023      \tl_set:Nn \l_stex_get_symbol_def_tl {#6}
3024      \tl_set:Nn \l_stex_get_symbol_type_tl {#7}
3025      \tl_set:Nn \l_stex_get_symbol_return_tl {#8}
3026      \tl_set:Nn \l_stex_get_symbol_invoke_cs {#9}
3027    }
3028  }
```

(*End definition for* \stex_get_in_morphism:n. *This function is documented on page* *120.*)

13.7 Inheritance

13.7.1 \importmodule/\usemodule

```
3029 ⟨@@=stex_importmodule⟩
```

\usemodule

```
3030 \stex_new_stylable_cmd:nnnn {usemodule} { O{} m } {
3031   \stex_import_module_uri:nn { #1 }{ #2 }
3032   \stex_import_require_module:ooo
3033     \l_stex_import_archive_str
3034     \l_stex_import_path_str
3035     \l_stex_import_name_str
3036   \stex_if_do_html:T {
3037     \hbox{\stex_annotate_invisible:nn
3038       {shtml:usemodule=\l_stex_import_ns_str} {}}
3039   }
3040   \stex_if_smsmode:F{
3041     \group_begin:
3042     \tl_set_eq:NN \thismoduleuri \l_stex_import_ns_str
3043     \tl_set_eq:NN \thismodulename \l_stex_import_name_str
3044     \tl_clear:N \thisstyle
3045     \stex_style_apply:
3046     \group_end:
3047   }
3048 }{}
```

(*End definition for* \usemodule. *This function is documented on page 82.*)

\stex_import_module_uri:nn

```
3049 \cs_new_protected:Nn \stex_import_module_uri:nn {
3050   \stex_debug:nn{importmodule}{URI:~>#1<~>#2<}
3051   \exp_args:NNnx \seq_set_split:Nnn \__stex_importmodule_seq ? { \tl_to_str:n{ #2 } }
3052   \seq_pop_right:NN \__stex_importmodule_seq \l_stex_import_name_str
3053   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \__stex_importmodule_seq ? }
3054   \tl_if_empty:nTF { #1 } {
3055     \stex_debug:nn{importmodule}{No~archive}
3056     \prop_if_exist:NTF \l_stex_current_archive_prop {
3057       \stex_debug:nn{importmodule}{Picking~current~archive}
3058       \str_set:Nx \l_stex_import_archive_str {
3059         \prop_item:Nn \l_stex_current_archive_prop { id }
3060       }
3061     }{
3062       \str_clear:N \l_stex_import_archive_str
3063       \str_set:Nn \l_stex_import_uri_str {file:}
3064       \str_if_empty:NTF \l_stex_import_path_str {
3065         \stex_debug:nn{importmodule}{Empty~Path}
3066         \stex_file_split_off_ext:NN \l__stex_importmodule_path_seq \g_stex_current_file
3067         \stex_file_split_off_lang:NN \l__stex_importmodule_path_seq \l__stex_importmodule_pa
3068         \str_set:Nx \l_stex_import_path_str {
3069           \stex_file_use:N \l__stex_importmodule_path_seq
3070         }
3071       }{
3072         \stex_debug:nn{importmodule}{Resolving~path~\l_stex_import_path_str~relative~to~\ste
3073         \stex_file_resolve:Nx \l__stex_importmodule_seq { \stex_file_use:N \g_stex_current_f
```

201

```
3074          \str_set:Nx \l_stex_import_path_str {
3075            \stex_file_use:N \l__stex_importmodule_seq
3076          }
3077          \stex_debug:nn{importmodule}{...yields~\l_stex_import_path_str}
3078        }
3079      }
3080    } {
3081      \stex_debug:nn{importmodule}{Archive~#1}
3082      \str_set:Nx \l_stex_import_archive_str { #1 }
3083      \stex_require_archive:o \l_stex_import_archive_str
3084      \str_set:Nx \l_stex_import_uri_str { \prop_item:cn{ c_stex_mathhub_ \l_stex_import_archi
3085    }
3086 }
```

(*End definition for* `\stex_import_module_uri:nn`*. This function is documented on page* *119.*)

`\stex_import_require_module:nnn`
`\stex_import_require_module:ooo`

```
3087 \cs_new_protected:Npn \stex_import_require_module:nnn #1 {
3088    \tl_if_empty:nTF { #1 } {
3089      \str_clear:N \l__stex_importmodule_archive_str
3090      \str_set:Nn \l_stex_import_uri_str {file:}
3091      \__stex_importmodule_get_module:nnn {}
3092    }{
3093      \stex_require_archive:n { #1 }
3094      \str_set:Nx \l__stex_importmodule_archive_str {#1}
3095      \str_set:Nx \l_stex_import_uri_str { \prop_item:cn{ c_stex_mathhub_ #1 _manifest_prop}
3096      \str_set:Nx \l__stex_importmodule_str { \stex_file_use:N \c_stex_mathhub_file / #1 / sou
3097      \exp_args:No \__stex_importmodule_get_module:nnn \l__stex_importmodule_str
3098    }
3099 }
3100 \cs_generate_variant:Nn \stex_import_require_module:nnn {ooo}
3101
3102 \cs_new_protected:Npn \stex_import_require_module_safe:nnn #1 {
3103    \tl_if_empty:nTF { #1 } {
3104      \str_clear:N \l__stex_importmodule_archive_str
3105      \str_set:Nn \l_stex_import_uri_str {file:}
3106      \__stex_importmodule_get_module_safe:nnn {}
3107    }{
3108      \stex_require_archive:n { #1 }
3109      \str_set:Nx \l__stex_importmodule_archive_str {#1}
3110      \str_set:Nx \l_stex_import_uri_str { \prop_item:cn{ c_stex_mathhub_ #1 _manifest_prop}
3111      \str_set:Nx \l__stex_importmodule_str { \stex_file_use:N \c_stex_mathhub_file / #1 / sou
3112      \exp_args:No \__stex_importmodule_get_module_safe:nnn \l__stex_importmodule_str
3113    }
3114 }
3115
3116 \cs_new_protected:Nn \__stex_importmodule_get_module_uri:nnn {
3117    \tl_if_empty:nF {#2}{
3118      \str_set:Nx \l_stex_import_uri_str {\l_stex_import_uri_str / #2}
3119    }
3120    \stex_debug:nn{importmodule}{~>#1<^^J>#2<^^J>#3<^^J>\l_stex_import_uri_str<^^J
3121      Current~file:\stex_file_use:N \g_stex_current_file^^J
3122      Current~namespace:\stex_uri_use:N \l_stex_current_ns_uri
3123    }
```

```
3124    \stex_if_module_exists:nTF {\l_stex_import_uri_str?#3} {
3125      \str_set:Nx \l_stex_import_ns_str {\l_stex_import_uri_str?#3}
3126    }{
3127      \stex_if_module_exists:nTF{\stex_uri_use:N \l_stex_current_ns_uri ? #3}{
3128        \str_set:Nx \l_stex_import_ns_str {\stex_uri_use:N \l_stex_current_ns_uri ? #3}
3129      }{
3130        \__stex_importmodule_get_from_file:nnn{#1}{#2}{#3}
3131        \str_set:Nx \l_stex_import_ns_str {\l_stex_import_uri_str?#3}
3132      }
3133    }
3134 }
3135 \cs_new_protected:Nn \__stex_importmodule_get_module_uri_safe:nnn {
3136    \tl_if_empty:nF {#2}{
3137      \str_set:Nx \l_stex_import_uri_str {\l_stex_import_uri_str / #2}
3138    }
3139    \stex_debug:nn{importmodule}{~>#1<^^J>#2<^^J>#3<^^J>\l_stex_import_uri_str<^^J
3140      Current~file:\stex_file_use:N \g_stex_current_file^^J
3141      Current~namespace:\stex_uri_use:N \l_stex_current_ns_uri
3142    }
3143    \stex_if_module_exists:nTF {\l_stex_import_uri_str?#3} {
3144      \str_set:Nx \l_stex_import_ns_str {\l_stex_import_uri_str?#3}
3145    }{
3146      \stex_if_module_exists:nTF{\stex_uri_use:N \l_stex_current_ns_uri ? #3}{
3147        \str_set:Nx \l_stex_import_ns_str {\stex_uri_use:N \l_stex_current_ns_uri ? #3}
3148      }{
3149        \__stex_importmodule_get_from_file_safe:nnn{#1}{#2}{#3}
3150        \str_set:Nx \l_stex_import_ns_str {\l_stex_import_uri_str?#3}
3151      }
3152    }
3153 }
3154
3155 \cs_new_protected:Nn \__stex_importmodule_get_module:nnn {
3156    \stex_debug:nn{importmodule}{Requiring~[#1]#2?#3<}
3157    \__stex_importmodule_get_module_uri:nnn{#1}{#2}{#3}
3158    \stex_activate_module:o \l_stex_import_ns_str
3159 }
3160
3161 \cs_new_protected:Nn \__stex_importmodule_get_module_safe:nnn {
3162    \stex_debug:nn{importmodule}{Requiring~[#1]#2?#3<}
3163    \__stex_importmodule_get_module_uri_safe:nnn{#1}{#2}{#3}
3164 }
3165
3166 \cs_new_protected:Nn \__stex_importmodule_get_from_file:nnn {
3167    \stex_file_resolve:Nx \l__stex_importmodule_seq { \tl_if_empty:nF{ #1 }{ #1 / } #2 }
3168    \str_set:Nx \l__stex_importmodule_str {\stex_file_use:N \l__stex_importmodule_seq}
3169    \stex_debug:nn{imports}{Looking~for~\l_stex_import_uri_str?#3...}
3170    \__stex_importmodule_check_file:nn{ /#3.tex }{
3171      \__stex_importmodule_check_file:nn{/#3.\l_stex_current_language_str .tex}{
3172        \__stex_importmodule_check_file:nn{/#3.en.tex}{
3173          \__stex_importmodule_check_file:nn{.tex}{
3174            \__stex_importmodule_check_file:nn{.\l_stex_current_language_str .tex}{
3175              \__stex_importmodule_check_file:nn{.en.tex}{
3176                \msg_error:nnx{stex}{error/unknownmodule}{\l_stex_import_uri_str?#3}
3177              }
```

203

```
3178            }
3179          }
3180        }
3181      }
3182    }
3183    \stex_if_smsmode:TF{
3184      \exp_args:NNo \exp_args:Nnx \str_if_eq:nnTF{\l__stex_importmodule_str}{\stex_file_use:N
3185        \stex_debug:nn{imports}{Skipping~current~file}
3186      }{
3187        \__stex_importmodule_load_file:n{#3}
3188      }
3189    }{
3190      \__stex_importmodule_load_file:n{#3}
3191    }
3192 }
3193 \cs_new_protected:Nn \__stex_importmodule_get_from_file_safe:nnn {
3194    \stex_file_resolve:Nx \l__stex_importmodule_seq { \tl_if_empty:nF{ #1 }{ #1 / } #2 }
3195    \str_set:Nx \l__stex_importmodule_str {\stex_file_use:N \l__stex_importmodule_seq}
3196    \stex_debug:nn{imports}{Looking~for~\l_stex_import_uri_str?#3...}
3197    \__stex_importmodule_check_file:nn{ /#3.tex }{
3198      \__stex_importmodule_check_file:nn{/#3.\l_stex_current_language_str .tex}{
3199        \__stex_importmodule_check_file:nn{/#3.en.tex}{
3200          \__stex_importmodule_check_file:nn{.tex}{
3201            \__stex_importmodule_check_file:nn{.\l_stex_current_language_str .tex}{
3202              \__stex_importmodule_check_file:nn{.en.tex}{
3203              }
3204            }
3205          }
3206        }
3207      }
3208    }
3209    \stex_if_smsmode:TF{
3210      \exp_args:NNo \exp_args:Nnx \str_if_eq:nnTF{\l__stex_importmodule_str}{\stex_file_use:N
3211        \stex_debug:nn{imports}{Skipping~current~file}
3212      }{
3213        \IfFileExists{ \l__stex_importmodule_str }{
3214          \__stex_importmodule_load_file:n{#3}
3215        }{}
3216      }
3217    }{
3218      \IfFileExists{ \l__stex_importmodule_str }{
3219        \__stex_importmodule_load_file:n{#3}
3220      }{}
3221    }
3222 }
3223
3224 \cs_new_protected:Nn \__stex_importmodule_load_file:n {
3225    \stex_file_in_smsmode:on \l__stex_importmodule_str {
3226      \str_if_empty:NF \l__stex_importmodule_archive_str {
3227        \stex_set_current_archive:n \l__stex_importmodule_archive_str
3228      }
3229      \stex_debug:nn{modules}{Loading~\l__stex_importmodule_str}
3230    }
3231    \stex_if_module_exists:nF {\l_stex_import_uri_str?#1} {
```

```
3232        \msg_error:nnx{stex}{error/unknownmodule}{\l_stex_import_uri_str?#1}
3233     }
3234  }
3235
3236  \cs_new_protected:Npn \__stex_importmodule_check_file:nn #1 {
3237     \stex_debug:nn{imports}{Checking~ \l__stex_importmodule_str #1}
3238     \IfFileExists{ \l__stex_importmodule_str #1 }{
3239        \stex_debug:nn{imports}{Success}
3240        \str_set:Nx \l__stex_importmodule_str { \l__stex_importmodule_str #1 }
3241     }
3242  }
```

(*End definition for* `\stex_import_require_module:nnn`*. This function is documented on page* *119*.)

\importmodule

```
3243  \stex_new_stylable_cmd:nnnn{importmodule} { O{} m } {
3244     \__stex_importmodule_import_module:nn {#1}{#2}
3245     \stex_smsmode_do:
3246  }{}
3247  \stex_deactivate_macro:Nn \importmodule {module~environments}
3248
3249  \cs_new_protected:Nn \__stex_importmodule_import_module:nn {
3250     \stex_import_module_uri:nn { #1 }{ #2 }
3251     \stex_import_require_module:ooo
3252        \l_stex_import_archive_str
3253        \l_stex_import_path_str
3254        \l_stex_import_name_str
3255     \stex_execute_in_module:x{
3256        \stex_activate_module:n{\l_stex_import_ns_str}
3257     }
3258     \stex_module_add_morphism:nonn
3259        {}{\l_stex_import_ns_str}{import}{}
3260     \stex_if_do_html:T {
3261        \stex_annotate_invisible:nn
3262           {shtml:import=\l_stex_import_ns_str} {}
3263     }
3264     \stex_if_smsmode:F{
3265        \group_begin:
3266        \tl_set:Nn \thisarchive {#1}
3267        \tl_set_eq:NN \thismoduleuri \l_stex_import_ns_str
3268        \tl_set_eq:NN \thismodulename \l_stex_import_name_str
3269        \tl_clear:N \thisstyle
3270        \stex_style_apply:
3271        \group_end:
3272     }
3273  }
3274
3275  \cs_new_protected:Nn \__stex_importmodule_import_module_presms:nn {
3276     \stex_import_module_uri:nn { #1 }{ #2 }
3277     \tl_gput_right:Nx \g_stex_sms_import_code {
3278        \stex_import_require_module_safe:nnn
3279           {\l_stex_import_archive_str}
3280           {\l_stex_import_path_str}
3281           {\l_stex_import_name_str}
```

```
3282        }
3283    }
3284
3285    \stex_sms_allow_escape:N \importmodule
3286    \stex_every_module:n {\stex_reactivate_macro:N \importmodule}
3287    \stex_sms_allow_import:Nn \importmodule {
3288        \stex_reactivate_macro:N \importmodule
3289        \let \__stex_importmodule_import_module:nn \__stex_importmodule_import_module_presms:nn
3290    }
3291
3292    \stex_new_stylable_cmd:nnnn{requiremodule} { O{} m } {
3293        \stex_import_module_uri:nn { #1 }{ #2 }
3294        \stex_import_require_module:ooo
3295            \l_stex_import_archive_str
3296            \l_stex_import_path_str
3297            \l_stex_import_name_str
3298        \stex_do_up_to_module:x{
3299            \stex_activate_module:n{\l_stex_import_ns_str}
3300        }
3301        \stex_if_do_html:T {
3302            \stex_annotate_invisible:nn
3303                {shtml:import=\l_stex_import_ns_str} {}
3304        }
3305        \stex_if_smsmode:F{
3306            \group_begin:
3307            \tl_set:Nn \thisarchive {#1}
3308            \tl_set_eq:NN \thismoduleuri \l_stex_import_ns_str
3309            \tl_set_eq:NN \thismodulename \l_stex_import_name_str
3310            \tl_clear:N \thisstyle
3311            \stex_style_apply:
3312            \group_end:
3313        }
3314        \stex_smsmode_do:
3315    }{}
3316    \stex_deactivate_macro:Nn \requiremodule {module~environments}
```

(*End definition for* \importmodule. *This function is documented on page* *82*.)

13.7.2 Theory Morphisms

```
3317    ⟨@@=stex_morphisms⟩

3318    \stex_new_stylable_cmd:nnnn {assign} { m m }{
3319        \stex_get_in_morphism:n{#1}
3320        \_stex_assign_do:n{#2}
3321        \stex_smsmode_do:
3322    }{}
3323    \stex_sms_allow_escape:N\assign
3324
3325    \cs_new_protected:Nn \_stex_assign_do:n{
3326        \stex_debug:nn{assign}{Assigning~\l_stex_get_symbol_name_str~to~\tl_to_str:n{#1}}
3327        \tl_if_empty:NF \l_stex_get_symbol_def_tl {
3328            %\msg_error:nnxx{stex}{error/symbolalreadydefined}{\l_stex_get_symbol_name_str}{
3329            %   morphism~\l_stex_feature_name_str
3330            %}
```

206

```
3331      }
3332      \stex_check_term:n{#1}
3333      \stex_debug:nn{HERE!}{
3334        \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str^^J
3335        \tl_to_str:n{#1}
3336      }
3337      \stex_if_do_html:T{
3338        \stex_annotate_invisible:nn{shtml:assign={\l_stex_get_symbol_mod_str?\l_stex_get_symbol_
3339          \_stex_annotate_force_break:n{
3340            \mode_if_math:T\hbox{$\stex_annotate:nn{shtml:definiens={}}{#1}$}
3341          }
3342        }
3343      }
3344      \exp_args:Ne \stex_metagroup_do_in:nx{
3345        \l_stex_current_module_str / \l_stex_feature_name_str
3346      }{
3347        \prop_put:Nnn \exp_not:N \l_stex_morphism_symbols_prop
3348        {[\l_stex_get_symbol_mod_str]/[\l_stex_get_symbol_name_str]}
3349        {
3350          {\l_stex_get_symbol_macro_str}
3351          {\int_use:N \l_stex_get_symbol_arity_int}
3352          {\l_stex_get_symbol_args_tl}
3353          {\exp_not:n{#1}}
3354          {\exp_args:No\exp_not:n\l_stex_get_symbol_type_tl}
3355          {\exp_args:No\exp_not:n\l_stex_get_symbol_return_tl}
3356          {\l_stex_get_symbol_invoke_cs}
3357        }
3358      }
3359    }
3360
3361
3362    \stex_new_stylable_cmd:nnnn {renamedecl} { m O{} m }{
3363      \stex_get_in_morphism:n{#1}
3364      \_stex_renamedecl_do:nn{#2}{#3}
3365      \stex_smsmode_do:
3366    }{}
3367    \stex_sms_allow_escape:N\renamedecl
3368
3369    \cs_new_protected:Nn \_stex_renamedecl_do:nn {
3370      \stex_debug:nn{renamedecl}{Renaming~\l_stex_get_symbol_name_str~to~[#1]{#2}}
3371      \stex_if_do_html:T{
3372        \exp_args:Ne \stex_annotate_invisible:nn{
3373          shtml:rename={\l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str},
3374          shtml:macroname={#2}
3375          \str_if_empty:nF{#1}{ ,shtml:to={#1} }
3376        }{}
3377      }
3378      \exp_args:Ne \stex_metagroup_do_in:nx{
3379        \l_stex_current_module_str / \l_stex_feature_name_str
3380      }{
3381        \prop_put:Nnn \exp_not:N \l_stex_morphism_renames_prop
3382        {\l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str}{{#2}{
3383          \tl_if_empty:nTF{#1}{\l_stex_feature_name_str/\l_stex_get_symbol_name_str}{#1}
3384        }}
```

```
3385      }
3386  }
3387
3388  \stex_new_stylable_cmd:nnnn {assignMorphism} { m m }{
3389    \str_clear:N \l__stex_morphisms_morphism_dom_str
3390    \stex_iterate_morphisms:nn\l_stex_current_domain_str{
3391      \stex_debug:nn{assignMorphism}{
3392        Checking:~#1~vs:^^J##1^^J##2^^J##3^^J##4
3393      }
3394      \str_if_eq:nnTF{#1}{##1}{
3395        \__stex_morphisms_do_morph_assign:nnn{##1}{##2}{##4}
3396      }{
3397        \stex_str_if_ends_with:nnT{##2}{#1}{
3398          \__stex_morphisms_do_morph_assign:nnn{##1}{##2}{##4}
3399        }
3400      }
3401    }
3402    \str_if_empty:NT \l__stex_morphisms_morphism_dom_str {
3403      \msg_error:nnn{stex}{error/nomorphism}{#1}
3404    }
3405    \bool_set_false:N \l_tmpa_bool
3406    \stex_iterate_morphisms:nn \l_stex_current_module_str {
3407      \stex_debug:nn{assignMorphism}{
3408        Checking:~#2~vs:^^J##1^^J##2^^J##3^^J##4
3409      }
3410      \str_if_eq:nnTF{#2}{##1}{
3411        \stex_debug:nn{assignMorphism}{match!}
3412        \stex_iterate_break:n{
3413          \stex_annotate_invisible:nn{
3414            shtml:assignMorphismFrom={\l__stex_morphisms_morphism_dom_str}
3415            ahtml:assignMorphismTo={\l_stex_current_module_str?##1}
3416          }{}
3417          \bool_set_true:N \l_tmpa_bool
3418        }
3419      }{
3420        \stex_str_if_ends_with:nnT{##2}{#2}{
3421          \stex_debug:nn{assignMorphism}{match!}
3422          \stex_iterate_break:n{
3423            \stex_annotate_invisible:nn{
3424              shtml:assignMorphismFrom={\l__stex_morphisms_morphism_dom_str},
3425              shtml:assignMorphismTo={\l_stex_current_module_str?##1}
3426            }{}
3427            \bool_set_true:N \l_tmpa_bool
3428          }
3429        }
3430      }
3431    }
3432    \bool_if:NF \l_tmpa_bool {
3433      \msg_error:nnn{stex}{error/nomorphism}{#2}
3434    }
3435  }{}
3436  \cs_new_protected:Nn \__stex_morphisms_do_morph_assign:nnn {
3437    \stex_iterate_break:n{
3438      \str_set:Nx \l__stex_morphisms_morphism_dom_str { \l_stex_current_domain_str ? #1 }
```

```
3439     \stex_debug:nn{assignMorphism}{match!}
3440     \stex_iterate_symbols:nn{#2}{
3441        \stex_debug:nn{assignMorphism}{removing~##1?##3}
3442        % TODO: non-trivial assignments
3443        \prop_remove:Nn \l_stex_morphism_symbols_prop {
3444           [##1]/[##3]
3445        }
3446     }
3447   }
3448 }

3449

3450 \stex_deactivate_macro:Nn \assign {morphism~environments}
3451 \stex_deactivate_macro:Nn \renamedecl {morphism~environments}
3452 \stex_deactivate_macro:Nn \assignMorphism {morphism~environments}

3453 \stex_new_stylable_env:nnnnnnn {copymodule}{m O{} m}{

3454

3455     \stex_structural_feature_morphism:nnnnn{#1}{morphism}{#2}{#3}{,shtml:total=false}

3456

3457     \stex_if_smsmode:F {
3458        \tl_set:Nn \thiscopyname { #1 }
3459        \tl_set_eq:NN \thismoduleuri \l_stex_current_domain_str
3460        \stex_style_apply:
3461     }
3462     \stex_smsmode_do:
3463 }{
3464     \stex_if_smsmode:F {
3465        \stex_style_apply:
3466     }
3467     \stex_structural_feature_morphism_end:
3468 }{}{}{}
3469 \stex_deactivate_macro:Nn \copymodule {module~environments}
3470 \stex_every_module:n {
3471     \stex_reactivate_macro:N \copymodule
3472 }
3473 \stex_sms_allow_env:n{copymodule}

3474

3475 \stex_new_stylable_env:nnnnnnn {interpretmodule}{m O{} m}{
3476     \stex_structural_feature_morphism:nnnnn{#1}{morphism}{#2}{#3}{,shtml:total=true}
3477     \stex_if_smsmode:F {
3478        \tl_set:Nn \thiscopyname { #1 }
3479        \tl_set_eq:NN \thismoduleuri \l_stex_current_domain_str
3480        \stex_style_apply:
3481     }
3482     \stex_smsmode_do:
3483 }{
3484     \stex_structural_feature_morphism_check_total:
3485     \stex_if_smsmode:F {
3486        \stex_style_apply:
3487     }
3488     \stex_structural_feature_morphism_end:
3489 }{}{}{}
3490 \stex_deactivate_macro:Nn \interpretmodule {module~environments}
3491 \stex_every_module:n {
3492     \stex_reactivate_macro:N \interpretmodule
```

```latex
3493 }
3494 \stex_sms_allow_env:n{interpretmodule}
3495 \stex_new_stylable_env:nnnnnnn {realization}{O{} m}{
3496
3497   \stex_structural_feature_morphism:nnnnn{}{morphism}{#1}{#2}{,shtml:total=true}
3498   %\stex_execute_in_module:x{
3499   %  \stex_activate_module:n{\l_stex_current_domain_str}
3500   %}
3501   \stex_if_smsmode:F {
3502     \tl_set:Nn \thiscopyname { #2 }
3503     \tl_set_eq:NN \thismoduleuri \l_stex_current_domain_str
3504     \stex_style_apply:
3505   }
3506   \stex_smsmode_do:
3507 }{
3508   \stex_structural_feature_morphism_check_total:
3509   \stex_if_smsmode:F {
3510     \stex_style_apply:
3511   }
3512   \stex_structural_feature_morphism_end:
3513 }{}{}{}
3514 \stex_deactivate_macro:Nn \realization {module~environments}
3515 \stex_every_module:n {
3516   \stex_reactivate_macro:N \realization
3517 }
3518 \stex_sms_allow_env:n{realization}
3519 \cs_new_protected:Nn \__stex_morphisms_parse_assign:n {
3520   \str_clear:N \l__stex_morphisms_name_str
3521   \str_clear:N \l__stex_morphisms_newname_str
3522   \tl_clear:N \l__stex_morphisms_ass_tl
3523   \exp_args:NNe \seq_set_split:Nnn \l__stex_morphisms_seq {\tl_to_str:n{@}} {#1}
3524   \int_compare:nNnTF {\seq_count:N \l__stex_morphisms_seq} = 1 {
3525     \seq_pop_left:NN \l__stex_morphisms_seq \l__stex_morphisms_next_tl
3526   }{
3527     \seq_pop_left:NN \l__stex_morphisms_seq \l__stex_morphisms_name_str
3528     \exp_args:NNo \str_set:Nn \l__stex_morphisms_name_str \l__stex_morphisms_name_str
3529     \tl_set:Nx \l__stex_morphisms_next_tl {\seq_use:Nn \l__stex_morphisms_seq @}
3530   }
3531   \exp_args:NNNo \seq_set_split:Nnn \l__stex_morphisms_seq = \l__stex_morphisms_next_tl
3532   \str_if_empty:NTF \l__stex_morphisms_name_str {
3533     \seq_pop_left:NN \l__stex_morphisms_seq \l__stex_morphisms_name_str
3534     \exp_args:NNo \str_set:Nn \l__stex_morphisms_name_str \l__stex_morphisms_name_str
3535     \tl_set:Nx \l__stex_morphisms_ass_tl {\seq_use:Nn \l__stex_morphisms_seq =}
3536   }{
3537     \seq_pop_left:NN \l__stex_morphisms_seq \l__stex_morphisms_newname_str
3538     \exp_args:NNo \str_set:Nn \l__stex_morphisms_newname_str \l__stex_morphisms_newname_str
3539     \tl_set:Nx \l__stex_morphisms_ass_tl {\seq_use:Nn \l__stex_morphisms_seq =}
3540   }
3541   \__stex_morphisms_do_parsed_assign:
3542 }
3543
3544 \cs_new_protected:Nn \__stex_morphisms_do_parsed_assign: {
3545   \exp_args:No \stex_get_in_morphism:n \l__stex_morphisms_name_str
```

```
3546    \str_if_empty:NF \l__stex_morphisms_newname_str {
3547      \exp_after:wN \__stex_morphisms_do_parsed_newname: \l__stex_morphisms_newname_str \__ste
3548    }
3549    \tl_if_empty:NF \l__stex_morphisms_ass_tl {
3550      \exp_args:No \_stex_assign_do:n \l__stex_morphisms_ass_tl
3551    }
3552  }
3553
3554  \cs_new_protected:Nn \__stex_morphisms_do_parsed_newname: {
3555    \peek_charcode:NTF [ {
3556      \__stex_morphisms_do_parsed_newname:w
3557    }{
3558      \__stex_morphisms_do_parsed_newname:w []
3559    }
3560  }
3561
3562  \cs_new_protected:Npn \__stex_morphisms_do_parsed_newname:w [#1] #2 \__stex_morphisms_end: {
3563    \_stex_renamedecl_do:nn{#1}{#2}
3564  }
3565
3566  \stex_new_stylable_cmd:nnnn{copymod}{m O{} m m}{
3567    \stex_structural_feature_morphism:nnnnn{#1}{morphism}{#2}{#3}{,shtml:total=false}
3568
3569    \clist_map_function:nN{#4}\__stex_morphisms_parse_assign:n
3570
3571    \stex_if_smsmode:F {
3572      \tl_set:Nn \thiscopyname { #1 }
3573      \tl_set_eq:NN \thismoduleuri \l_stex_current_domain_str
3574      \stex_style_apply:
3575    }
3576    \stex_structural_feature_morphism_end:
3577    \stex_smsmode_do:
3578  }{}
3579  \stex_deactivate_macro:Nn \copymod {module~environments}
3580  \stex_every_module:n {
3581    \stex_reactivate_macro:N \copymod
3582  }
3583  \stex_sms_allow_escape:N\copymod
3584
3585
3586  \stex_new_stylable_cmd:nnnn{interpretmod}{m O{} m m}{
3587    \stex_structural_feature_morphism:nnnnn{#1}{morphism}{#2}{#3}{,shtml:total=true}
3588
3589    \clist_map_function:nN{#4}\__stex_morphisms_parse_assign:n
3590
3591    \stex_if_smsmode:F {
3592      \tl_set:Nn \thiscopyname { #1 }
3593      \tl_set_eq:NN \thismoduleuri \l_stex_current_domain_str
3594      \stex_style_apply:
3595    }
3596    \stex_structural_feature_morphism_check_total:
3597    \stex_structural_feature_morphism_end:
3598    \stex_smsmode_do:
3599  }{}
```

```
3600 \stex_deactivate_macro:Nn \interpretmod {module~environments}
3601 \stex_every_module:n {
3602   \stex_reactivate_macro:N \iterpretmod
3603 }
3604 \stex_sms_allow_escape:N\interpretmod
3605

3606
3607 \stex_new_stylable_cmd:nnnn{realize}{O{} m m}{
3608   \stex_structural_feature_morphism:nnnnn{}{morphism}{#1}{#2}{,shtml:total=true}
3609
3610   \clist_map_function:nN{#3}\__stex_morphisms_parse_assign:n
3611
3612   \stex_if_smsmode:F {
3613     \tl_set:Nn \thiscopyname { #1 }
3614     \tl_set_eq:NN \thismoduleuri \l_stex_current_domain_str
3615     \stex_style_apply:
3616   }
3617   \stex_structural_feature_morphism_check_total:
3618   \stex_structural_feature_morphism_end:
3619   \stex_smsmode_do:
3620 }{}
3621 \stex_deactivate_macro:Nn \realize {module~environments}
3622 \stex_every_module:n {
3623   \stex_reactivate_macro:N \realize
3624 }
3625 \stex_sms_allow_escape:N\realize
```

13.8 Symbols

13.8.1 Declarations

```
3626 ⟨@@=stex_symdecl⟩
```

Some setup:

\stex_if_check_terms_p:
\stex_if_check_terms:*TF*

```
3627 \stex_if_html_backend:TF {
3628   \prg_new_conditional:Nnn \stex_if_check_terms: {p, T, F, TF} {
3629     \prg_return_false:
3630   }
3631 }{
3632   \stex_get_env:Nn\__stex_symdecl_env_str{STEX_CHECKTERMS}
3633   \str_if_empty:NF\__stex_symdecl_env_str{
3634     \exp_args:No \str_if_eq:nnF \__stex_symdecl_env_str{false}{
3635       \bool_set_true:N \c_stex_check_terms_bool
3636     }
3637   }
3638   \bool_if:NTF \c_stex_check_terms_bool {
3639     \prg_new_conditional:Nnn \stex_if_check_terms: {p, T, F, TF} {
3640       \prg_return_true:
3641     }
3642   }{
3643     \prg_new_conditional:Nnn \stex_if_check_terms: {p, T, F, TF} {
3644       \prg_return_false:
3645     }
```

```
3646    }
3647 }
```

(*End definition for* `\stex_if_check_terms:TF`*. This function is documented on page 113.*)

\stex_check_term:n

```
3648 \stex_if_check_terms:TF{
3649    \cs_new_protected:Nn \stex_check_term:n {
3650       \hbox_set:Nn \l_tmpa_box {
3651          \group_begin:
3652             $#1$
3653          \group_end:
3654       }
3655    }
3656 }{
3657    \cs_new_protected:Nn \stex_check_term:n {}
3658 }
```

(*End definition for* `\stex_check_term:n`*. This function is documented on page 113.*)

symdecl arguments:

```
3659 \stex_keys_define:nnnn{symargs}{
3660    \str_clear:N \l_stex_key_args_str
3661    \str_clear:N \l_stex_key_role_str
3662    \str_clear:N \l_stex_key_reorder_str
3663    \str_clear:N \l_stex_key_assoc_str
3664 }{
3665    args      .str_set:N  = \l_stex_key_args_str ,
3666    reorder   .str_set:N  = \l_stex_key_reorder_str ,
3667    assoc     .choices:nn  = {bin,binl,binr,pre,conj,pwconj}
3668      {\str_set:Nx \l_stex_key_assoc_str \l_keys_choice_tl},
3669    role      .str_set:N     = \l_stex_key_role_str
3670 }{}
3671
3672 \stex_keys_define:nnnn{decl}{
3673    \str_clear:N \l_stex_key_name_str
3674    \str_clear:N \l_stex_key_args_str
3675    \tl_clear:N \l_stex_key_type_tl
3676    \tl_clear:N \l_stex_key_def_tl
3677    \tl_clear:N \l_stex_key_return_tl
3678    \clist_clear:N \l_stex_key_argtypes_clist
3679 }{
3680    name      .str_set:N  = \l_stex_key_name_str ,
3681
3682    return    .tl_set:N     = \l_stex_key_return_tl ,
3683    argtypes  .clist_set:N = \l_stex_key_argtypes_clist ,
3684
3685    type      .tl_set:N     = \l_stex_key_type_tl  ,
3686    def       .tl_set:N     = \l_stex_key_def_tl   ,
3687
3688    align     .code:n       = {},
3689    gfc       .code:n       = {}
3690 }{style,deprecate,symargs}
3691 % \_stex_do_deprecation:n{#2}
```

