The sTeX3 Package Collection*

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

2023-10-13

Contents

*Version 3.4.0 (last revised 2023-10-13)

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.

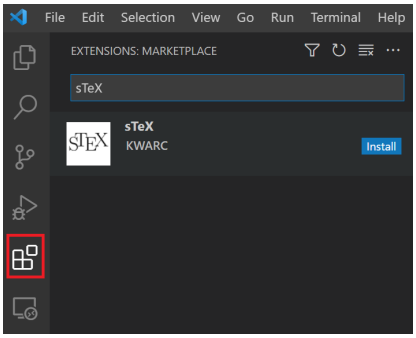
One considerable weakness of LaTeX is the way source files are referenced: they need to be either in a `texmf` directory, or else be referenced via file paths relative to the main `.tex`-file being compiled. This is highly inconvenient if we want to collaboratively develop many highly interrelated document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



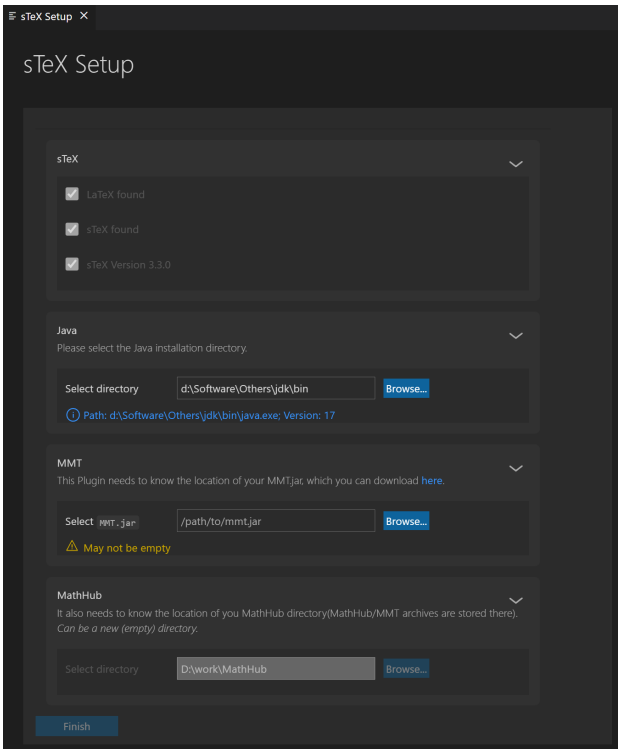


Figure 1: Installing the sTeX IDE

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 RusTeX 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
https://stexmmt.mathhub.info/:
sTeX/fullhtml?archive=sTeX/Documentation&filepath=tutorial.en.xhtml

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:

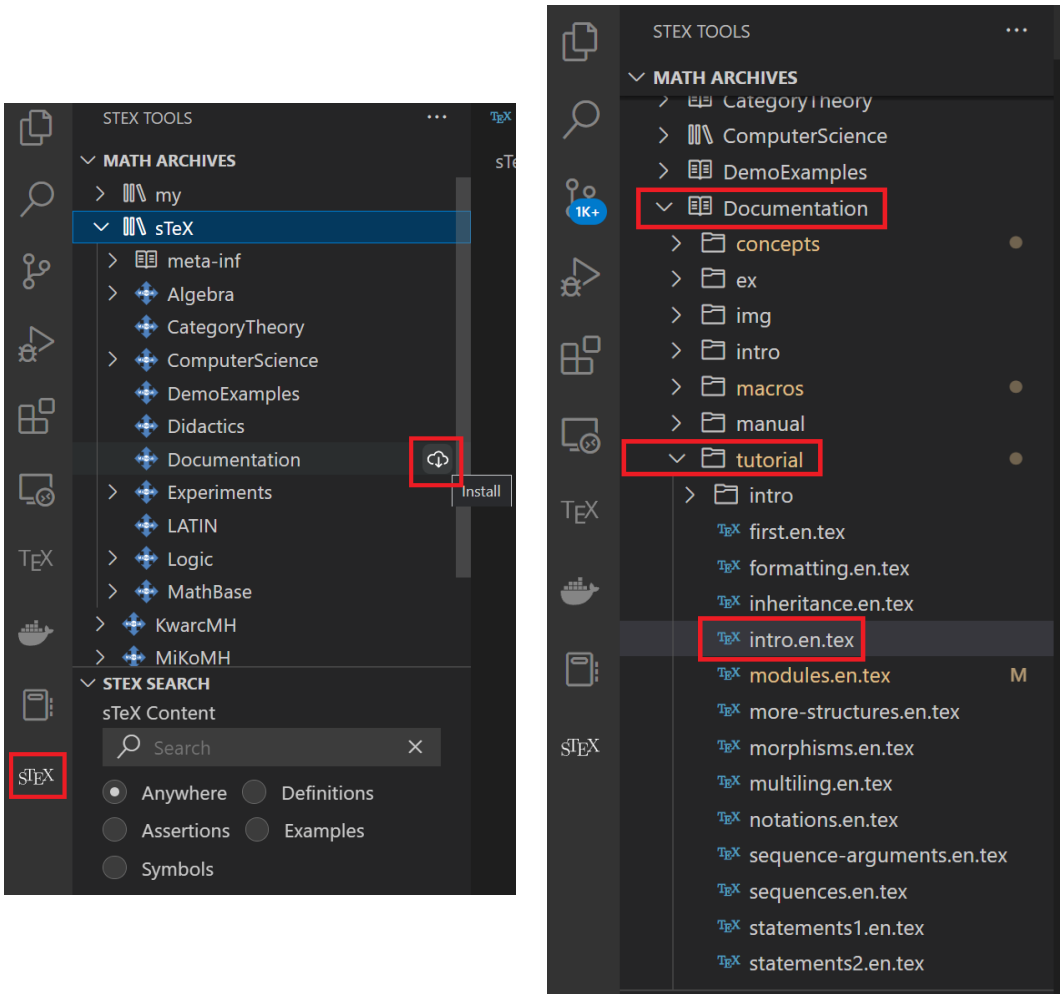


Figure 2: Installing Math Archives

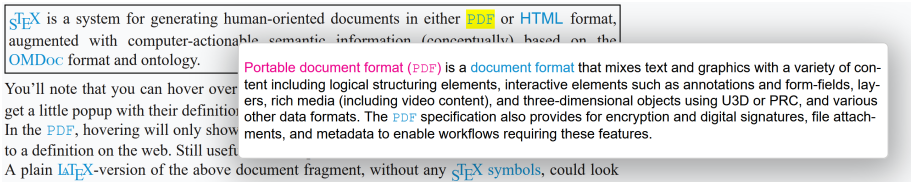


Figure 3: Definition on Hover

File [sTeX/Documentation]tutorial/intro/intro1plain.en.tex

```
\documentclass{article}
\usepackage{stex-logo}
\begin{document}

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

\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).

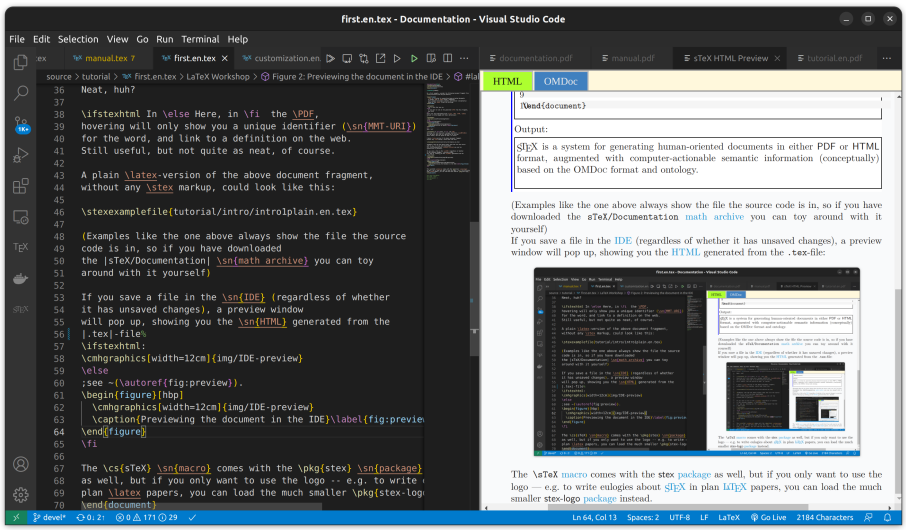


Figure 4: Previewing the document in the IDE

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*:

> sTeX 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).

> sTeX 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.

> sTeX 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:

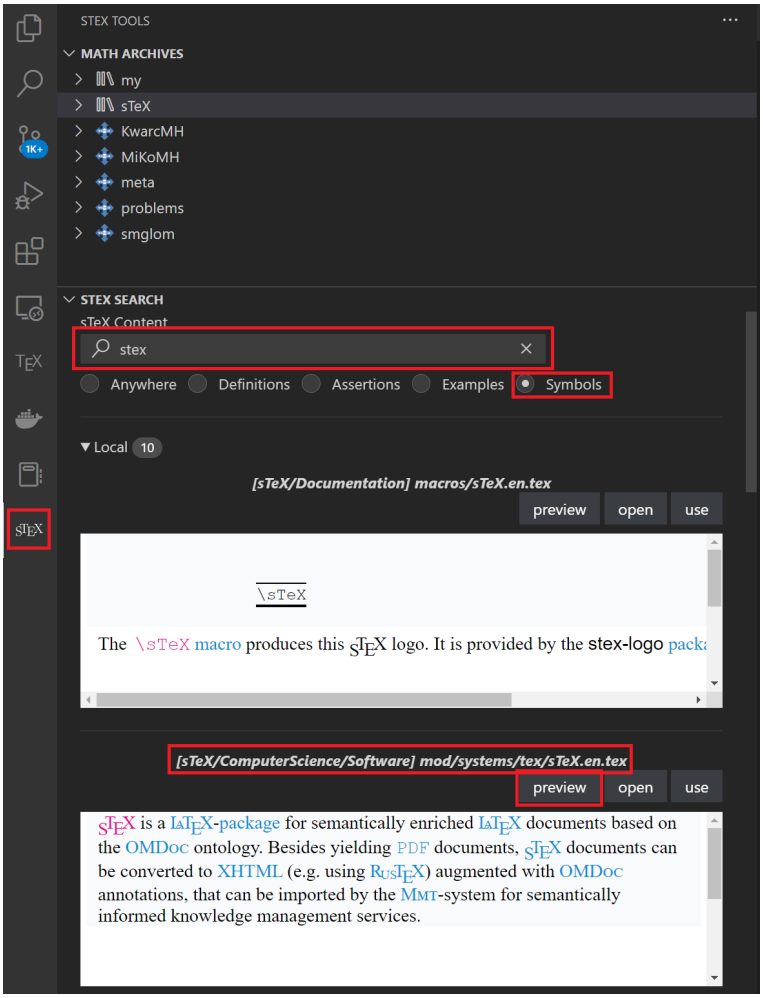


Figure 5: Search in the sTeX IDE

Figure 6: OMDoc Preview

Example 2

Input:

File [sTeX/Documentation]tutorial/intro/intro1stex.en.tex

```
\documentclass{article}
\usepackage{stex}
\begin{document}
  \usemodule[sTeX/ComputerScience/Software]{mod/systems/tex?sTeX}
  \usemodule[sTeX/ComputerScience/Software]{mod/formats?PDF}
  \usemodule[sTeX/ComputerScience/Software]{mod/formats?HTML}
  \usemodule[sTeX/ComputerScience/Software]{mod/formats?OMDoc}

  \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.
\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.

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

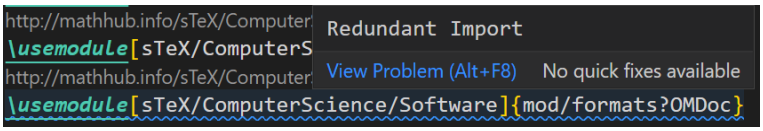


Figure 7: Redundant Imports

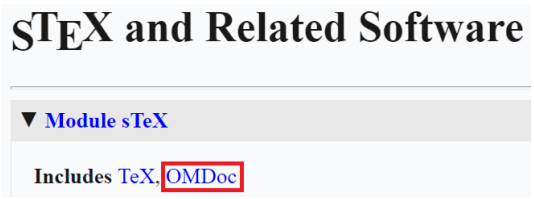


Figure 8: Includes in the OMDoc Preview

- 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:

File [sTeX/Documentation]tutorial/intro/intro2plain.en.tex

```
\documentclass{article}
\usepackage{stex}
\begin{document}

  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 \texttt{pdflatex}
  over \sTeX-annotated documents formats them into normal-looking
  \textsf{PDF}.

\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,

2. the name of its semantic macro,
3. or any suffix of its MMT-URI containing at least the module name.

The second option is often short – and therefore convenient to write; for example, to achieve "formulae", we can also write `\symref{wff}{formulae}`, since `\wff` is the semantic macro for "`well-formed formula`".
The third option allows for distinguishing between multiple symbols with the same name – the IDE can help in the latter case, by underlining ambiguous symbol references in yellow, and offering the `Quick Fix` functionality to let you select and autocomplete the specific symbol you want to reference.

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