```
3692 \str_new:N \l_stex_macroname_str
3693 \stex_new_stylable_cmd:nnnn {symdecl} { s m O{}} {
3694   \stex_keys_set:nn{decl}{#3}
3695   \IfBooleanTF #1 {
3696     \str_clear:N \l_stex_macroname_str
3697   }{
3698     \str_set:Nx \l_stex_macroname_str { #2 }
3699   }
3700   \stex_symdecl_top:n{#2}
3701
3702   \stex_if_smsmode:F{
3703     \group_begin:
3704     \tl_set:Nx \thisdecluri {\l_stex_current_module_str ? \l_stex_key_name_str}
3705     \tl_set_eq:NN \thisdeclname \l_stex_key_name_str
3706     \tl_set_eq:NN \thistype \l_stex_key_type_tl
3707     \tl_set_eq:NN \thisdefiniens \l_stex_key_def_tl
3708     \tl_set_eq:NN \thisargs \l_stex_key_args_str
3709     \tl_clear:N \thisstyle
3710     \stex_style_apply:
3711     \group_end:
3712   }
3713   \stex_smsmode_do:
3714 }{}
3715 \stex_deactivate_macro:Nn \symdecl {module~environments}
3716 \stex_every_module:n {\stex_reactivate_macro:N \symdecl}
3717 \stex_sms_allow_escape:N \symdecl
```

(*End definition for* \symdecl. *This function is documented on page 76.*)

```
3718 \cs_new_protected:Nn \stex_symdecl_top:n {
3719   \str_if_empty:NT \l_stex_key_name_str {
3720     \str_set:Nx \l_stex_key_name_str { #1 }
3721   }
3722   \stex_symdecl_do:
3723   \_stex_symdecl_check_terms:
3724   \__stex_symdecl_add_decl:
3725   \stex_if_do_html:T {
3726     \_stex_symdecl_html:
3727   }
3728 }
3729
3730 \cs_new_protected:Nn \__stex_symdecl_add_decl: {
3731   \exp_args:Nnx \use:nn {\stex_module_add_symbol:nnnnnnnN}{
3732     {\l_stex_macroname_str}
3733     {\l_stex_key_name_str}
3734     {\int_use:N \l_stex_get_symbol_arity_int}
3735     {\l_stex_get_symbol_args_tl}
3736     {\tl_if_empty:NF \l_stex_key_def_tl{DEFED} }%{\exp_args:No \exp_not:n \l_stex_key_def_tl
3737     {}%{\exp_args:No \exp_not:n \l_stex_key_type_tl}
3738     {\exp_args:No \exp_not:n \l_stex_key_return_tl}
3739     \stex_invoke_symbol:
3740   }
```

214

```
3741     \exp_args:Ne \stex_ref_new_symbol:n
3742       {\l_stex_current_module_str?\l_stex_key_name_str}
3743 }
3744
3745 \cs_new:Nn \_stex_return_args:nn {
3746   {\svar{ARGUMENT_#1}\_stex_eat_exclamation_point:}
3747 }
3748
3749 \cs_new_protected:Nn \_stex_symdecl_html: {
3750   \exp_args:Ne \stex_annotate_invisible:nn {
3751     shtml:symdecl = {\l_stex_current_module_str ? \l_stex_key_name_str},
3752     shtml:args = {\l_stex_key_args_str}
3753     \str_if_empty:NF \l_stex_macroname_str {,
3754       shtml:macroname={\l_stex_macroname_str}
3755     }
3756     \str_if_empty:NF \l_stex_key_assoc_str {,
3757       shtml:assoctype={\l_stex_key_assoc_str}
3758     }
3759     \str_if_empty:NF \l_stex_key_reorder_str {,
3760       shtml:reorderargs={\l_stex_key_reorder_str}
3761     }
3762     \str_if_empty:NF \l_stex_key_role_str {,
3763       shtml:role={\l_stex_key_role_str}
3764     }
3765   }{\hbox\bgroup\_stex_annotate_force_break:n{
3766       \bool_set_true:N \stex_in_invisible_html_bool
3767       \tl_if_empty:NF \l_stex_key_type_tl {
3768         $\stex_annotate:nn{shtml:type={}}{\l_stex_key_type_tl}$
3769       }
3770       \tl_if_empty:NF \l_stex_key_def_tl {
3771         $\stex_annotate:nn{shtml:definiens={}}{\l_stex_key_def_tl}$
3772       }
3773       \tl_if_empty:NF \l_stex_key_return_tl{
3774         \exp_args:Nno \use:n{
3775         \cs_generate_from_arg_count:NNnn \l__stex_symdecl_cs
3776         \cs_set:Npn \l_stex_get_symbol_arity_int} \l_stex_key_return_tl
3777         \tl_set:Nx \l__stex_symdecl_args_tl {\_stex_map_args:N \_stex_return_args:nn}
3778         $\stex_annotate:nn{shtml:returntype={}}{
3779           \exp_after:wN \l__stex_symdecl_cs \l__stex_symdecl_args_tl!
3780         }$
3781       }
3782       \clist_if_empty:NF \l_stex_key_argtypes_clist {
3783         \stex_annotate:nn{shtml:argtypes={}}{\_stex_annotate_force_break:n{
3784           \clist_map_inline:Nn \l_stex_key_argtypes_clist {
3785             $\stex_annotate:nn{shtml:type={}}{##1}$
3786           }
3787         }}
3788       }
3789   }\egroup}
3790 }
```

(*End definition for* `\stex_symdecl_top:n`. *This function is documented on page* *114.*)

\stex_symdecl_do: Requires the above keys and \l_stex_macroname_str to be set first

```
3791  \cs_new_protected:Nn \stex_symdecl_do: {
3792    \_stex_do_deprecation:n \l_stex_key_name_str
3793    \__stex_symdecl_parse_arity:
3794    \__stex_symdecl_do_args:
3795  }
3796
3797  \int_new:N \l_stex_assoc_args_count
3798
3799  \cs_new_protected:Nn \__stex_symdecl_parse_arity: {
3800    \int_zero:N \l_stex_get_symbol_arity_int
3801    \int_zero:N \l_stex_assoc_args_count
3802    \str_map_inline:Nn \l_stex_key_args_str {
3803      \str_case:nnF ##1 {
3804        0 { \str_map_break: }
3805        1 { \str_map_break:n{
3806          \int_set:Nn \l_stex_get_symbol_arity_int {1}
3807          \str_set:Nn \l_stex_key_args_str {i}
3808        } }
3809        2 { \str_map_break:n{
3810          \int_set:Nn \l_stex_get_symbol_arity_int {2}
3811          \str_set:Nn \l_stex_key_args_str {ii}
3812        } }
3813        3 { \str_map_break:n{
3814          \int_set:Nn \l_stex_get_symbol_arity_int {3}
3815          \str_set:Nn \l_stex_key_args_str {iii}
3816        } }
3817        4 { \str_map_break:n{
3818          \int_set:Nn \l_stex_get_symbol_arity_int {4}
3819          \str_set:Nn \l_stex_key_args_str {iiii}
3820        } }
3821        5 { \str_map_break:n{
3822          \int_set:Nn \l_stex_get_symbol_arity_int {5}
3823          \str_set:Nn \l_stex_key_args_str {iiiii}
3824        } }
3825        6 { \str_map_break:n{
3826          \int_set:Nn \l_stex_get_symbol_arity_int {6}
3827          \str_set:Nn \l_stex_key_args_str {iiiiii}
3828        } }
3829        7 { \str_map_break:n{
3830          \int_set:Nn \l_stex_get_symbol_arity_int {7}
3831          \str_set:Nn \l_stex_key_args_str {iiiiiii}
3832        } }
3833        8 { \str_map_break:n{
3834          \int_set:Nn \l_stex_get_symbol_arity_int {8}
3835          \str_set:Nn \l_stex_key_args_str {iiiiiiii}
3836        } }
3837        9 { \str_map_break:n{
3838          \int_set:Nn \l_stex_get_symbol_arity_int {9}
3839          \str_set:Nn \l_stex_key_args_str {iiiiiiiii}
3840        } }
3841        i {\int_incr:N \l_stex_get_symbol_arity_int}
3842        b {\int_incr:N \l_stex_get_symbol_arity_int}
3843        a {\int_incr:N \l_stex_get_symbol_arity_int \int_incr:N \l_stex_assoc_args_count}
3844        B {\int_incr:N \l_stex_get_symbol_arity_int \int_incr:N \l_stex_assoc_args_count}
```

```
3845      }{
3846        \msg_error:nnxx{stex}{error/wrongargs}{
3847          \l_stex_current_module_str ? \l_stex_key_name_str
3848        }{##1}
3849      }
3850    }
3851  }
3852
3853  \cs_new_protected:Nn \__stex_symdecl_do_args: {
3854    \tl_clear:N \l_stex_get_symbol_args_tl
3855    \int_step_inline:nn \l_stex_get_symbol_arity_int {
3856      \tl_put_right:Nn \l_stex_get_symbol_args_tl {##1}
3857      \tl_put_right:Nx \l_stex_get_symbol_args_tl {
3858        \str_item:Nn \l_stex_key_args_str {##1}
3859      }
3860    }
3861  }
```

(*End definition for* \stex_symdecl_do:. *This function is documented on page 113.*)

_stex_symdecl_check_terms:

```
3862  \cs_new_protected:Nn \_stex_symdecl_check_terms: {
3863    \stex_check_term:n{
3864      \stex_debug:nn{check_terms}{Checking~type...}
3865      \group_begin:\l_stex_key_type_tl\group_end:
3866      \stex_debug:nn{check_terms}{Checking~definiens...}
3867      \group_begin:\l_stex_key_def_tl\group_end:
3868      \stex_debug:nn{check_terms}{Checking~return...}
3869      \group_begin:\l_stex_key_return_tl!\group_end:
3870      \stex_debug:nn{check_terms}{Checking~argument~types...}
3871      \group_begin:\l_stex_key_argtypes_clist\group_end:
3872    }
3873  }
```

(*End definition for* _stex_symdecl_check_terms:. *This function is documented on page 114.*)

\textsymdecl

```
3874
3875  \stex_keys_define:nnnn{textsymdecl}{
3876    \str_clear:N \l_stex_key_name_str
3877    \tl_clear:N \l_stex_key_type_tl
3878    \tl_clear:N \l_stex_key_def_tl
3879  }{
3880    name      .str_set:N  = \l_stex_key_name_str ,
3881    type      .tl_set:N    = \l_stex_key_type_tl ,
3882    def       .tl_set:N    = \l_stex_key_def_tl
3883  }{style,deprecate}
3884
3885  \stex_new_stylable_cmd:nnnn {textsymdecl} {m O{} m} {
3886    \stex_keys_set:nn{symdef}{}
3887    \stex_keys_set:nn{textsymdecl}{#2}
3888    \str_set:Nx \l_stex_macroname_str { #1 }
3889    \str_if_empty:NT \l_stex_key_name_str {
3890      \str_set:Nn \l_stex_key_name_str {#1}
3891    }%{
```

```
3892    %  \str_set:Nx \l_stex_key_name_str {\l_stex_key_name_str-sym}
3893    %}
3894    \str_set:Nn \l_stex_key_role_str {textsymdecl}
3895
3896    \stex_symdecl_do:
3897    \_stex_symdecl_check_terms:
3898    \exp_args:Nnx \use:nn {\stex_module_add_symbol:nnnnnnnN}{
3899      {\l_stex_macroname_str}
3900      {\l_stex_key_name_str}
3901      {0}{}
3902      {\tl_if_empty:NF \l_stex_key_def_tl{DEFED} }
3903      {}% type
3904      {\use:c{#1name_nospace}}% return
3905      \stex_invoke_text_symbol:
3906    }
3907    \exp_args:Ne \stex_ref_new_symbol:n
3908      {\l_stex_current_module_str?\l_stex_key_name_str}
3909    \stex_if_do_html:T {
3910      \_stex_symdecl_html:
3911    }
3912
3913    \int_set:Nn \l_stex_get_symbol_arity_int 0
3914    \tl_clear:N \l_stex_key_op_tl
3915    \str_clear:N \l_stex_key_intent_str
3916    \str_clear:N \l_stex_key_prec_str
3917    \str_set_eq:NN \l_stex_get_symbol_mod_str \l_stex_current_module_str
3918    \str_set_eq:NN \l_stex_get_symbol_name_str \l_stex_key_name_str
3919    \stex_notation_parse:n{\hbox{#3}}
3920    \_stex_notation_add:
3921    \stex_if_do_html:T {
3922      \def\comp{\_comp}
3923      \_stex_notation_do_html:n{\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
3924    }
3925    \stex_execute_in_module:x{
3926      \__stex_symdecl_set_textsymdecl_macro:nnn{#1}{\l_stex_current_module_str?\l_stex_key_nam
3927      \exp_not:n{{#3}}
3928    }
3929
3930    \stex_if_smsmode:F{
3931      \group_begin:
3932      \tl_set:Nx \thisdecluri {\l_stex_current_module_str ? \l_stex_key_name_str}
3933      \tl_set_eq:NN \thisdeclname \l_stex_key_name_str
3934      \tl_clear:N \thisstyle
3935      \stex_style_apply:
3936      \group_end:
3937    }
3938    \stex_smsmode_do:
3939 }{}
3940 \stex_deactivate_macro:Nn \textsymdecl {module~environments}
3941 \stex_every_module:n {\stex_reactivate_macro:N \textsymdecl}
3942 \stex_sms_allow_escape:N \textsymdecl
3943
3944 \cs_new_protected:Nn \__stex_symdecl_set_textsymdecl_macro:nnn {
3945    \cs_set_protected:cpn{#1name_nospace}{#3}
```

```
3946        \cs_set_protected:cpn{#1name}{
3947          \mode_if_vertical:T{\hbox_unpack:N\c_empty_box}
3948          \mode_if_math:T\hbox{\let\xspace\relax #3}
3949          \mode_if_math:F{\cs_if_exist:NT\xspace\xspace}
3950        }
3951      }
3952
3953      \cs_new_protected:Nn \stex_invoke_text_symbol: {
3954        \mode_if_vertical:T{\hbox_unpack:N\c_empty_box}
3955        \_stex_term_oms_or_omv:nnn{}{}{\maincomp{\let\xspace\relax\l_stex_current_return_tl}}
3956        \group_end:\mode_if_math:F{\cs_if_exist:NT\xspace\xspace}
3957      }
```

(*End definition for* \textsymdecl. *This function is documented on page* *77.*)

\stex_get_symbol:n

```
3958      \cs_new_protected:Nn \stex_get_symbol:n {
3959        \_stex_get_symbol:n{ #1 }
3960        \str_if_empty:NT \l_stex_get_symbol_name_str {
3961          \msg_error:nnn{stex}{error/unknownsymbol}{#1}
3962        }
3963      }
3964
3965      \cs_new_protected:Nn \_stex_get_symbol:n {
3966        \str_clear:N \l_stex_get_symbol_mod_str
3967        \str_clear:N \l_stex_get_symbol_name_str
3968        \cs_if_exist:cTF { #1 }{
3969          \cs_set_eq:Nc \l__stex_symdecl_cs { #1 }
3970          % command name
3971          \exp_args:Ne \tl_if_empty:nTF { \cs_argument_spec:N \l__stex_symdecl_cs }{
3972            % ...that takes no arguments
3973            \exp_args:Ne \cs_if_eq:NNTF {\tl_head:N \l__stex_symdecl_cs}
3974              \_stex_invoke_symbol:nnnnnnnN
3975              \__stex_symdecl_get_symbol_from_cs:
3976              {\__stex_symdecl_get_symbol_from_string:n { #1 }}
3977          }{
3978            \__stex_symdecl_get_symbol_from_string:n { #1 }
3979          }
3980        }{
3981          \__stex_symdecl_get_symbol_from_string:n { #1 }
3982        }
3983      }
3984
3985      \int_new:N \l_stex_get_symbol_arity_int
3986      \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
3987        \stex_debug:nn{symbols}{Getting~from~cs...}
3988        \stex_pseudogroup_with:nn{\_stex_invoke_symbol:nnnnnnnN}{
3989          \cs_set:Npn \_stex_invoke_symbol:nnnnnnnN ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 {
3990            \str_set:Nn \l_stex_get_symbol_mod_str {##1}
3991            \str_set:Nn \l_stex_get_symbol_name_str {##2}
3992            \int_set:Nn \l_stex_get_symbol_arity_int {##3}
3993            \tl_set:Nn \l_stex_get_symbol_args_tl {##4}
3994            \tl_set:Nn \l_stex_get_symbol_def_tl {##5}
3995            \tl_set:Nn \l_stex_get_symbol_type_tl {##6}
```

219

```
3996        \tl_set:Nn \l_stex_get_symbol_return_tl {##7}
3997        \tl_set:Nn \l_stex_get_symbol_invoke_cs {##8}
3998      }
3999      \l__stex_symdecl_cs
4000    }
4001  }
4002
4003  \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
4004    \stex_debug:nn{symbols}{Getting~from~string~#1...}
4005    \seq_set_split:Nnn \l__stex_symdecl_seq ? {#1}
4006    \seq_pop_right:NN \l__stex_symdecl_seq \l__stex_symdecl_name
4007    \seq_if_empty:NTF \l__stex_symdecl_seq {
4008      \exp_args:No \__stex_symdecl_get_from_one_string:n {#1}
4009    }{
4010      \exp_args:NNe \exp_args:Nno \__stex_symdecl_get_symbol_from_modules:nn {
4011        \seq_use:Nn \l__stex_symdecl_seq ?
4012      } \l__stex_symdecl_name
4013    }
4014  }
4015
4016  \cs_new_protected:Nn \__stex_symdecl_sym_from_str_i:nnnn {
4017    \bool_lazy_any:nTF{
4018      {\str_if_eq_p:nn{#2}{#3}}
4019      {\str_if_eq_p:nn{#2}{#4}}
4020      {\stex_str_if_ends_with_p:nn{#4}{/#2}}
4021    }{
4022      \__stex_symdecl_sym_i_finish:nnnnnnnN{#1}{#4}
4023    }{
4024      \__stex_symdecl_sym_i_gobble:nnnnn
4025    }
4026  }
4027  \cs_new_protected:Nn \__stex_symdecl_sym_i_gobble:nnnnn {}
4028
4029  \cs_new_protected:Nn \__stex_symdecl_sym_i_finish:nnnnnnnN {
4030    \prop_map_break:n{\seq_map_break:n{
4031      \str_set:Nn \l_stex_get_symbol_mod_str {#1}
4032      \str_set:Nn \l_stex_get_symbol_name_str {#2}
4033      \int_set:Nn \l_stex_get_symbol_arity_int {#3}
4034      \tl_set:Nn \l_stex_get_symbol_args_tl {#4}
4035      \tl_set:Nn \l_stex_get_symbol_def_tl {#5}
4036      \tl_set:Nn \l_stex_get_symbol_type_tl {#6}
4037      \tl_set:Nn \l_stex_get_symbol_return_tl {#7}
4038      \tl_set:Nn \l_stex_get_symbol_invoke_cs {#8}
4039    }}
4040  }
4041
4042  \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_modules:nn {
4043    \stex_debug:nn{symbols}{Getting~#2~in~#1...}
4044    \seq_map_inline:Nn \l_stex_all_modules_seq {
4045      \stex_str_if_ends_with:nnT{##1}{#1}{
4046        \prop_map_inline:cn{c_stex_module_##1_symbols_prop}{
4047          \__stex_symdecl_sym_from_str_i:nnnn{##1}{#2} ####2
4048        }
4049      }
```

```
4050        }
4051    }
4052
4053    \cs_new_protected:Nn \__stex_symdecl_get_from_one_string:n {
4054        \stex_debug:nn{symbols}{Getting~#1~anywhere...}
4055        \stex_iterate_symbols:n{
4056            %\stex_debug:nn{symbols}{>#1==##2~|~#1==##3<...}
4057            \bool_lazy_any:nT{
4058                {\str_if_eq_p:nn{#1}{##2}}
4059                {\str_if_eq_p:nn{#1}{##3}}
4060                {\stex_str_if_ends_with_p:nn{##3}{/#1}}
4061            }{
4062                \stex_iterate_break:n{
4063                    \str_set:Nn \l_stex_get_symbol_mod_str {##1}
4064                    \str_set:Nn \l_stex_get_symbol_name_str {##3}
4065                    \int_set:Nn \l_stex_get_symbol_arity_int {##4}
4066                    \tl_set:Nn \l_stex_get_symbol_args_tl {##5}
4067                    \tl_set:Nn \l_stex_get_symbol_def_tl {##6}
4068                    \tl_set:Nn \l_stex_get_symbol_type_tl {##7}
4069                    \tl_set:Nn \l_stex_get_symbol_return_tl {##8}
4070                    \tl_set:Nn \l_stex_get_symbol_invoke_cs {##9}
4071                }
4072            }
4073        }
4074    }
```

(*End definition for* `\stex_get_symbol:n`. *This function is documented on page* *113.*)

13.8.2   Notations

```
4075    ⟨@@=stex_notations⟩
```

_stex_map_args:N
_stex_map_notation_args:N

```
4076    \cs_new:Nn \_stex_map_args:N {
4077        \tl_if_empty:NF \l_stex_get_symbol_args_tl {
4078            \exp_after:wN \__stex_notations_map_args_i:w \exp_after:wN
4079            #1 \l_stex_get_symbol_args_tl \__stex_notations_args_end:
4080        }
4081    }
4082    \cs_new:Npn \__stex_notations_map_args_i:w #1 #2 #3 #4 \__stex_notations_args_end: {
4083        #1 #2 #3
4084        \tl_if_empty:nF{#4}{
4085            \__stex_notations_map_args_i:w #1 #4 \__stex_notations_args_end:
4086        }
4087    }
4088
4089    \cs_new:Nn \_stex_map_notation_args:N {
4090        \tl_if_empty:NF \l_stex_notation_args_tl {
4091            \exp_after:wN \__stex_notations_map_args_ii:w \exp_after:wN
4092            #1 \l_stex_get_symbol_args_tl \__stex_notations_args_end:
4093        }
4094    }
4095    \cs_new:Npn \__stex_notations_map_args_ii:w #1 #2 #3 #4 #5 #6 \__stex_notations_args_end: {
4096        #1 #2 #3 #4 #5
4097        \tl_if_empty:nF{#6}{
```

```
4098        \__stex_notations_map_args_ii:w #1 #6 \__stex_notations_args_end:
4099      }
4100  }
```

(*End definition for* _stex_map_args:N *and* _stex_map_notation_args:N. *These functions are documented on page* **??**.)

notation arguments:

```
4101  \stex_keys_define:nnnn{notation}{
4102    \str_clear:N \l_stex_key_variant_str
4103    \str_clear:N \l_stex_key_prec_str
4104    \str_clear:N \l_stex_key_op_tl
4105    \str_clear:N \l_stex_key_intent_str
4106    \clist_clear:N \l_stex_key_intent_args_clist
4107  }{
4108    variant     .str_set_x:N  = \l_stex_key_variant_str ,
4109    prec        .str_set_x:N  = \l_stex_key_prec_str ,
4110    op          .tl_set:N     = \l_stex_key_op_tl ,
4111    intent      .str_set:N    = \l_stex_key_intent_str ,
4112    argnames    .clist_set:N  = \l_stex_key_intent_args_clist ,
4113    unknown     .code:n       = {
4114      \str_set_eq:NN \l_stex_key_variant_str \l_keys_key_str
4115    }
4116  }{style}
```

\notation

```
4117  \stex_new_stylable_cmd:nnnn {notation} { s m O{} m } {
4118    \stex_keys_set:nn{notation}{#3}
4119    \stex_get_symbol:n{#2}
4120    \stex_notation_parse:n{#4}
4121    \stex_if_check_terms:T{ \_stex_notation_check: }
4122    \_stex_notation_add:
4123    \stex_if_do_html:T {
4124      \def\comp{\_comp}
4125      \_stex_notation_do_html:n{\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
4126    }
4127    \IfBooleanTF#1{
4128      \_stex_notation_set_default:n{
4129        \l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str
4130      }
4131    }{}
4132    \stex_if_smsmode:F{
4133      \group_begin:
4134      \__stex_notations_styledefs:
4135      \stex_style_apply:
4136      \group_end:
4137    }
4138    \stex_smsmode_do:
4139  }{}
4140
4141  \cs_new_protected:Nn \__stex_notations_styledefs: {
4142    \str_set_eq:NN\thisnotationvariant\l_stex_key_variant_str
4143    \str_set:Nn \thisdeclname \l_stex_get_symbol_name_str
4144    \tl_set:Nx \thisdecluri {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
4145    \def\thisnotation{
```

```
4146        $
4147        \tl_set_eq:NN \l_stex_current_symbol_str\thisdecluri
4148        \exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs}{
4149          \_stex_notation_make_args:
4150        }$
4151      }
4152    }
4153
4154 \stex_deactivate_macro:Nn \notation {module~environments}
4155 \stex_every_module:n {\stex_reactivate_macro:N \notation}
4156 \stex_sms_allow_escape:N \notation
```

(*End definition for* \notation. *This function is documented on page* *79.*)

\stex_notation_parse:n   requires the above keys, \l_stex_get_symbol_arity_int, and \l_stex_get_symbol_-
args_tl

```
4157 \cs_new_protected:Nn \stex_notation_parse:n {
4158    \tl_if_empty:NF \l_stex_key_op_tl {
4159      \tl_set:Nx \l_stex_key_op_tl { \exp_not:N\maincomp {
4160        \exp_args:No \exp_not:n \l_stex_key_op_tl
4161      } }
4162    }
4163    \seq_clear:N \l__stex_notations_precs_seq
4164    \tl_clear:N \l_stex_notation_args_tl
4165    \int_compare:nNnTF \l_stex_get_symbol_arity_int = 0 {
4166      \__stex_notations_const_precs:
4167      \tl_if_empty:NT \l_stex_key_op_tl {
4168        \tl_set:Nn \l_stex_key_op_tl { \maincomp{#1} }
4169      }
4170    }{
4171      \__stex_notations_fun_precs:
4172      \str_set:Nn \l__stex_notations_missing_str {#1}
4173      \tl_clear:N \l__stex_notations_missing_tl
4174      \_stex_map_args:N \__stex_notations_add_missing_args:nn
4175      \tl_if_empty:NT \l_stex_key_op_tl {
4176        \hbox_set:Nn \l_tmpa_box {
4177          \str_set:Nn \l_stex_current_symbol_str {}
4178          \cs_set:Npn \l_tmpa_cs ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 ##9 { #1 }
4179          \cs_set:Npn \maincomp ##1 {
4180            \tl_gset:Nn \l_stex_key_op_tl { \maincomp{##1} }
4181            ##1
4182          }
4183          \cs_set:Npn \argsep ##1 ##2 {##1 ##2}
4184          \cs_set:Npn \argmap ##1 ##2 ##3 {##1 ##3}
4185          \cs_set:Npn \argarraymap ##1 ##2 ##3 ##4 {
4186            ##1 ##2
4187          }
4188          \stex_suppress_html:n{$\l_tmpa_cs abcdefghj$}
4189        }
4190      }
4191    }
4192    \exp_args:NNe
4193    \tl_set:Nn \l_stex_notation_macrocode_cs {
4194      \STEXInternalNotation
```

223

```
4195        { \l_stex_key_variant_str }
4196        { \l__stex_notations_opprec_tl }
4197        { \l_stex_key_intent_str }
4198        { \l_stex_notation_args_tl }
4199        {
4200          \int_compare:nNnTF \l_stex_get_symbol_arity_int = 0
4201          { \exp_not:n { \maincomp{ #1 } } }
4202          { \exp_not:n { #1 } \l__stex_notations_missing_tl }
4203        }
4204    }
4205    \stex_debug:nn{notation}{Notation:~\meaning\l_stex_notation_macrocode_cs}
4206 }
4207
4208 \cs_new_protected:Nn \__stex_notations_const_precs: {
4209    \str_if_empty:NTF \l_stex_key_prec_str {
4210      \tl_set:No \l__stex_notations_opprec_tl { \neginfprec }
4211    }{
4212      \str_if_eq:onTF \l_stex_key_prec_str {nobrackets}{
4213        \tl_set:No \l__stex_notations_opprec_tl { \neginfprec }
4214      }{
4215        \tl_set_eq:NN \l__stex_notations_opprec_tl \l_stex_key_prec_str
4216      }
4217    }
4218 }
4219
4220 \cs_new_protected:Nn \__stex_notations_fun_precs: {
4221    \str_if_empty:NTF \l_stex_key_prec_str {
4222      \tl_set:No \l__stex_notations_opprec_tl { \neginfprec }
4223    }{
4224      \str_if_eq:onTF \l_stex_key_prec_str {nobrackets}{
4225        \tl_set:No \l__stex_notations_opprec_tl { \neginfprec }
4226      }{
4227        \tl_set_eq:NN \l__stex_notations_opprec_tl \l_stex_key_prec_str
4228      }
4229    }
4230    \str_if_empty:NTF \l_stex_key_prec_str {
4231      \tl_set:Nn \l__stex_notations_opprec_tl { 0 }
4232      \int_step_inline:nn \l_stex_get_symbol_arity_int {
4233        \seq_put_right:Nn \l__stex_notations_precs_seq {0}
4234      }
4235    }{
4236      \str_if_eq:onTF \l_stex_key_prec_str {nobrackets}{
4237        \stex_debug:nn{notation}{No~brackets}
4238        \tl_set:No \l__stex_notations_opprec_tl { \neginfprec }
4239        \int_step_inline:nn \l_stex_get_symbol_arity_int {
4240          \exp_args:NNo \seq_put_right:Nn \l__stex_notations_precs_seq \infprec
4241        }
4242      }\__stex_notations_parse_precs:
4243    }
4244    \__stex_notations_do_argnames:
4245 }
4246
4247 \cs_new_protected:Nn \__stex_notations_parse_precs: {
4248    \stex_debug:nn{notation}{parsing~precedence~\l_stex_key_prec_str}
```

```
4249    \seq_set_split:NnV \l__stex_notations_seq ; \l_stex_key_prec_str
4250    \seq_pop_left:NNTF \l__stex_notations_seq \l__stex_notations_str {
4251      \tl_set_eq:NN \l__stex_notations_opprec_tl \l__stex_notations_str
4252      \seq_pop_left:NNT \l__stex_notations_seq \l__stex_notations_str {
4253        \exp_args:NNo \seq_set_split:NnV \l__stex_notations_seq
4254          {\tl_to_str:n{x}} \l__stex_notations_str
4255      }
4256    }{
4257      \tl_set:No \l__stex_notations_opprec_tl { 0 }
4258    }
4259    \int_step_inline:nn \l_stex_get_symbol_arity_int {
4260      \seq_pop_left:NNTF \l__stex_notations_seq \l__stex_notations_str {
4261        \seq_put_right:No \l__stex_notations_precs_seq \l__stex_notations_str
4262      }{
4263        \seq_put_right:No \l__stex_notations_precs_seq \l__stex_notations_opprec_tl
4264      }
4265    }
4266  }
4267
4268  \cs_new_protected:Nn \__stex_notations_do_argnames: {
4269    \tl_clear:N \l_stex_notation_args_tl
4270    \_stex_map_args:N \__stex_notations_do_argname:nn
4271  }
4272
4273  \cs_new_protected:Nn \__stex_notations_do_argname:nn {
4274    \clist_if_empty:NTF \l_stex_key_intent_args_clist {
4275      \tl_put_right:Nx \l_stex_notation_args_tl {
4276        #1#2{\seq_item:Nn \l__stex_notations_precs_seq #1}{
4277          \str_if_empty:NF \l_stex_key_intent_str {#1}
4278        }
4279      }
4280    }{
4281      \tl_put_right:Nx \l_stex_notation_args_tl {
4282        #1#2{\seq_item:Nn \l__stex_notations_precs_seq #1}
4283        {\c_dollar_str\clist_item:Nn \l_stex_key_intent_args_clist 1}
4284      }
4285      \clist_pop:NN \l_stex_key_intent_args_clist \l_tmpa_tl
4286    }
4287  }
4288
4289  \cs_new:Nn \__stex_notations_add_missing_args:nn {
4290    \exp_args:NNe \str_if_in:NnF \l__stex_notations_missing_str {\c_hash_str\c_hash_str#1}{
4291      \tl_put_right:Nn \l__stex_notations_missing_tl{\STEXinvisible{## #1}}
4292    }
4293  }
```

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

```
\_stex_notation_check:       4294  \cs_new_protected:Nn \_stex_notation_check: {
\_stex_notation_add:         4295    \stex_check_term:n{
\_stex_notation_do_html:n    4296      \str_set:Nn \l_stex_current_symbol_str {test}
\_stex_notation_make_args:   4297      \cs_set:Npn \comp ##1 {##1}
                             4298      \stex_debug:nn{check_terms}{Checking~notation...}
```

225

```
4299        \exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs{}}{
4300            \_stex_notation_make_args:
4301        }
4302    }
4303 }
4304
4305 \cs_new_protected:Nn \_stex_notation_add: {
4306    \stex_module_add_notation:eoeoo{
4307        \l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str
4308        }\l_stex_key_variant_str
4309        {\int_use:N \l_stex_get_symbol_arity_int}
4310        \l_stex_notation_macrocode_cs
4311        \l_stex_key_op_tl
4312 }
4313
4314 \cs_new_protected:Nn \_stex_notation_do_html:n {
4315    \hbox{\stex_annotate_invisible:nn {
4316        shtml:notation={#1},
4317        shtml:notationfragment={\l_stex_key_variant_str},
4318        shtml:precedence={\l__stex_notations_opprec_tl},
4319        shtml:argprecs={\seq_use:Nn \l__stex_notations_precs_seq ,}
4320    }{
4321        \cs_set_protected:Npn \argsep ##1 ##2 {
4322            \stex_annotate:nn{shtml:argsep={}}{
4323                ##1 ##2
4324            }
4325        }
4326        \cs_set_protected:Npn \argmap ##1 ##2 ##3 {
4327            \cs_set:Npn \__stex_notations_map_cs: ####1 { ##2 }
4328            \stex_annotate:nn{shtml:argmap={}}{
4329                \__stex_notations_map_cs:{##1} ##3
4330            }
4331        }
4332        \cs_set_protected:Npn \maincomp {
4333            \_do_comp:nNn {maincomp}\compemph@uri
4334        }
4335        $
4336            \str_set:Nx \l_stex_current_symbol_str {#1}
4337            \stex_annotate:nn{shtml:notationcomp={}}{
4338                \exp_args:Nne \use:nn {
4339                    \l_stex_notation_macrocode_cs {}
4340                }{
4341                    \_stex_map_args:N \__stex_notations_make_arg_html:nn
4342                }
4343            }
4344        $
4345        \tl_if_empty:NF \l_stex_key_op_tl {
4346            $
4347            \str_set:Nx \l_stex_current_symbol_str {#1}
4348            \stex_annotate:nn{shtml:notationopcomp={}}{
4349                \_stex_term_oms:nnn{\l_stex_key_variant_str}{}{\l_stex_key_op_tl}
4350            }
4351            $
4352        }
```

226

```
4353        }}
4354    }
4355
4356    \cs_new:Nn \__stex_notations_make_arg_html:nn {
4357 %   \str_case:nnF #2 {
4358 %      a {{
4359 %         \stex_annotate:nn{shtml:argnum=#1a}{x},
4360 %         \stex_annotate:nn{shtml:argnum=#1b}{x}
4361 %      }}
4362 %      B {{
4363 %         \stex_annotate:nn{shtml:argnum=#1a}{x},
4364 %         \stex_annotate:nn{shtml:argnum=#1b}{x}
4365 %      }}
4366 %   }{
4367        {
4368           \stex_annotate:nn{shtml:argnum=#1}{x}
4369        }
4370 %   }
4371    }
4372
4373    \cs_new:Nn \_stex_notation_make_args: {
4374      \_stex_map_notation_args:N \__stex_notations_make_arg:nnnn
4375    }
4376
4377
4378    \cs_new:Nn \__stex_notations_make_arg:nnnn  {
4379      \str_case:nnF #2 {
4380        a {{
4381           a\c_math_subscript_token{#1,1},
4382           a\c_math_subscript_token{#1,2}
4383        }}
4384        B {{
4385           B\c_math_subscript_token{#1,1},
4386           B\c_math_subscript_token{#1,2}
4387        }}
4388      }{
4389        \_stex_term_arg:nnnnn{#1}{#2}{#3}{#4}
4390          {{#2}\c_math_subscript_token{#1}}
4391      }
4392    }
```

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

\setnotation
_stex_notation_set_default:n