```
\usemodule[sTeX/ComputerScience/Software]{mod/systems/tex?sTeX}
\usemodule[sTeX/Logic/General]{mod/syntax?Formula}
\usemodule[sTeX/MathBase/General]{mod?Mathematics}
\usemodule[sTeX/ComputerScience/Software]{mod/formats?PDF}

  At its core is the \stex \sn{package} for \latex, that allows for
  semantically marking up document fragments; in particular
  concepts, \sr{wff}{formulae} and \sr{mathematics}{mathematical}
  statements (such as definitions, theorems and proofs). Running
  \texttt{pdflatex} over \stex-annotated documents formats them
  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

```
\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 statements (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

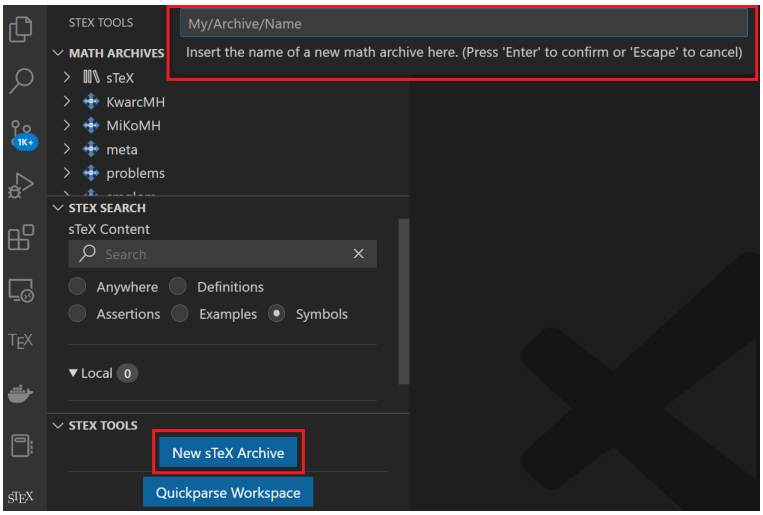


Figure 9: New Math Archive in the IDE

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 directory:

```
- 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 "LaTeX" 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.
>
> **LaTeX** is a (nowadays) default collection of TeX macros developed by Leslie Lamport. Among other things, LaTeX 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, LaTeX, 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) –

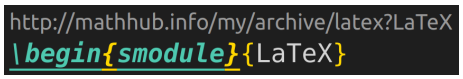


Figure 10: VS Code Code Lense

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).

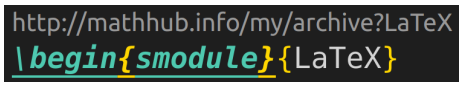


Figure 11: VS Code Code Lense

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).

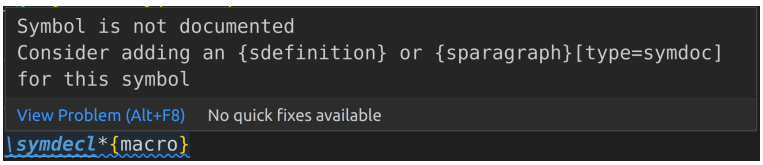


Figure 12: Undocumented Symbols

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 SᴛEX 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 SᴛEX 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
```