```
4393    \cs_new_protected:Npn \setnotation #1 #2 {
4394      \stex_get_symbol:n{#1}
4395      \cs_if_exist:cTF{l_stex_notation_
4396        \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
4397        _#2_cs
4398      }{
4399        \tl_set_eq:Nc \l_stex_notation_macrocode_cs {l_stex_notation_
4400          \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
4401          _#2_cs
4402        }
```

227

```
4403        \cs_if_exist:cTF{l_stex_notation_
4404          \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
4405          _op_#2_cs
4406        }{
4407          \tl_set_eq:Nc \l_stex_key_op_tl {l_stex_notation_
4408            \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
4409            _op_#2_cs
4410          }
4411        }{
4412          \tl_clear:N \l_stex_key_op_tl
4413        }
4414        \_stex_notation_set_default:n{
4415          \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
4416        }
4417      }{
4418        \msg_error:nnxx{stex}{unknownnotation}{#2}{
4419          \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
4420        }
4421      }
4422    }
4423
4424  \cs_new_protected:Nn \_stex_notation_set_default:n{
4425    \stex_module_add_notation:eoeoo{#1}{}
4426      {\int_use:N \l_stex_get_symbol_arity_int}
4427      \l_stex_notation_macrocode_cs
4428      \l_stex_key_op_tl
4429  }
```

(*End definition for* \setnotation *and* _stex_notation_set_default:n. *These functions are documented on page 80.*)

\varnotation

```
4430  \stex_new_stylable_cmd:nnnn {varnotation} { s m O{} m} {
4431    \stex_keys_set:nn{notation}{#3}
4432    \stex_get_var:n{#2}
4433    \str_set_eq:NN \l_stex_key_name_str \l_stex_get_symbol_name_str
4434    \stex_notation_parse:n{#4}
4435    \stex_if_check_terms:T{ \_stex_notation_check: }
4436    \_stex_vardecl_notation_macro:
4437    \IfBooleanTF#1{
4438      \_stex_notation_set_default:n{\l_stex_get_symbol_name_str}
4439    }{}
4440    \group_begin:
4441    \tl_set_eq:NN \thisvarname \l_stex_get_symbol_name_str
4442    \tl_clear:N \thisstyle
4443    \str_set_eq:NN\thisnotationvariant\l_stex_key_variant_str
4444    \def\thisnotation{
4445      $\let\l_stex_current_symbol_str\thisvarname
4446        \def\comp{\_varcomp}\exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs{}}{
4447        \_stex_notation_make_args:
4448      }$
4449    }
4450    \stex_style_apply:
4451    \group_end:
4452  }{}
```

228

(End definition for `\varnotation`*. This function is documented on page 84.)*

`\symdef`

```
4453 \stex_keys_define:nnnn{symdef}{}{}{decl,notation}
4454
4455 \cs_new_protected:Nn \_stex_symdef_styledefs: {
4456   \tl_set:Nx \thisdecluri {\l_stex_current_module_str ? \l_stex_key_name_str}
4457   \tl_set_eq:NN \thisdeclname \l_stex_key_name_str
4458   \tl_set_eq:NN \thistype \l_stex_key_type_tl
4459   \tl_set_eq:NN \thisdefiniens \l_stex_key_def_tl
4460   \tl_set_eq:NN \thisargs \l_stex_key_args_str
4461   \tl_clear:N \thisstyle
4462   \str_set_eq:NN\thisnotationvariant\l_stex_key_variant_str
4463   \def\thisnotation{
4464     $\let\l_stex_current_symbol_str\thisdecluri
4465       \def\comp{\_comp}\exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs}{
4466       \_stex_notation_make_args:
4467     }$
4468   }
4469 }
4470
4471 \stex_new_stylable_cmd:nnnn {symdef} { m O{} m} {
4472   \stex_keys_set:nn{symdef}{#2}
4473   \str_set:Nx \l_stex_macroname_str { #1 }
4474   \stex_symdecl_top:n{#1}
4475   \stex_debug:nn{symdef}{Doing~\l_stex_current_module_str ? \l_stex_key_name_str}
4476   \str_set_eq:NN \l_stex_get_symbol_mod_str \l_stex_current_module_str
4477   \str_set_eq:NN \l_stex_get_symbol_name_str \l_stex_key_name_str
4478   \stex_notation_parse:n{#3}
4479   \stex_debug:nn{Here!}{\meaning\l_stex_notation_args_tl}
4480   \_stex_notation_check:
4481   \_stex_notation_add:
4482   \stex_if_do_html:T{
4483     \_stex_notation_do_html:n{\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
4484   }
4485   \stex_if_smsmode:F{
4486     \group_begin:
4487     \_stex_symdef_styledefs:
4488     \stex_style_apply:
4489     \group_end:
4490   }
4491   \stex_smsmode_do:
4492 }{}
4493
4494 \stex_deactivate_macro:Nn \symdef {module~environments}
4495 \stex_every_module:n {\stex_reactivate_macro:N \symdef}
4496 \stex_sms_allow_escape:N \symdef
```

(End definition for `\symdef`*. This function is documented on page 77.)*

`\stex_do_default_notation_op:`

```
4497 \cs_new_protected:Nn \stex_do_default_notation: {
4498   \stex_do_default_notation_op:
4499   \tl_if_empty:NTF \l_stex_current_args_tl {
```

```
4500        \tl_clear:N \l__stex_notations_args_tl
4501     }{
4502        \__stex_notations_make_name:
4503        \tl_set_eq:NN \l_stex_get_symbol_args_tl \l_stex_current_args_tl
4504        \tl_set:Nx \l__stex_notations_args_tl {
4505          \_stex_map_args:N \__stex_notations_augment_arg:nn
4506        }
4507        \tl_put_right:Nn \l_stex_default_notation {\comp(}
4508        \seq_clear:N \l_tmpa_seq
4509        \int_step_inline:nn \l_stex_current_arity_str {
4510          \seq_put_right:Nn \l_tmpa_seq {#### ##1}
4511        }
4512        \tl_put_right:Nx \l_stex_default_notation {
4513          \seq_use:Nn \l_tmpa_seq {\mathpunct{\comp{,}}}
4514        }
4515        \tl_put_right:Nn \l_stex_default_notation {\comp)}
4516     }
4517     \tl_set:Nx \l_stex_default_notation {\STEXInternalNotation{}{0}{}{\l__stex_notations_args_
4518        \exp_args:No \exp_not:n \l_stex_default_notation
4519     }}
4520  }
4521
4522  \cs_new:Nn \__stex_notations_augment_arg:nn {
4523     #1#2{0}{}
4524  }
4525
4526  \cs_new_protected:Nn \__stex_notations_make_name: {
4527     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq ? \l_stex_current_symbol_str
4528     \seq_pop_right:NN \l_tmpa_seq \l__stex_notations_name_str
4529     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq / \l__stex_notations_name_str
4530     \seq_pop_right:NN \l_tmpa_seq \l__stex_notations_name_str
4531  }
4532
4533  \cs_new_protected:Nn \stex_do_default_notation_op: {
4534     \__stex_notations_make_name:
4535     \tl_set:Nx \l_stex_default_notation {\exp_not:N \maincomp{ \exp_not:N \mathrm {\l__stex_no
4536  }
```

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

```
4537  % 1: variant 2: operator precedence 3: intent 4: arguments 5: code 6: next
4538
4539  \cs_new_protected:Npn \STEXInternalNotation #1 #2 #3 #4 #5 #6 {
4540     \__stex_notations_process_notation:nnnnnn{#1}{#2}{#3}{#4}{#5}{
4541        \l__stex_notations_code_tl
4542        #6
4543     }
4544  }
4545
4546  \cs_new_protected:Npn \__stex_notations_process_notation:nnnnnn #1 #2 #3 #4 {
4547     \tl_if_empty:nTF{#4}{
4548        \__stex_notations_simple:nnnnn{#1}{#2}{#3}
4549     }{
```

```
4550        \__stex_notations_complex:nnnnnn{#1}{#2}{#3}{#4}
4551      }
4552  }
4553
4554  \cs_new_protected:Nn \__stex_notations_simple:nnnnn {
4555    \stex_debug:nn{Notation~code}{\tl_to_str:n{#4}}
4556    \tl_set:Nn \l__stex_notations_code_tl {
4557      \cs_set:Npn \l__stex_notations_cs {
4558        \_stex_maybe_brackets:nn{#2}{
4559          \_stex_term_oms_or_omv:nnn{#1}{#3}{#4}
4560        }
4561      }
4562      \l__stex_notations_cs
4563    }
4564    \stex_debug:nn{Here:Notation}{\meaning \l__stex_notations_code_tl }
4565    #5
4566  }
4567
4568  \cs_new_protected:Nn \__stex_notations_complex:nnnnnn {
4569    \stex_debug:nn{Notation~code}{\tl_to_str:n{#5}}
4570    \int_zero:N \l_tmpa_int
4571    \tl_set:Nn \l__stex_notations_pre_tl {\cs_set_eq:NN \_stex_term_oma_or_omb:nnn \_stex_term
4572    \tl_set:Nn \l__stex_notations_code_tl {
4573      \cs_generate_from_arg_count:NNnn \l__stex_notations_cs \cs_set:Npn \l_tmpa_int
4574      {
4575        \_stex_maybe_brackets:nn{#2}{
4576          \_stex_term_oma_or_omb:nnn{#1}{#3}{
4577            \bool_set_false:N \l_stex_brackets_dones_bool
4578            #5
4579          }
4580        }
4581      }
4582      \l__stex_notations_cs
4583    }
4584    \tl_set:Nn \l__stex_notations_after_tl{
4585      \exp_args:NNo
4586      \tl_put_left:Nn \l__stex_notations_code_tl \l__stex_notations_pre_tl
4587      \tl_put_left:Nx \l__stex_notations_code_tl {
4588        \int_set:Nn \l_tmpa_int {\int_use:N \l_tmpa_int}
4589      }
4590      \stex_debug:nn{Here:Notation}{\meaning \l__stex_notations_code_tl }
4591      #6
4592    }
4593    \__stex_notations_parse_notation_args:nnnnw #4 \__stex_notations_args_end:
4594  }
4595
4596  \cs_new_protected:Npn \__stex_notations_parse_notation_args:nnnnw #1 #2 #3 #4 #5 \__stex_not
4597    \tl_if_empty:nTF{#5}{
4598      \__stex_notations_add_last:nnnnn{#1}{#2}{#3}{#4}
4599    }{
4600      \__stex_notations_add_next:nnnnnn{#1}{#2}{#3}{#4}{#5}
4601    }
4602  }
4603
```

231

```
4604 \cs_new_protected:Nn \__stex_notations_add_next:nnnnnn {
4605   \__stex_notations_add:nnnnn{#1}{#2}{#3}{#4}{#6}
4606   \__stex_notations_parse_notation_args:nnnnw #5 \__stex_notations_args_end:
4607 }
4608
4609 \cs_new_protected:Nn \__stex_notations_add_last:nnnnn {
4610   \__stex_notations_add:nnnnn{#1}{#2}{#3}{#4}{#5}
4611   \l__stex_notations_after_tl
4612 }
4613
4614 \cs_new_protected:Nn \__stex_notations_add:nnnnn {
4615   \int_incr:N \l_tmpa_int
4616   \str_case:nn{#2}{
4617     i {
4618       \tl_put_right:Nn \l__stex_notations_code_tl {
4619         {\_stex_term_arg:nnnnn{#1}{#2}{#3}{#4}{#5}}
4620       }
4621     }
4622     b {
4623       \tl_set:Nn \l__stex_notations_pre_tl {
4624         \cs_set_eq:NN \_stex_term_oma_or_omb:nnn \_stex_term_omb:nnn
4625       }
4626       \tl_put_right:Nn \l__stex_notations_code_tl {
4627         {\_stex_term_arg:nnnnn{#1}{#2}{#3}{#4}{#5}}
4628       }
4629     }
4630     a {
4631       \tl_put_right:Nn \l__stex_notations_code_tl {
4632         {\_stex_term_arg_aB:nnnnn{#1}{#2}{#3}{#4}{#5}}
4633       }
4634     }
4635     B  {
4636       \tl_set:Nn \l__stex_notations_pre_tl {
4637         \cs_set_eq:NN \_stex_term_oma_or_omb:nnn \_stex_term_omb:nnn
4638       }
4639       \tl_put_right:Nn \l__stex_notations_code_tl {
4640         {\_stex_term_arg_aB:nnnnn{#1}{#2}{#3}{#4}{#5}}
4641       }
4642     }
4643   }
4644 }
```

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

a/B-mode argument handling

\argsep

```
4645 \cs_new_protected:Nn \__stex_notations_check_aB_arg:Nn {
4646   \exp_args:Ne \cs_if_eq:NNF {\tl_head:n{#2}}
4647     \_stex_term_arg_aB:nnnnn {
4648     \msg_error:nnx{stex}{error/assocarg}{\tl_to_str:n{#1}}
4649   }
4650 }
4651
```

```
4652 \cs_new_protected:Npn \argsep #1 #2 {
4653   \__stex_notations_check_aB_arg:Nn\argsep{#1}
4654   \stex_pseudogroup_with:nn{\_stex_term_do_aB_clist:}{
4655     \tl_set:Nn \_stex_term_do_aB_clist: {
4656       \seq_use:Nn \l_stex_aB_args_seq {#2}
4657     }
4658     #1
4659   }
4660 }
```

(*End definition for* \argsep. *This function is documented on page 81.*)

```
4661 \cs_new_protected:Npn \argmap #1 #2 #3 {
4662   \__stex_notations_check_aB_arg:Nn\argmap{#1}
4663   \stex_pseudogroup_with:nn{
4664     \_stex_term_do_aB_clist:
4665     \__stex_notations_map_cs:
4666   }{
4667     \cs_set:Npn \__stex_notations_map_cs: ##1 { #2 }
4668     \tl_set:Nn \_stex_term_do_aB_clist: {
4669       \seq_clear:N \l_tmpa_seq
4670       \seq_map_inline:Nn \l_stex_aB_args_seq {
4671         \tl_if_eq:nnTF{##1}{\ellipses}{
4672           \seq_put_right:Nn \l_tmpa_seq \ellipses
4673         }{
4674           \seq_put_right:Nx \l_tmpa_seq {
4675             \exp_after:wN \exp_not:n \exp_after:wN { \__stex_notations_map_cs: {##1} }
4676           }
4677         }
4678       }
4679       \seq_set_eq:NN \l_stex_aB_args_seq \l_tmpa_seq
4680       \seq_use:Nn \l_stex_aB_args_seq {#3}
4681     }
4682     #1
4683   }
4684 }
```

(*End definition for* \argmap. *This function is documented on page 81.*)

```
4685 \int_new:N \l__stex_notations_clist_count_int
4686 \cs_new_protected:Npn \argarraymap #1 #2 #3 #4 {
4687   \__stex_notations_check_aB_arg:Nn\argarraymap{#1}
4688   \stex_pseudogroup_with:nn{
4689     \_stex_term_do_aB_clist:
4690     \__stex_notations_map_cs:
4691   }{
4692     \cs_set:Npn \__stex_notations_map_cs: ##1 { #3 }
4693     \int_set:Nn \l__stex_notations_clist_count_int {\exp_args:No\clist_count:n{\tl_to_str:n{
4694     \tl_set:Nn \_stex_term_do_aB_clist: {
4695       \tl_clear:N \l_tmpa_tl
4696       \int_zero:N \l_tmpa_int
4697       \seq_map_inline:Nn \l_stex_aB_args_seq {
4698         \int_incr:N \l_tmpa_int
```

233

```
4699            \int_compare:nNnT \l_tmpa_int > \l__stex_notations_clist_count_int {
4700              \int_set:Nn \l_tmpa_int 1
4701            }
4702            \tl_put_right:Nx \l_tmpa_tl {
4703              \exp_after:wN \exp_not:n \exp_after:wN { \__stex_notations_map_cs: {##1} }
4704              \clist_item:nn{#4}\l_tmpa_int
4705            }
4706          }
4707          \seq_set_eq:NN \l_stex_aB_args_seq \l_tmpa_seq
4708          \begin{array}{#2}
4709              \l_tmpa_tl
4710          \end{array}
4711        }
4712        #1
4713      }
4714    }
```

(*End definition for* \argarraymap. *This function is documented on page* *81*.)

13.8.3   Variables

4715   ⟨@@=stex_vars⟩

\vardef

```
4716  \tl_new:N \l_stex_variables_prop
4717  \bool_new:N \l__stex_vars_bind_bool
4718  \cs_new_protected:Nn \_stex_variable:nnnnnnnN {}
4719  \stex_keys_define:nnnn{vardef}{
4720    \bool_set_false:N \l__stex_vars_bind_bool
4721  }{
4722    bind  .bool_set:N  = \l__stex_vars_bind_bool
4723  }{symdef}
4724
4725  \stex_new_stylable_cmd:nnnn {vardef} { m O{} m } {
4726    \stex_keys_set:nn{vardef}{#2}
4727    \str_set:Nx \l_stex_macroname_str { #1 }
4728    \str_if_empty:NT \l_stex_key_name_str {
4729      \str_set:Nx \l_stex_key_name_str { #1 }
4730    }
4731
4732    \stex_symdecl_do:
4733    \_stex_symdecl_check_terms:
4734    \__stex_vars_add:
4735    \__stex_vars_macro:
4736    \stex_if_do_html:T \__stex_vars_html:
4737
4738    \int_set:Nn \l_stex_get_symbol_arity_int {\l_stex_get_symbol_arity_int}
4739    \stex_debug:nn{vardef}{Doing~\l_stex_key_name_str}
4740    \tl_set_eq:NN \l_stex_get_symbol_return_tl \l_stex_key_return_tl
4741    \stex_notation_parse:n{#3}
4742    \stex_if_check_terms:T{ \_stex_notation_check: }
4743    \_stex_vardecl_notation_macro:
4744    \stex_if_do_html:T {
4745      \def\comp{\_varcomp}
4746      \_stex_notation_do_html:n \l_stex_key_name_str
```

234

```
4747      }
4748    \group_begin:
4749    \tl_set_eq:NN \thisvarname \l_stex_key_name_str
4750    \tl_clear:N \thisstyle
4751    \str_set_eq:NN\thisnotationvariant\l_stex_key_variant_str
4752    \def\thisnotation{
4753      $\let\l_stex_current_symbol_str\thisvarname
4754        \def\comp{\_varcomp}\exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs{}}{
4755        \_stex_notation_make_args:
4756      }$
4757    }
4758    \stex_style_apply:
4759    \group_end:\ignorespaces
4760 }{}
4761
4762 \cs_new_protected:Nn \__stex_vars_add: {
4763    \exp_args:NNNo \exp_args:NNnx
4764    \prop_put:Nnn \l_stex_variables_prop \l_stex_key_name_str {
4765      {\l_stex_macroname_str}
4766      {\l_stex_key_name_str}
4767      {\int_use:N \l_stex_get_symbol_arity_int}
4768      {\l_stex_get_symbol_args_tl}
4769      {\exp_args:No \exp_not:n \l_stex_key_def_tl}
4770      {\exp_args:No \exp_not:n \l_stex_key_type_tl}
4771      {\exp_args:No \exp_not:n \l_stex_key_return_tl}
4772      \stex_invoke_symbol:
4773    }
4774 }
4775
4776 \cs_new_protected:Nn \__stex_vars_macro: {
4777    \tl_set:cx{\l_stex_macroname_str}{
4778      \_stex_invoke_variable:nnnnnnN
4779        {\l_stex_key_name_str}
4780        {\int_use:N \l_stex_get_symbol_arity_int}
4781        {\l_stex_get_symbol_args_tl}
4782        {\exp_args:No \exp_not:n \l_stex_key_def_tl}
4783        {\exp_args:No \exp_not:n \l_stex_key_type_tl}
4784        {\exp_args:No \exp_not:n \l_stex_key_return_tl}
4785        \stex_invoke_symbol:
4786    }
4787 }
4788
4789 \cs_new_protected:Nn \__stex_vars_html: {
4790    \stex_if_do_html:T {
4791      \hbox\bgroup\exp_args:Ne \stex_annotate_invisible:nn {
4792        shtml:vardef = {\l_stex_key_name_str},
4793        shtml:args = {\l_stex_key_args_str}
4794        \str_if_empty:NF \l_stex_macroname_str {,
4795          shtml:macroname={\l_stex_macroname_str}
4796        }
4797        \str_if_empty:NF \l_stex_key_assoc_str {,
4798          shtml:assoctype={\l_stex_key_assoc_str}
4799        }
4800        \str_if_empty:NF \l_stex_key_role_str {,
```

```
4801          shtml:role={\l_stex_key_role_str}
4802        }
4803      \str_if_empty:NF \l_stex_key_reorder_str {,
4804          shtml:reorderargs={\l_stex_key_reorder_str}
4805      }
4806      \bool_if:NT \l__stex_vars_bind_bool {,
4807          shtml:bind={}
4808      }
4809    }{
4810      \_stex_annotate_force_break:n{
4811        \bool_set_true:N \stex_in_invisible_html_bool
4812        \tl_if_empty:NF \l_stex_key_type_tl {
4813          \stex_annotate:nn{shtml:type={}}{$\l_stex_key_type_tl$}
4814        }
4815        \tl_if_empty:NF \l_stex_key_def_tl {
4816          \stex_annotate:nn{shtml:definiens={}}{$\l_stex_key_def_tl$}
4817        }
4818        \tl_if_empty:NF \l_stex_key_return_tl{
4819          \exp_args:Nno \use:n{
4820          \cs_generate_from_arg_count:NNnn \l__stex_vars_cs
4821          \cs_set:Npn \l_stex_get_symbol_arity_int} \l_stex_key_return_tl
4822          \tl_set:Nx \l__stex_vars_args_tl {\_stex_map_args:N \_stex_return_args:nn}
4823          $\stex_annotate:nn{shtml:returntype={}}{
4824            \exp_after:wN \l__stex_vars_cs \l__stex_vars_args_tl!}$
4825        }
4826        \tl_if_empty:NF \l_stex_key_argtypes_clist {
4827          \stex_annotate:nn{shtml:argtypes={}}{
4828            \_stex_annotate_force_break:n{
4829              \clist_map_inline:Nn \l_stex_key_argtypes_clist {
4830                $\stex_annotate:nn{shtml:type={}}{##1}$
4831              }
4832            }
4833          }
4834        }
4835      }
4836    }\egroup
4837  }
4838 }
```

(*End definition for* \vardef. *This function is documented on page* *84.*)

```
4839 \cs_new_protected:Nn \_stex_vardecl_notation_macro: {
4840   \tl_set_eq:cN {l_stex_notation_
4841     \l_stex_key_name_str _
4842     \l_stex_key_variant_str _cs
4843   }\l_stex_notation_macrocode_cs
4844   \cs_if_exist:cF {l_stex_notation_\l_stex_key_name_str __cs}{
4845     \tl_set_eq:cN{l_stex_notation_\l_stex_key_name_str __cs}
4846       \l_stex_notation_macrocode_cs
4847   }
4848   \tl_if_empty:NF \l_stex_key_op_tl {
4849     \tl_set_eq:cN {l_stex_notation_\l_stex_key_name_str _op_
4850     \l_stex_key_variant_str _cs}\l_stex_key_op_tl
```

236

```
4851       \cs_if_exist:cF {l_stex_notation_\l_stex_key_name_str _op__cs}{
4852          \cs_set_eq:cN{l_stex_notation_\l_stex_key_name_str _op__cs}
4853          \l_stex_key_op_tl
4854       }
4855    }
4856 }
```

(*End definition for* _stex_vardecl_notation_macro:. *This function is documented on page 114.*)

```
4857 \cs_new_protected:Nn \__stex_vars_set_vars:nnnnnnN {
4858    \stex_debug:nn{symbols}{Variable~#1~found}
4859    \cs_set:Npn \_stex_variable:nnnnnnnN ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 {}
4860    \str_clear:N \l_stex_get_symbol_mod_str
4861    \str_set:Nn \l_stex_get_symbol_name_str {#1}
4862    \int_set:Nn \l_stex_get_symbol_arity_int {#2}
4863    \tl_set:Nn \l_stex_get_symbol_args_tl {#3}
4864    \tl_set:Nn \l_stex_get_symbol_def_tl {#4}
4865    \tl_set:Nn \l_stex_get_symbol_type_tl {#5}
4866    \tl_set:Nn \l_stex_get_symbol_return_tl {#6}
4867    \tl_set:Nn \l_stex_get_symbol_invoke_cs {#7}
4868 }
4869
4870 \cs_new_protected:Nn \__stex_vars_get_var:n {
4871    \prop_map_inline:Nn \l_stex_variables_prop {
4872       \__stex_vars_check_var:nnnnnnnnN {#1} ##2
4873    }
4874 }
4875
4876 \cs_new_protected:Nn \__stex_vars_check_var:nnnnnnnnN {
4877    \str_if_eq:nnTF{#1}{#2}{
4878       \prop_map_break:n{\__stex_vars_set_vars:nnnnnnN {#3}{#4}{#5}{#6}{#7}{#8}#9}
4879    }{
4880       \str_if_eq:nnT{#1}{#3}{
4881          \prop_map_break:n{\__stex_vars_set_vars:nnnnnnN {#3}{#4}{#5}{#6}{#7}{#8}#9}
4882       }
4883    }
4884 }
4885
4886 \cs_new_protected:Nn \stex_get_var:n {
4887    \str_clear:N \l_stex_get_symbol_name_str
4888    \__stex_vars_get_var:n{#1}
4889    \str_if_empty:NT \l_stex_get_symbol_name_str {
4890       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
4891    }
4892 }
4893
4894 \cs_new_protected:Nn \stex_get_symbol_or_var:n {
4895    \str_clear:N \l_stex_get_symbol_name_str
4896    \__stex_vars_get_var:n{#1}
4897    \str_if_empty:NT \l_stex_get_symbol_name_str {
4898       \stex_debug:nn{symbols}{No~variable~#1~found}
4899       \stex_get_symbol:n{#1}
4900    }
4901 }
```

(End definition for `\stex_get_symbol_or_var:n` *and* `\stex_get_var:n`*. These functions are documented on page [114](#).)*

\svar

```
4902 \NewDocumentCommand \svar {O{} m}{
4903   \group_begin:
4904     \tl_if_empty:nTF{#1}{
4905       \str_set:Nn \l_stex_current_symbol_str {#2}
4906     }{
4907       \str_set:Nn \l_stex_current_symbol_str {#1}
4908     }
4909     \bool_if:NTF \l_stex_allow_semantic_bool{
4910         \tl_clear:N \l_stex_current_term_tl
4911         \_stex_term_omv:nnn{}{}{\_varcomp{#2}}
4912     }{
4913       \msg_error:nnxx{stex}{error/notallowed}{Variable}{\l_stex_current_symbol_str}
4914     }
4915   \group_end:
4916 }
```

(End definition for `\svar`*. This function is documented on page [84](#).)*

13.8.4   Sequences

```
4917 ⟨@@=stex_seqs⟩
```

\varseq

```
4918 \stex_new_stylable_cmd:nnnn {varseq}{m O{} m m} {
4919   \stex_keys_set:nn{symdef}{#2}
4920   \str_set:Nx \l_stex_macroname_str { #1 }
4921   \str_if_empty:NT \l_stex_key_name_str {
4922     \str_set:Nx \l_stex_key_name_str { #1 }
4923   }
4924   \str_if_empty:NT \l_stex_key_args_str {
4925     \str_set:Nn \l_stex_key_args_str {1}
4926   }
4927   \stex_symdecl_do:
4928
4929   \tl_set_eq:NN \l_stex_get_symbol_return_tl \l_stex_key_return_tl
4930   \clist_set:Nn \l__stex_seqs_range_clist {#3}
4931   \tl_if_empty:NTF \l_stex_key_op_tl {
4932     \stex_notation_parse:n{#4}
4933     \tl_clear:N \l_stex_key_op_tl
4934   }{
4935     \stex_notation_parse:n{#4}
4936   }
4937   \stex_if_do_html:T \__stex_seqs_html:
4938   \stex_if_check_terms:T \__stex_seqs_check_terms:
4939   \__stex_seqs_add:
4940   \__stex_seqs_macro:
4941   \stex_if_check_terms:T \_stex_notation_check:
4942   \_stex_vardecl_notation_macro:
4943   \group_begin:
4944   \tl_set_eq:NN \thisvarname \l_stex_key_name_str
4945   \tl_clear:N \thisstyle
```

```
4946    \str_set_eq:NN\thisnotationvariant\l_stex_key_variant_str
4947    \def\thisnotation{
4948      $\let\l_stex_current_symbol_str\thisvarname
4949        \def\comp{\_varcomp}\exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs{}}{
4950        \__stex_seqs_make_args:
4951      }$
4952    }
4953    \stex_style_apply:
4954    \group_end:\ignorespaces
4955 }{}
4956
4957 \cs_new_protected:Nn \__stex_seqs_add: {
4958    \exp_args:NNNo \exp_args:NNnx
4959    \prop_put:Nnn \l_stex_variables_prop \l_stex_key_name_str {
4960      {\l_stex_macroname_str}
4961      {\l_stex_key_name_str}
4962      {\int_use:N \l_stex_get_symbol_arity_int}
4963      {\l_stex_get_symbol_args_tl}
4964      {\exp_args:No \exp_not:n \l_stex_key_def_tl}
4965      {\exp_args:No \exp_not:n \l__stex_seqs_range_clist}
4966      {\exp_args:No \exp_not:n \l_stex_key_return_tl}
4967      \stex_invoke_sequence:
4968    }
4969 }
4970
4971 \cs_new_protected:Nn \__stex_seqs_macro: {
4972    \tl_set:cx{\l_stex_macroname_str}{
4973      \_stex_invoke_variable:nnnnnnN
4974        {\l_stex_key_name_str}
4975        {\int_use:N \l_stex_get_symbol_arity_int}
4976        {\l_stex_get_symbol_args_tl}
4977        {\exp_args:No \exp_not:n \l_stex_key_def_tl}
4978        {\exp_args:No \exp_not:n \l__stex_seqs_range_clist}
4979        {\exp_args:No \exp_not:n \l_stex_key_return_tl}
4980        \stex_invoke_sequence:
4981    }
4982 }
4983
4984 \cs_new_protected:Nn \__stex_seqs_make_args: { \TODO }
4985 \cs_new_protected:Nn \__stex_seqs_check_terms: { \TODO }
4986
4987 \cs_new_protected:Nn \__stex_seqs_html: {
4988    \exp_args:Ne \stex_annotate_invisible:nn {
4989      shtml:varseq = {\l_stex_key_name_str},
4990      shtml:args = {\l_stex_key_args_str}
4991      \str_if_empty:NF \l_stex_macroname_str {,
4992        shtml:macroname={\l_stex_macroname_str}
4993      }
4994      \str_if_empty:NF \l_stex_key_assoc_str {,
4995        shtml:assoctype={\l_stex_key_assoc_str}
4996      }
4997      \str_if_empty:NF \l_stex_key_role_str {,
4998        shtml:role={\l_stex_key_role_str}
4999      }
```

```
5000        \str_if_empty:NF \l_stex_key_reorder_str {,
5001          shtml:reorderargs={\l_stex_key_reorder_str}
5002        }
5003      }{\hbox\bgroup
5004        \_stex_annotate_force_break:n{
5005          \tl_if_empty:NF \l_stex_key_type_tl {
5006            \stex_annotate:nn{shtml:type={}}{$\l_stex_key_type_tl$}
5007          }
5008          \tl_if_empty:NF \l_stex_key_def_tl {
5009            \stex_annotate:nn{shtml:definiens={}}{$\l_stex_key_def_tl$}
5010          }
5011          \tl_if_empty:NF \l_stex_key_return_tl{
5012            \exp_args:Nno \use:n{
5013            \cs_generate_from_arg_count:NNnn \l__stex_seqs_cs
5014            \cs_set:Npn \l_stex_get_symbol_arity_int} \l_stex_key_return_tl
5015            \tl_set:Nx \l__stex_seqs_args_tl {\_stex_map_args:N \_stex_return_args:nn}
5016            \stex_annotate:nn{shtml:returntype={}}{
5017              $\exp_after:wN \l__stex_seqs_cs \l__stex_seqs_args_tl!$}
5018          }
5019          \tl_if_empty:NF \l_stex_key_argtypes_clist {
5020            \stex_annotate:nn{shtml:argtypes={}}{
5021              \_stex_annotate_force_break:n{
5022                \clist_map_inline:Nn \l_stex_key_argtypes_clist {
5023                  \stex_annotate:nn{shtml:type={}}{$##1$}
5024                }
5025              }
5026            }
5027          }
5028        }
5029      \egroup}
5030 }
```

(*End definition for* \varseq. *This function is documented on page 84.*)

\stex_invoke_sequence:

```
5031 \cs_new_protected:Nn \stex_invoke_sequence: {
5032   \peek_charcode_remove:NTF ! {
5033     \peek_charcode:NTF [ \__stex_seqs_do_op:w { \__stex_seqs_do_op:w [] }
5034   }\__stex_seqs_do_first:
5035 }
5036
5037 \cs_new_protected:Npn \__stex_seqs_do_op:w [#1] {
5038   \cs_if_exist:cTF {l_stex_notation_\l_stex_current_symbol_str _op_#1_cs}{
5039     \_stex_maybe_brackets:nn{\neginfprec}{
5040       \_stex_term_oms_or_omv:nnn{#1}{}
5041       {\use:c{l_stex_notation_\l_stex_current_symbol_str _op_#1_cs}}
5042     }
5043     \group_end:
5044   }{
5045     \__stex_seqs_get_index_notation:n{#1}
5046     \peek_charcode:NTF [ \__stex_seqs_doop_range:w { \__stex_seqs_doop_range:w[] }
5047   }
5048 }
5049
```

240

```
5050 \cs_new_protected:Npn \__stex_seqs_doop_range:w [#1] {
5051   \bool_set_true:N \l_stex_allow_semantic_bool
5052   \clist_clear:N \l__stex_seqs_clist
5053   \clist_map_function:NN \l_stex_current_type_tl \__stex_seqs_doop_arg:n
5054     \stex_annotate:nn{
5055       shtml:term=OMV,
5056       shtml:head={\l_stex_current_symbol_str},
5057       shtml:notationid={}
5058     }{
5059       \l__stex_seqs_clist
5060     }
5061     \group_end:
5062 }
5063
5064 \cs_new_protected:Nn \__stex_seqs_doop_arg:n {
5065   \tl_if_eq:nnTF{#1}{\ellipses}{
5066     \clist_put_right:Nn \l__stex_seqs_clist {
5067         \ellipses
5068     }
5069   }{
5070     \clist_put_right:Nn \l__stex_seqs_clist {
5071       \exp_args:No \str_if_eq:nnTF \l_stex_current_arity_str {1}{
5072         \group_begin:
5073           \l__stex_seqs_cs \group_end: {#1}
5074
5075       }{
5076         \group_begin:
5077           \l__stex_seqs_cs \group_end: #1
5078       }
5079     }
5080   }
5081 }
5082
5083 \cs_new_protected:Nn \__stex_seqs_get_index_notation:n {
5084   \cs_if_exist:cTF {l_stex_notation_\l_stex_current_symbol_str _#1_cs}{
5085     \cs_set_eq:Nc \l__stex_seqs_cs {l_stex_notation_\l_stex_current_symbol_str _#1_cs}
5086   }{
5087     \stex_do_default_notation:
5088     \cs_set_eq:NN \l__stex_seqs_cs \l_stex_default_notation
5089   }
5090 }
5091
5092
5093 \cs_new:Nn \__stex_seqs_do_first_arg:n {{\exp_not:n{## #1}}}
5094
5095 \cs_new_protected:Nn \__stex_seqs_do_first: {
5096   \exp_args:Nnx \use:nn{
5097   \cs_generate_from_arg_count:NNnn \l__stex_seqs_cs \cs_set:Npn
5098   \l_stex_current_arity_str} {{
5099     \tl_set:Nn \exp_not:N \l__stex_seqs_first_args_tl {
5100       \int_step_function:nN \l_stex_current_arity_str \__stex_seqs_do_first_arg:n
5101     }
5102     \exp_not:N \__stex_seqs_do_first_next:
5103   }}
```

241