```
 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

```
\documentclass{stex}
\libinput{preamble}
\begin{document}
\begin{smodule}{sTeX}
  \usemodule{OMDoc}
  \usemodule{PDF}
  \usemodule{HTML}
  \textsymdecl{stex}{\sTeX}
  \begin{sparagraph}[style=symdoc]
    \definiendum{stex}{\stexname} 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.
  \end{sparagraph}
\end{smodule}
\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).

Lösung: 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?

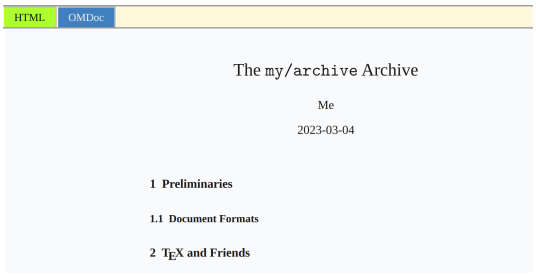


Figure 13: Missing Fragments in the HTML Preview

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 \inputrefed fragment *somewhere*. Meaning: we have to compile all of the fragments

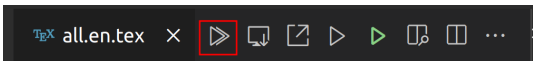


Figure 14: The `Build PDF/XHTML/OMDoc` Button

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`.
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).

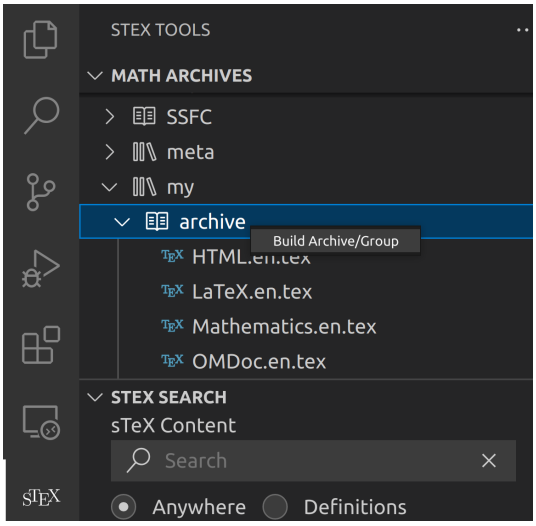


Figure 15: Building Archives in the IDE

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).

Figure 16: Exporting HTML in the IDE

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 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	TEX 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	, ;

TEX "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 TEX 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.

Lösung: 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

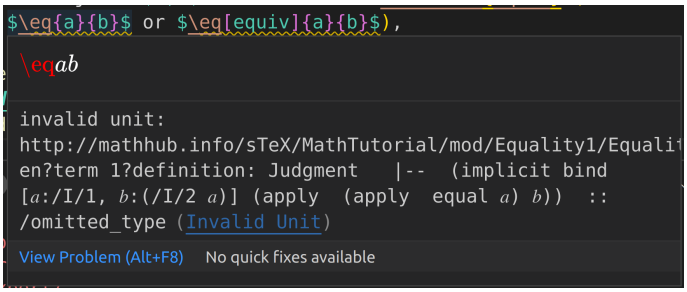


Figure 17: Type Checking Warning

a somewhat cryptic stacktrace-like window (Figure 18). The reason being, that MMT

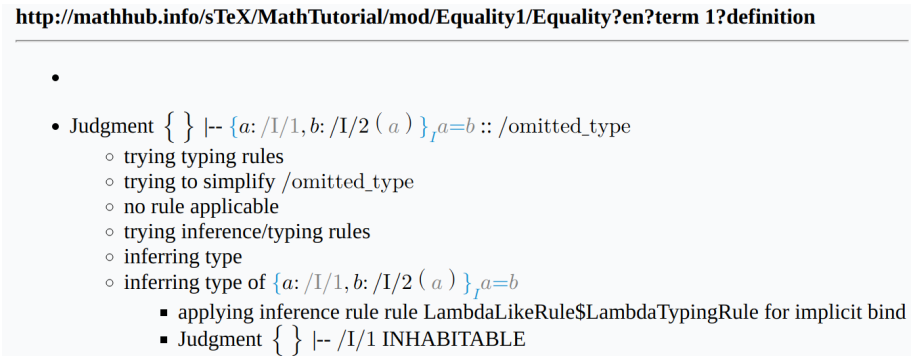


Figure 18: Type Checking Proof Tree

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.

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.

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.

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

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

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	$(A : \mathtt{SET}, B : \mathtt{SET}) \to \mathtt{SET}$	
Notations	**id**	**notation**
	arrowtimes	$a_1 \times ... \times a_n \to b$
	arrowcurry	$a_1 \to ... \to a_n \to b$
	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 $a_1 \times ... \times a_n \to b$.

We want addition to have type $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$, 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.

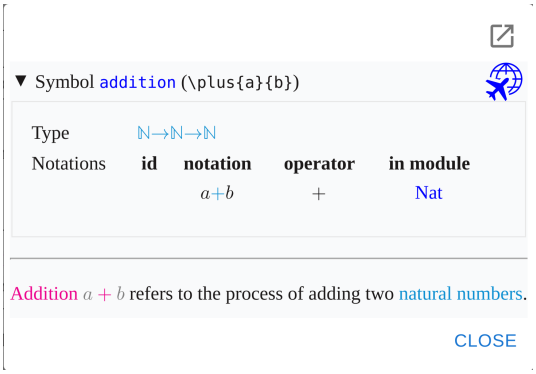


Figure 20: On-Click Popup in the HTML

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** 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].

[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 STEX; 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.

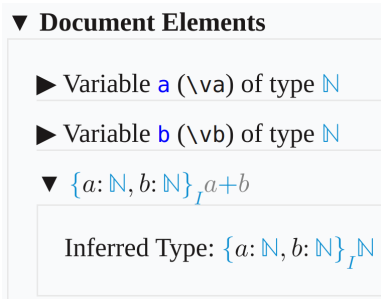


Figure 21: Inferred Type

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.

sTEX

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 ex-

pression, we hence do `\symdecl{add}[args=biii]`,

- 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 $a \cdot b \cdot c$.

Lösung: 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 SᴛEX directly; we say that it is part of the SᴛEX.)

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 SᴛEX.

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

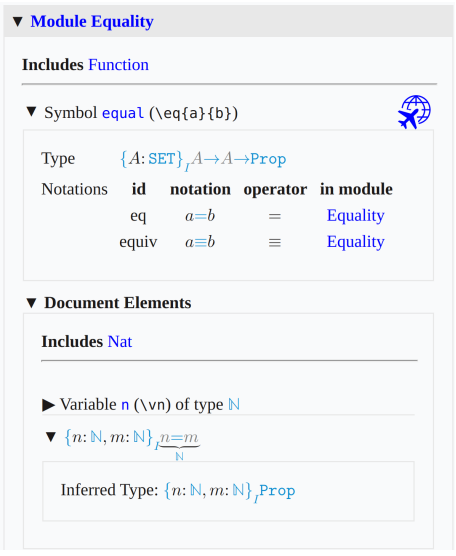


Figure 22: Implicit Arguments

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

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:

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

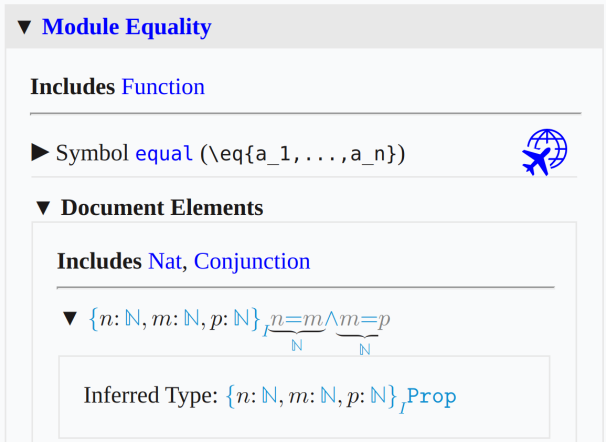


Figure 23: Conjunction of Equalities

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

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`

```
\begin{smodule}{Associative}
  \importmodule{mod?Equality}

  \symdecl*{associative}[args=1]
  \begin{sdefinition}[for=associative]
    \vardef{vA}[name=A,type=\collection]{A}
    \vardef{vop}[name=op,type=\funspace{\vA,\vA}\vA,args=a,assoc=bin]
      {\argsep{#1}{\mathbin{\maincomp{\circ}}}}
    %
    A binary operation $\fun{\vop!}{\vA,\vA}\vA$ is called
    \definame{associative}, if
    $\eq{
      \vop{(\vop{a,b}),c},
      \vop{a,(\vop{b,c})}
    }$ for all $\inset{a,b,c}\vA$.
  \end{sdefinition}
\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 TEX 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 MMT 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

```
\begin{sdefinition}[name=associative,args=1]
  \vardef{vA}[name=A,type=\collection]{A}
  \vardef{vop}[name=op,type=\funspace{\vA,\vA}\vA,
    args=a,assoc=bin,bind % <- argument for the symbol
  ]{\argsep{#1}{\mathbin{\maincomp{\circ}}}}
  \vardef{va}[name=a,type=\vA]{a}
  \vardef{vb}[name=b,type=\vA]{b}
  \vardef{vc}[name=c,type=\vA]{c}
  %
  A binary operation $\fun{\vop!}{\vA,\vA}\vA$ is called
  \definame{associative}, if
  \definiens{\foral{$\arg[2]{\eq{
    \vop{(\vop{\va,\vb}),\vc},
    \vop{\va,(\vop{\vb,\vc})}
  }}$ \comp{for all} $\arg[1]{\inset{\va,\vb,\vc}\vA}$}}.
\end{sdefinition}
%
```

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).