```
5104    \l__stex_seqs_cs
5105  }
5106
5107  \cs_new_protected:Nn \__stex_seqs_do_first_next: {
5108    \peek_charcode_remove:NTF ! {
5109      \peek_charcode:NTF [ \__stex_seqs_do_one:w {\__stex_seqs_do_one:w []}
5110    }{
5111      \peek_charcode:NTF [ \__stex_seqs_do_all:w {\__stex_seqs_do_all:w []}
5112    }
5113  }
5114
5115  \cs_new_protected:Npn \__stex_seqs_do_one:w [#1] {
5116    \__stex_seqs_get_index_notation:n{#1}
5117    \stex_debug:nn{HERE~seq~one}{\meaning\l__stex_seqs_cs^^J\meaning\l__stex_seqs_first_args_t
5118    \exp_args:Nno\use:nn{\l__stex_seqs_cs\group_end:}\l__stex_seqs_first_args_tl
5119  }
5120
5121  \cs_new_protected:Npn \__stex_seqs_do_all:w [#1] {
5122    \stex_debug:nn{HERE~seq~all}{\meaning\l__stex_seqs_first_args_tl}
5123    \exp_args:Nno\use:nn{\_stex_invoke_notation:w [#1]}\l__stex_seqs_first_args_tl
5124  }
```

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

`\seqmap`

```
5125  \cs_new_protected:Npn \seqmap #1 #2 {
5126    \symuse{Metatheory?sequence~expression}{\seqmap{#1}{#2}}%\l_tmpa_tl {#2}
5127  }
```

(*End definition for* `\seqmap`*. This function is documented on page* *85*.*)

13.8.5 Expressions

```
5128  ⟨@@=stex_expr⟩
```

Various variables:

```
5129  \bool_new:N \l_stex_allow_semantic_bool
5130  \bool_set_true:N \l_stex_allow_semantic_bool
5131  \tl_new:N \l_stex_current_term_tl
5132  \tl_set:Nn \l_stex_every_symbol_tl {
5133    \bool_set_false:N \l_stex_allow_semantic_bool
5134  }
```

`\_stex_next_symbol:n`

```
5135  \tl_new:N \l__stex_expr_reset_tl
5136  \cs_new_protected:Nn \_stex_next_symbol:n {
5137    \tl_set:Nx \l_stex_every_symbol_tl {
5138      \tl_gset:Nn \exp_not:N \l__stex_expr_reset_tl {
5139        \tl_set:Nn \exp_not:N \l_stex_every_symbol_tl  {
5140          \exp_args:No \exp_not:n \l_stex_every_symbol_tl
5141        }
5142        \tl_gset:Nn \exp_not:N \l__stex_expr_reset_tl {
5143          \exp_args:No \exp_not:n \l__stex_expr_reset_tl
5144        }
5145      }
```

```
5146      \tl_set:Nn \exp_not:N \l_stex_every_symbol_tl  {
5147        \exp_args:No \exp_not:n \l_stex_every_symbol_tl
5148      }
5149      \exp_not:n{ \aftergroup \l__stex_expr_reset_tl }
5150      \exp_not:N \l_stex_every_symbol_tl
5151      \exp_not:n{ #1 }
5152   }
5153 }
5154 \cs_generate_variant:Nn \_stex_next_symbol:n {e}
```

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

\STEXinvisible

```
5155 \cs_new_protected:Npn \STEXinvisible #1 {
5156   \stex_annotate_invisible:n { #1 }
5157 }
```

(*End definition for* \STEXinvisible. *This function is documented on page* *70.*)

Invoking Semantic Macros

_stex_invoke_symbol:nnnnnnnN

```
5158 \cs_new_protected:Nn \_stex_invoke_symbol:nnnnnnnN {
5159   \bool_if:NTF \l_stex_allow_semantic_bool{
5160     \stex_if_html_backend:T{\ifvmode\indent\fi}
5161     \__stex_expr_setup:nnnnnn{\_comp}{#1?#2}{#3}{#4}{#7}{#6}
5162     \cs_set_eq:NN \_stex_term_oms_or_omv:nnn \_stex_term_oms:nnn
5163     \tl_put_right:Nn \l_stex_current_redo_tl{
5164       \cs_set_eq:NN \_stex_term_oms_or_omv:nnn \_stex_term_oms:nnn
5165     }
5166     #8
5167   }{
5168     \msg_error:nnxx{stex}{error/notallowed}{#1?#2}{\l_stex_current_symbol_str}
5169   }
5170 }
5171 \cs_generate_variant:Nn \_stex_invoke_symbol:nnnnnnnN {ooxooooN}
5172
5173 \cs_new_protected:Nn \__stex_expr_setup:nnnnnn {
5174   \group_begin:
5175   \tl_clear:N \l_stex_return_notation_tl
5176   \tl_set:Nn \l_stex_current_redo_tl {
5177     \let \this \stex_current_this:
5178     \def\comp{#1}
5179     \def\maincomp{\comp}
5180     \str_set:Nn \l_stex_current_symbol_str {#2}
5181     \str_set:Nn \l_stex_current_arity_str{ #3 }
5182     \tl_set:Nn \l_stex_current_args_tl{ #4 }
5183     \tl_set:Nn \l_stex_current_return_tl{ #5 }
5184     \tl_set:Nn \l_stex_current_type_tl{ #6 }
5185     \tl_clear:N \l_stex_current_term_tl
5186   }
5187   \tl_put_right:Nx \l_stex_current_redo_tl {
5188     \exp_args:No \exp_not:n \l_stex_every_symbol_tl
5189   }
```

243

```
5190      \l_stex_current_redo_tl
5191 }
```

(*End definition for* _stex_invoke_symbol:nnnnnnnN. *This function is documented on page* **??**.)

_stex_invoke_variable:nnnnnn

```
5192 \cs_new_protected:Nn \_stex_invoke_variable:nnnnnnnN {
5193    \bool_if:NTF \l_stex_allow_semantic_bool{
5194       \stex_if_html_backend:T{\ifvmode\indent\fi}
5195       \__stex_expr_setup:nnnnnn{\_varcomp}{#1}{#2}{#3}{#6}{#5}
5196       \cs_set_eq:NN \_stex_term_oms_or_omv:nnn \_stex_term_omv:nnn
5197       \tl_put_right:Nn \l_stex_current_redo_tl {
5198          \cs_set_eq:NN \_stex_term_oms_or_omv:nnn \_stex_term_omv:nnn
5199       }
5200       #7
5201    }{
5202       \msg_error:nnxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
5203    }
5204 }
```

(*End definition for* _stex_invoke_variable:nnnnnn. *This function is documented on page* **??**.)

\symuse

```
5205 \cs_new_protected:Npn \symuse #1 {
5206    \stex_get_symbol:n{#1}
5207    \exp_args:Nno \use:n {\_stex_invoke_symbol:ooxooooN
5208    \l_stex_get_symbol_mod_str
5209    \l_stex_get_symbol_name_str
5210    {\int_use:N \l_stex_get_symbol_arity_int}
5211    \l_stex_get_symbol_args_tl
5212    \l_stex_get_symbol_def_tl
5213    \l_stex_get_symbol_type_tl
5214    \l_stex_get_symbol_return_tl}
5215    \l_stex_get_symbol_invoke_cs
5216 }
```

(*End definition for* \symuse. *This function is documented on page 78.*)

\stex_invoke_symbol: Top-Level: Check whether text/math mode or custom notation, whether delimited by !, return code etc.

```
5217 \cs_new_protected:Nn \stex_invoke_symbol: {
5218    \stex_debug:nn{expressions}{Invoking~\l_stex_current_symbol_str}
5219    \mode_if_math:TF \__stex_expr_invoke_math: \__stex_expr_invoke_text:
5220 }
5221
5222 \cs_new_protected:Nn \__stex_expr_invoke_text: {
5223    \stex_debug:nn{expressions}{text~mode}
5224    \peek_charcode_remove:NTF ! \__stex_expr_invoke_op_custom:n \__stex_expr_invoke_custom:n
5225 }
5226
5227 \cs_new_protected:Nn \__stex_expr_invoke_math: {
5228    \stex_debug:nn{expressions}{math~mode}
5229    \peek_charcode_remove:NTF ! {
5230       % operator
5231       \peek_charcode_remove:NTF * \__stex_expr_invoke_op_custom:n {
```

```
5232        % op notation
5233        \peek_charcode:NTF [ \__stex_expr_invoke_op_notation:w {
5234          \__stex_expr_invoke_op_notation:w []
5235        }
5236      }
5237    }{
5238      \peek_charcode_remove:NTF * \__stex_expr_invoke_custom:n {
5239        % normal
5240        \peek_charcode:NTF [ \_stex_invoke_notation:w {
5241          \_stex_invoke_notation:w []
5242        }
5243      }
5244    }
5245 }
```

Notations:

```
5246 \cs_new_protected:Npn \_stex_invoke_notation:w [#1] {
5247   \stex_debug:nn{expressions}{using~notation~#1~for~\l_stex_current_symbol_str}
5248   \cs_if_exist:cTF{l_stex_notation_\l_stex_current_symbol_str _#1_cs}{
5249     \tl_if_empty:NTF \l_stex_current_return_tl {
5250       \stex_debug:nn{expressions}{return~empty}
5251       \use:c{l_stex_notation_\l_stex_current_symbol_str _#1_cs}{\group_end:\_stex_eat_exclam
5252     }{
5253       \stex_debug:nn{expressions}{return?}
5254       \exp_after:wN\exp_after:wN\exp_after:wN
5255       \__stex_expr_invoke_return_maybe:n
5256       \exp_after:wN\exp_after:wN\exp_after:wN
5257       {\cs:w l_stex_notation_\l_stex_current_symbol_str _#1_cs \cs_end: {}}
5258     }
5259   }{
5260     \stex_do_default_notation:
5261     \tl_if_empty:NTF \l_stex_current_return_tl {
5262       \l_stex_default_notation{\group_end:\_stex_eat_exclamation_point:}
5263     }{
5264       \exp_after:wN
5265       \__stex_expr_invoke_return_maybe:n
5266       \exp_after:wN
5267       {\l_stex_default_notation {}}
5268     }
5269   }
5270 }
5271
5272 \cs_new_protected:Npn \__stex_expr_invoke_op_notation:w [#1] {
5273   \stex_debug:nn{expressions}{op~notation~for~\l_stex_current_symbol_str}
5274   \cs_if_exist:cTF{l_stex_notation_\l_stex_current_symbol_str _op_#1_cs}{
5275     \_stex_maybe_brackets:nn{\neginfprec}{
5276       \_stex_term_oms_or_omv:nnn{#1}{}
5277       {\use:c{l_stex_notation_\l_stex_current_symbol_str _op_#1_cs}}
5278     }
5279     \group_end:
5280   }{
5281     \int_compare:nNnTF \l_stex_current_arity_str = 0 {
5282       \tl_clear:N \l_stex_current_return_tl
5283       \_stex_invoke_notation:w [#1]
```

245

```
5284        }{
5285          \stex_do_default_notation_op:
5286          \_stex_maybe_brackets:nn{\neginfprec}{
5287            \_stex_term_oms_or_omv:nnn{#1}{}
5288            {\l_stex_default_notation}
5289          }
5290          \group_end:
5291        }
5292      }
5293  }
```

Return:

```
5294  \cs_new_protected:Nn \__stex_expr_invoke_return_maybe:n {
5295    \tl_clear:N \l__stex_expr_return_args_tl
5296    \tl_set:Nn \l__stex_expr_return_this_tl {#1}
5297    \exp_args:Nnx \use:n {
5298    \cs_generate_from_arg_count:NNnn \__stex_expr_ret_cs
5299      \cs_set:Npn \l_stex_current_arity_str } {
5300        \int_step_function:nN \l_stex_current_arity_str \__stex_expr_return_arg:n
5301        \__stex_expr_invoke_return_next:
5302      }
5303    \__stex_expr_ret_cs
5304  }
5305
5306  \cs_new:Nn \__stex_expr_return_arg:n {
5307    \tl_put_right:Nn \exp_not:N \l__stex_expr_return_args_tl {{#### #1}}
5308  }
5309
5310  \cs_new_protected:Nn \__stex_expr_invoke_return_next: {
5311    \peek_charcode_remove:NTF ! {
5312      \exp_after:wN \l__stex_expr_return_this_tl \l__stex_expr_return_args_tl \group_end:
5313    }\__stex_expr_invoke_return:
5314  }
5315
5316  \cs_new_protected:Nn \__stex_expr_invoke_return: {
5317    \tl_set:Nx \l__stex_expr_return_this_tl {
5318      \__stex_expr_return_notation:n {
5319        \exp_after:wN \exp_after:wN \exp_after:wN
5320        \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
5321          \exp_after:wN \l__stex_expr_return_this_tl \l__stex_expr_return_args_tl
5322        }
5323      }
5324    }
5325    \stex_debug:nn{return}{Notation:~\meaning\l__stex_expr_return_this_tl}
5326    \tl_put_left:Nx \l__stex_expr_return_this_tl {
5327      \group_begin:\exp_args:No \exp_not:n \l_stex_current_redo_tl
5328    }
5329    \exp_args:Nnx \use:n {
5330    \cs_generate_from_arg_count:NNnn \__stex_expr_ret_cs
5331      \cs_set:Npn \l_stex_current_arity_str } {
5332        \exp_args:No \exp_not:n \l_stex_current_return_tl
5333      }
5334    \stex_debug:nn{return}{
5335      \meaning\__stex_expr_ret_cs^^J
```

```
5336      \meaning\l__stex_expr_return_this_tl^^J
5337      \exp_args:No \exp_not:n \l__stex_expr_return_args_tl^^J
5338    }
5339    \exp_args:Nnx \use:nn {
5340      \exp_after:wN \group_end: \__stex_expr_ret_cs
5341    }{
5342      \exp_args:No \exp_not:n \l__stex_expr_return_args_tl
5343      {
5344        \exp_args:No \exp_not:n \l__stex_expr_return_this_tl
5345        \group_end:
5346      }
5347    }
5348  }
5349
5350
5351  \cs_new_protected:Nn \__stex_expr_return_notation:n {
5352    \tl_if_empty:NTF \l_stex_return_notation_tl { #1 }{
5353      \l_stex_return_notation_tl
5354    }
5355  }
5356
```

Custom Notations:

```
5357  \cs_new_protected:Nn \__stex_expr_invoke_op_custom:n {
5358    \stex_debug:nn{expressions}{custom~op}
5359    \bool_set_true:N \l_stex_allow_semantic_bool
5360    \_stex_term_oms_or_omv:nnn{}{}{\maincomp{#1}}
5361    \group_end:
5362  }
5363
5364  \int_new:N \l__stex_expr_arg_counter_int
5365  \cs_new_protected:Nn \__stex_expr_invoke_custom:n {
5366    \stex_debug:nn{custom}{custom~notation~for~\l_stex_current_symbol_str}
5367    \stex_pseudogroup:nn{
5368      \bool_set_true:N \l_stex_allow_semantic_bool
5369      \prop_gclear:N \l__stex_expr_customs_prop
5370      \seq_gclear:N \l__stex_expr_customs_seq
5371      \int_gzero:N \l__stex_expr_arg_counter_int
5372      \tl_if_empty:NF \l_stex_current_args_tl {
5373        \exp_after:wN \__stex_expr_add_prop_arg:nnw \l_stex_current_args_tl \_stex_args_end:
5374        \cs_set_eq:NN \arg \__stex_expr_arg:n
5375      }
5376      \tl_set_eq:NN \l_stex_get_symbol_args_tl \l_stex_current_args_tl
5377      \cs_set_eq:NN \__stex_expr_do_ab_next:nnn \_stex_term_oma:nnn
5378      \_stex_map_args:N \__stex_expr_check_b:nn
5379      \__stex_expr_do_ab_next:nnn{}{}{#1}
5380    }{
5381      \prop_if_exist:NT \l__stex_expr_customs_prop {
5382        \prop_gset_from_keyval:Nn \exp_not:N \l__stex_expr_customs_prop {
5383          \prop_to_keyval:N \l__stex_expr_customs_prop
5384        }
5385      }
5386      \int_gset:Nn \l__stex_expr_arg_counter_int { \int_use:N \l__stex_expr_arg_counter_int}
5387      \seq_if_exist:NT \l__stex_expr_customs_seq {
```

```
5388      \seq_gset_split:Nnn \exp_not:N \l__stex_expr_customs_seq , {
5389        \seq_use:Nn \l__stex_expr_customs_seq ,
5390      }
5391    }
5392  }
5393  % TODO check that all arguments are present
5394  \group_end:
5395 }
5396
5397 \cs_new_protected:Npn \__stex_expr_add_prop_arg:nnw #1 #2 #3\_stex_args_end: {
5398   \prop_gput:Nnn \l__stex_expr_customs_prop {#1} {}
5399   \seq_gput_right:Nn \l__stex_expr_customs_seq {#2}
5400   \tl_if_empty:nF{#3}{\__stex_expr_add_prop_arg:nnw #3 \_stex_args_end:}
5401 }
5402
5403 \cs_new:Nn \__stex_expr_check_b:nn {
5404   \str_case:nn #2 {
5405     b {\cs_set_eq:NN \__stex_expr_do_ab_next:nnn \_stex_term_omb:nnn}
5406     B {\cs_set_eq:NN \__stex_expr_do_ab_next:nnn \_stex_term_omb:nnn}
5407   }
5408 }
5409
5410 \NewDocumentCommand \__stex_expr_arg:n {s O{} m} {
5411   \IfBooleanTF #1 {
5412     \stex_annotate_invisible:n{
5413       \__stex_expr_arg_inner:nn{#2}{#3}
5414     }
5415   }{
5416     \__stex_expr_arg_inner:nn{#2}{#3}
5417   }
5418 }
5419
5420 \cs_new_protected:Nn \__stex_expr_arg_inner:nn {
5421   \tl_if_empty:nTF{#1}{
5422     \int_gincr:N \l__stex_expr_arg_counter_int
5423     \exp_args:Ne \__stex_expr_check:nTF{ \int_use:N \l__stex_expr_arg_counter_int }{
5424       \__stex_expr_arg_do:oon \l_tmpa_tl \l_tmpb_tl
5425     }{
5426       \__stex_expr_arg_inner:nn{}
5427     }{ #2 }
5428   }{
5429     \__stex_expr_check:nTF {#1}{
5430       \__stex_expr_arg_do:oon \l_tmpa_tl \l_tmpb_tl { #2 }
5431     }{
5432       \exp_args:No \str_case:nnTF \l_tmpb_tl {
5433         {a}{
5434           \exp_args:NNnx \prop_gput:Nnn \l__stex_expr_customs_prop {#1}{
5435             \l_tmpa_tl X
5436           }
5437           \tl_set:Nx \l_tmpa_tl { #1 \int_eval:n {\tl_count:N \l_tmpa_tl + 1} }
5438         }
5439         {B}{
5440           \exp_args:NNnx \prop_gput:Nnn \l__stex_expr_customs_prop {#1}{
5441             \l_tmpa_tl X
```

248

```
5442                }
5443              \tl_set:Nx \l_tmpa_tl { #1 \int_eval:n {\tl_count:N \l_tmpa_tl + 1} }
5444            }
5445          }{
5446            \__stex_expr_arg_do:oon \l_tmpa_tl \l_tmpb_tl { #2 }
5447          }{
5448            \msg_error:nnxx{stex}{error/invalidarg}{#1}{\l_stex_current_symbol_str}
5449          }
5450        }
5451      }
5452 }
5453
5454 \prg_new_conditional:Nnn \__stex_expr_check:n {TF} {
5455    \exp_args:NNe \prop_get:NnNTF \l__stex_expr_customs_prop {#1} \l_tmpa_tl {
5456      \tl_set:Nx \l_tmpb_tl {\seq_item:Nn \l__stex_expr_customs_seq {#1} }
5457      \tl_if_empty:NTF \l_tmpa_tl {
5458        \exp_args:NNe \prop_gput:Nnn \l__stex_expr_customs_prop
5459          { #1 }{X}
5460        \exp_args:No \str_case:nnF \l_tmpb_tl {
5461          {a}{
5462            \tl_set:Nx \l_tmpa_tl{ #1 1 }
5463          }
5464          {B}{
5465            \tl_set:Nx \l_tmpa_tl{ #1 1 }
5466          }
5467        }{
5468          \tl_set:Nx \l_tmpa_tl{ #1 }
5469        }
5470        \prg_return_true:
5471      }{
5472        \prg_return_false:
5473      }
5474    }{
5475      \msg_error:nnxx{stex}{error/invalidarg}{#1}{\l_stex_current_symbol_str}
5476      \prg_return_false:
5477    }
5478 }
5479
5480 % #1 argnum #2 argmode #3 code
5481 \cs_new_protected:Nn \__stex_expr_arg_do:nnn {
5482    \stex_debug:nn{custom}{Doing~argument~#1~of~mode~#2:~\tl_to_str:n{#3}}
5483    \group_begin:
5484      \bool_set_true:N \l_stex_allow_semantic_bool
5485      \_stex_term_arg:nnn {#2}{#1}{#3}
5486    \group_end:
5487 }
5488 \cs_generate_variant:Nn \__stex_expr_arg_do:nnn {oon}
```

(*End definition for* `\stex_invoke_symbol:`. *This function is documented on page 120.*)

Argument Handling and Annotating

`\_stex_term_arg:nnnnn`
`\_stex_term_arg:nnn`

```
5489 % 1: argnum 2: argmode 3: precedence 4: argname 5: code
```

```
5490 \cs_new_protected:Nn \_stex_term_arg:nnnnn {
5491   \group_begin:
5492     \str_clear:N \l_stex_current_symbol_str
5493     \tl_clear:N \l_stex_current_term_tl
5494     \int_set:Nn \l_stex_notation_downprec { #3 }
5495     \bool_set_true:N \l_stex_allow_semantic_bool
5496     \_stex_term_arg:nnn {#2}{#1}{
5497       \tl_if_empty:nTF{#4}{
5498         #5
5499       }{
5500         \stex_annotate:nn{mml:arg={#4}}{#5}
5501       }
5502     }
5503   \group_end:
5504 }
5505
5506 \cs_new_protected:Nn \_stex_term_arg:nnn {
5507   \stex_annotate:nn{ shtml:arg={#2}, shtml:argmode={#1}}{
5508     \_stex_annotate_force_break:n{ #3 }
5509   }
5510 }
```

(*End definition for* _stex_term_arg:nnnnn *and* _stex_term_arg:nnn. *These functions are documented on page* **??**.)

_stex_term_arg_aB:nnnnn

```
5511 \tl_set:Nn \__stex_expr_do_aB_clist: {
5512   \seq_use:Nn \l_stex_aB_args_seq {
5513     \mathpunct{\comp{,}}
5514   }
5515 }
5516 \tl_set_eq:NN \_stex_term_do_aB_clist: \__stex_expr_do_aB_clist:
5517
5518 \int_new:N \l__stex_expr_count_int
5519 \cs_new_protected:Nn \_stex_term_arg_aB:nnnnn {
5520   \tl_if_empty:nTF{#5}{
5521     \_stex_term_arg:nnnnn{#1}{#2}{#3}{#4}{}
5522   }{
5523     \seq_clear:N \l_stex_aB_args_seq
5524     \int_zero:N \l__stex_expr_count_int
5525     \clist_map_inline:nn{#5}{
5526       \__stex_expr_aB_arg:nnnnn{##1}{#1}{#2}{#3}{#4}
5527     }
5528     \_stex_term_do_aB_clist:
5529   }
5530 }
5531
5532 % 1: code 2: argnum 3: argmode 4: precedence 5: argname
5533 \cs_new_protected:Npn \__stex_expr_aB_arg:nnnnn #1 {
5534   \int_incr:N \l__stex_expr_count_int
5535   \__stex_expr_is_varseq:nTF{#1}{
5536     \exp_after:wN \exp_after:wN \exp_after:wN
5537     \__stex_expr_assoc_seq:nnnnnnn
5538     \exp_after:wN
```

```
5539        \__stex_expr_gobble:nnnnnnnn #1 \__stex_expr_end:
5540      }{
5541        \__stex_expr_is_seqmap:nTF{#1}{
5542          \exp_args:NNe \use:nn \__stex_expr_do_seqmap:nnnnnn {\tl_tail:n{#1}}
5543        }{
5544          \__stex_expr_aB_simple_arg:nnnnn{#1}
5545        }
5546      }
5547 }
5548
5549 \cs_new:Npn \__stex_expr_gobble:nnnnnnnn #1 #2 #3 #4 #5 #6 #7 #8 #9 \__stex_expr_end: {
5550   {#2} #3 {#6}
5551 }
5552
5553 \cs_new_protected:Nn \__stex_expr_aB_simple_arg:nnnnn{
5554   \seq_put_right:Nx \l_stex_aB_args_seq {
5555     \_stex_term_arg:nnnnn{#2\int_use:N\l__stex_expr_count_int}{#3}{#4}{#5}{
5556       \exp_not:n{
5557         \tl_set_eq:NN \_stex_term_do_aB_clist: \__stex_expr_do_aB_clist:
5558         #1
5559       }
5560     }
5561   }
5562 }
5563
5564 \cs_new_protected:Nn \_stex_is_sequentialized:n {
5565   \group_begin: #1 \group_end:
5566 }
```

Conditionals: Is the argument a sequence variable or a \seqmap?

```
5567 \prg_new_conditional:Nnn \__stex_expr_is_varseq:n {TF} {
5568   \int_compare:nNnTF {\tl_count:n{#1}} = 1 {
5569     \exp_args:Ne \cs_if_eq:NNTF {\exp_args:No \tl_head:n{#1}}
5570     \_stex_invoke_variable:nnnnnnN {
5571       \exp_args:Ne \cs_if_eq:NNTF {\exp_args:No\tl_item:nn{#1}{8}}
5572         \stex_invoke_sequence:
5573           \prg_return_true:\prg_return_false:
5574     }\prg_return_false:
5575   }\prg_return_false:
5576 }
5577
5578 \prg_new_conditional:Nnn \__stex_expr_is_seqmap:n {TF} {
5579   \int_compare:nNnTF {\tl_count:n{#1}} = 3 {
5580     \exp_args:Ne \tl_if_eq:nnTF {\tl_head:n{#1}} {\seqmap}
5581       \prg_return_true:\prg_return_false:
5582   }\prg_return_false:
5583 }
```

Sequence variable:

```
5584 % 1: name 2: arity 3: clist 4: argnum 5: argmode 6: precedence 7: argname
5585 \cs_new_protected:Nn \__stex_expr_assoc_seq:nnnnnnn {
5586   \group_begin:
5587     \seq_clear:N \l_stex_aB_args_seq
5588     \__stex_expr_assoc_make_seq:nnn{#1}{#3}{#2}
```

```
5589    \exp_args:NNe \use:nn \group_end: {
5590      \seq_put_right:Nn \exp_not:N \l_stex_aB_args_seq {
5591        \_stex_is_sequentialized:n{
5592          \_stex_term_arg:nnnnn{#4\int_use:N\l__stex_expr_count_int}{#5}{#6}{#7}{
5593            \bool_set_true:N \l_stex_allow_semantic_bool
5594            \str_set:Nn \exp_not:N \l_stex_current_symbol_str
5595              {\l_stex_current_symbol_str}
5596            \tl_if_empty:NF \l_stex_current_term_tl {
5597              \tl_set:Nn \exp_not:N \l_stex_current_term_tl {
5598                \exp_args:No \exp_not:n \l_stex_current_term_tl
5599              }
5600            }
5601            \stex_annotate:nn{
5602              shtml:term=OMV,
5603              shtml:head={#1},
5604              shtml:notationid={}
5605            }{
5606              \_stex_annotate_force_break:n{
5607                \_stex_term_do_aB_clist:
5608              }
5609            }
5610          }
5611        }
5612      }
5613    }
5614 }
5615
5616 % #1: name, #2: clist, #3:arity
5617 \cs_new_protected:Nn \__stex_expr_assoc_make_seq:nnn {
5618    \cs_if_exist:cTF{l_stex_notation_#1__cs}{
5619      \cs_set_eq:Nc \l__stex_expr_cs {l_stex_notation_#1__cs}
5620    }{
5621      \stex_do_default_notation:
5622      \cs_set_eq:NN \l__stex_expr_cs \l_stex_default_notation
5623    }
5624    \clist_map_inline:nn{#2}{
5625      \tl_if_eq:nnTF{##1}{\ellipses}{
5626        \seq_put_right:Nn \l_stex_aB_args_seq { ##1 }
5627      }{
5628        \int_compare:nNnTF {#3} = 1 {
5629          \tl_set:Nn \l__stex_expr_iarg_tl { {##1} }
5630        }{
5631          \tl_set:Nn \l__stex_expr_iarg_tl { ##1 }
5632        }
5633        \seq_put_right:Nx \l_stex_aB_args_seq {
5634          \group_begin:
5635          \exp_not:n {
5636            \tl_set_eq:NN \_stex_term_do_aB_clist: \__stex_expr_do_aB_clist:
5637            \def\comp{\_varcomp}
5638            \str_set:Nn \l_stex_current_symbol_str{#1}
5639          }
5640          \exp_after:wN \exp_after:wN \exp_after:wN \exp_not:n
5641          \exp_after:wN \exp_after:wN \exp_after:wN {
5642            \exp_after:wN \l__stex_expr_cs \exp_after:wN \group_end: \l__stex_expr_iarg_tl
```

```
5643                   }
5644               }
5645           }
5646       }
5647 }

     \seqmap:

5648 % 1: fun 2: clist 3: argnum 4: argmode 5: precedence 6: argname
5649 \cs_new_protected:Nn \__stex_expr_do_seqmap:nnnnnn {
5650   \group_begin:
5651     \cs_set:Npn \l_tmpa_cs ##1 {#1}
5652     \seq_clear:N \l_stex_aB_args_seq
5653     \clist_map_inline:nn{#2}{
5654       \__stex_expr_is_varseq:nTF{##1}{
5655         \exp_after:wN
5656         \__stex_expr_varseq_in_map:nnnnnnnn ##1
5657       }{
5658         \seq_put_right:Nn \l_stex_aB_args_seq {##1}
5659       }
5660     }
5661     \seq_clear:N \l__stex_expr_old_seq
5662     \seq_map_inline:Nn \l_stex_aB_args_seq {
5663       \tl_if_eq:nnTF{##1}{\ellipses}{
5664         \seq_put_right:Nn \l__stex_expr_old_seq {##1}
5665       }
5666       % TODO \_stex_is_sequentialized:n
5667       {
5668         \exp_args:NNo \seq_put_right:Nn \l__stex_expr_old_seq {
5669           \l_tmpa_cs {##1}
5670         }
5671       }
5672     }
5673     \seq_set_eq:NN \l_stex_aB_args_seq \l__stex_expr_old_seq
5674   \exp_args:NNe \use:nn \group_end: {
5675     \seq_put_right:Nn \exp_not:N \l_stex_aB_args_seq {
5676       \_stex_is_sequentialized:n{
5677         \_stex_term_arg:nnnnn{#4\int_use:N\l__stex_expr_count_int}{#4}{#5}{#6}{
5678           \bool_set_true:N \l_stex_allow_semantic_bool
5679           \str_set:Nn \exp_not:N \l_stex_current_symbol_str
5680             {\l_stex_current_symbol_str}
5681           \tl_set:Nn \exp_not:N \l_stex_current_term_tl {
5682             \symuse{Metatheory?sequence~map}
5683             {\exp_args:No \exp_not:n \l_tmpa_tl}
5684             {\_stex_term_do_aB_clist:}
5685         }
5686         \__stex_expr_do_headterm:nn{}{
5687           \_stex_term_do_aB_clist:
5688         }
5689       }
5690     }
5691   }
5692   }
5693 }

5694
```

```
5695 \cs_new_protected:Nn \__stex_expr_varseq_in_map:nnnnnnnn {
5696   \__stex_expr_assoc_make_seq:nnn{#2}{#6}{#3}
5697 }
```

(End definition for _stex_term_arg_aB:nnnnn. This function is documented on page **??**.)

Term HTML Annotations

_stex_term_oms_or_omv:nnn
_stex_term_oms:nnn
_stex_term_omv:nnn

```
5698 \cs_new_protected:Nn \_stex_eat_exclamation_point: {
5699   \peek_charcode_remove:NT ! {
5700     \_stex_eat_exclamation_point:
5701   }
5702 }
5703
5704 \bool_new:N \stex_in_invisible_html_bool
5705 \bool_set_false:N \stex_in_invisible_html_bool
5706 \stex_if_html_backend:TF {
5707   % 1: variant 2: intent 3: code
5708   \cs_new_protected:Nn \_stex_term_oms:nnn {
5709     \tl_if_empty:NTF \l_stex_current_term_tl {
5710       \stex_annotate:nn{
5711         shtml:term=OMID,
5712         shtml:head={\l_stex_current_symbol_str},
5713         shtml:notationid={#1},
5714       }{
5715         \_stex_annotate_force_break:n{#3}
5716       }
5717     }{
5718       \__stex_expr_do_headterm:nn{#1}{#3}
5719     }
5720   }
5721   \cs_new_protected:Nn \_stex_term_omv:nnn {
5722     \tl_if_empty:NTF \l_stex_current_term_tl {
5723       \stex_annotate:nn{
5724         shtml:term=OMV,
5725         shtml:head={\l_stex_current_symbol_str},
5726         shtml:notationid={#1}
5727       }{
5728         \_stex_annotate_force_break:n{#3}
5729       }
5730     }{
5731       \__stex_expr_do_headterm:nn{#1}{#3}
5732     }
5733   }
5734   \cs_new_protected:Nn \__stex_expr_do_headterm:nn {
5735     \bool_if:NTF \stex_in_invisible_html_bool {
5736       {\bool_set_true:N \l_stex_allow_semantic_bool
5737         \ensuremath{\l_stex_current_term_tl}
5738       }
5739     }{
5740       \stex_annotate:nn{
5741         shtml:term=complex,
5742         shtml:head={\l_stex_current_symbol_str},
```

254

```
5743         shtml:notationid={#1}
5744       }{
5745         \_stex_annotate_force_break:n{
5746           \stex_annotate_invisible:nn{shtml:headterm={}}{
5747             {\bool_set_true:N \l_stex_allow_semantic_bool
5748               \ensuremath{\l_stex_current_term_tl}
5749             }
5750           }
5751         }
5752         #2
5753       }
5754     }
5755   }
5756 }{
5757   \cs_new_protected:Nn \_stex_term_oms:nnn {#3}
5758   \cs_new_protected:Nn \_stex_term_omv:nnn {#3}
5759   \cs_new_protected:Nn \__stex_expr_do_headterm:nn { #2 }
5760 }
5761 \cs_set_eq:NN \_stex_term_oms_or_omv:nnn \_stex_term_oms:nnn
```

(End definition for `\_stex_term_oms_or_omv:nnn`*,* `\_stex_term_oms:nnn`*, and* `\_stex_term_omv:nnn`*. These functions are documented on page* **??***.)*

`\_stex_term_oma:nnn`

```
5762 \stex_if_html_backend:TF {
5763   \cs_new_protected:Nn \_stex_term_oma:nnn {
5764     \tl_if_empty:NTF \l_stex_current_term_tl {
5765       \stex_annotate:nn{
5766         shtml:term=OMA,
5767         shtml:head={\l_stex_current_symbol_str},
5768         shtml:notationid={#1}
5769       }{
5770         \_stex_annotate_force_break:n{#3}
5771       }
5772     }{
5773       \stex_annotate:nn{
5774         shtml:term=OMA,
5775         shtml:head={\l_stex_current_symbol_str},
5776         shtml:notationid={#1}
5777       }{
5778         \_stex_annotate_force_break:n{
5779           \stex_annotate_invisible:nn{shtml:headterm={}}{{
5780             \bool_set_true:N \l_stex_allow_semantic_bool
5781             \l_stex_current_term_tl
5782           }}
5783           #3
5784         }
5785       }
5786     }
5787   }
5788 }{
5789   \cs_new_protected:Nn \_stex_term_oma:nnn {#3}
5790 }
```

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

255

`\_stex_term_omb:nnn`

```
5791 \stex_if_html_backend:TF {
5792   \cs_new_protected:Nn \_stex_term_omb:nnn {
5793     \tl_if_empty:NTF \l_stex_current_term_tl {
5794       \stex_annotate:nn{
5795         shtml:term=OMBIND,
5796         shtml:head={\l_stex_current_symbol_str},
5797         shtml:notationid={#1}
5798       }{
5799         \_stex_annotate_force_break:n{#3}
5800       }
5801     }{
5802       \stex_annotate:nn{
5803         shtml:term=OMBIND,
5804         shtml:head={\l_stex_current_symbol_str},
5805         shtml:notationid={#1}
5806       }{
5807         \_stex_annotate_force_break:n{
5808           \stex_annotate_invisible:nn{shtml:headterm={}}{{
5809             \bool_set_true:N \l_stex_allow_semantic_bool
5810             \l_stex_current_term_tl
5811           }}
5812           #3
5813         }
5814       }
5815     }
5816   }
5817 }{
5818   \cs_new_protected:Nn \_stex_term_omb:nnn { #3 }
5819 }
```

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

Automated Bracketing

`\infprec`
`\neginfprec`
```
5820 \tl_const:Nx \infprec {\int_use:N \c_max_int}
5821 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
```

(*End definition for* `\infprec` *and* `\neginfprec`*. These functions are documented on page* *80*.)

`\dobrackets`
```
5822 \int_new:N \l_stex_notation_downprec
5823 \int_set:Nn \l_stex_notation_downprec \infprec
5824 \tl_set:Nn \l__stex_expr_left_bracket_str (
5825 \tl_set:Nn \l__stex_expr_right_bracket_str )
5826 \bool_new:N \l_stex_brackets_dones_bool
5827
5828 \cs_new_protected:Nn \stex_maybe_brackets:nn {
5829   \bool_if:NTF \l_stex_brackets_dones_bool {
5830     \bool_set_false:N \l_stex_brackets_dones_bool
5831     #2
5832   } {
5833     \stex_debug:nn{brackets}{#1>\int_eval:n \l_stex_notation_downprec?}
```

```
5834    \int_compare:nNnTF { #1 } > \l_stex_notation_downprec {
5835      %\bool_if:NTF \l_stex_inparray_bool { #2 }{
5836        \dobrackets {
5837          #2
5838        }
5839      %}
5840    }{
5841      #2
5842    }
5843  }
5844 }
5845
5846 %\RequirePackage{scalerel}
5847 \cs_new_protected:Npn \dobrackets #1 {
5848   %\ThisStyle{\if D\m@switch
5849   %    \exp_args:Nnx \use:nn
5850   %    { \exp_after:wN \left\l_stex_expr_left_bracket_str #1 }
5851   %    { \exp_not:N\right\l__stex_expr_right_bracket_str }
5852   %  \else
5853   \group_begin:
5854   %\stex_pseudogroup_with:nn{\l_stex_brackets_dones_bool\l_stex_notation_downprec}{
5855     \bool_set_true:N \l_stex_brackets_dones_bool
5856     %\int_set:Nn \l_stex_notation_downprec \infprec
5857     \mathopen{\cs_if_exist:NT\l_stex_current_symbol_str\comp
5858       \l__stex_expr_left_bracket_str
5859     }
5860     #1
5861   \group_end:%}
5862   \mathclose{\cs_if_exist:NT\l_stex_current_symbol_str\comp
5863     \l__stex_expr_right_bracket_str
5864   }
5865   %\fi}
5866 }
```

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

`\withbrackets`

```
5867 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
5868   \stex_pseudogroup_with:nn{\l__stex_expr_left_bracket_str\l__stex_expr_right_bracket_str}{
5869     \tl_set:Nn \l__stex_expr_left_bracket_str { #1 }
5870     \tl_set:Nn \l__stex_expr_right_bracket_str { #2 }
5871     #3
5872   }
5873 }
```

(*End definition for* `\withbrackets`. *This function is documented on page* *81.*)

`\dowithbrackets`

```
5874 \cs_new_protected:Npn \dowithbrackets #1 #2 #3 {
5875   \withbrackets{#1}{#2}{\dobrackets{#3}}
5876 }
```

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

257

Symname and Variants

```
5877 \def\maincomp{\comp}
5878
5879 \stex_keys_define:nnnn{symname}{
5880   \tl_clear:N \l_stex_key_pre_tl
5881   \tl_clear:N \l_stex_key_post_tl
5882   %\tl_clear:N \l_stex_key_proot_tl
5883 }{
5884   pre    .tl_set:N  = \l_stex_key_pre_tl ,
5885   post   .tl_set:N  = \l_stex_key_post_tl ,
5886   root   .code:n    = {}%.tl_set:N   = \l_stex_key_root_tl
5887 }{}
5888
5889 \NewDocumentCommand \symref { O{} m m } {
5890   \group_begin:
5891   \stex_keys_set:nn{symname}{#1}
5892   \stex_get_symbol:n{#2}
5893   \__stex_expr_do_ref:nNn{#3}\symrefemph@uri\_stex_term_oms:nnn
5894 }
5895 \let\sr\symref
5896
5897 \NewDocumentCommand \symname { O{} m } {
5898   \group_begin:
5899   \stex_keys_set:nn{symname}{#1}
5900   \stex_get_symbol:n{#2}
5901   \__stex_expr_do_ref:nNn{
5902     \l_stex_key_pre_tl\l_stex_get_symbol_name_str\l_stex_key_post_tl
5903   }\symrefemph@uri\_stex_term_oms:nnn
5904 }
5905 \let\sn\symname
5906 \protected\def\sns{\symname[post=s]}
5907
5908 \NewDocumentCommand \Symname { O{} m } {
5909   \group_begin:
5910   \stex_keys_set:nn{symname}{#1}
5911   \stex_get_symbol:n{#2}
5912   \__stex_expr_do_ref:nNn{
5913     \l_stex_key_pre_tl\exp_after:wN\_stex_capitalize:n\l_stex_get_symbol_name_str\l_stex_key
5914   }\symrefemph@uri\_stex_term_oms:nnn
5915 }
5916 \cs_new_protected:Nn \_stex_capitalize:n {
5917   \uppercase{#1}
5918 }
5919 \let\Sn\Symname
5920 \protected\def\Sns{\Symname[post=s]}
5921
5922 \cs_new:Npn \_stex_split_slash: #1/#2/#3\_stex_args_end: {
5923 \tl_if_empty:nTF{#3}{
5924   #2
5925 }{
5926   \_stex_split_slash: #2 / #3 \_stex_args_end:
5927 }
```

```
5928  }
5929
5930  \NewDocumentCommand \varref { O{} m m} {
5931    \group_begin:
5932    \stex_keys_set:nn{symname}{#1}
5933    \stex_get_var:n{#2}
5934    \__stex_expr_do_ref:nNn{#3}\varemph@uri{
5935      \str_set_eq:NN \l_stex_current_symbol_str\l_stex_get_symbol_name_str
5936      \def\comp{\_varcomp}
5937      \_stex_term_omv:nnn
5938    }
5939  }
5940
5941  \NewDocumentCommand \varname { O{} m} {
5942    \group_begin:
5943    \stex_keys_set:nn{symname}{#1}
5944    \stex_get_var:n{#2}
5945    \__stex_expr_do_ref:nNn{
5946      \l_stex_key_pre_tl\l_stex_get_symbol_name_str\l_stex_key_post_tl
5947    }\varemph@uri{
5948      \str_set_eq:NN \l_stex_current_symbol_str\l_stex_get_symbol_name_str
5949      \def\comp{\_varcomp}
5950      \_stex_term_omv:nnn
5951    }
5952  }
5953
5954  \NewDocumentCommand \Varname { O{} m} {
5955    \group_begin:
5956    \stex_keys_set:nn{symname}{#1}
5957    \stex_get_var:n{#2}
5958    \__stex_expr_do_ref:nNn{
5959      \l_stex_key_pre_tl\exp_after:wN\_stex_capitalize:n\l_stex_get_symbol_name_str\l_stex_key
5960    }\varemph@uri{
5961      \str_set_eq:NN \l_stex_current_symbol_str\l_stex_get_symbol_name_str
5962      \def\comp{\_varcomp}
5963      \_stex_term_omv:nnn
5964    }
5965  }
5966
5967
5968  \cs_new_protected:Nn \__stex_expr_do_ref:nNn {
5969    \stex_if_html_backend:T{\ifvmode\indent\fi}
5970    \bool_if:NTF \l_stex_allow_semantic_bool{
5971      \str_set:Nx\l_stex_current_symbol_str
5972        {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
5973      \str_if_in:NnT \l_stex_get_symbol_name_str / {
5974        \str_set:Nx \l_stex_get_symbol_name_str {
5975          \exp_after:wN \_stex_split_slash: \l_stex_get_symbol_name_str
5976          /\_stex_args_end:
5977        }
5978      }
5979      \tl_clear:N \l_stex_current_term_tl
5980      \def\comp{\_comp}
5981      \let\compemph@uri#2
```

259

```
5982        #3{}{}{\comp{#1}}
5983      }{
5984        \msg_error:nnxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
5985      }
5986      \group_end:
5987    }
```

(*End definition for* `\symname` *and others. These functions are documented on page 78.*)

Highlighting

```
5988    ⟨@@=stex_notationcomps⟩
```

(margin labels)

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

```
5989    \cs_new_protected:Nn \_do_comp:nNn {
5990      \stex_pseudogroup_with:nn{\comp}{
5991        \def\comp##1{##1}
5992        \str_if_empty:NTF \l_stex_current_symbol_str {
5993          #3
5994        }{
5995          \stex_if_html_backend:TF {
5996            \stex_annotate:nn { shtml:#1 = \l_stex_current_symbol_str}{ #3 }
5997          }{
5998            \exp_args:Nno #2 { #3 } \l_stex_current_symbol_str
5999          }
6000        }
6001      }
6002    }
6003
6004    \cs_new_protected:Npn \_comp {
6005      \_do_comp:nNn {comp}\compemph@uri
6006    }
6007
6008    \cs_new_protected:Npn \_varcomp {
6009      \_do_comp:nNn {varcomp}\varemph@uri
6010    }
6011
6012    \cs_new_protected:Npn \_defcomp {
6013      \_do_comp:nNn {definiendum}\defemph@uri
6014    }
6015
6016    \cs_set_protected:Npn \comp {}
6017
6018    \cs_new_protected:Npn \compemph@uri #1 #2 {
6019        \compemph{ #1 }
6020    }
6021
6022    \cs_new_protected:Npn \compemph #1 {
6023        #1
6024    }
6025
6026    \cs_new_protected:Npn \defemph@uri #1 #2 {
6027        \defemph{#1}
6028    }
6029
```

```
6030 \cs_new_protected:Npn \defemph #1 {
6031     \ifmmode\else\expandafter\textbf\fi{#1}
6032 }
6033
6034 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
6035     \symrefemph{#1}
6036 }
6037
6038 \cs_new_protected:Npn \symrefemph #1 {
6039     \emph{#1}
6040 }
6041
6042 \cs_new_protected:Npn \varemph@uri #1 #2 {
6043     \varemph{#1}
6044 }
6045
6046 \cs_new_protected:Npn \varemph #1 {
6047     #1
6048 }
```

(*End definition for* `\comp` *and others. These functions are documented on page 79.*)

13.9   Mathematical Structures

```
6049 ⟨@@=stex_structures⟩
```

\this

```
6050 \cs_new_protected:Npn \stex_current_this: {
6051   { \bool_set_true:N \l_stex_allow_semantic_bool
6052     \tl_if_empty:NTF \l_stex_current_this_tl {{}}{
6053       \str_set:Nx \l_stex_current_symbol_str {
6054         \l_stex_current_module_str ? \l__stex_structures_name_str
6055       }
6056       \maincomp{\l_stex_current_this_tl}
6057     }
6058   }
6059 }
6060 \let \this \stex_current_this:
```

(*End definition for* `\this`*. This function is documented on page 85.*)

mathstructure (*env.*)

```
6061 \stex_new_stylable_env:nnnnnnn {mathstructure}{m O{}}{
6062   \__stex_structures_begin:nn{#1}{#2}
6063   \stex_smsmode_do:
6064 }{
6065   \stex_structural_feature_module_end:
6066   \__stex_structures_do_externals:
6067 }{}{}{}
6068
6069 \stex_keys_define:nnnn{mathstructure}{
6070   \tl_clear:N \l_stex_current_this_tl
6071   \str_clear:N \l__stex_structures_name_str
6072 }{
```

```
6073    this     .tl_set:N = \l_stex_current_this_tl ,
6074    unknown .code:n   = {
6075      \str_set_eq:NN \l__stex_structures_name_str \l_keys_key_str
6076    }
6077 }{}
6078
6079 \cs_new_protected:Nn \__stex_structures_begin:nn {
6080    \stex_keys_set:nn {mathstructure}{#2}
6081    \str_if_empty:NT \l__stex_structures_name_str {
6082      \str_set:Nn \l__stex_structures_name_str {#1}
6083    }
6084    \def\comp{\_comp}
6085
6086    \exp_args:Nne \use:nn { \stex_module_add_symbol:nnnnnnnN }
6087      { {#1}{\l__stex_structures_name_str}{0}{}{defed}{
6088        \l_stex_current_module_str / \l__stex_structures_name_str-module
6089      }}
6090      {}\stex_invoke_structure:
6091    \str_set:Nx \l_stex_macroname_str {#1}
6092    \stex_execute_in_module:x{
6093      \seq_clear:c{l_stex_structure_macros_\l_stex_current_module_str / \l__stex_structures_na
6094      \seq_put_right:cn{l_stex_structure_macros_\l_stex_current_module_str / \l__stex_structur
6095    }
6096    \exp_args:No \stex_structural_feature_module:nn
6097      {\l__stex_structures_name_str}{structure}
6098 }
6099
6100 \stex_sms_allow_env:n{mathstructure}
6101 \stex_deactivate_macro:Nn \mathstructure {module~environments}
6102 \stex_every_module:n {\stex_reactivate_macro:N \mathstructure}
6103
6104 \cs_new_protected:Nn \__stex_structures_do_externals: {
6105    \tl_set:Nn \l__stex_structures_replace_this_tl {####1}
6106    \exp_args:No \stex_iterate_symbols:nn{\g_stex_last_feature_str}{
6107      \__stex_structures_external_decl:nnnn{##5}{##4}{##3}{##8}
6108    }
6109 }
6110
6111 \cs_new_protected:Nn \__stex_structures_external_decl:nnnn {
6112    %\stex_debug:nn{structure}{
6113    %  Generating~external~declaration~\l__stex_structures_name_str/#3~in~
6114    %  \l_stex_current_module_str^^J
6115    %  \tl_to_str:n{#1}^^J\tl_to_str:n{#2}^^J\tl_to_str:n{#4}
6116    %}
6117    %\tl_set:Nn \l_stex_get_symbol_args_tl {#1}
6118    %\exp_args:Nnx \use:nn {\stex_module_add_symbol:nnnnnnnN} {
6119    %  {}{\l__stex_structures_name_str/#3}{\int_eval:n{#2 + 1}}
6120    %  {1i\tl_if_empty:nF{#1}{\_stex_map_args:N \__stex_structures_shift_argls:nn}}
6121    %  {defed}{typed}
6122    %}{#4}\stex_invoke_outer_field:
6123 }
6124
6125 \cs_new:Nn \__stex_structures_shift_argls:nn {
6126    \int_eval:n{#1+1}#2
```

```
6127   }
```

```
6128   \cs_new_protected:Nn \stex_get_mathstructure:n {
6129     \_stex_get_mathstructure:n{#1}
6130     \str_if_empty:NT \l_stex_get_structure_module_str {
6131       \msg_error:nnn{stex}{error/unknownstructure}{#1}
6132     }
6133   }
6134   \cs_new_protected:Nn \_stex_get_mathstructure:n {
6135     \str_clear:N \l_stex_get_structure_module_str
6136     \_stex_get_symbol:n{#1}
6137     \str_if_empty:NF \l_stex_get_symbol_name_str {
6138       \exp_args:No \tl_if_eq:NNT \l_stex_get_symbol_invoke_cs \stex_invoke_structure: {
6139         \str_set_eq:NN \l_stex_get_structure_module_str \l_stex_get_symbol_type_tl
6140       }
6141     }
6142   }
```

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

```
6143   \stex_new_stylable_env:nnnnnnn {extstructure}{m O{} m}{
6144     \seq_clear:N \l__stex_structures_imports_seq
6145     \clist_map_inline:nn{#3}{
6146       \stex_get_mathstructure:n{##1}
6147       \clist_map_inline:Nn \l_stex_get_symbol_type_tl {
6148         \exp_args:Ne \stex_if_module_exists:nT{\tl_to_str:n{####1}}{
6149           \seq_put_right:Nn \l__stex_structures_imports_seq{####1}
6150         }
6151       }
6152       \stex_execute_in_module:x{
6153         \seq_put_right:cn{l_stex_structure_macros_\l_stex_get_structure_module_str _seq}{#1}
6154       }
6155     }
6156     \__stex_structures_begin:nn{#1}{#2}
6157     \seq_map_inline:Nn\l__stex_structures_imports_seq{
6158       \stex_if_do_html:T {
6159         \hbox{\stex_annotate_invisible:nn
6160           {shtml:import={##1}} {}}
6161       }
6162       \stex_module_add_morphism:nonn
6163         {}{##1}{import}{}
6164       \stex_execute_in_module:x{
6165         \stex_activate_module:n{##1}
6166       }
6167     }
6168     \stex_smsmode_do:
6169   }{
6170     \stex_structural_feature_module_end:
6171     \__stex_structures_do_externals:
6172   }{}{}{}
6173
6174   \stex_sms_allow_env:n{extstructure}
```

```
6175 \stex_deactivate_macro:Nn \extstructure {module~environments}
6176 \stex_every_module:n {
6177   \stex_reactivate_macro:N \extstructure
6178 }
6179
6180 \cs_new:Nn \__stex_structures_extend_structure_i:NnnnnnnnN {
6181   \exp_not:n{#1{#2}{#3}{#4}{#5}{defed}}{\l__stex_structures_extmod_str,#7}\exp_not:n{{#8}#9}
6182 }
6183 \cs_new_protected:Nn \__stex_structures_extend_structure:nn {
6184   \stex_debug:nn{ext}{Extending~#1~by~#2}
6185   \str_set:Nn \l__stex_structures_extmod_str{#2}
6186   \tl_set:cx{#1}{
6187     \exp_after:wN \exp_after:wN \exp_after:wN
6188     \__stex_structures_extend_structure_i:NnnnnnnnN \cs:w #1 \cs_end:
6189   }
6190 }
6191
6192 \stex_new_stylable_env:nnnnnnn {extstructure*}{m}{
6193   \__stex_structures_new_extstruct_name:
6194   \seq_clear:N \l__stex_structures_imports_seq
6195   \stex_get_mathstructure:n{#1}
6196
6197   \clist_map_inline:Nn \l_stex_get_symbol_type_tl {
6198     \exp_args:Ne \stex_if_module_exists:nT{\tl_to_str:n{##1}}{
6199       \seq_put_right:Nn \l__stex_structures_imports_seq{##1}
6200     }
6201   }
6202
6203   \stex_execute_in_module:x{
6204     \seq_map_inline:cn{l_stex_structure_macros_\l_stex_get_structure_module_str _seq}{
6205       \exp_not:N \tl_if_exist:cT{####1}{
6206         \__stex_structures_extend_structure:nn{####1}{\l_stex_current_module_str/\l__stex_st
6207       }
6208     }
6209   }
6210
6211   \exp_args:No \__stex_structures_begin:nn\l__stex_structures_exstruct_name_str{}
6212
6213   \seq_map_inline:Nn \l__stex_structures_imports_seq {
6214     \stex_if_do_html:T {
6215       \stex_annotate_invisible:nn
6216         {shtml:import= {##1}} {}
6217     }
6218     \stex_module_add_morphism:nonn
6219       {}{##1}{import}{}
6220     \stex_execute_in_module:x{
6221       \stex_activate_module:n{##1}
6222     }
6223   }
6224
6225   \stex_smsmode_do:
6226 }{
6227   \prop_map_inline:cn{
6228     c_stex_module_ \l_stex_current_module_str _symbols_prop
```

```
6229     }{
6230       \__stex_structures_check_def:nnnnnnnn ##2
6231     }
6232     \stex_structural_feature_module_end:
6233     \__stex_structures_do_externals:
6234 }{}{}{}
6235
6236 \stex_sms_allow_env:n{extstructure*}
6237 \exp_after:wN \stex_deactivate_macro:Nn
6238   \cs:w extstructure*\cs_end: {module~environments}
6239 \stex_every_module:n {
6240   \exp_after:wN \stex_reactivate_macro:N \cs:w extstructure*\cs_end:
6241 }
6242
6243 \cs_new_protected:Nn \__stex_structures_check_def:nnnnnnnn {
6244   \tl_if_empty:nT{#5}{
6245     \msg_error:nnnn{stex}{error/needsdefiniens}{#2}{extstructure*}
6246   }
6247 }
6248
6249 \stex_every_module:n{ \str_set:Nn \l__stex_structures_extname_count 0}
6250
6251 \cs_new_protected:Nn \__stex_structures_new_extstruct_name: {
6252   \stex_do_up_to_module:n {
6253     \str_set:Nx \l__stex_structures_extname_count {\int_eval:n{\l__stex_structures_extname_c
6254   }
6255   \str_set:Nx \l__stex_structures_exstruct_name_str {EXTSTRUCT_\l__stex_structures_extname_c
6256 }
6257
```

Invoking structures:

```
6258 \cs_new_protected:Nn \stex_invoke_structure: {
6259   \tl_set:Nn \l__stex_structures_set_comp_tl {\__stex_structures_set_thiscomp:}
6260   \__stex_structures_invoke_top:n {}
6261 }
6262
6263 \cs_new_protected:Nn \__stex_structures_invoke_top:n {
6264   \stex_debug:nn{structure}{
6265     invoking~structure~{\l_stex_current_type_tl}<\tl_to_str:n{#1}>
6266   }
6267   \peek_charcode:NTF [ {
6268     \__stex_structures_merge:nw{#1}
6269   }{
6270     \__stex_structures_invokation_type:n {#1}
6271     \tl_set:Nn \l__stex_structures_this_tl {}
6272     \peek_charcode_remove:NTF ! {
6273       \peek_charcode:NTF [ {
6274         \__stex_structures_maybe_notation:w
6275       }{
6276         \__stex_structures_maybe_notation:w []
6277       }
6278     }{
6279       \__stex_structures_invoke_this:n
6280     }
```

```
6281      }
6282 }
6283
6284 \cs_new_protected:Npn \__stex_structures_merge:nw #1 [ #2 ] {
6285    \exp_args:Ne \stex_str_if_starts_with:nnTF {\tl_to_str:n{#2}}{comp=}{
6286      \__stex_structures_set_customcomp: #2 \__stex_structures_end:
6287      \__stex_structures_invoke_top:n{#1}
6288    }{
6289      \exp_args:Ne \stex_str_if_starts_with:nnTF {\tl_to_str:n{#2}}{this=}{
6290        \__stex_structures_set_thisnotation: #2 \__stex_structures_end:
6291        \__stex_structures_invoke_top:n{#1}
6292      }{
6293        \tl_if_empty:nTF{#1}{
6294          \__stex_structures_invoke_top:n{#2}
6295        }{
6296          \tl_if_empty:nTF{#2}{
6297            \__stex_structures_invoke_top:n{#1}
6298          }{
6299            \__stex_structures_invoke_top:n{#1,#2}
6300          }
6301        }
6302      }
6303    }
6304 }
6305
6306 \cs_new_protected:Npn \__stex_structures_set_thisnotation: this= #1 \__stex_structures_end:
6307    \tl_set:Nn \l_stex_return_notation_tl { \comp{#1} }
6308    \tl_set:Nn \l__stex_structures_set_comp_tl {}
6309 }
6310
6311 \cs_new_protected:Npn \__stex_structures_set_customcomp: comp= #1 \__stex_structures_end: {
6312    \tl_set:Nn \l__stex_structures_set_comp_tl {
6313      \__stex_structures_set_custom_comp:n{#1}
6314    }
6315    \tl_set:Nn \l_stex_return_notation_tl { \comp{} }
6316 }
```

The structure type:

```
6317 \cs_new_protected:Nn \__stex_structures_invokation_type:n {
6318    \__stex_structures_do_assign_list:n{#1}
6319    \clist_if_empty:NTF \l__stex_structures_fields_clist {
6320      \int_compare:nNnTF {\clist_count:N \l_stex_current_type_tl}
6321        = 1 {
6322          \tl_set:Nx \l__stex_structures_current_type_tl {
6323            \exp_args:No \exp_not:n \l_stex_current_redo_tl
6324            \_stex_term_oms_or_omv:nnn{}{}{}
6325          }
6326        }{
6327          \exp_args:No \__stex_structures_make_type:n \l_stex_current_type_tl
6328        }
6329    }{
6330      \int_compare:nNnTF {\clist_count:N \l_stex_current_type_tl}
6331        = 1 {
6332          \__stex_structures_make_type:n {}
6333        }{
```

```
6334        \exp_args:No \__stex_structures_make_type:n \l_stex_current_type_tl
6335      }
6336    }
6337 }
6338
6339 \cs_new_protected:Nn \__stex_structures_do_assign_list:n {
6340    \clist_clear:N \l__stex_structures_fields_clist
6341    \tl_if_empty:nF {#1} {
6342      \keyval_parse:NNn\TODO\__stex_structures_do_assign:nn{#1}
6343    }
6344 }
6345
6346 \cs_new_protected:Nn \__stex_structures_do_assign:nn {
6347    \clist_put_right:Nn \l__stex_structures_fields_clist {{#1}{#2}}
6348 }
6349
6350 \cs_new_protected:Nn \__stex_structures_make_type:n {
6351    \tl_if_empty:nTF{#1}{
6352      \seq_clear:N \l_tmpa_seq
6353    }{
6354      \seq_set_split:Nnn \l_tmpa_seq ,{#1}
6355      \seq_pop_right:NN \l_tmpa_seq \l_tmpa_tl
6356      \seq_reverse:N \l_tmpa_seq
6357    }
6358    \tl_set:Nx \l__stex_structures_current_type_tl {
6359      \symuse{Metatheory?module~type~merge}{
6360        {
6361          \exp_args:No \exp_not:n \l_stex_current_redo_tl
6362          \_stex_term_oms_or_omv:nnn{}{}{}
6363        }
6364        \seq_map_function:NN \l_tmpa_seq \__stex_structures_make_mod:n
6365        \clist_if_empty:NF \l__stex_structures_fields_clist {
6366          ,\symuse{Metatheory?anonymous~record}{
6367            \exp_args:Ne \tl_tail:n{
6368              \clist_map_function:NN \l__stex_structures_fields_clist \__stex_structures_make_
6369            }
6370          }
6371        }
6372      }
6373    }
6374 }
6375
6376 \cs_new:Nn \__stex_structures_make_mod:n {
6377    ,\symuse{Metatheory?module~type}{
6378      \stex_annotate:nn{shtml:term=OMMOD,shtml:head={#1}}{}
6379    }
6380 }
6381
6382 \cs_new:Nn \__stex_structures_make_oml:n {
6383    \__stex_structures_make_oml:nn #1
6384 }
6385 \cs_new:Nn \__stex_structures_make_oml:nn {
6386    ,\stex_annotate:nn{
6387      shtml:term=OML,
```

```
6388        shtml:head={#1}
6389      }{
6390        \_stex_annotate_force_break:n{
6391          \stex_annotate:nn{shtml:definiens={}}{\exp_not:n{#2!}}
6392        }
6393      }
6394    }
```

Insert the structure type as a term:

```
6395  \cs_new:Nn \__stex_structures_current_type: {
6396    %\exp_args:No \exp_not:n \l_stex_current_redo_tl
6397    \tl_set:Nn \exp_not:N \l_stex_current_term_tl {
6398      \exp_args:No\exp_not:n\l__stex_structures_current_type_tl
6399    }
6400    \_stex_term_oms_or_omv:nnn{}{}{}
6401  }
```

The structure type itself:

```
6402  \cs_new_protected:Npn \__stex_structures_maybe_notation:w [ #1 ] {
6403    \tl_set_eq:NN \l_stex_current_term_tl \l__stex_structures_current_type_tl
6404    \cs_if_exist:cTF{l_stex_notation_\l_stex_current_symbol_str _#1_cs}{
6405      \use:c{l_stex_notation_\l_stex_current_symbol_str _#1_cs}\group_end:
6406    }{
6407      \__stex_structures_make_prop:
6408      \__stex_structures_make_prop_assign:
6409      \__stex_structures_present_i:w [#1]
6410    }
6411  }
6412
6413  \cs_new_protected:Nn \__stex_structures_present: {
6414    \peek_charcode:NTF [ {
6415      \__stex_structures_present_i:w
6416    }{
6417      \__stex_structures_present:nn{}{}
6418    }
6419  }
6420
6421  \cs_new_protected:Npn \__stex_structures_present_i:w [#1] {
6422    \int_compare:nNnTF{\clist_count:n{#1}} = 1 {
6423      \__stex_structures_present:nn{}{#1}
6424    }{
6425      \peek_charcode:NTF [ {
6426        \__stex_structures_present_ii:nw{#1}
6427      }{
6428        \__stex_structures_present:nn{#1}{}
6429      }
6430    }
6431  }
6432
6433  %First: clist, second:notation-id
6434  \cs_new_protected:Npn \__stex_structures_present_ii:nw #1 [#2] {
6435    \__stex_structures_present:nn{#1}{#2}
6436  }
6437
6438  \cs_new_protected:Nn \__stex_structures_present:nn {
```

```
6439    \clist_clear:N \l__stex_structures_clist
6440    \tl_if_empty:nTF{#1}{
6441      \cs_set:Npn \l__stex_structures_cs ##1 ##2 ##3 {
6442        \tl_if_empty:nF{##2}{
6443          \__stex_structures_present_entry:nn {##1}{##3}
6444        }
6445      }
6446    }{
6447      \cs_set:Npn \l__stex_structures_cs ##1 ##2 ##3 {
6448        \exp_args:Ne \clist_if_in:nnT{\tl_to_str:n{#1}}{##1}{
6449          \__stex_structures_present_entry:nn {##1}{##3}
6450        }
6451      }
6452    }
6453    \prop_map_inline:Nn \l__stex_structures_prop {
6454      \l__stex_structures_cs {##1} ##2
6455    }
6456    \_stex_term_oms_or_omv:nnn{}{}{
6457      \exp_args:Nno \use:n{
6458        \bool_set_true:N \l_stex_allow_semantic_bool
6459        \symuse{Metatheory?mathematical~structure}[#2]
6460      }{\l__stex_structures_clist}
6461    }\group_end:
6462  }
6463
6464  \cs_new_protected:Nn \__stex_structures_present_entry:nn {
6465    \seq_if_in:NnTF \l__stex_structures_assigned_seq {#1}{
6466      \clist_put_right:Nn \l__stex_structures_clist {#2!}
6467    }{
6468      \exp_args:NNe \clist_put_right:Nn \l__stex_structures_clist {
6469        \_stex_next_symbol:n {
6470          \exp_args:No \exp_not:n \l__stex_structures_set_comp_tl
6471          \tl_set:Nn \exp_not:N \l__stex_structures_this_tl {
6472            \exp_args:No \exp_not:n \l__stex_structures_this_tl
6473          }
6474          \exp_not:n {
6475            \tl_set_eq:NN \this \l__stex_structures_this_tl
6476          }
6477          \tl_set:Nn \exp_not:N \l_stex_return_notation_tl {
6478            \exp_args:No \exp_not:n \l_stex_return_notation_tl
6479          }
6480        }
6481        \exp_not:n{#2!}
6482      }
6483    }
6484  }
6485
6486  \cs_new_protected:Npn \_thiscomp #1 #2 {
6487    {\tl_set:cn{this}{{}}#1{#2}\c_math_subscript_token{
6488      \group_begin:
6489      \bool_set_true:N \l_stex_allow_semantic_bool
6490      \l__stex_structures_this_tl
6491      \group_end:
6492    }
```

```
6493        }
6494   }

6495
6496   \cs_new_protected:Nn \__stex_structures_set_thiscomp: {
6497     \exp_args:Ne \tl_if_eq:NNF {\tl_head:N \maincomp} \_thiscomp {
6498       \edef\maincomp {\_thiscomp{\comp}}
6499     }
6500   }

6501
6502   \cs_new_protected:Nn \__stex_structures_set_custom_comp:n {
6503     \exp_args:Ne \tl_if_eq:NNF {\tl_head:N \maincomp} \_customthiscomp {
6504       \cs_set_protected:Npx \_customthiscomp ##1 {
6505         \group_begin:
6506           \bool_set_true:N \l_stex_allow_semantic_bool
6507           \exp_not:n{
6508             \cs_set:Npn \l__stex_structures_comp_cs ##1  {
6509               #1
6510             }
6511             \def\maincomp
6512           }{\comp}
6513           \exp_not:N\l__stex_structures_comp_cs{\comp{##1}}
6514         \group_end:
6515       }
6516       \def\maincomp {\_customthiscomp}
6517     }
6518   }

6519
```

this (of type structure):

```
6520
6521   \cs_new_protected:Nn \__stex_structures_invoke_this:n {
6522     \peek_charcode_remove:NTF ! {
6523       \exp_args:Nne\use:nn{
6524         \group_end:\symuse{Metatheory?of~type}[invisible]{#1}
6525       }{
6526         {\__stex_structures_current_type:}
6527       }
6528     }{
6529       \__stex_structures_invoke_maybe_field:nn{#1}
6530     }
6531   }

6532
6533   \cs_new_protected:Nn \__stex_structures_invoke_maybe_field:nn {
6534     \__stex_structures_make_prop:
6535     \__stex_structures_set_this:n{#1}
6536     \tl_if_empty:nTF{#2}{
6537       \__stex_structures_make_prop_assign:
6538       \__stex_structures_present:
6539     }{
6540       \__stex_structures_invoke_field:n{#2}
6541     }
6542   }

6543
6544   \cs_new_protected:Nn \__stex_structures_set_this:n {
6545     \tl_if_empty:nTF{#1}{
```