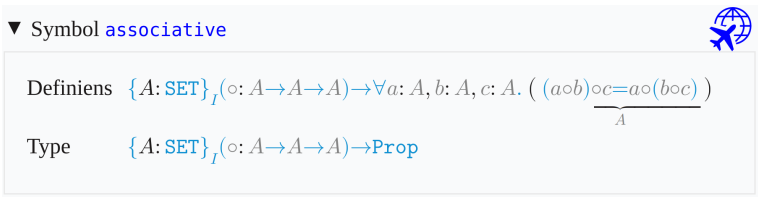


Figure 24: Type Inferred from Definiens

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`.

Lösung: 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:

[5] TODO: intent?

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

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 `sdefinitions`:

```
\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).

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`".

Lösung: 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 $\vdash \forall\varphi\colon \mathbb{N}\to \mathtt{Prop}.\varphi(0)\Rightarrow (\forall m\colon \mathbb{N}.\varphi(m)\Rightarrow\varphi(m+1)) \Rightarrow (\forall n\colon \mathbb{N}.\varphi(n))$

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*.

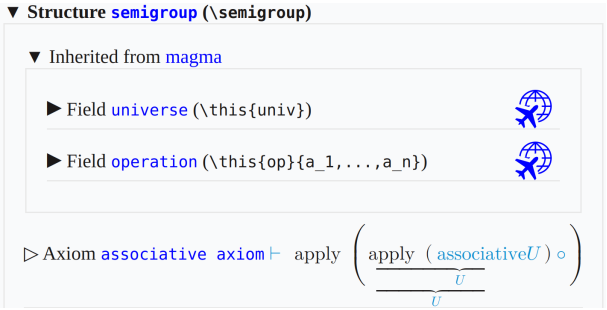


Figure 27: Axioms in OMDoc

Conservative Extensions

For structures, there is a *critical* distinction between *defined* and *undefined* symbols; and analogously between *theorems* and *axioms*.

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:

File [sTeX/MathTutorial]algebra/MagmaSquare.en.tex

```
\begin{extstructure*}{magma}
  \begin{sdefinition}[macro=sq,args=1]
    \notation{sq}[op=\cdot^2]{{#1}^{\comp 2}}
    \vardef{va}[name=a,type=\univ,bind]{a}
    Let $\inset{\va}{\univ}$. We define
    $\defnotation{\sq{\va}} := \definiens{\op{\va,\va}}$.
  \end{sdefinition}
\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., as semigroup extends magma, 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

```
\begin{mathstructure}{magmahom}[magma homomorphism,this=F]
  \symdef{dom}[name=domain,return={\magma[comp={##1}_1]}]{M_1}
  \symdef{cod}[name=codomain,return={\magma[comp={##1}_2]}]{M_2}
  \symdef{f}[op=\this,args=1,
    type=\funspace{\dom{univ}}{\cod{univ}}
  ]{\this \dobrackets{#1}}

  \begin{sdefinition}[for={magmahom,dom,cod,f}]
    \vardef{va}[name=a,type=\dom{univ}]{a}
    \vardef{vb}[name=b,type=\dom{univ}]{b}
    Let $\dom!=\dom{}$ and $\cod!=\cod{}$ \sns{magma}.
    A \definame{magmahom} is a function
    $\fun{\f!}{\dom{univ}}{\cod{univ}}$ such that
    \inlineass[name=homomorphism condition]{\conclusion{\foral{
      $\arg[2]{\eq{
        \f{\dom{op}}{\va,\vb}}, \cod{op}{\f{\va},\f{\vb}}
      }}$ \comp{for all} $\arg[1]{\inset{\va,\vb}{\dom{univ}}}$.
    }}}
  \end{sdefinition}
\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 $\circ$.

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

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 $b \in B$.

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

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 $b \in B$.

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

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

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

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

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

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

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

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

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

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 $a, b, c \in A$.

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

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 $a, b \in A$.

[sTeX/MathTutorial]orders/Graph.en.tex

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 $\vee$ absorbs $\wedge$ and vice versa; i.e. $a \vee (a \wedge b) = a$ and $a \wedge (a \vee b) = a$.
The operations $\wedge$ and $\vee$ 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 axioms` 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

```
\begin{extstructure*}{semilattice}
  \realize{meetsl}{
    univ = \univ,
    meet = \op!,
    rel @ [order]order = \map{a,b}{\eq{\op{a,b},a}},
    reflexive axiom = trivial,
    transitive axiom = trivial,
    antisymmetric axiom = trivial,
    infimum axiom = trivial
  }
\end{extstructure*}

\vardef{mysl}[return=\semilattice]{S}
$\mysl{order}{a,b} \qquad \mysl{}[univ,op,order]$
```

Output:

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

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. In fact, educational documents have been one of the initial document categories sTeX has been developed for. The idea is that if we can mark up the meaning and didactic role of learning objects, we can base learning support services on that and embed them into the documents.

Another reason educational documents are particularly interesting is that in a sense all academic communication is educational, as all documents try to "teach" the reader new concepts and results.

Concretely, we cover a document class for combining slides and course notes (section 4.1) and functionality for marking up problems and excercises (section 4.2) and for marking up homework assignments and exams (section 4.3).

4.1 Slides and Course Notes

TODO[6]

Contents

4.2 Problems and Exercises

4.2.1 Background

Problems/exercises are text fragments that contain a task assigned to the learner – e.g. computing the value of a specified quantity, simplifying an expression, modeling a described situation in a mathematical structure, or judging the veracity of a given statement. Problems/exercises are used in very different contexts, in

[6] TODO: notesslides.sty

- *homework assignments and formal exams*, where they are graded and points are awarded for course credit,
- *self-study materials* that allow learners explore applications of some concepts and/or practice,
- *automated tutoring systems* to determine learner competencies to trigger computer supported learning services.

In all of these contexts, diagnosis the correctness or suitability of an answer plays a great role, e.g. for grading, the generation of feedback, or the suggestion of remedial actions that may "heal" any diagnosed competency gaps or misconceptions. To support that we might want to specify "answer classes" that classify answers received for automating/triggering feedback and grading.

There are various types of problems/exercises in practical use. They are mainly distinguished by how they support diagnosis of student answers: there are

- open problems, where expected answers are free-form, and classification into answer classes is usually a manual process; see subsection 4.2.3
- single/multiple choice questions where the answer classes and thus feedback/grading correspond to the selection pattern of items; see subsection 4.2.5.
- fill-in-the-blanks questions where we may want to specify how the answer string is interpreted; see subsection 4.2.6.

Additionally, Problems and exercises often come with text fragments that serve auxiliary functions: hints, notes, and and grading specifications. Furthermore, we can specify how long solving a given problem is estimated to take and how many points will be awarded for a perfect solution – e.g. in an exam. See subsection 4.2.9 for details.

4.2.2 The Package, Options, and Configuration

The problem package provides functionality for marking up problems and exercises as semantic sources from which the presentations for the various contexts can be generated. E.g. documents without solutions for paper or online exams, and the corresponding exams with master solutions for exam reviews. Similarly with/without hints, or points. Their visibility is specified in the options of the problem package, which can be used in any LaTeX class. The following is a typical preamble for a problem file:

```
\documentclass[lang=de]{article}
\usepackage[solutions,hints,pts,min]{problem}
```

Here we have specified the options `solutions` (solutions should be shown), `hints` (hints should be given), `pts` (display the points awarded for solving the problem?), `min` (display the estimated minutes for problem solving). Leaving out the options would make the corresponding functionality invisible.

EdN:1 The `problem` package generates content and labels according to the language specified in the `lang` key of the main document[1], these can be localized by in the files `ldf/problem-<lang>.ldf`[7].

[1] EdNote: MK: document the attribute somewhere

[7] The current crop of language definition files is quite limited. Please feel free to contribute the to the localization effort

The `problem` package also supplies the `test` option, which specifies that the content is intended for use in an assessment situation, where certain additional content (e.g. exam legalse) should be shown while other content (e.g. solutions) should not be.

4.2.3 (Open) Problems

The main environment provided by the problem package is (surprise surprise) the `sproblem` environment. It is used to mark up problems and exercises. The following example shows the main functionality:

Example 16

Input:

File [sTeX/Documentation]tutorial/ext/simple-problem.en.tex

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

Output:

Exercise (Fitting Elefants)

How many Elefants can you fit into a Volkswagen beetle?

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

The `sproblem` environment takes an optional key/value 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.

The `sproblem` can contain several `solution` environments, which take the well-known `id`, `style`, and `title` attributes, and also the `testspace` and `answerclass`. The former

The additional functionality is specified in the `hint exnote` (notes in exercises), `solution`, and `gnote` (grading notes) environments. Here, the first three are shown whereas the grading notes are hidden, since the corresponding option was not given in the `\usepackage[...]{problem}`. All of these environments can occur any number

of times in the `sproblem` environment. The `solution` environment takes an optional argument that is interpreted as the identifier.

4.2.4 Structured Problems

Problems can be structured into subproblems via the `subproblem` environment. It takes the same arguments and allows the same content model as as `sproblem`.

Example 17

Input:

File [sTeX/Documentation]tutorial/ext/structured-problem.en.tex

```
\begin{document}
\begin{sproblem}[id=prob.elefants-mod,title=Fitting Elefants Modularly]
  Consider the problem of fitting elephants into a Volkswagen beetle.
  \begin{subproblem}[pts=1,min=2]
    Estimate the number of elephants on the front seats.
    \begin{solution}
      Two on each side one.
    \end{solution}
  \end{subproblem}
  \begin{subproblem}[pts=1,min=2]
    Estimate the number of elephants on the back seats.
    \begin{solution}
      Three little elephants.
    \end{solution}
  \end{subproblem}
```

Output:

Exercise (Fitting Elefants Modularly)

Consider the problem of fitting elephants into a Volkswagen beetle.

1. Estimate the number of elephants on the front seats.

   Lösung: Two on each side one.

2. Estimate the number of elephants on the back seats.

   Lösung: Three little elephants.

3. What is the total number?

   Lösung: A family of five; we do not want elephants in the frunk.

By default, subproblems are numbered subordinate to the "mother problem". The `problem` package does not make any assumptions about whether subproblems can be used independently from their sister problems, or on order requirements.

4.2.5 Single/Multiple Choice Blocks

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 correct answers, `F` (the default value) for false ones,
- `Ttext` the verdict for correct 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 18

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} \mcc[F,feedback=that is for C and C++]{function}
6     \mcc[F,feedback=that is for Standard ML]{fun}
7     \mcc[F,Ftext=Noooooooooo,feedback=that is for Java]{public static void}
8   \end{mcb}
9 \end{sproblem}
```

Output:

Exercise (Functions)

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

- ☐ def[a]
- ☐ function[b]
- ☐ fun[c]
- ☐ public static void[d]

[a] Korrekt
[b] Falsch
that is for C and C++
[c] Falsch
that is for Standard ML
[d] Noooooooooo
that is for Java

In "exam mode" where disable solutions (here via `\stopsolutions`) we get the questions without solutions (that is what the students see during the exam/quiz).

Example 19

Input:

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

Output:

Exercise (Functions)

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

- ☐ def
- ☐ function
- ☐ fun
- ☐ public static void

What we have seen is the default rendering of the multiple choice block, which is as an itemize like environment with check boxes. There are alternatives to that which we can choose with the `style` key in the optional attribute of the `mcb` environment. Currently the only alternative is `inline` style:

Example 20

Input:

```
\startsolutions
\begin{sproblem}[title=Functions,name=functions1]
  What is the keyword to introduce a function definition in python?

  \begin{mcb}[style=inline]
    \mcc[T]{def} \mcc[F,feedback=that is for C and C++]{function}
    \mcc[F,feedback=that is for Standard ML]{fun}
    \mcc[F,Ftext=Nooooooooo,feedback=that is for Java]{public static void}
  \end{mcb}
\end{sproblem}
```

Output:

Exercise (Functions)

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

☐ def[a] ☐ function[b] ☐ fun[c] ☐ public static void[d]

[a]Korrekt
[b]Falsch
that is for C and C++
[c]Falsch
that is for Standard ML
[d]Noooooooooo
that is for Java

This is the solutions mode, i.e. with solutions, otherwise, the footnotes will fully disappear.

Analogously, single choice blocks are marked up by the `scb` environment and the individual choices by the `\scc` macro. In PDF, there is little difference to the multiple choice case. In interactive formats like HTML, single choice blocks are typically realized via radio buttons to enforce single choice.[2]

EdN:2

4.2.6 Filling-In Concrete Solutions

The next simplest situation, where we can implement auto-grading is the case where we have fill-in-the-blanks

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:

```
\stopsolutions
\begin{sproblem}[id=elefants.fillin,title=Fitting Elefants]
  How many Elefants can you fit into a Volkswagen beetle? \fillinsol{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:

[2]EdNote: MK: are there any differences between mcc and scc we need to discuss here?