```
6546      %\tl_put_right:Nn \l_stex_current_redo_tl {
6547      %  \tl_clear:N \l__stex_structures_this_tl
6548      %}
6549    }{
6550      \tl_set:Nx \l__stex_structures_this_tl {{
6551        \bool_set_true:N \l_stex_allow_semantic_bool
6552        \tl_set:Nn \exp_not:N \this {
6553          \exp_args:No \exp_not:n \this
6554        }
6555        \exp_not:n{#1}
6556      }}
6557      \tl_set_eq:NN \this \l__stex_structures_this_tl
6558      % \l_stex_return_notation_tl
6559    }
6560  }
6561
6562  \cs_new_protected:Nn \__stex_structures_get_field_name:n {
6563    \str_set:Nx \l__stex_structures_field_name_str {
6564      \exp_args:Nne \use:n {\exp_after:wN \use_i:nn \use:n}
6565      {\prop_item:Nn \l__stex_structures_prop {#1}}
6566    }
6567    \str_if_empty:NT \l__stex_structures_field_name_str {
6568      \str_set:Nn \l__stex_structures_field_name_str {#1}
6569    }
6570  }
6571
6572  \cs_new_protected:Nn \__stex_structures_invoke_field:n {
6573    \prop_if_in:NnTF \l__stex_structures_prop {#1}{
6574      \__stex_structures_get_field_name:n{#1}
6575      \tl_clear:N \l__stex_structures_more_nextsymbol_tl
6576      %\exp_args:NNe \seq_if_in:NnF \l__stex_structures_assigned_seq {\tl_to_str:n{#1}}{
6577        \tl_set:Nx \l__stex_structures_more_nextsymbol_tl {
6578          \tl_set:Nn \exp_not:N \l__stex_structures_this_tl {
6579            \exp_args:No \exp_not:n \l__stex_structures_this_tl
6580          }
6581          \exp_not:n {
6582            \tl_set_eq:NN \this \l__stex_structures_this_tl
6583          }
6584          \tl_set:Nn \exp_not:N \l_stex_return_notation_tl {
6585            \exp_args:No \exp_not:n \l_stex_return_notation_tl
6586          }
6587          \exp_args:No \exp_not:n \l__stex_structures_set_comp_tl
6588        }
6589      %}
6590      \exp_args:NNx \use:nn \group_end: {
6591        \_stex_next_symbol:n {
6592          \exp_args:No \exp_not:n \l__stex_structures_redo_tl
6593          \tl_set:Nn \exp_not:N \l_stex_current_term_tl {
6594            \symuse{Metatheory?record~field}{
6595              \symuse{Metatheory?of~type}{
6596                \exp_args:No \exp_not:n \l__stex_structures_this_tl
6597              }{  \__stex_structures_current_type: }
6598            }{
6599              \stex_annotate:nn{shtml:term=OML,shtml:head={\l__stex_structures_field_name_str}
```

```
6600                    }
6601                  }
6602              \exp_args:No \exp_not:n \l__stex_structures_more_nextsymbol_tl
6603            }
6604          \exp_not:N \use_ii:nn
6605          \prop_item:Nn \l__stex_structures_prop {#1}
6606        }
6607    }{
6608      \msg_error:nnn{stex}{error/unknownfield}{#1}
6609    }
6610  }
6611
6612  \cs_new_protected:Nn \__stex_structures_make_prop: {
6613    \prop_clear:N \l__stex_structures_prop
6614    \seq_clear:N \l__stex_structures_seq
6615    \seq_clear:N \l__stex_structures_assigned_seq
6616    \tl_clear:N \l__stex_structures_redo_tl
6617    \__stex_structures_prop_do_decls:
6618    \__stex_structures_prop_do_notations:
6619  }
6620
6621  \cs_new_protected:Nn \__stex_structures_make_prop_assign: {
6622    \clist_if_empty:NF \l__stex_structures_fields_clist {
6623      \clist_map_inline:Nn \l__stex_structures_fields_clist {
6624        \__stex_structures_make_prop_assign:nn ##1
6625      }
6626    }
6627  }
6628
6629  \cs_new_protected:Nn \__stex_structures_make_prop_assign:nn {
6630    \prop_if_in:NnTF \l__stex_structures_prop {#1}{
6631      \exp_args:NNe \seq_put_right:Nn \l__stex_structures_assigned_seq {\tl_to_str:n{#1}}
6632      \exp_args:Nne \use:nn {\__stex_structures_make_prop_assign_replace:nnnn {#1}{#2}}
6633      {\prop_item:Nn \l__stex_structures_prop {#1}}
6634    }{
6635      \msg_error:nnn{stex}{error/unknownfieldass}{#1}
6636    }
6637  }
6638  \cs_new_protected:Nn \__stex_structures_make_prop_assign_replace:nnnn {
6639    \prop_put:Nnn \l__stex_structures_prop {#1}{{#3}{#2}}
6640    \tl_if_empty:nF{#3}{
6641      \tl_set:cn{#1}{ #2 }
6642      \tl_put_right:Nn \l__stex_structures_redo_tl {
6643        \tl_set:cn{#1}{ #2 }
6644      }
6645    }
6646  }
6647
6648  \cs_new_protected:Nn \__stex_structures_prop_do_decls: {
6649    \exp_args:No \stex_iterate_symbols:nn \l_stex_current_type_tl {
6650      \tl_if_empty:nTF{##2}{
6651        \__stex_structures_do_decl_nomacro:nnnnnnnnn{##3}
6652      }{
6653        \__stex_structures_do_decl:nnnnnnnnn{##2}
```

```
6654        }
6655        {##1}{##3}{##4}{##5}{##6}{##7}{##8}{##9}
6656      }
6657 }
6658
6659 \cs_new_protected:Nn \__stex_structures_do_decl_nomacro:nnnnnnnnn {
6660    \prop_if_in:NnF \l__stex_structures_prop {#1} {
6661      \seq_put_left:Nx \l__stex_structures_seq {\tl_to_str:n{#2?#3}}
6662      \prop_put:Nnn \l__stex_structures_prop {#1}{
6663        {}{
6664          \_stex_invoke_symbol:nnnnnnnN
6665          {#2}
6666          {#3}
6667          {#4}{#5}{#6}{#7}{#8}#9
6668        }
6669      }
6670    }
6671 }
6672
6673 \cs_new_protected:Nn \__stex_structures_do_decl:nnnnnnnnn {
6674    \prop_if_in:NnF \l__stex_structures_prop {#1} {
6675      \seq_put_left:Nx \l__stex_structures_seq {\tl_to_str:n{#2?#3}}
6676      \prop_put:Nnn \l__stex_structures_prop {#1}{
6677        {#3}{
6678          \_stex_invoke_symbol:nnnnnnnN
6679          {#2}
6680          {#3}
6681          {#4}{#5}{#6}{#7}{#8}#9
6682        }
6683      }
6684    }
6685    %\tl_set:cn{#1}{
6686    %  \_stex_invoke_symbol:nnnnnnnN
6687    %  {#2}{#3}{#4}{#5}{#6}{#7}{#8}#9
6688    %}
6689    %\tl_put_right:Nn \l__stex_structures_redo_tl {
6690    %  \tl_set:cn{#1}{
6691    %    \_stex_invoke_symbol:nnnnnnnN
6692    %    {#2}{#3}{#4}{#5}{#6}{#7}{#8}#9
6693    %  }
6694    %}
6695 }
6696
6697 \cs_new_protected:Nn \__stex_structures_prop_do_notations: {
6698    \exp_args:No \stex_iterate_notations:nn\l_stex_current_type_tl{
6699      \exp_args:NNe \seq_if_in:NnT \l__stex_structures_seq {\tl_to_str:n{##1}}{
6700        \tl_put_right:Nn \l__stex_structures_redo_tl {
6701          \cs_if_exist:cF{l_stex_notation_##1 _##2_cs}{
6702            \tl_set:cn{l_stex_notation_##1 _##2_cs}{##4}
6703          }
6704          \cs_if_exist:cF{l_stex_notation_##1 __cs}{
6705            \tl_set:cn{l_stex_notation_##1 __cs}{##4}
6706          }
6707        }
```

```
6708        \cs_if_exist:cF{l_stex_notation_##1 _##2_cs}{
6709          \tl_set:cn{l_stex_notation_##1 _##2_cs}{##4}
6710        }
6711        \cs_if_exist:cF{l_stex_notation_##1 __cs}{
6712          \tl_set:cn{l_stex_notation_##1 __cs}{##4}
6713        }
6714        \tl_if_empty:nF{##5}{
6715          \tl_put_right:Nn \l__stex_structures_redo_tl {
6716            \cs_if_exist:cF{l_stex_notation_##1 _op_##2_cs}{
6717              \tl_set:cn{l_stex_notation_##1 _op_##2_cs}{##5}
6718            }
6719            \cs_if_exist:cF{l_stex_notation_##1 _op__cs}{
6720              \tl_set:cn{l_stex_notation_##1 _op__cs}{##5}
6721            }
6722          }
6723          \cs_if_exist:cF{l_stex_notation_##1 _op_##2_cs}{
6724            \tl_set:cn{l_stex_notation_##1 _op_##2_cs}{##5}
6725          }
6726          \cs_if_exist:cF{l_stex_notation_##1 _op__cs}{
6727            \tl_set:cn{l_stex_notation_##1 _op__cs}{##5}
6728          }
6729        }
6730      }
6731    }
6732 }
```

\usestructure

```
6733 \cs_new_protected:Npn \usestructure #1 {
6734   \stex_get_mathstructure:n{ #1 }
6735   \seq_clear:N \l__stex_structures_imports_seq
6736   \clist_map_inline:Nn \l_stex_get_symbol_type_tl {
6737     \exp_args:Ne \stex_if_module_exists:nT{\tl_to_str:n{##1}}{
6738       \seq_put_right:Nn \l__stex_structures_imports_seq{##1}
6739     }
6740   }
6741   \seq_map_inline:Nn \l__stex_structures_imports_seq {
6742     \stex_if_do_html:T {
6743       \hbox{\stex_annotate_invisible:nn
6744         {shtml:usemodule=##1} {}}
6745     }
6746     \stex_activate_module:n {##1}
6747   }
6748 }
```

(*End definition for* \usestructure. *This function is documented on page 85.*)

13.10   Statements

```
6749 ⟨@@=stex_statements⟩
6750
6751 \stex_keys_define:nnnn{statement}{
6752   \str_clear:N \l_stex_key_name_str
6753   \str_clear:N \l_stex_key_macroname_str
```

274

```
6754    \clist_clear:N \l_stex_key_for_clist
6755    \str_clear:N \l_stex_key_args_str
6756    \tl_clear:N \l_stex_key_type_tl
6757    \tl_clear:N \l_stex_key_def_tl
6758    \tl_clear:N \l_stex_key_return_tl
6759    \clist_clear:N \l_stex_key_argtypes_clist
6760 }{
6761    name     .str_set:N = \l_stex_key_name_str ,
6762    for      .clist_set:N = \l_stex_key_for_clist ,
6763    macro    .str_set:N = \l_stex_key_macroname_str ,
6764    % start   .str_set:N = \l_stex_key_title_str , % TODO remove
6765    type  .tl_set:N = \l_stex_key_type_tl ,
6766    judgment .code:n = {},
6767    from .code:n= {}, % TODO remove
6768    to .code:n={} % TODO remove
6769 }{id,title,style,symargs}
```

\stex_new_statement:nn

```
6770 \cs_new_protected:Npn \_stex_do_for_list: {
6771    \seq_clear:N \l_stex_fors_seq
6772    \clist_map_inline:Nn \l_stex_key_for_clist {
6773       \exp_args:Ne\stex_get_symbol:n{\tl_to_str:n{##1}}
6774       \seq_put_right:Nx \l_stex_fors_seq
6775          {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
6776    }
6777 }
6778
6779 \cs_new_protected:Nn \__stex_statements_setup:nn {
6780    \str_if_empty:NF \l_stex_key_macroname_str {
6781       \str_if_empty:NT \l_stex_key_name_str {
6782          \str_set_eq:NN \l_stex_key_name_str \l_stex_key_macroname_str
6783       }
6784    }
6785    \_stex_do_for_list:
6786    \str_if_empty:NF \l_stex_key_name_str {
6787       \__stex_statements_force_id:
6788       \seq_put_right:Nx \l_stex_fors_seq {
6789          \l_stex_current_module_str ? \l_stex_key_name_str
6790       }
6791       \str_set_eq:NN \l_stex_macroname_str \l_stex_key_macroname_str
6792       \str_set:Nn \l_stex_key_role_str {#2}
6793       \stex_symdecl_do:
6794       \exp_args:Nnx \use:nn {\stex_module_add_symbol:nnnnnnnN}{
6795          {\l_stex_key_macroname_str}{\l_stex_key_name_str}
6796          {\int_use:N \l_stex_get_symbol_arity_int}
6797          {\l_stex_get_symbol_args_tl}
6798          {#1}{}{}\stex_invoke_symbol:
6799       }
6800       \stex_if_do_html:T \_stex_symdecl_html:
6801    }
6802    \str_clear:N \l__stex_statements_uri_str
6803    \str_if_empty:NTF \l_stex_key_name_str {
6804       \stex_debug:nn{statement}{no~name}
6805       \int_compare:nNnTF {\seq_count:N \l_stex_fors_seq} = 1 {
```

```
6806        \str_set:Nx \l__stex_statements_uri_str {\seq_item:Nn \l_stex_fors_seq 1}
6807        \stex_debug:nn{statement}{for:~\l__stex_statements_uri_str}
6808      }{
6809        \stex_debug:nn{statement}{no~for}
6810      }
6811    }{
6812      \str_set:Nx \l__stex_statements_uri_str {\l_stex_current_module_str ? \l_stex_key_name_s
6813      \stex_debug:nn{statement}{name:~\l__stex_statements_uri_str}
6814    }
6815  }
6816
6817  \cs_new:Nn \__stex_statements_html_keyvals:nn {
6818    shtml:#1={},
6819    shtml:inline={#2},
6820    \seq_if_empty:NF \l_stex_fors_seq {,
6821      shtml:fors={\seq_use:Nn \l_stex_fors_seq ,}
6822    }
6823    \str_if_empty:NF \l_stex_key_id_str {,
6824      shtml:id={\stex_uri_use:N \l_stex_current_doc_uri ? \l_stex_key_id_str}
6825    }
6826    \clist_if_empty:NF \l_stex_key_style_clist {,
6827      shtml:styles={\l_stex_key_style_clist}
6828    }
6829  }
6830
6831  \cs_new_protected:Nn \stex_new_statement:nnn {
6832    \stex_new_stylable_env:nnnnnnn {#1}{O{}}{
6833      \stex_keys_set:nn{statement}{##1}
6834      #3
6835
6836      \stex_if_smsmode:F {
6837        \exp_args:Nne \begin{stex_annotate_env}{
6838          \__stex_statements_html_keyvals:nn{#1}{false}
6839        }
6840        \tl_set_eq:NN \thistitle \l_stex_key_title_tl
6841        \str_set_eq:NN \thisname \l_stex_key_name_str
6842        \clist_set_eq:NN \thisfor \l_stex_key_for_str
6843        \stex_if_html_backend:TF {
6844          \noindent
6845          \stex_annotate:nn{shtml:statementtitle={}}{\_stex_annotate_force_break:n\l_stex_key_
6846        }
6847        \stex_style_apply:
6848      }
6849      \_stex_do_id:
6850      \stex_smsmode_do:
6851    }{
6852      \stex_if_smsmode:F {
6853        \stex_if_html_backend:F \stex_style_apply:
6854        \end{stex_annotate_env}
6855      }
6856    }{}{}{s}
6857    \stex_sms_allow_env:n{s#1}
6858
6859    \tl_if_empty:nF{#2}{
```

```
\exp_after:wN \NewDocumentCommand \cs:w inline#2\cs_end: { O{} m}{
  \group_begin:
    \stex_keys_set:nn{statement}{##1}
    #3
    \_stex_do_id:
    \stex_if_smsmode:F{
      \exp_args:Ne \stex_annotate:nn{\__stex_statements_html_keyvals:nn{#1}{true}}{
        \_stex_annotate_force_break:n{##2}
      }
    }
  \group_end:
  \stex_smsmode_do:
}
\exp_after:wN \stex_sms_allow_escape:N\cs:w inline#2\cs_end:
}
}

\cs_new_protected:Nn \__stex_statements_setup_def: {
  \stex_if_smsmode:F{
    \seq_map_inline:Nn \l_stex_fors_seq {
      \stex_ref_new_sym_target:n{##1}
    }
  }
  \stex_reactivate_macro:N \definiendum
  \stex_reactivate_macro:N \defnotation
  \stex_reactivate_macro:N \definame
  \stex_reactivate_macro:N \Definame
  \stex_reactivate_macro:N \varbind
}

\cs_new_protected:Nn \__stex_statements_force_id: {
  \str_if_empty:NT \l_stex_key_id_str {
    \_stex_ref_new_id:n{}
    \str_set_eq:NN \l_stex_key_id_str \l__stex_refs_str
  }
}

\stex_new_statement:nnn{definition}{def}{
  \__stex_statements_force_id:
  \__stex_statements_setup:nn{}{}
  \__stex_statements_setup_def:
  \stex_reactivate_macro:N \definiens
}
\stex_new_statement:nnn{assertion}{ass}{
  \__stex_statements_setup:nn{}{assertion}
  \stex_if_smsmode:F{
    \seq_map_inline:Nn \l_stex_fors_seq {
      \stex_ref_new_sym_target:n{##1}
    }
  }
  \stex_reactivate_macro:N \varbind
  \stex_reactivate_macro:N \conclusion
  \stex_reactivate_macro:N \premise
  \stex_reactivate_macro:N \definiendum
```

```
6914    \stex_reactivate_macro:N \defnotation
6915    \stex_reactivate_macro:N \definame
6916    \stex_reactivate_macro:N \Definame
6917 }
6918 \stex_new_statement:nnn{example}{ex}{\__stex_statements_setup:nn{}{example}}
6919 \stex_new_statement:nnn{paragraph}{}{
6920    \clist_if_in:NnTF \l_stex_key_style_clist {symdoc}{
6921      \__stex_statements_force_id:
6922      \__stex_statements_setup:nn{}{}
6923      \__stex_statements_setup_def:
6924    }{
6925      \__stex_statements_setup:nn{}{}
6926    }
6927 }
```

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

definiendum

```
6928 \cs_new_protected:Nn \__stex_statements_do_defref:nn {
6929    \stex_if_html_backend:T{\ifvmode\indent\fi}
6930    \group_begin:
6931    \stex_get_symbol:n{#1}
6932    \bool_if:NTF \l_stex_allow_semantic_bool{
6933      \str_set:Nx\l_stex_current_symbol_str
6934        {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
6935      \str_if_in:NnT \l_stex_get_symbol_name_str / {
6936        \str_set:Nx \l_stex_get_symbol_name_str {
6937          \exp_after:wN \_stex_split_slash: \l_stex_get_symbol_name_str
6938          /\_stex_args_end:
6939        }
6940      }
6941      \exp_args:No \stex_ref_new_sym_target:n \l_stex_current_symbol_str
6942      \def\comp{\_defcomp}
6943      \stex_annotate:nn{shtml:definiendum=\l_stex_current_symbol_str}{\comp{#2}}
6944    }{
6945      \msg_error:nnxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
6946    }
6947    \group_end:
6948 }
6949
6950 \NewDocumentCommand \defnotation{ m } {
6951    \_stex_next_symbol:n { \def\comp{\_defcomp}}#1
6952 }
6953 \stex_deactivate_macro:Nn \defnotation {definition~environments}
6954
6955 \NewDocumentCommand \definiendum { O{} m m} {
6956    \stex_keys_set:nn{symname}{ #1 }
6957    \__stex_statements_do_defref:nn{#2}{#3}
6958 }
6959 \stex_deactivate_macro:Nn \definiendum {definition~environments}
6960
6961 \NewDocumentCommand \definame { O{} m } {
6962    \stex_keys_set:nn{symname}{#1}
6963    \__stex_statements_do_defref:nn{#2}{
```

```
6964        \l_stex_key_pre_tl\l_stex_get_symbol_name_str\l_stex_key_post_tl
6965    }
6966 }
6967 \stex_deactivate_macro:Nn \definame {definition~environments}
6968
6969 \NewDocumentCommand \Definame { O{} m } {
6970    \stex_keys_set:nn{symname}{#1}
6971    \__stex_statements_do_defref:nn{#2}{
6972      \l_stex_key_pre_tl\exp_after:wN\_stex_capitalize:n\l_stex_get_symbol_name_str\l_stex_key
6973    }
6974 }
6975 \stex_deactivate_macro:Nn \Definame {definition~environments}
6976
6977
6978 \NewDocumentCommand \definiens { O{} m }{
6979    \group_begin:
6980    \str_clear:N \l_stex_get_symbol_name_str
6981    \tl_if_empty:nF {#1} {
6982      \stex_get_symbol:n { #1 }
6983      \str_set:Nx \l__stex_statements_uri_str
6984        {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
6985    }
6986    \str_if_empty:NT \l__stex_statements_uri_str {
6987      \msg_error:nn{stex}{error/definiensfor}
6988    }
6989    \stex_debug:nn{definiens}{Checking~\l__stex_statements_uri_str}
6990
6991    \exp_args:No \_stex_add_definiens:nn \l__stex_statements_uri_str{#2}
6992
6993    \group_end:
6994    \stex_smsmode_do:
6995 }
6996 \stex_deactivate_macro:Nn \definiens {definition~environments}
6997 \stex_sms_allow_escape:N \definiens
6998
6999 \cs_new_protected:Nn \_stex_add_definiens:nn {
7000    \exp_args:Nno \stex_str_if_starts_with:nnT{#1} \l_stex_current_module_str {
7001      \prop_map_inline:cn{c_stex_module_\l_stex_current_module_str _symbols_prop}{
7002        \stex_debug:nn{definiens}{#1 == \l_stex_current_module_str?##1}
7003        \str_if_eq:noT {#1} {\l_stex_current_module_str?##1}{
7004          \prop_map_break:n{\_stex_add_definiens_inner:nnnnnnnn ##2}
7005        }
7006      }
7007    }
7008    \stex_if_smsmode:F{
7009      \stex_annotate:nn{ shtml:definiens={#1}}{
7010        #2 %\_stex_annotate_force_break:n{ #2 }
7011      }
7012    }
7013 }
7014
7015 \cs_new_protected:Nn \_stex_add_definiens_inner:nnnnnnnn {
7016    \stex_debug:nn{definiens}{Adding~definiens~to~\l_stex_current_module_str?#2}
7017    \prop_gput:cnn{c_stex_module_\l_stex_current_module_str _symbols_prop}
```

```
7018        {#2}{{#1}{#2}{#3}{#4}{defed}{#6}{#7}{#8}}
7019  }
7020
7021  \NewDocumentCommand \varbind {m} {
7022    \clist_map_inline:nn {#1} {
7023      \stex_get_var:n {##1}
7024      \stex_if_do_html:T {
7025        \stex_annotate_invisible:nn {shtml:bind=\l_stex_get_symbol_name_str}{}
7026      }
7027    }
7028  }
7029  \stex_deactivate_macro:Nn \varbind {definition~or~assertion~environments}
7030
7031  \NewDocumentCommand \conclusion { O{} m } {
7032    \group_begin:
7033    \str_clear:N \l_stex_get_symbol_name_str
7034    \tl_if_empty:nF {#1} {
7035      \stex_get_symbol:n { #1 }
7036      \str_set:Nx \l__stex_statements_uri_str
7037        {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
7038    }
7039    \str_if_empty:NT \l__stex_statements_uri_str {
7040      \msg_error:nn{stex}{error/conclusionfor}
7041    }
7042    \stex_annotate:nn{ shtml:conclusion=\l__stex_statements_uri_str}{
7043      #2 %\_stex_annotate_force_break:n{ #2 }
7044    }
7045    \group_end:
7046  }
7047  \stex_deactivate_macro:Nn \conclusion {assertion~environments}
7048
7049  \NewDocumentCommand \premise {O{} m} {
7050    \tl_if_empty:nF {#1} {
7051      \stex_debug:nn{Here:}{Variable~#1}
7052      \exp_args:NNx\exp_args:Nnx\vardef{v#1}{#1}
7053    }
7054    \stex_annotate:nn{shtml:premise={#1}}{#2}
7055  }
7056  \stex_deactivate_macro:Nn \premise {assertion~environments}
```

(*End definition for* `definiendum`. *This function is documented on page* *89*.)

13.11   Proofs

We first define some keys for the sproof environment.

```
7057  ⟨@@=stex_proof⟩
7058  \stex_keys_define:nnnn{ spf }{
7059  \tl_clear:N \l_stex_key_for_clist
7060  \tl_clear:N \l_stex_key_from_tl
7061  \tl_set_eq:NN \l_stex_key_proofend_tl \__stex_proof_proof_box_tl
7062  \tl_clear:N \l_stex_key_continues_tl
7063  \tl_clear:N \l_stex_key_term_tl
7064  \tl_clear:N \l_stex_key_functions_tl
```

```
7065 \tl_clear:N \l_stex_key_method_tl
7066   \bool_set_false:N \l_stex_key_hide_bool
7067 }{
7068   for          .clist_set:N  = \l_stex_key_for_clist ,
7069   from         .tl_set:N     = \l_stex_key_from_tl ,
7070   proofend     .tl_set:N     = \l_stex_key_proofend_tl,
7071   continues    .tl_set:N     = \l_stex_key_continues_tl,
7072   functions    .tl_set:N     = \l_stex_key_functions_tl,
7073   term         .tl_set:N     = \l_stex_key_term_tl,
7074   method       .tl_set:N     = \l_stex_key_method_tl,
7075   hide         .bool_set:N   = \l_stex_key_hide_bool
7076 }{id,style,title}
7077
7078 \bool_set_true:N \l__stex_proof_inc_counter_bool
```

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.

```
7079 \intarray_new:Nn\l__stex_proof_counter_intarray{50}
7080 \cs_new_protected:Npn \__stex_proof_insert_number: {
7081   \int_set:Nn \l_tmpa_int {1}
7082   \bool_while_do:nn {
7083     \int_compare_p:nNn {
7084       \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
7085     } > 0
7086   }{
7087     \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int .
7088     \int_incr:N \l_tmpa_int
7089   }
7090 }
7091 \cs_new_protected:Nn \__stex_proof_number_as_string:N {
7092   \str_clear:N #1
7093   \int_set:Nn \l_tmpa_int {1}
7094   \bool_while_do:nn {
7095     \int_compare_p:nNn {
7096       \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
7097     } > 0
7098   }{
7099     \str_put_right:Nx #1 {\intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int .}
7100     \int_incr:N \l_tmpa_int
7101   }
7102 }
7103
7104 \cs_new_protected:Npn \__stex_proof_inc_counter: {
7105   \int_set:Nn \l_tmpa_int {1}
7106   \bool_while_do:nn {
7107     \int_compare_p:nNn {
7108       \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
7109     } > 0
```

```
7110    }{
7111      \int_incr:N \l_tmpa_int
7112    }
7113    \int_compare:nNnF \l_tmpa_int = 1 {
7114      \int_decr:N \l_tmpa_int
7115    }
7116    \intarray_gset:Nnn \l__stex_proof_counter_intarray \l_tmpa_int {
7117      \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int + 1
7118    }
7119  }
7120
7121  \cs_new_protected:Npn \__stex_proof_add_counter: {
7122    \int_set:Nn \l_tmpa_int {1}
7123    \bool_while_do:nn {
7124      \int_compare_p:nNn {
7125        \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
7126      } > 0
7127    }{
7128      \int_incr:N \l_tmpa_int
7129    }
7130    \intarray_gset:Nnn \l__stex_proof_counter_intarray \l_tmpa_int { 1 }
7131  }
7132
7133  \cs_new_protected:Npn \__stex_proof_remove_counter: {
7134    \int_set:Nn \l_tmpa_int {1}
7135    \bool_while_do:nn {
7136      \int_compare_p:nNn {
7137        \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
7138      } > 0
7139    }{
7140      \int_incr:N \l_tmpa_int
7141    }
7142    \int_decr:N \l_tmpa_int
7143    \intarray_gset:Nnn \l__stex_proof_counter_intarray \l_tmpa_int { 0 }
7144  }
```

spfsketch

```
7145  \newenvironment{spfsketchenv}{}{}
7146  \stex_new_stylable_cmd:nnnn{spfsketch}{O{} m}{\par
7147    \begin{spfsketchenv}
7148    \stex_keys_set:nn{spf}{#1}
7149    \_stex_do_for_list:
7150    \_stex_do_id:
7151    \exp_args:Ne \stex_annotate:nn{
7152      shtml:proofsketch={
7153        \seq_if_empty:NF \l_stex_fors_seq {
7154          \seq_use:Nn \l_stex_fors_seq ,
7155        }
7156      }
7157    }{
7158      \stex_style_apply:
7159      #2
7160    }
7161    \end{spfsketchenv}
```

```
7162 }{
7163   \noindent\emph{\spfsketchenvautorefname :}~
7164 }
```

(*End definition for* `spfsketch`. *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

```
7165 \tl_set:Nn \__stex_proof_proof_box_tl {
7166   \ltx@ifpackageloaded{amssymb}{$\square$}{
7167     \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
7168   }
7169 }
7170
7171 \tl_set:Nn \sproofend {
7172   \tl_if_empty:NF \l_stex_key_proofend_tl {
7173     \hfil\null\nobreak\hfill\l_stex_key_proofend_tl\par\smallskip
7174   }
7175 }
```

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

\stexcommentfont

```
7176 \cs_new_protected:Npn \stexcommentfont {
7177   \small\itshape
7178 }
```

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

sproof (*env.*)

```
7179 \cs_new_protected:Nn \__stex_proof_start_list:n {
7180   \begin{list}{}{
7181     \setlength\topsep{0pt}
7182     \setlength\parsep{0pt}
7183     \setlength\rightmargin{0pt}
7184   }\item[#1]
7185 }
7186 \cs_new_protected:Nn \__stex_proof_end_list: {
7187   \end{list}
7188 }
7189
7190 \cs_new_protected:Nn \__stex_proof_html: {
7191   \stex_annotate_invisible:n{\hbox{
7192     \tl_if_empty:NF \l_stex_key_term_tl {
7193       $\stex_annotate:nn{shtml:proofterm={}}{\l_stex_key_term_tl}$
7194     }
7195     \tl_if_empty:NF \l_stex_key_method_tl {
7196       \stex_annotate:nn{shtml:proofmethod={}}{\l_stex_key_method_tl}
7197     }
7198   }}
7199 }
7200
7201 \cs_new_protected:Nn \__stex_proof_html_env:n {
7202   \exp_args:Nne \begin{stex_annotate_env}{
7203     shtml:#1={
```

```
7204      \seq_if_empty:NF \l_stex_fors_seq {
7205         \seq_use:Nn \l_stex_fors_seq ,
7206      }
7207    }
7208    \bool_if:NT \l_stex_key_hide_bool {,
7209       shtml:proofhide=true
7210    }
7211  }
7212  \__stex_proof_html:
7213 }
7214
7215 \bool_set_false:N \l__stex_proof_in_spfblock_bool
7216 \cs_new_protected:Nn \__stex_proof_begin_proof:nn {\par
7217   \intarray_gzero:N \l__stex_proof_counter_intarray
7218   \intarray_gset:Nnn \l__stex_proof_counter_intarray 1 1
7219   \stex_keys_set:nn{spfsteps}{#1}
7220   \_stex_do_for_list:
7221   \stex_if_do_html:T {
7222      \__stex_proof_html_env:n{proof}
7223   }
7224   \seq_map_inline:Nn \l_stex_fors_seq {
7225      \stex_debug:nn{definiens}{Adding~definiens~to~##1}
7226      \_stex_add_definiens:nn {##1}{\STEXinvisible{proven}}
7227   }
7228   \stex_style_apply:
7229   \_stex_do_id:
7230   \stex_reactivate_macro:N \subproof
7231   \stex_reactivate_macro:N \spfstep
7232   \stex_reactivate_macro:N \conclude
7233   \stex_reactivate_macro:N \assumption
7234   \stex_reactivate_macro:N \eqstep
7235   \stex_reactivate_macro:N \yield
7236   \stex_reactivate_macro:N \spfblock
7237   \stex_reactivate_macro:N \spfjust
7238   \stex_annotate:nn{shtml:prooftitle={}}{#2}
7239   \stex_if_do_html:T{
7240      \begin{stex_annotate_env}{shtml:proofbody={}}
7241   }
7242 }
7243 \stex_new_stylable_env:nnnnnnn{proof}{O{} m}{
7244   \__stex_proof_begin_proof:nn{#1}{#2}
7245   \bool_set_true:N\l__stex_proof_in_spfblock_bool\__stex_proof_start_list:n{}
7246   \group_begin:\stexcommentfont
7247 }{
7248   \stex_style_apply:
7249   \stex_if_do_html:T{\end{stex_annotate_env}\end{stex_annotate_env}}
7250 }{
7251   \emph{\sproofautorefname :}~
7252 }{
7253   \sproofend
7254 }{s}
7255 \AddToHook{env/sproof/end}{
7256   \bool_if:NT\l__stex_proof_in_spfblock_bool {
7257      \group_end:\__stex_proof_end_list:
```

```
7258      }
7259 }
7260
7261 \stex_new_stylable_env:nnnnnnn{proof*}{O{}}{
7262   \__stex_proof_begin_proof:nn{#1}{}
7263   \bool_set_false:N\l__stex_proof_in_spfblock_bool
7264 }{
7265   \stex_style_apply:
7266   \stex_if_do_html:T{\end{stex_annotate_env}\end{stex_annotate_env}}
7267 }{
7268   \emph{Proof:}~
7269 }{
7270   \sproofend
7271 }{s}
```

subproof (*env.*)

```
7272 \str_set_eq:NN \subproofautorefname \spfstepautorefname
7273 \stex_new_stylable_env:nnnnnnn{subproof}{s O{} m}{\par
7274   \stex_keys_set:nn{spf}{#2}
7275   \__stex_do_for_list:
7276   \stex_if_do_html:T {
7277     \__stex_proof_html_env:n{subproof}
7278   }
7279   \seq_map_inline:Nn \l_stex_fors_seq {
7280     \stex_debug:nn{definiens}{Adding~definiens~to~##1}
7281     \__stex_add_definiens:nn {##1}{\STEXinvisible{proven}}
7282   }
7283
7284   \IfBooleanTF #1 {
7285     \stex_style_apply:
7286     \str_if_empty:NF \l_stex_key_id_str {
7287       \__stex_proof_number_as_string:N \@currentlabel
7288       \str_set:Nx \@currentHref{subproof.\@currentlabel}
7289       \_stex_do_id:
7290     }
7291     \bool_set_false:N \l__stex_proof_in_spfblock_bool
7292     \stex_annotate:nn{shtml:prooftitle={}}{#3}
7293   }{
7294     \bool_if:NTF \l__stex_proof_in_spfblock_bool {
7295       \str_if_empty:NF \l_stex_key_id_str {
7296         \__stex_proof_number_as_string:N \@currentlabel
7297         \str_set:Nx \@currentHref{subproof.\@currentlabel}
7298         \_stex_do_id:
7299       }
7300       \__stex_proof_start_list:n\__stex_proof_insert_number:
7301       \stex_annotate:nn{shtml:prooftitle={}}{#3}
7302       \__stex_proof_add_counter:
7303       \stex_style_apply:
7304     }{
7305       \stex_annotate:nn{shtml:prooftitle={}}{#3}
7306       \stex_style_apply:
7307       \_stex_do_id:
7308     }
7309   }
```

```
7310    \stex_if_do_html:T{
7311       \begin{stex_annotate_env}{shtml:proofbody={}}
7312    }
7313    \bool_if:NT \l__stex_proof_in_spfblock_bool {\group_begin:\stexcommentfont}
7314 }{
7315    \stex_style_apply:
7316    \bool_if:NT \l__stex_proof_in_spfblock_bool \__stex_proof_inc_counter:
7317    \stex_if_do_html:T{\end{stex_annotate_env}}
7318    \bool_if:NT\l__stex_proof_in_spfblock_bool \__stex_proof_end_list:
7319    \stex_if_do_html:T{\end{stex_annotate_env}}
7320    \aftergroup \__stex_proof_inblock_restore:
7321 }{}{}{}
7322 \AddToHook{env/subproof/before}{
7323    \bool_if:NT \l__stex_proof_in_spfblock_bool \group_end:
7324 }
7325 \AddToHook{env/subproof/end}{
7326    \bool_if:NT\l__stex_proof_in_spfblock_bool {
7327       \group_end:\__stex_proof_remove_counter:
7328       %\__stex_proof_end_list:
7329    }
7330 }
7331 \stex_deactivate_macro:Nn \subproof {sproof~environments}
7332
7333 \cs_new_protected:Nn \__stex_proof_inblock_restore: {
7334    \bool_if:NT\l__stex_proof_in_spfblock_bool {
7335       \group_begin:\stexcommentfont
7336    }
7337 }
```

\spfstep
\conclude
\assumption
\have
\eqstep

```
7338
7339 \stex_keys_define:nnnn { spfsteps } {
7340 \clist_clear:N \l_stex_key_for_clist
7341    \str_clear:N \l_stex_key_name_str
7342 \tl_clear:N \l_stex_key_method_tl
7343 \tl_clear:N \l_stex_key_term_tl
7344 }{
7345    for          .clist_set:N  = \l_stex_key_for_clist ,
7346    method       .tl_set:N     = \l_stex_key_method_tl,
7347    term         .tl_set:N     = \l_stex_key_term_tl,
7348    name         .str_set_x:N  = \l_stex_key_name_str
7349    % todo: style=inline
7350 }{id,style,title}
7351
7352 \newenvironment{spfstepenv}{
7353    \str_set_eq:NN \spfstepenvautorefname \spfstepautorefname
7354 }{}
7355
7356 \cs_new_protected:Nn \__stex_proof_step_html:nn {
7357    \stex_if_do_html:TF{
7358       \exp_args:Ne \stex_annotate:nn{
7359          shtml:spf#1={
7360             \seq_if_empty:NF \l_stex_fors_seq {
7361                \seq_use:Nn \l_stex_fors_seq ,
```

```
7362              }
7363            }
7364          \str_if_empty:NF \l_stex_key_name_str {,
7365            shtml:stepname={\l_stex_key_name_str}
7366          }
7367        }{
7368          \__stex_proof_html:
7369          #2
7370        }
7371      }{ #2 }
7372  }
7373
7374  \cs_new_protected:Nn \__stex_proof_make_step_macro:Nnnnn {
7375    \NewDocumentCommand #1 {s O{} +m} {
7376      \bool_if:NT \l__stex_proof_in_spfblock_bool \group_end:
7377      \stex_keys_set:nn{spfsteps}{##2}
7378      \str_if_empty:NF \l_stex_key_name_str {
7379        \stex_debug:nn{Here:}{Variable~\l_stex_key_name_str}
7380        \exp_args:NNx\exp_args:Nnx\vardef{v\l_stex_key_name_str}{\l_stex_key_name_str}
7381      }
7382
7383      \begin{spfstepenv}
7384        \str_if_empty:NF \l_stex_key_id_str {
7385          \__stex_proof_number_as_string:N \@currentlabel
7386          \str_set:Nx \@currentHref{spfstep.\@currentlabel}
7387          \_stex_do_id:
7388        }
7389
7390        \bool_if:NTF \l__stex_proof_in_spfblock_bool {
7391          \IfBooleanTF ##1 {
7392            \__stex_proof_step_html:nn{#2}{##3}
7393          }{
7394            \__stex_proof_step_html:nn{#2}{\__stex_proof_start_list:n{#3} ##3 \__stex_proof_end_
7395            #5
7396          }
7397          \end{spfstepenv}
7398          \group_begin:\stexcommentfont
7399        }{
7400          \__stex_proof_step_html:nn{#2}{##3}
7401          \end{spfstepenv}
7402        }
7403      }
7404    \stex_deactivate_macro:Nn #1 {sproof~environments}
7405  }
7406
7407  \__stex_proof_make_step_macro:Nnnnn \assumption {assumption} \__stex_proof_insert_number: {}
7408  \__stex_proof_make_step_macro:Nnnnn \conclude {conclusion} {$\Rightarrow$} {} {}
7409  \__stex_proof_make_step_macro:Nnnnn \spfstep {step} \__stex_proof_insert_number: {} \__stex_
7410
7411  \NewDocumentCommand \eqstep {s m}{
7412    \bool_if:NTF \l__stex_proof_in_spfblock_bool {
7413      \group_end:
7414      \IfBooleanTF #1 {
7415        \__stex_proof_step_html:nn{eqstep}{$= #2$}
```

287

```
7416      }{
7417        \__stex_proof_step_html:nn{eqstep}{\__stex_proof_start_list:n{$=$} $#2$ \__stex_proof_
7418      }
7419      \group_begin:\stexcommentfont
7420    }{
7421      \__stex_proof_step_html:nn{eqstep}{$= #2$}
7422    }
7423  }
7424  \stex_deactivate_macro:Nn \eqstep {sproof~environments}
7425
7426  \NewDocumentCommand \yield {+m}{
7427    \stex_annotate:nn{shtml:proofterm={}}{ #1 }
7428  }
7429  \stex_deactivate_macro:Nn \yield {sproof~environments}
7430
7431  \NewDocumentEnvironment{spfblock}{}{
7432    \bool_set_false:N \l__stex_proof_in_spfblock_bool
7433  }{
7434    \aftergroup\__stex_proof_inblock_restore:
7435  }
7436  \stex_deactivate_macro:Nn \spfblock {sproof~environments}
7437  \AddToHook{env/spfblock/before}{
7438    \bool_if:NT \l__stex_proof_in_spfblock_bool \group_end:
7439  }
7440
7441
7442  \newcommand\spfjust[1]{
7443    \stex_annotate:nn{spfjust={}}{ #1 }
7444  }
7445  \stex_deactivate_macro:Nn \spfjust {sproof~environments}
```

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

13.12   Metatheory

```
7446  ⟨@@=stex_meta⟩
7447  \group_begin:
7448    \cs_set:Npn \__stex_modules_persist_module: {}
7449    \cs_set:Npn \stex_check_term:n #1 {}
7450    \cs_set:Npn \_stex_sref_do_aux:n #1 { #1 }
7451    \bool_set_false:N \_stex_html_do_output_bool
7452    \bool_set_false:N \c_stex_check_terms_bool
7453    \stex_uri_resolve:Nn \l_stex_current_ns_uri {http://mathhub.info/sTeX/meta}
7454    \stex_module_setup:n{Metatheory}
7455
7456    \symdef{of~type}[args=ii,invisible]{#1}
7457    \notation{of~type}[colon]{#1 \mathbin{\comp{:}} #2}
7458
7459    \symdef{apply}[args=ia,prec=0;\infprec x\infprec]{#1\mathopen{\comp(} #2 \mathclose{\comp)
7460    \notation{apply}[lambda]{#1\; \argsep{#2}{\;}}
7461    \notation{apply}[infixop]{\argsep{#2}{\mathbin{#1}}}
7462    \notation{apply}[infixrel]{\argsep{#2}{\mathrel{#1}}}
7463
7464    % structures
```

288

```
7465  \symdef{module~type}[args=i,op=\mathtt{MOD}]
7466  {\mathopen{\comp{\mathtt{MOD}(}}#1\mathclose{\comp{)}}}}
7467  \symdef{module~type~merge}[args=a,op=\oplus]
7468    {\argsep{#1}{\mathbin{\comp{\oplus}}}}}
7469  \symdef{anonymous~record}[args=a]
7470    {\mathopen{\comp{[[}}#1\mathclose{\comp{]]}}}}
7471  \symdef{record~field}[args=2]{#1\comp{.}#2}
7472  \symdecl*{record~type}
7473
7474  \symdecl{mathstruct}[name=mathematical~structure,args=a] % TODO
7475  \notation{mathstruct}[angle,prec=nobrackets]
7476    {\mathopen{\comp\langle} #1 \mathclose{\comp\rangle}}
7477  \notation{mathstruct}[parens,prec=nobrackets]
7478    {\mathopen{\comp(} #1 \mathclose{\comp)}}
7479
7480  % sequences
7481  \symdef{ellipses}[ldots]{\ldots}
7482  \symdef{sequence~expression}[comma,args=a]{#1}
7483  \symdef{sequence~type}[args=1]{#1^{\comp\ast}}
7484  \symdef{sequence~map}[args=ia]{
7485    \comp{\mathrm{map}}\mathopen{\comp{(}}#1\mathpunct{\comp{,}}
7486    #2\mathclose{\comp{)}}}
7487  }
7488  \iffalse
7489  % binder (\forall, \Pi, \lambda etc.)
7490  \symdef{pibind}[name=dependent function type,prec=nobrackets,
7491    op=(\cdot)\;\to\;\cdot,args=Bi,assoc=pre]
7492    {\argmap{#1}{
7493      \mathopen{\comp(} ##1 \mathclose{\comp)}}
7494      }{\mathbin{\comp{\to}}} \mathbin{\comp{\to}} #2
7495  \notation{pibind}[forall]{\comp\forall #1\mathpunct{\comp.} #2}
7496  \notation{pibind}[Pi]{\mathop{\comp\prod}\c_math_subscript_token{#1}#2}
7497
7498  \symdef{mapbind}[name=lambda,mapsto,prec=nobrackets,op=\mapsto,args=Bi,assoc=pre]
7499    { #1 \mathrel{\comp\mapsto} #2}
7500  \notation{mapbind}[lambda,prec=nobrackets,op=\lambda]
7501    {\comp\lambda #1 \mathpunct{\comp.} #2}
7502  \fi
7503  \symdecl{bind}[args=Bi,assoc=pre]
7504  \notation{bind}[depfun,prec=nobrackets,op=(\cdot)\;\to\;\cdot]
7505    {\mathopen{\comp(} #1 \mathclose{\comp)}\mathbin{\comp{\to}}} #2
7506  \notation{bind}[forall]{\comp\forall #1.\;#2}
7507  \notation{bind}[Pi]{\mathop{\comp\prod}\c_math_subscript_token{#1}#2}
7508
7509  \symdef{implicit~bind}[args=Bi,assoc=pre]{\mathopen{\comp\{} #1 \mathclose{\comp{\}}\c_math
7510
7511  \symdecl*{integer~literal}
7512  \notation{integer~literal}{\mathbb Z}
7513
7514  \symdecl*{ordinal}
7515  \notation{ordinal}{\mathtt{Ord}}
7516
7517  % propositions
7518  \symdef{prop}[name=proposition]{\mathtt{Prop}}
```

```
7519    \symdef{judgment~holds}[args=i,role=judgment]{\comp\vdash\;#1}

7520

7521    % any object
7522    \symdef{object}{\mathtt{Obj}}

7523

7524    % TODO DELETE
7525    \symdef{aseqdots}[args=a,prec=nobrackets]
7526      {#1\comp{,\ldots}}%{##1\comp,##2}
7527    \symdef{aseqfromto}[args=ai,prec=nobrackets]
7528      {#1\comp{,\ldots,}#2}%{##1\comp,##2}
7529    \symdef{aseqfromtovia}[args=aii,prec=nobrackets]
7530      {#1\comp{,\ldots,}#2\comp{,\ldots,}#3}%{##1\comp,##2}

7531

7532

7533    \stex_close_module:
7534    \stex_uri_add_module:NNn \l_stex_metatheory_uri \l_stex_current_ns_uri {Metatheory}
7535    \global \let \l_stex_metatheory_uri \l_stex_metatheory_uri
7536    \global \let \c_stex_default_metatheory \l_stex_metatheory_uri
7537 \group_end:
```

13.13   MMT Interfaces

```
7538    ⟨@@=todo⟩
7539 \cs_new_protected:Npn \MSC #1 {}
```

\MMTinclude

```
7540 \stex_new_stylable_cmd:nnnn{MMTinclude}{m}{
7541    \stex_annotate_invisible:nn{shtml:import={#1}}{}
7542 }{}
7543 \stex_deactivate_macro:Nn \MMTinclude {module~environments}
7544 \stex_every_module:n {\stex_reactivate_macro:N \MMTinclude}
```

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

\MMTrule

```
7545 \NewDocumentCommand \MMTrule {m m}{
7546    \tl_if_empty:nTF{#2}{\seq_clear:N \l_tmpa_seq}{
7547      \seq_set_split:Nnn \l_tmpa_seq , {#2}
7548    }
7549    \int_zero:N \l_tmpa_int
7550    \stex_annotate_invisible:n{
7551      $
7552      \stex_annotate:nn{shtml:rule={scala://#1}}{
7553        \_stex_annotate_force_break:n{
7554          \seq_if_empty:NF \l_tmpa_seq {
7555            \seq_map_inline:Nn \l_tmpa_seq {
7556              \int_incr:N \l_tmpa_int
7557              \stex_annotate:nn{
7558                shtml:argmode=i,
7559                shtml:arg={\int_use:N \l_tmpa_int}
7560              }{ ##1 }
7561            }
7562          }
7563        }
7564      }$
```

```
7565        }
7566    }
7567  \stex_deactivate_macro:Nn \MMTrule {module~environments}
7568  \stex_every_module:n{\stex_reactivate_macro:N \MMTrule}
```

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

mmtinterface (*env.*)

```
7569  \NewDocumentEnvironment { mmtinterface } {  O{} m m } {
7570    \_stex_module_setup_top_nosig:n { #3 }
7571    \str_set_eq:NN \l__todo_mmt_module_str \l_stex_current_module_str
7572    \str_clear:N \l_stex_current_module_str
7573    \stex_keys_set:nn { smodule }{ #1 }
7574    \stex_module_setup:n{ #2 }
7575    \str_set_eq:NN \l__todo_stex_module_str \l_stex_current_module_str
7576    \stex_debug:nn{mmt}{Interface~\l__todo_stex_module_str^^Jfor~\l__todo_mmt_module_str}
7577
7578    \stex_if_do_html:T {
7579      \exp_args:Nne \begin{stex_annotate_env} {
7580        shtml:theory={\l_stex_current_module_str},
7581        shtml:language={ \l_stex_current_language_str},
7582        shtml:signature={}
7583        \tl_if_empty:NF \l_stex_metatheory_uri {,
7584          shtml:metatheory={\stex_uri_use:N \l_stex_metatheory_uri}
7585        }
7586      }
7587      \stex_annotate_invisible:n{}
7588      \stex_annotate_invisible:nn
7589        {shtml:import=\l__todo_mmt_module_str} {}
7590    }
7591    \stex_module_add_code:x{
7592      \stex_activate_module:n{ \l__todo_mmt_module_str }
7593    }
7594    \stex_module_add_morphism:nonn
7595      {}{\l__todo_mmt_module_str}{import}{}
7596    \stex_reactivate_macro:N \mmtdef
7597    \stex_smsmode_do:
7598  }{
7599      \str_set_eq:NN \l_stex_current_module_str \l__todo_mmt_module_str
7600      \stex_close_module:
7601      \str_set_eq:NN \l_stex_current_module_str \l__todo_stex_module_str
7602      \stex_close_module:
7603      \stex_if_do_html:T { \end{stex_annotate_env} }
7604  }
7605  \stex_sms_allow_env:n{mmtinterface}
```

\mmtdef

```
7606  \NewDocumentCommand \mmtdef {m O{} m} {
7607    \stex_keys_set:nn{symdef}{#2}
7608    \str_set:Nx \l_stex_macroname_str { #1 }
7609    \str_if_empty:NT \l_stex_key_name_str {
7610      \str_set:Nx \l_stex_key_name_str { #1 }
7611    }
7612    \stex_symdecl_do:
```

```
7613
7614    \str_set_eq:NN \l_stex_current_module_str \l__todo_mmt_module_str
7615    \cs_set_eq:NN \l__todo_old_metagroup_cd \stex_metagroup_do_in:nn
7616    \cs_set_protected:Npn \stex_metagroup_do_in:nn ##1 ##2 {##2}
7617    \exp_args:Nnx \use:nn {\stex_module_add_symbol:nnnnnnnN}{
7618       {\l_stex_macroname_str}
7619       {\l_stex_key_name_str}
7620       {\int_use:N \l_stex_get_symbol_arity_int}
7621       {\l_stex_get_symbol_args_tl}
7622       {}
7623       {}
7624       {}
7625       \stex_invoke_symbol:
7626    }
7627    \cs_set_eq:NN \stex_metagroup_do_in:nn \l__todo_old_metagroup_cd
7628    \str_set_eq:NN \l_stex_current_module_str \l__todo_stex_module_str
7629
7630    \str_set_eq:NN \l_stex_get_symbol_mod_str \l__todo_mmt_module_str
7631    \str_set_eq:NN \l_stex_get_symbol_name_str \l_stex_key_name_str
7632    \stex_notation_parse:n{#3}
7633    \_stex_notation_check:
7634    \_stex_notation_add:
7635    \stex_if_do_html:T{
7636       \_stex_notation_do_html:n{\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
7637    }
7638    \stex_smsmode_do:
7639 }
7640 \stex_deactivate_macro:Nn \mmtdef {mmtinterface~environments}
7641 \stex_sms_allow_escape:N \mmtdef
```

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

```
7642 \seq_if_empty:NT \g_stex_current_file {
7643    \seq_gset_eq:NN \g_stex_current_file \c_stex_main_file
7644 }
7645 \_stex_persist_read_now:
7646 \_stex_every_file:
7647 \cs_new_protected:Nn \__todo_newlabel:n {
7648    \exp_args:Ne\__todo_old_newlabel:{\tl_to_str:n{#1}}
7649 }
7650 \AtBeginDocument{
7651    \iow_now:Nn \@auxout {
7652       \ExplSyntaxOn
7653       \let\__todo_old_newlabel:\newlabel
7654       \let\newlabel\__todo_newlabel:n
7655       \ExplSyntaxOff
7656    }
7657 }
7658 ⟨/package⟩
```

Chapter 14

Additional Packages

14.1 Implementation: The notesslides Package

14.1.1 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.

```
7659 ⟨*cls⟩
7660 ⟨@@=notesslides⟩
7661 \ProvidesExplClass{notesslides}{2023/03/19}{3.3.0}{notesslides Class}
7662 \RequirePackage{l3keys2e}
7663
7664 \str_const:Nn \c__notesslides_class_str {article}
7665
7666 \keys_define:nn{notesslides / cls}{
7667   class    .str_set_x:N = \c__notesslides_class_str,
7668   notes    .bool_set:N  = \c_notesslides_notes_bool ,
7669   slides   .code:n      = { \bool_set_false:N \c_notesslides_notes_bool },
7670   %docopt  .str_set_x:N = \c__notesslides_docopt_str,
7671   unknown .code:n       = {
7672     \PassOptionsToClass{\CurrentOption}{beamer}
7673     \PassOptionsToClass{\CurrentOption}{\c__notesslides_class_str}
7674     \PassOptionsToPackage{\CurrentOption}{notesslides}
7675     \PassOptionsToPackage{\CurrentOption}{stex}
7676   }
7677 }
7678 \ProcessKeysOptions{ notesslides / cls }
7679
7680 \RequirePackage{stex}
7681 \stex_if_html_backend:T {
7682   \bool_set_true:N\c_notesslides_notes_bool
7683 }
7684
7685 \bool_if:NTF \c_notesslides_notes_bool {
7686   \PassOptionsToPackage{notes=true}{notesslides}
7687   \message{notesslides.cls:~Formatting~document~in~notes~mode}
```

```
7688  }{
7689    \PassOptionsToPackage{notes=false}{notesslides}
7690    \message{notesslides.cls:~Formatting~document~in~slides~mode}
7691  }
7692
7693  \bool_if:NTF \c_notesslides_notes_bool {
7694    \LoadClass{\c__notesslides_class_str}
7695  }{
7696    \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
7697    %\newcounter{Item}
7698    %\newcounter{paragraph}
7699    %\newcounter{subparagraph}
7700    %\newcounter{Hfootnote}
7701  }
7702  \RequirePackage{notesslides}
7703  ⟨/cls⟩
```

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

```
7704  ⟨*package⟩
7705  \ProvidesExplPackage{notesslides}{2023/03/19}{3.3.0}{notesslides Package}
7706  \RequirePackage{l3keys2e}
7707
7708  \keys_define:nn{notesslides / pkg}{
7709    notes          .bool_set:N  = \c_notesslides_notes_bool ,
7710    slides         .code:n      = { \bool_set_false:N \c_notesslides_notes_bool },
7711    sectocframes   .bool_set:N  = \c_notesslides_sectocframes_bool ,
7712    topsect        .str_set_x:N = \c_notesslides_topsect_str,
7713    unknown        .code:n      = {
7714      \PassOptionsToPackage{\CurrentOption}{stex}
7715      \PassOptionsToPackage{\CurrentOption}{tikzinput}
7716    }
7717  }
7718  \ProcessKeysOptions{ notesslides / pkg }
7719
7720  \RequirePackage{stex}
7721  \stex_if_html_backend:T {
7722    \bool_set_true:N\c_notesslides_notes_bool
7723  }
7724
7725  \cs_set:Npn \sectiontitleemph #1 {
7726    \textbf{\Large #1}
7727  }
7728
7729  \newif\ifnotes
7730  \bool_if:NTF \c_notesslides_notes_bool {
7731    \notestrue
7732    \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
7733    \RequirePackage[noamsthm,hyperref]{beamerarticle}
7734    \RequirePackage{mdframed}
7735    \str_if_empty:NTF \c_notesslides_topsect_str{
7736      %\setsectionlevel{section}
7737    } {
7738      \exp_args:No \setsectionlevel \c_notesslides_topsect_str
7739    }
```

294

```
7740 }{
7741   \notesfalse
7742
7743   \cs_new_protected:Nn \__notesslides_do_sectocframes: {
7744     \cs_set_protected:Nn \__notesslides_do_label:n {
7745       \str_case:nnF{##1}{
7746         {part}   {
7747           \tl_set:Nx\l__notesslides_num{\thepart}
7748           \tl_set:cx{@ @ label}{
7749             \cs_if_exist:NTF\parttitlename{\exp_not:N\parttitlename}{\exp_not:N\partname}{}~
7750         }
7751         {chapter} {
7752           \tl_set:Nx\l__notesslides_num{\thechapter}
7753           \tl_set:cx{@ @ label}{
7754             \cs_if_exist:NTF\chaptertitlename{\exp_not:N\chaptertitlename}{\exp_not:N\chapte
7755         }
7756         {section} {
7757           \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\thechapter.}\thesection}
7758           \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
7759         }
7760         {subsection} {
7761           \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\thechapter.}\thesection.
7762           \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
7763         }
7764         {subsubsection} {
7765           \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\thechapter.}\thesection.
7766           \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
7767         }
7768         {paragraph} {
7769           \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\thechapter.}\thesection.
7770           \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
7771         }
7772       }{
7773         \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\thechapter.}\thesection.\t
7774         \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
7775       }
7776     }
7777     \cs_set_protected:Nn \_sfragment_do_level:nn {
7778       \tl_if_exist:cT{c@##1}{\stepcounter{##1}}
7779       \addcontentsline{toc}{##1}{\protect\numberline{\use:c{the##1}}##2}
7780       \__notesslides_do_label:n{##1}
7781       \pdfbookmark[\int_use:N \l_stex_docheader_sect]{\l__notesslides_num\ ##2}{##1.\l__note
7782       \begin{frame}[noframenumbering]
7783         \vfill\centering
7784         \sectiontitleemph{
7785           \use:c{@ @ label} ##2
7786         }
7787       \end{frame}
7788       \int_incr:N \l_stex_docheader_sect
7789       \tl_set:Nn \stex_current_section_level{##1}
7790     }
7791   }
7792
7793   \AtBeginDocument{
```

295

```
7794        \str_if_empty:NTF \c_notesslides_topsect_str {
7795          \setsectionlevel{section}
7796        } {
7797          \exp_args:No \setsectionlevel \c_notesslides_topsect_str
7798          \exp_args:No \str_if_eq:nnTF \c_notesslides_topsect_str {chapter} {
7799            \__notesslides_define_chapter:
7800          }{
7801            \exp_args:No \str_if_eq:nnT \c_notesslides_topsect_str {part} {
7802              \__notesslides_define_chapter:
7803              \__notesslides_define_part:
7804            }
7805          }
7806        }
7807      }
7808
7809      \bool_if:NT \c_notesslides_sectocframes_bool {
7810        \__notesslides_do_sectocframes:
7811      }
7812    }
7813
7814    \cs_new_protected:Nn \__notesslides_define_chapter: {
7815      \cs_if_exist:NF \chaptername {
7816        \cs_set_protected:Npn \chaptername {Chapter}
7817      }
7818      \cs_if_exist:NF \chapter {
7819        \cs_set_protected:Npn \chapter {INVALID}
7820      }
7821      \cs_if_exist:NF \c@chapter {
7822        \newcounter{chapter}\counterwithin*{section}{chapter}
7823      }
7824    }
7825
7826    \cs_new_protected:Nn \__notesslides_define_part: {
7827      \cs_if_exist:NF \partname {
7828        \cs_set_protected:Npn \partname {Part}
7829      }
7830      \cs_if_exist:NF \part {
7831        \cs_set_protected:Npn \part {INVALID}
7832      }
7833      \cs_if_exist:NF \c@part {
7834        \newcounter{part}\counterwithin*{chapter}{part}
7835      }
7836    }
```

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

```
7837    \def \c__notesslides_document_str{document}
7838    \newcommand\afterprematurestop{}
7839    \def\prematurestop@endsfragment{
7840      \unless\ifx\@currenvir\c__notesslides_document_str
7841        \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafte
7842        \expandafter\prematurestop@endsfragment
7843      \fi
7844    }
```

296

```
7845  \providecommand\prematurestop{
7846    \stex_if_html_backend:F{
7847      \message{Stopping~sTeX~processing~prematurely}
7848      \prematurestop@endsfragment
7849      \afterprematurestop
7850      \end{document}
7851    }
7852  }
```

(*End definition for* \prematurestop. *This function is documented on page 97.*)

14.1.2   Notes and Slides

For the notes case, we also provide the \usetheme macro that would otherwise come from the the beamer class.

```
7853  \bool_if:NT \c_notesslides_notes_bool {
7854    \renewcommand\usetheme[2][]{\usepackage[#1]{beamertheme#2}}
7855  }
7856  \NewDocumentCommand \libusetheme {O{} m} {
7857    \libusepackage[#1]{beamertheme#2}
7858  }
```

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

```
7859  \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
7860  \newlength{\slideheight}\setlength{\slideheight}{9cm}
```

We first set up the slide boxes in notes mode. We set up sizes and provide a box register for the frames and a counter for the slides.

```
7861  \ifnotes
7862
7863  \newlength{\slideframewidth}
7864  \setlength{\slideframewidth}{1.5pt}
```

frame (*env.*) We first define the keys.

```
7865  \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
7866    \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
7867      \bool_set_true:N #1
7868    }{
7869      \bool_set_false:N #1
7870    }
7871  }
7872
7873  \stex_keys_define:nnnn{notesslides / frame}{
7874    \str_clear:N \l__notesslides_frame_label_str
7875    \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
7876    \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
7877    \bool_set_true:N \l__notesslides_frame_fragile_bool
7878    \bool_set_true:N \l__notesslides_frame_shrink_bool
7879    \bool_set_true:N \l__notesslides_frame_squeeze_bool
7880    \bool_set_true:N \l__notesslides_frame_t_bool
7881  }{
7882    label                .str_set_x:N  = \l__notesslides_frame_label_str,
7883    allowframebreaks     .code:n       = {
```

```
7884      \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
7885    },
7886    allowdisplaybreaks  .code:n      = {
7887      \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
7888    },
7889    fragile            .code:n      = {
7890      \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
7891    },
7892    shrink             .code:n      = {
7893      \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
7894    },
7895    squeeze            .code:n      = {
7896      \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
7897    },
7898    t                  .code:n      = {
7899      \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
7900    },
7901    unknown    .code:n      = {}
7902 }{}
```

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

```
7903 \cs_new_protected:Nn \__notesslides_setup_itemize: {
7904    \def\itemize@level{outer}
7905    \def\itemize@outer{outer}
7906    \def\itemize@inner{inner}
7907    %\newcommand\metakeys@show@keys[2]{\marginnote{{\scriptsize ##2}}}
7908    \renewenvironment{itemize}{
7909      \ifx\itemize@level\itemize@outer
7910        \def\itemize@label{$\rhd$}
7911      \fi
7912      \ifx\itemize@level\itemize@inner
7913        \def\itemize@label{$\scriptstyle\rhd$}
7914      \fi
7915      \begin{list}
7916      {\itemize@label}
7917      {\setlength{\labelsep}{.3em}
7918        \setlength{\labelwidth}{.5em}
7919        \setlength{\leftmargin}{1.5em}
7920      }
7921      \edef\itemize@level{\itemize@inner}
7922    }{
7923      \end{list}
7924    }
7925 }
```

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

```
7926 \stex_if_html_backend:TF {
7927    \cs_new_protected:Nn \__notesslides_frame_box_begin: {
7928      \vbox\bgroup
7929      \begin{stex_annotate_env}{shtml:frame={}}
7930        \mdf@patchamsthm\notesslidesfont
7931    }
7932    \cs_new_protected:Nn \__notesslides_frame_box_end: {
7933      %^^A \notesslides@slidelabel
```

```
7934      \medskip\par\noindent\tiny\notesslidesfooter
7935      \end{stex_annotate_env}\egroup
7936   }
7937 }{
7938   \cs_new_protected:Nn \__notesslides_frame_box_begin: {
7939      \begin{mdframed}[
7940         linewidth=\slideframewidth,
7941         skipabove=1ex,
7942         skipbelow=1ex,
7943         userdefinedwidth=\slidewidth,
7944         align=center
7945      ]\notesslidesfont
7946   }
7947   \cs_new_protected:Nn \__notesslides_frame_box_end: {
7948      \medskip\par\noindent\tiny\notesslidesfooter%^^A\notesslides@slidelabel
7949      \end{mdframed}
7950   }
7951 }
```

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

```
7952 \renewenvironment{frame}[1][]{
7953   \stex_keys_set:nn{notesslides / frame}{#1}
7954   \stepcounter{framenumber}
7955   \renewcommand\newpage{\addtocounter{framenumber}{1}}
7956   \def\@currentlabel{\theframenumber}
7957   \str_if_empty:NF \l__notesslides_frame_label_str {
7958      \label{\l__notesslides_frame_label_str}
7959   }
7960   \__notesslides_setup_itemize:
7961   \__notesslides_frame_box_begin:
7962 }{
7963   \__notesslides_frame_box_end:
7964 }
```

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

\frametitle

```
7965 \renewcommand{\frametitle}[1]{
7966   \stexdoctitle { #1 }
7967   \notesslidestitleemph{#1}\medskip
7968 }
```

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

\pause

```
7969 \newcommand\pause{}
```

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

We redefine the columns and column environments:

```
7970 \renewenvironment{columns}[1][]{
7971   \par\noindent
7972   \begin{minipage}
7973   \slidewidth\centering\leavevmode
7974 %  \stex_if_html_backend:T{
```

```
7975 %     \cs_if_exist:NT \rustex_if:T {
7976 %        \rustex_if:T {\par
7977 %          \rustex_direct_HTML:n{<table><tr><td>}
7978 %        }
7979 %     }
7980 %  }
7981 }{
7982 %  \stex_if_html_backend:T{
7983 %     \cs_if_exist:NT \rustex_if:T {
7984 %        \rustex_if:T {\par
7985 %          \rustex_direct_HTML:n{</td></tr></table>}
7986 %        }
7987 %     }
7988 %  }
7989    \end{minipage}\par\noindent
7990 }
7991 \newsavebox\columnbox
7992 \renewenvironment<>{column}[2][]{
7993    \begin{lrbox}{\columnbox}
7994 %     \stex_if_html_backend:T{
7995 %        \cs_if_exist:NT \rustex_if:T {
7996 %          \rustex_if:T {\par
7997 %             \rustex_direct_HTML:n{</td><td>}
7998 %          }
7999 %        }
8000 %     }
8001    \begin{minipage}{#2}
8002 }{
8003    \end{minipage}
8004 %  \stex_if_html_backend:T{
8005 %     \cs_if_exist:NT \rustex_if:T {
8006 %        \rustex_if:T {\par
8007 %          \rustex_direct_HTML:n{</td><td>}
8008 %        }
8009 %     }
8010 %  }
8011    \end{lrbox}\usebox\columnbox
8012 }
8013 \fi
```

14.1.3   Environment and Macro Patches

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 to produce no output.

```
8014 \bool_if:NTF \c_notesslides_notes_bool {
8015    \renewenvironment{note}{\ignorespaces}{}
8016 }{
8017    \renewenvironment{note}{\setbox \l_tmpa_box\vbox\bgroup}{\egroup}
8018 }
```

For other environments we introduce variants prefixed with `n`, which are excluded in `slides` mode.

```
8019 \cs_new_protected:Nn \__notesslides_notes_env:nnnn {
8020   \bool_if:NTF \c_notesslides_notes_bool {
8021     \newenvironment{#1}#2{#3}{#4}
8022   }{
8023     \newenvironment{#1}#2{
8024       \cs_set:Npn \__notesslides_eat: ####1 \end ####2 {
8025         \str_if_eq:nnTF{#1}{####2}{
8026           \end{#1}
8027         }{
8028           \__notesslides_eat:
8029         }
8030       }
8031       \__notesslides_eat:
8032       %\setbox\l_tmpa_box\vbox\bgroup#3
8033     }{
8034       %#4\egroup
8035     }
8036   }
8037 }
8038
8039 \__notesslides_notes_env:nnnn{nparagraph}{[1][]}{\begin{sparagraph}[#1]}{\end{sparagraph}}
8040 \__notesslides_notes_env:nnnn{nfragment}{[2][]}{\begin{sfragment}[#1]{#2}}{\end{sfragment}}
8041 \__notesslides_notes_env:nnnn{ndefinition}{[1][]}{\begin{sdefinition}[#1]}{\end{sdefinition}
8042 \__notesslides_notes_env:nnnn{nassertion}{[1][]}{\begin{sassertion}[#1]}{\end{sassertion}}
8043 \__notesslides_notes_env:nnnn{nproof}{[2][]}{\begin{sproof}[#1]{#2}}{\end{sproof}}
8044 \__notesslides_notes_env:nnnn{nexample}{[1][]}{\begin{sexample}[#1]}{\end{sexample}}
8045
8046 \RequirePackage{graphicx}
8047
8048 \NewDocumentCommand\frameimage{s O{} m}{
8049   \IfBooleanTF #1 {
8050     \begin{frame}[plain]
8051   }{
8052     \begin{frame}
8053   }
8054   \bool_if:NTF \c_notesslides_notes_bool {
8055     \slidewidth=\dimexpr\slidewidth-(2\slideframewidth)\relax
8056   }{
8057     \slidewidth=\textwidth\relax
8058   }
8059   \def\Gin@ewidth{}\setkeys{Gin}{#2}
8060   \tl_if_empty:NTF \Gin@ewidth {
8061     \mhgraphics[width=\slidewidth,#2]{#3}
8062   }{
8063     \mhgraphics[#2]{#3}
8064   }
8065   \end{frame}
8066 }
```

hacking inputref:

\inputref*

```
8067 \cs_set_eq:NN\__notesslides_inputref:\inputref
8068 \cs_set_protected:Npn\inputref{\@ifstar\ninputref\__notesslides_inputref:}
```

301

```
8069  \bool_if:NTF \c_notesslides_notes_bool {
8070    \newcommand\ninputref[2][]{
8071      \__notesslides_inputref:[#1]{#2}
8072    }
8073  }{
8074    \newcommand\ninputref[2][]{}
8075  }
```

(*End definition for* `\inputref*`*. This function is documented on page* *96*.)

14.1.4 Styling Across Notes/Slides

```
8076  \def\notesslidestitleemph#1{
8077    {\Large\bf\sf#1}
8078    \vskip0.1\baselineskip
8079    \leaders\vrule width \textwidth
8080    \vskip0.4pt%
8081    \nointerlineskip
8082  }
8083
8084  \def\notesslidesfooter{}
8085
8086  \let\notesslidesfont\sffamily
```

14.1.5 Beamer Compatibility

All of this should be removed and made part of a template

```
8087
8088  \bool_if:NT \c_notesslides_notes_bool {
8089    \def\author{\@dblarg\ns@author}
8090    \long\def\ns@author[#1]#2{%
8091      \tl_if_empty:nTF{#1}{
8092        \def\beamer@shortauthor{#2}
8093      }{
8094        \def\beamer@shortauthor{#1}
8095      }
8096      \def\@author{#2}
8097    }
8098    \def\title{\@dblarg\ns@title}
8099    \long\def\ns@title[#1]#2{%
8100      \tl_if_empty:nTF{#1}{
8101        \def\beamer@shorttitle{#2}
8102      }{
8103        \def\beamer@shorttitle{#1}
8104      }
8105      \def\@title{#2}
8106      \stexdoctitle{#2}
8107    }
8108    \def\insertshortauthor{
8109      \hbox\bgroup\def\\{}\cs_if_exist:NT\beamer@shortauthor\beamer@shortauthor\egroup
8110    }
8111    \def\insertshorttitle{
8112      \hbox\bgroup\def\\{}\cs_if_exist:NT\beamer@shorttitle\beamer@shorttitle\egroup
8113    }
8114    \stex_if_html_backend:TF{
```

```
8115        \def\insertframenumber{\stex_annotate:nn{shtml:framenumber={}}{}}
8116    }{
8117        \def\insertframenumber{\@arabic\c@framenumber}
8118    }
8119    \def\insertshortdate{\today}
8120 }
```

14.1.6 TODO 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.