```
1 \startsolutions
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? 4

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. [8]

4.2.7 Answer Classes

EdN:3

[3].

4.2.8 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.

4.2.9 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

[8] 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 in MCQ. This still needs a bit of design. Also we want to make the formatting of the answer in solutions/test mode configurable.

[3] EdNote: MK: describe them

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 vertical space accordingly. Specific instances exist: `\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 empty page with the cautionary message that this page was intentionally left empty.

`\testnewpage` The `solution` environment takes an TODO[9]

`\testemptypage`

`solution`

4.3 Homework Assignments and Exams

4.3.1 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.

4.3.2 Package Options

`solutions` 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).

`notes`

`hints`

`gnotes`

`pts`

`min`

`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.

4.3.3 Assignments

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

`number`

`title`

`type`

`given`

`due`

[9] TODO: check what is still undescribed problem.sty and make examples for it.

4.3.4 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.

4.3.5 Typesetting Exams

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

`duration`
`min`
`reqpts`

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

Will result in

Name: Matriculation Number:

320101 General Computer Science (Fall 2010)

2023-10-13

You have one hour (sharp) for the test;
Write the solutions to the sheet.
The estimated time for solving this exam is 23 minutes, leaving you 37 minutes for revising your exam.
You can reach 23 points if you solve all problems. You will only need 27 points for a perfect score, i.e. -4 points are bonus points.

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

	To be used for grading, do not write here								
prob.	4.1	1.1	1.2	2.1	2.2	2.3	2.4	2.5	2.6
total	0	0	0	10	0	0	0	0	0
reached									

good luck

10

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

Part II
User Manual

The dynamic HTML version of this part can be found at
https://stexmmt.mathhub.info/:
sTeX/fullhtml?archive=sTeX/Documentation&filepath=manual.xhtml

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 `\input`s 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 \inputrefs a section section1.tex, which references a definition with label some_definition in section2.tex (subsequently also \inputrefed 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`.[11]

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⟩}

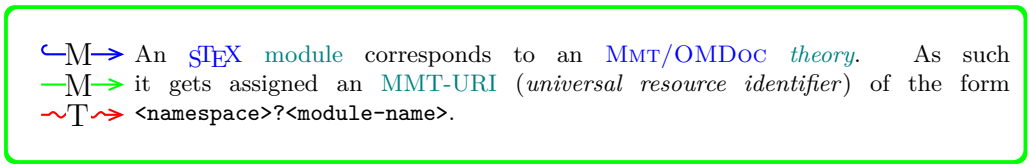
Like \sref, but with the third argument mandatory, \extref will *always* produce the output as if ⟨*label*⟩ would *not* occur in the current document.

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).



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

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

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

The `smodule`-environment takes several 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.

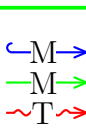
7.2 Symbol Declarations

\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.



\symdecl introduces a new MMT/OMDoc constant in the current module (i.e. MMT/OMDoc theory). Correspondingly, they get assigned the MMT-URI <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[12].

`\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

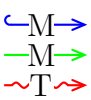
`\symref` `\sr`

`\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.

[12] 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` `\sn` `\Symname` `\Sn` `\sns` `\Sns`

`\symname[pre=`⟨*pre*⟩`,post=`⟨*post*⟩`]{`⟨*symbol*⟩`}`

If the symbol referenced by ⟨*symbol*⟩ has name `name`, this is a shortcut for `\symref{`⟨*symbol*⟩`}{`⟨*pre*⟩`name`⟨*post*⟩`}`.

For example, given a symbol `agroup` with name `abelian group`, we can do `\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` `\srefsymuri`

`\srefsym{`⟨*symbol*⟩`}{`⟨*text*⟩`}`

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

`\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
\maincomp

`\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` `\neginfprec` `\infprec` and `\neginfprec` represent *infinitely large* and *infinitely small* precedences, 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 23
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.

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[13]

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`.

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.

In MMT, every *document* and every module induces an MMT theory. `\usemodule` induces and MMT include in the document theory, `\importmodule` and `\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 accordingly. Largely, users need not concern themselves with namespaces at all, but for completenesses sake, we describe how they are constructed:

- If `\begin{smodule}{Foo}` occurs in a file `/path/to/file/Foo[.`⟨*lang*⟩`].tex` which does not belong to an 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 current 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` declared 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).

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

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.

Variables induces the same MMT/OMDoc terms as symbols do, except for the 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.

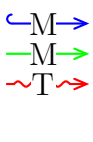
mathstructure (env.) \begin{mathstructure}{⟨mname⟩}[⟨name⟩,this=⟨code⟩] opens a new mathematical structure with name ⟨mname⟩ (if provided) or ⟨mname⟩ (otherwise), and semantic macro \mname. It subsequently behaves like the smodule environment.

\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 24

```
\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:
  - `$\struct{}{}[`$\langle$*fieldname*$\rangle$`,...]$` 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{}{}[`$\langle$*id*$\rangle$`]$` lets you pick the notation of the "mathematical structure" symbol to use to typeset the structure; e.g. `$\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, StTeX 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 xb_Sy.
- For the sake of completion, `$\struct{first}!$` simply produces the given argument; e.g. `$\struct{S}!$` simply produces S.

More precisely: `$\struct{`$\langle$*code*$\rangle$`}$` acts like a "type coercion" of $\langle$*code*$\rangle$ 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 StTeX does when using `\struct` in the `return` of a symbol (or variable).

Continuing:

- `$\struct[comp=`$\langle$*code*$\rangle$`]{...}{...}$` applies $\langle$*code*$\rangle$ (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 xb^Sy.
- `$\struct[this=`$\langle$*code*$\rangle$`]{...}{...}$` 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, `$\struct[`$\langle$*fieldname*$\rangle$`=`$\langle$*code*$\rangle$`]{...}{...}$` *assigns* the field $\langle$*fieldname*$\rangle$ to the term $\langle$*code*$\rangle$. $\langle$*code*$\rangle$ 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 (primarily 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, the `for=` attribute specifies the concept this is an example for. For counterexamples use `style=counterexample`

`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 SᴛEX 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/OMDoc 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 $t := \texttt{imply}(p_1, ..., p_n, c)$.
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 $t := \texttt{judgment}(t)$.
If no `forall` symbol is found, it will first apply the `judgment` symbol (if existent) and then use the bind symbol that ships with SᴛEX 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[14]

[14] TODO: proofs

Chapter 9

Customizing Typesetting

There are two kinds of typesetting that can be customized in SᴛEX: 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 \symrefs, 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 `sdefinition`s, 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 `sasssertion`, 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}}
```

\stexstylemodule
\stexstylecopymodule
\stexstyleinterpretmodule
\stexstylerealization
\stexstylemathstructure
\stexstyleextstructure
\stexstyledefinition
\stexstyleassertion
\stexstyleexample
\stexstyleparagraph
\stexstyleproof
\stexstylesubproof

TODO[15]

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 $\mathbb{N}$.