```
8121 \gdef\printexcursions{}
8122 \newcommand\excursionref[2]{% label, text
8123    \bool_if:NT \c_notesslides_notes_bool {
8124        \begin{sparagraph}[title=Excursion]
8125            #2 \sref[fallback=the appendix]{#1}.
8126        \end{sparagraph}
8127    }
8128 }
8129 \newcommand\activate@excursion[2][]{
8130    \tl_gput_right:Nn\printexcursions{\inputref[#1]{#2}}
8131 }
8132 \newcommand\excursion[4][]{% repos, label, path, text
8133    \bool_if:NT \c_notesslides_notes_bool {
8134        \activate@excursion[#1]{#3}
8135        \excursionref{#2}{#4}
8136    }
8137 }
```

(*End definition for* \excursion. *This function is documented on page 98.*)

\excursiongroup
```
8138 \keys_define:nn{notesslides / excursiongroup }{
8139    id        .str_set_x:N  = \l__notesslides_excursion_id_str,
8140    intro     .tl_set:N     = \l__notesslides_excursion_intro_tl,
8141    archive   .str_set_x:N  = \l__notesslides_excursion_mhrepos_str
8142 }
8143 \cs_new_protected:Nn \__notesslides_excursion_args:n {
8144    \tl_clear:N \l__notesslides_excursion_intro_tl
8145    \str_clear:N \l__notesslides_excursion_id_str
8146    \str_clear:N \l__notesslides_excursion_mhrepos_str
8147    \keys_set:nn {notesslides / excursiongroup }{ #1 }
8148 }
8149 \newcommand\excursiongroup[1][]{
8150    \__notesslides_excursion_args:n{ #1 }
8151    \tl_if_empty:NF\printexcursions
8152    {\IfInputref{}{\begin{note}
8153        \begin{sfragment}{Excursions}% TODO pass on id
8154            \ifdefempty\l__notesslides_excursion_intro_tl{}{
8155                \exp_args:NNe \use:nn \inputref[\l__notesslides_excursion_mhrepos_str]{
8156                    \l__notesslides_excursion_intro_tl
8157                }}
8158            }
```

```
8159        \printexcursions%
8160      \end{sfragment}
8161    \end{note}}}
8162  }
8163  \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
```

(*End definition for* \excursiongroup. *This function is documented on page* 98.)

```
8164  \prop_new:N \g__notesslides_variables_prop
8165  \cs_set_protected:Npn \setSGvar #1 #2 {
8166    \prop_gput:Nnn \g__notesslides_variables_prop {#1}{#2}
8167  }
8168  \cs_set_protected:Npn \useSGvar #1 {
8169    \prop_item:Nn \g__notesslides_variables_prop {#1}
8170  }
8171  \cs_set_protected:Npn \ifSGvar #1 #2 #3 {
8172    \prop_get:NnNF \g__notesslides_variables_prop {#1} \l__notesslides_tmp {
8173      \PackageError{document-structure}
8174      {The sTeX Global variable #1 is undefined}
8175      {set it with \protect\setSGvar}\TODO better error
8176    }
8177    \tl_if_eq:NnT \l__notesslides_tmp {#2}{ #3 }
8178  }
8179
8180
8181  ⟨/package⟩
```

14.2 Implementation: The problem Package

14.2.1 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.

```
8182  ⟨*package⟩
8183  ⟨@@=problems⟩
8184  \ProvidesExplPackage{problem}{2023/03/19}{3.3.0}{Semantic Markup for Problems}
8185  \RequirePackage{l3keys2e}
8186
8187  \keys_define:nn { problem / pkg }{
8188    notes      .default:n   = { true },
8189    notes      .bool_set:N  = \c__problems_notes_bool,
8190    gnotes     .default:n   = { true },
8191    gnotes     .bool_set:N  = \c__problems_gnotes_bool,
8192    hints      .default:n   = { true },
8193    hints      .bool_set:N  = \c__problems_hints_bool,
8194    solutions  .default:n   = { true },
8195    solutions  .bool_set:N  = \c__problems_solutions_bool,
8196    pts        .default:n   = { true },
8197    pts        .bool_set:N  = \c__problems_pts_bool,
8198    min        .default:n   = { true },
8199    min        .bool_set:N  = \c__problems_min_bool,
8200    %boxed      .default:n    = { true },
8201    %boxed      .bool_set:N   = \c__problems_boxed_bool,
```

```
8202    test      .default:n    = { true },
8203    test      .bool_set:N   = \c__problems_test_bool,
8204    unknown       .code:n        = {
8205      \PassOptionsToPackage{\CurrentOption}{stex}
8206    }
8207 }
8208 \newif\ifsolutions
8209
8210 \ProcessKeysOptions{ problem / pkg }
8211 \bool_if:NTF \c__problems_solutions_bool {
8212    \solutionstrue
8213 }{
8214    \solutionsfalse
8215 }
8216 \RequirePackage{stex}
```

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

```
8217 \AddToHook{begindocument}{
8218    \ExplSyntaxOn\makeatletter
8219    \input{problem-english.ldf}
8220    \ltx@ifpackageloaded{babel}{
8221        \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
8222        \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
8223          \input{problem-ngerman.ldf}
8224        }
8225        \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
8226          \input{problem-finnish.ldf}
8227        }
8228        \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
8229          \input{problem-french.ldf}
8230        }
8231        \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
8232          \input{problem-russian.ldf}
8233        }
8234    }{}
8235    \makeatother\ExplSyntaxOff
8236 }
```

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

14.2.2   Problems and Solutions

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

```
8237 \stex_keys_define:nnnn{ problem }{
8238    \tl_set:Nn \l_stex_key_pts_tl 0
8239    \tl_set:Nn \l_stex_key_min_tl 0
8240    \str_clear:N \l_stex_key_name_str
8241    \str_clear:N \l_stex_key_mhrepos_str
8242 }{
8243    pts      .tl_set:N    = \l_stex_key_pts_tl,
8244    min      .tl_set:N    = \l_stex_key_min_tl,
```

```
8245   name     .str_set:N    = \l_stex_key_name_str,
8246   archive .str_set:N    = \l_stex_key_mhrepos_str,
8247   creators .code:n = {}
8248   %imports .tl_set:N     = \l__problems_prob_imports_tl,
8249   %refnum  .int_set:N    = \l__problems_prob_refnum_int,
8250 }{id,title,style,uses}
```

Then we set up a counter for problems.

\numberproblemsin

```
8251 \newcounter{sproblem}[section]
8252 \newcommand\numberproblemsin[1]{
8253   \@addtoreset{sproblem}{#1}
8254   \def\thesproblem{\arabic{#1}.\arabic{sproblem}}}
8255 }
8256 \numberproblemsin{section}
8257 %\def\theplainsproblem{\arabic{sproblem}}
8258 %\def\thesproblem{\thesection.\theplainsproblem}
```

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

sproblem (*env.*)

```
8259 \newcounter{pts}
8260 \newcounter{min}
8261 \stex_new_stylable_env:nnnnnnn {problem} {O{}}{
8262   \cs_if_exist:NTF \l_problem_inputproblem_keys_tl {
8263     \tl_put_left:Nn \l_problem_inputproblem_keys_tl {#1,}
8264     \exp_args:Nno \stex_keys_set:nn{problem}{
8265       \l_problem_inputproblem_keys_tl
8266     }
8267   }{
8268     \stex_keys_set:nn{problem}{#1}
8269   }
8270   \refstepcounter{sproblem}
8271   \str_if_empty:NT \l_stex_key_name_str {
8272     \stex_file_split_off_lang:NN \l__problems_path_seq \g_stex_current_file
8273     \seq_get_right:NN \l__problems_path_seq \l_stex_key_name_str
8274   }
8275
8276   \exp_args:No \stex_module_setup:n \l_stex_key_name_str
8277
8278   \stex_if_do_html:T {
8279     \exp_args:Nne \begin{stex_annotate_env} {
8280       shtml:problem={\l_stex_current_module_str},
8281       shtml:language={ \l_stex_current_language_str},
8282       shtml:signature={\l_stex_key_sig_str}
8283       \tl_if_empty:NF \l_stex_metatheory_uri {,
8284         shtml:metatheory={\stex_uri_use:N \l_stex_metatheory_uri}
8285       }
8286     }
8287     \stex_annotate_invisible:n{}
8288     \tl_if_empty:NF \l_stex_key_title_tl {
8289       \exp_args:No \stexdoctitle \l_stex_key_title_tl
8290     }
8291   }
```

306

```
8292    \stex_if_smsmode:F {
8293      \str_set_eq:NN \thismoduleuri \l_stex_current_module_str
8294      \tl_set_eq:NN \thismodulename \l_stex_key_name_str
8295      \tl_set_eq:NN \thistitle \l_stex_key_title_tl
8296      \stex_style_apply:
8297      \addtocounter{pts}{\l_stex_key_pts_tl}
8298      \addtocounter{min}{\l_stex_key_min_tl}
8299      \_stex_do_id:
8300      \__problems_record_problem:
8301    }
8302    \stex_reactivate_macro:N \solution
8303    \stex_reactivate_macro:N \mcb
8304    \stex_reactivate_macro:N \fillinsol
8305    \stex_smsmode_do:
8306  }{
8307    \stex_close_module:
8308    \stex_if_smsmode:F \stex_style_apply:
8309    \stex_if_do_html:T{ \end{stex_annotate_env} }
8310  }{
8311    \par\noindent\problemheader
8312    \bool_if:NT \c__problems_pts_bool {
8313      \marginpar{\l_stex_key_pts_tl{}~\problem@kw@points\smallskip}
8314    }
8315    \bool_if:NT \c__problems_min_bool {
8316      \marginpar{\l_stex_key_min_tl{}~\problem@kw@minutes\smallskip}
8317    }
8318    \\
8319    \stex_ignore_spaces_and_pars:
8320  }{
8321    \par\bigskip
8322  % \bool_if:NT \c__problems_test_bool \pagebreak
8323  }{s}
8324  \stex_sms_allow_env:n{sproblem}
8325
8326  \tl_set:Nn \problemheader {
8327    \textbf{\sproblemautorefname{~}\thesproblem
8328      \tl_if_empty:NF \thistitle {
8329        {~}(\thistitle)
8330      }
8331    }
8332  }
8333
8334  \cs_new_protected:Nn \__problems_record_problem: {
8335    \exp_args:NNe \iow_now:Nn \@auxout {
8336      \problem@restore {\thesproblem}{\l_stex_key_pts_tl}{\l_stex_key_min_tl}
8337    }
8338  }
8339
8340  \cs_new_protected:Npn \problem@restore #1 #2 #3 {}
```

\includeproblem

```
8341  \stex_keys_define:nnnn{ includeproblem }{
8342    \str_clear:N \l_stex_key_mhrepos_str
8343  }{
```

307

```
8344    archive .str_set:N    = \l_stex_key_mhrepos_str,
8345    unknown .code:n = {}
8346 }{}
8347
8348 \NewDocumentCommand\includeproblem{O{} m}{
8349    \group_begin:
8350    \tl_set:Nn \l_problem_inputproblem_keys_tl {#1}
8351    \stex_keys_set:nn{includeproblem}{#1}
8352    \exp_args:Nno \use:nn{\inputref[]\l_stex_key_mhrepos_str]{#2}
8353    \group_end:
8354 }
8355
```

(*End definition for* `\includeproblem`. *This function is documented on page* *104*.)

solution (*env.*)

```
8356 \int_new:N \g_problem_id_counter
8357
8358 \cs_new_protected:Nn \__problems_solution_start:n {
8359    \str_set:Nn \l_stex_key_title_tl {#1}
8360    \str_set:Nn \l_stex_key_id_str {#1}
8361    \str_if_empty:NT \l_stex_key_id_str {
8362       \int_gincr:N \g_problem_id_counter
8363       \str_set:Nx \l_stex_key_id_str {
8364          SOLUTION_\int_use:N \g_problem_id_counter
8365       }
8366    }
8367    \stex_if_do_html:T{
8368       \begin{stex_annotate_env}{
8369          shtml:solution=\l_stex_key_id_str
8370       }
8371    }
8372    \stex_style_apply:
8373 }
8374
8375 \stex_new_stylable_env:nnnnnnn { solution }{ O{} }{
8376    \stex_if_do_html:TF{
8377       \__problems_solution_start:n{#1}
8378    }{
8379       \ifsolutions
8380          \__problems_solution_start:n{#1}
8381       \else
8382          \setbox\l_tmpa_box\vbox\bgroup
8383       \fi
8384    }
8385 }{
8386    \stex_if_do_html:TF{
8387       \stex_style_apply:
8388       \end{stex_annotate_env}
8389    }{
8390       \ifsolutions
8391          \stex_style_apply:
8392          \stex_if_do_html:T{
8393             \end{stex_annotate_env}
```

```
8394          }
8395        \else
8396          \egroup
8397        \fi
8398    }
8399 }{
8400    \par\smallskip\hrule\smallskip
8401    \noindent\emph{Solution\str_if_empty:NF \l_stex_key_title_tl{
8402      {~}\l_stex_key_title_tl
8403    } :~}
8404 }{
8405    \par\smallskip\hrule
8406 }{}
8407
8408 \stex_deactivate_macro:Nn \solution {sproblem~environments}
```

\startsolutions

\startsolutions
\stopsolutions

```
8409 \cs_new_protected:Npn \startsolutions{
8410    \global\solutionstrue
8411 }
8412 \cs_new_protected:Npn \stopsolutions{
8413    \global\solutionsfalse
8414 }
```

(*End definition for* \startsolutions *and* \stopsolutions. *These functions are documented on page* 100.)

hint (*env.*)

```
8415 \cs_new_protected:Nn \__problems_hint_start:n {
8416    \str_set:Nn \l_stex_key_title_tl {#1}
8417    \str_set:Nn \l_stex_key_id_str {#1}
8418    \str_if_empty:NT \l_stex_key_id_str {
8419      \int_gincr:N \g_problem_id_counter
8420      \str_set:Nx \l_stex_key_id_str {
8421        HINT_\int_use:N \g_problem_id_counter
8422      }
8423    }
8424    \stex_if_do_html:T{
8425      \begin{stex_annotate_env}{
8426        shtml:problemhint=\l_stex_key_id_str
8427      }
8428    }
8429    \stex_style_apply:
8430 }
8431
8432 \stex_new_stylable_env:nnnnnnn { hint }{ O{} }{
8433    \stex_if_do_html:TF{
8434      \__problems_hint_start:n{#1}
8435    }{
8436      \bool_if:NTF \c__problems_hints_bool {
8437        \__problems_hint_start:n{#1}
8438      }{
8439        \setbox\l_tmpa_box\vbox\bgroup
8440      }
```

```
8441        }
8442  }{
8443     \stex_if_do_html:TF{
8444        \stex_style_apply:
8445        \end{stex_annotate_env}
8446     }{
8447        \bool_if:NTF \c__problems_hints_bool {
8448           \stex_style_apply:
8449           \stex_if_do_html:T{
8450              \end{stex_annotate_env}
8451           }
8452        }{
8453           \egroup
8454        }
8455     }
8456  }{
8457     \par\smallskip\hrule\smallskip
8458     \noindent\emph{Hint\str_if_empty:NF \l_stex_key_title_tl{
8459        {~}\l_stex_key_title_tl
8460     } :~}
8461  }{
8462     \par\smallskip\hrule
8463  }{}
```

exnote (*env.*)

```
8464  \cs_new_protected:Nn \__problems_exnote_start:n {
8465     \str_set:Nn \l_stex_key_title_tl {#1}
8466     \str_set:Nn \l_stex_key_id_str {#1}
8467     \str_if_empty:NT \l_stex_key_id_str {
8468        \int_gincr:N \g_problem_id_counter
8469        \str_set:Nx \l_stex_key_id_str {
8470           EXNOTE_\int_use:N \g_problem_id_counter
8471        }
8472     }
8473     \stex_if_do_html:T{
8474        \begin{stex_annotate_env}{
8475           shtml:problemnote=\l_stex_key_id_str
8476        }
8477     }
8478     \stex_style_apply:
8479  }
8480
8481  \stex_new_stylable_env:nnnnnnn { exnote }{ O{} }{
8482     \stex_if_do_html:TF{
8483        \__problems_exnote_start:n{#1}
8484     }{
8485        \bool_if:NTF \c__problems_notes_bool {
8486           \__problems_exnote_start:n{#1}
8487        }{
8488           \setbox\l_tmpa_box\vbox\bgroup
8489        }
8490     }
8491  }{
8492     \stex_if_do_html:TF{
```

```
8493        \stex_style_apply:
8494        \end{stex_annotate_env}
8495    }{
8496      \bool_if:NTF \c__problems_notes_bool {
8497        \stex_style_apply:
8498        \stex_if_do_html:T{
8499          \end{stex_annotate_env}
8500        }
8501      }{
8502        \egroup
8503      }
8504    }
8505  }{
8506    \par\smallskip\hrule\smallskip
8507    \noindent\emph{Note\str_if_empty:NF \l_stex_key_title_tl{
8508      {~}\l_stex_key_title_tl
8509    } :~}
8510  }{
8511    \par\smallskip\hrule
8512  }{}
```

gnote (*env.*)

```
8513  \cs_new_protected:Nn \__problems_gnote_start:n {
8514    \str_set:Nn \l_stex_key_title_tl {#1}
8515    \str_set:Nn \l_stex_key_id_str {#1}
8516    \str_if_empty:NT \l_stex_key_id_str {
8517      \int_gincr:N \g_problem_id_counter
8518      \str_set:Nx \l_stex_key_id_str {
8519        GNOTE_\int_use:N \g_problem_id_counter
8520      }
8521    }
8522    \stex_if_do_html:T{
8523      \begin{stex_annotate_env}{
8524        shtml:problemgnote=\l_stex_key_id_str
8525      }
8526    }
8527    \stex_style_apply:
8528  }
8529
8530  \stex_new_stylable_env:nnnnnnn { gnote }{ O{} }{
8531    \stex_if_do_html:TF{
8532      \__problems_gnote_start:n{#1}
8533    }{
8534      \bool_if:NTF \c__problems_gnotes_bool {
8535        \__problems_gnote_start:n{#1}
8536      }{
8537        \setbox\l_tmpa_box\vbox\bgroup
8538      }
8539    }
8540  }{
8541    \stex_if_do_html:TF{
8542      \stex_style_apply:
8543      \end{stex_annotate_env}
8544    }{
```

```
8545      \bool_if:NTF \c__problems_gnotes_bool {
8546        \stex_style_apply:
8547        \stex_if_do_html:T{
8548          \end{stex_annotate_env}
8549        }
8550      }{
8551        \egroup
8552      }
8553    }
8554  }{
8555    \par\smallskip\hrule\smallskip
8556    \noindent\emph{Grading\str_if_empty:NF \l_stex_key_title_tl{
8557      {~}\l_stex_key_title_tl
8558    } :~}
8559  }{
8560    \par\smallskip\hrule
8561  }{}
```

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

```
8562  \def\pts#1{
8563    \bool_if:NT \c__problems_pts_bool {
8564      \marginpar{#1~\problem@kw@points}
8565    }
8566  }
8567  \def\min#1{
8568    \bool_if:NT \c__problems_min_bool {
8569      \marginpar{#1~\problem@kw@minutes}
8570    }
8571  }
```

mcb (*env.*)

```
8572  \newenvironment{mcb}{\par
8573    \stex_if_do_html:T{
8574      \begin{stex_annotate_env}{shtml:multiple-choice-block={}}
8575    }
8576    \stex_deactivate_macro:Nn \mcb {sproblem~environments}
8577    \stex_deactivate_macro:Nn \solution {sproblem~environments}
8578    \stex_reactivate_macro:N \mcc
8579    \begin{enumerate}
8580  }{
8581    \end{enumerate}
8582    \stex_if_do_html:T{
8583      \end{stex_annotate_env}
8584    }
8585  }
8586  \stex_deactivate_macro:Nn \mcb {sproblem~environments}
```

we define the keys for the mcc macro

```
8587  \cs_new_protected:Nn \__problems_do_yes_param:Nn {
8588    \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
8589      \bool_set_true:N #1
8590    }{
8591      \bool_set_false:N #1
8592    }
```

```
8593 }
8594 \stex_keys_define:nnnn{mcc}{
8595   \tl_clear:N \l_stex_key_feedback_tl
8596   \bool_set_false:N \l_stex_key_T_bool
8597   \tl_clear:N \l_stex_key_Ttext_tl
8598   \tl_clear:N \l_stex_key_Ftext_tl
8599 }{
8600   feedback  .tl_set:N     = \l_stex_key_feedback_tl ,
8601   T         .default:n    = { false } ,
8602   T         .bool_set:N   = \l_stex_key_T_bool ,
8603   F         .default:n    = { false } ,
8604   F         .code:n       = {\bool_set_false:N \l_stex_key_T_bool} ,
8605   Ttext     .tl_set:N     = \l_stex_key_Ttext_tl ,
8606   Ftext     .tl_set:N     = \l_stex_key_Ftext_tl ,
8607 }{id}
8608
```

\mcc

```
8609 \tl_set:Nn \problem_mcc_box_tl {
8610   \ltx@ifpackageloaded{amssymb}{$\square$}{
8611     \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
8612   }
8613 }
8614 \newcommand\mcc[2][]{
8615   \stex_keys_set:nn{mcc}{#1}\par
8616   \tl_set:Nn \l_tmpb_tl {~---~
8617     \bool_if:NTF \l_stex_key_T_bool {
8618       \tl_if_empty:NTF \l_stex_key_Ttext_tl \problem@kw@correct \l_stex_key_Ttext_tl
8619     }{
8620       \tl_if_empty:NTF \l_stex_key_Ftext_tl \problem@kw@wrong \l_stex_key_Ftext_tl
8621     }
8622     \tl_if_empty:NF \l_stex_key_feedback_tl {
8623       \\\emph{\l_stex_key_feedback_tl}
8624     }
8625   }
8626   \tl_set:Nn \l_tmpa_tl {
8627     #2
8628     \stex_if_do_html:TF{
8629       \stex_annotate:nn{shtml:mcc-solution={}}{\l_tmpb_tl}
8630     }{
8631       \ifsolutions\l_tmpb_tl\fi
8632     }
8633   }
8634   \item[\problem_mcc_box_tl]{}~
8635   \stex_if_do_html:TF{
8636     \stex_annotate:nn{shtml:mcc={
8637       \bool_if:NTF \l_stex_key_T_bool {true}{false}
8638     }}{\l_tmpa_tl}
8639   }{\l_tmpa_tl}
8640 }
8641 \stex_deactivate_macro:Nn \mcc {mcb~environments}
```

(*End definition for* \mcc. *This function is documented on page* *101*.)

```
8642 \newcommand\fillinsol[2][]{
8643   \quad
8644   \stex_if_do_html:TF{
8645     \stex_annotate:nn{shtml:fillinsol={}}{ \_stex_annotate_force_break:n{#2} }
8646   }{
8647     \ifsolutions
8648       \textcolor{red}{\fbox{#2}}
8649     \else
8650       \fbox{
8651         \tl_if_empty:nTF{#1}{
8652           \phantom{\huge{#2}}
8653         }{
8654           \hspace{#1}
8655         }
8656       }
8657     \fi
8658   }
8659   \quad
8660 }
8661 \stex_deactivate_macro:Nn \fillinsol {sproblem~environments}
```

(*End definition for* \fillinsol. *This function is documented on page* *103*.)

```
8662 ⟨/package⟩
```

14.3 Implementation: The hwexam Package

14.3.1 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.

```
8663 ⟨*package⟩
8664 \ProvidesExplPackage{hwexam}{2023/03/19}{3.3.0}{homework assignments and exams}
8665 \RequirePackage{l3keys2e}
8666
8667 \keys_define:nn {hwexam / pkg}{
8668 multiple   .default:n  = { false },
8669 multiple .bool_set:N  = \c_hwexam_multiple_bool,
8670 unknown    .code:n  = {
8671 \PassOptionsToPackage{\CurrentOption}{problem}
8672 }
8673 }
8674 \ProcessKeysOptions{ hwexam /pkg }
8675 \RequirePackage{problem}
```

\hwexam_kw_* For multilinguality, we define internal macros for keywords that can be specialized in *.ldf files.

```
8676 \AddToHook{begindocument}{
8677 \ExplSyntaxOn\makeatletter
8678 \input{hwexam-english.ldf}
8679 \ltx@ifpackageloaded{babel}{
```

```
8680 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
8681 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
8682 \input{hwexam-ngerman.ldf}
8683 }
8684 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
8685 \input{hwexam-finnish.ldf}
8686 }
8687 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
8688 \input{hwexam-french.ldf}
8689 }
8690 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
8691 \input{hwexam-russian.ldf}
8692 }
8693 }{}
8694 \makeatother\ExplSyntaxOff
8695 }
```

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

14.3.2 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.

assignment (*env.*)

```
8696 \stex_keys_define:nnnn{ assignment }{
8697   \tl_clear:N \l_stex_key_number_tl
8698   \tl_clear:N \l_stex_key_given_tl
8699   \tl_clear:N \l_stex_key_due_tl
8700 }{
8701   number  .tl_set:N    = \l_stex_key_number_tl,
8702   given   .tl_set:N    = \l_stex_key_given_tl,
8703   due     .tl_set:N    = \l_stex_key_due_tl,
8704 unknown .code:n = {}
8705 }{id,title,style}
8706
8707 \newcounter{assignment}
8708 \stex_new_stylable_env:nnnnnnn {assignment}{O{}}{
8709   \cs_if_exist:NTF \l_hwexam_includeassignment_keys_tl {
8710     \tl_put_left:Nn \l_hwexam_includeassignment_keys_tl {#1,}
8711     \exp_args:Nno \stex_keys_set:nn{assignment}{
8712       \l_hwexam_includeassignment_keys_tl
8713     }
8714   }{
8715     \stex_keys_set:nn{assignment}{#1}
8716   }
8717 \tl_if_empty:NF \l_stex_key_number_tl {
8718 \global\setcounter{assignment}{\int_eval:n{\l_stex_key_number_tl-1}}
8719 }
8720 \global\refstepcounter{assignment}
8721 \setcounter{sproblem}{0}
8722 \def\thesproblem{\theassignment.\arabic{sproblem}}
8723 \stex_style_apply:
```

```
8724  \_stex_do_id:
8725  }{
8726  \stex_style_apply:
8727  }{
8728  \par\begin{center}
8729  \textbf{\Large\assignmentautorefname~\theassignment
8730  \tl_if_empty:NF \l_stex_key_title_tl {
8731  {~}---~\l_stex_key_title_tl
8732  }
8733  }\par\smallskip
8734  \textbf{
8735  \tl_if_empty:NF \l_stex_key_given_tl {
8736  \hwexam@kw@given:~\l_stex_key_given_tl\quad
8737  }
8738  \tl_if_empty:NF \l_stex_key_due_tl {
8739  \hwexam@kw@due:~\l_stex_key_due_tl\quad
8740  }
8741  }
8742  \end{center}
8743  \par\bigskip
8744  }{
8745  \par\pagebreak
8746  }{}
```

\includeassignment

```
8747  \NewDocumentCommand\includeassignment{O{} m}{
8748  \group_begin:
8749  \tl_set:Nn \l_hwexam_includeassignment_keys_tl {#1}
8750  \stex_keys_set:nn{includeproblem}{#1}
8751  \exp_args:Nno \use:nn{\inputref[]\l_stex_key_mhrepos_str]{#2}
8752  \group_end:
8753  }
```

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

Restoring information about problems:

```
8754  \prop_new:N \c_@@_problems_prop
8755  \tl_set:Nn \c_@@_total_mins_tl {0}
8756  \tl_set:Nn \c_@@_total_pts_tl {0}
8757  \int_new:N \c_@@_total_problems_int
8758  \cs_set_protected:Npn \problem@restore #1 #2 #3 {
8759  \int_gincr:N \c_@@_total_problems_int
8760  \prop_gput:Nnn \c_@@_problems_prop {#1}{{#2}{#3}}
8761  \tl_gset:Nx \c_@@_total_pts_tl { \int_eval:n { \c_@@_total_pts_tl + #2 }}
8762  \tl_gset:Nx \c_@@_total_mins_tl { \int_eval:n { \c_@@_total_mins_tl + #2 }}
8763  }
```

\correction@table This macro generates the correction table

```
8764  \newcommand\correction@table{
8765  \int_compare:nNnT \c_@@_total_problems_int = 0 {
8766  \int_incr:N \c_@@_total_problems_int
8767  \prop_put:Nnn \c_@@_problems_prop {~}{{~}{~}}
8768  }
8769  \tl_clear:N \l_tmpa_tl
8770  \tl_clear:N \l_tmpb_tl
```

```
8771 \tl_clear:N \l_tmpc_tl
8772 \prop_map_inline:Nn \c_@@_problems_prop {
8773 \tl_put_right:Nn \l_tmpa_tl { ##1 & }
8774 \tl_put_right:Nx \l_tmpb_tl { \use_i:nn ##2 & }
8775 \tl_put_right:Nn \l_tmpc_tl { & }
8776 }
8777 \resizebox{\textwidth}{!}{%
8778 \exp_args:Nne \begin{tabular}{|l|*{\int_use:N \c_@@_total_problems_int}{c|}c||l|}\hline
8779 &\exp_args:Ne \multicolumn{\int_eval:n{ \c_@@_total_problems_int + 1}}{c||}
8780 {\footnotesize\hwexam@kw@forgrading} &\\\hline
8781 \hwexam@kw@probs & \l_tmpa_tl \hwexam@kw@sum & \hwexam@kw@grade\\\hline
8782 \hwexam@kw@pts & \l_tmpb_tl \c_@@_total_pts_tl & \\\hline
8783 \hwexam@kw@reached & \l_tmpc_tl & \\[.7cm]\hline
8784 \end{tabular}}}
```

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

\testheading

```
8785 \def\hwexamheader{\input{hwexam-default.header}}
8786
8787 \def\hwexamminutes{
8788 \tl_if_empty:NTF \hwexam@duration {
8789 {\hwexam@min}~\hwexam@minutes@kw
8790 }{
8791 \hwexam@duration
8792 }
8793 }
8794
8795 \stex_keys_define:nnnn{ hwexam / testheading }{
8796 \tl_clear:N \hwexam@min
8797 \tl_clear:N \hwexam@duration
8798 \tl_clear:N \hwexam@reqpts
8799 \tl_clear:N \hwexam@tools
8800 }{
8801 min    .tl_set:N  = \hwexam@min,
8802 duration .tl_set:N  = \hwexam@duration,
8803 reqpts .tl_set:N  = \hwexam@reqpts,
8804 tools  .tl_set:N  = \hwexam@tools
8805 }{}
8806
8807 \newenvironment{testheading}[1][]{
8808 \stex_keys_set:nn { hwexam / testheading}{#1}
8809
8810 \tl_set_eq:NN \hwexam@totalpts \c_@@_total_pts_tl
8811 \tl_set_eq:NN \hwexam@totalmin \c_@@_total_mins_tl
8812 \tl_set:Nx \hwexam@checktime {\int_eval:n { \hwexam@min - \hwexam@totalmin }}
8813
8814 \newif\if@bonuspoints
8815 \tl_if_empty:NTF \hwexam@reqpts {
8816 \@bonuspointsfalse
8817 }{
8818 \tl_set:Nx \hwexam@bonuspts {
8819 \int_eval:n{\hwexam@totalpts - \hwexam@reqpts}
8820 }
```

8821 `\@bonuspointstrue`
8822 `}`
8823
8824 `\makeatletter\hwexamheader\makeatother`
8825 `}{`
8826 `\newpage`
8827 `}`

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

`\testemptypage`

8828 `\newcommand\testemptypage[1][]{%`
8829 `\bool_if:NT \c__problems_test_bool {\ \vfill\begin{center}\hwexam@kw@testemptypage\end{cente`
8830 `}`

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

`\testspace`

8831 `\newcommand\testspace[1]{\bool_if:NT \c__problems_test_bool {\vspace*{#1}}}`

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

`\testnewpage`

8832 `\newcommand\testnewpage{\bool_if:NT \c__problems_test_bool {\newpage}}`

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

8833 `⟨/package⟩`

14.3.3 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}
```

14.4 Tikzinput Implementation

```
8834 ⟨@@=tikzinput⟩
8835 ⟨*package⟩
8836
8837 %%%%%%%%%%%%   tikzinput.dtx   %%%%%%%%%%%%
8838
8839 \ProvidesExplPackage{tikzinput}{2023/03/19}{3.3.0}{tikzinput package}
8840 \RequirePackage{l3keys2e}
8841
8842 \keys_define:nn { tikzinput } {
8843   image    .bool_set:N  = \c_tikzinput_image_bool,
8844   image    .default:n   = false ,
8845   unknown    .code:n       = {}
8846 }
8847
8848 \ProcessKeysOptions { tikzinput }
8849
8850 \bool_if:NTF \c_tikzinput_image_bool {
8851   \RequirePackage{graphicx}
8852
8853   \providecommand\usetikzlibrary[]{}
8854   \newcommand\tikzinput[2][]{\includegraphics[#1]{#2}}
8855 }{
8856   \RequirePackage{tikz}
8857   \RequirePackage{standalone}
8858
8859   \newcommand \tikzinput [2] [] {
8860     \setkeys{Gin}{#1}
8861     \ifx \Gin@ewidth \Gin@exclamation
8862       \ifx \Gin@eheight \Gin@exclamation
8863         \input { #2 }
8864       \else
8865         \resizebox{!}{ \Gin@eheight }{
8866           \input { #2 }
8867         }
8868       \fi
8869     \else
8870       \ifx \Gin@eheight \Gin@exclamation
8871         \resizebox{ \Gin@ewidth }{!}{
8872           \input { #2 }
8873         }
8874       \else
8875         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
8876           \input { #2 }
8877         }
8878       \fi
8879     \fi
8880   }
8881 }
8882
8883 \newcommand \ctikzinput [2] [] {
8884   \begin{center}
8885     \tikzinput [#1] {#2}
```

```
8886      \end{center}
8887  }

8888  ⟨/package⟩
```

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

322

323

328

330

334

336

340

349

351

352