\stexstyleusemodule
\stexstyleimportmodule
\stexstylerequiremodule
\stexstyleassign
\stexstylerenamedecl
\stexstyleassignMorphism
\stexstylecopymod
\stexstyleinterpretmod
\stexstylerealize
\stexstylesymdecl
\stexstyletextsymdecl
\stexstylenotation
\stexstylevarnotation
\stexstylesymdef
\stexstylevardef
\stexstylevarseq
\stexstylespfsketch
\stexstyleMMTinclude

TODO[16]

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 STEX and OMDoc. To support semantic course notes, it extends the notion of mixing frames and explanatory text, but rather than treating the frames as images (or integrating their contents into the flowing text), the notesslides package displays the slides as such in the course notes to give students a visual anchor into the slide presentation in the course (and to distinguish the different writing styles in slides and course notes).

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

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).

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

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

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.

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

\setsource 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.

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

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
\useSGvar

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

\ifSGvar

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

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:

```
\excursion{founif}{../fragments/founif.en}
  {We will cover first-order unification in}
...
\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.

\excursion

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

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

\activateexcursion
\printexcursion
\excursionref

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

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

\excursiongroup

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

```
\begin{note}
\begin{sfragment}[id=<id>]{Excursions}
  \inputref{<path>}
  \printexcursions
\end{sfragment}
\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. 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`
`notes`
`hints`
`gnotes`
`pts`
`min`
`boxed`
`test`

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

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

`problem` (*env.*) The main environment provided by the problempackage 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 25
Input:

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

Output:

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

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

`\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 26

Input:

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

Output:

Exercise (Functions)

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

- ☐ def[a]
- ☐ function[b]
- ☐ fun[c]
- ☐ public static void[d]

---

[a] Korrekt

[b] Falsch

that is for C and C++

[c] Falsch

that is for Standard ML

[d] Nooooooooo

that is for Java

In "exam mode" where disable solutions (here via `\stopsolutions`)

Example 27

Input:

```
\stopsolutions
\begin{sproblem}[title=Functions,name=functions1]
  What is the keyword to introduce a function definition in python?
  \begin{mcb}
    \mcc[T]{def}
    \mcc[F,feedback=that is for C and C++]{function}
    \mcc[F,feedback=that is for Standard ML]{fun}
    \mcc[F,Ftext=Nooooooooo,feedback=that is for Java]{public static void}
  \end{mcb}
\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 28

Input:

```
\stopsolutions
\begin{sproblem}[id=elefants.fillin,title=Fitting Elefants]
How many Elefants can you fit into a Volkswagen beetle? \fillinsol{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 29

Input:

```
\begin{sproblem}[id=elefants.fillin,title=Fitting Elefants]
How many Elefants can you fit into a Volkswagen beetle? \fillinsol{4}
\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. [17]

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

[17] 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.

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` `\testsmallspace` `\testsmallspace` `\testsmallspace` `\testnewpage` `\testemptypage`

`\testspace` takes an argument that expands to a dimension, and leaves vertical space accordingly. Specific instances exist: `\testsmallspace`, `\testsmallspace`, `\testsmallspace` give small (1cm), medium (2cm), and big (3cm) vertical space.

`\testnewpage` makes a new page in `test` mode, and `\testemptypage` generates an empty page with the cautionary message that this page was intentionally left empty.

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 LocalWords: Functions,name F,feedback Nooooooooo,feedback 2,title includeproblem LocalWords: 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

`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.

10.3.3 Assignments

`assignment` (*env.*)
`number`
`title`
`type`
`given`
`due`

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

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.*)
`duration`
`min`
`reqpts`

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

```
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)

2023-10-13

You have one hour (sharp) for the test;
Write the solutions to the sheet.
The estimated time for solving this exam is 23 minutes, leaving you 37 minutes for revising your exam.
You can reach 23 points if you solve all problems. You will only need 27 points for a perfect score, i.e. -4 points are bonus points.

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

	To be used for grading, do not write here								
prob.	4.1	1.1	1.2	2.1	2.2	2.3	2.4	2.5	2.6
total	0	0	0	10	0	0	0	0	0
reached									

good luck

[18]

Local Variables: mode: latex TeX-master: `../stex-manual` End:
LocalWords: hwexam solutions,notes,hints,gnotes,pts,min gnotes testemptypage reqpts 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`.

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}
```

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

```
\usetikzpackage{...}
\begin{document}
  \begin{tikzpicture}
     ...
  \end{tikzpicture}
\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 throwaway 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 $\leq$ 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 correponding 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 30

Assume we have a symbol and notation:

```
\symdecl{somesymbol}[args=iai]
\notation{somesymbol}[prec=10;20x30x40,variant=foo]
{First: #1; Second: #2; Third: #3; End}
{(#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:

```
\STEXInternalTermMathOMAiii{foo}{10}{
  First: \STEXInternalTermMathArgiii{i1}{20}{#1};
  Second: \STEXInternalTermMathAssocArgiiiii{2}{30}{#2}{
    (\STEXInternalTermMathArgiii{i1}{20}{##1} --
```

```
    \STEXInternalAssocArgMarkerI split
    \STEXInternalAssocArgMarkerII --
    \STEXInternalTermMathArgiii{i3}{40}{##3})
 }{a};
 Third: \STEXInternalTermMathArgiii {i3}{40}{#3}; End
}\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 STEX 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_set:nn

\stex_keys_define:nnnn {⟨*id*⟩}{⟨*setup code*⟩}
{⟨*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 31

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 32

`\importmodule` is defined something like the following:

```
\stex_new_stylable_cmd:nnnn{importmodule}{O{} m}{
  ...
  \def\thismoduleuri{...}
  \def\thismodulename{...}
  \stex_style_apply:
  ...
}{}
```

A user can then customize the output generated by `\importmodule` (by default none) via `\stexstyleimportmodule{...\thismodulename...}`.

Example 33

`smodule` does something like

```
\stex_new_stylable_env:nnnnnnn{module}{O{} m}{
  ...
  \def\thismoduleuri{...}
  \def\thismodulename{...}
  \stex_style_apply:
  ...
}{
  ...
```

```
  \stex_style_apply:
  ...
}{}{}{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

\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:nnTF ⋆

\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:nnTF ⋆

\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

MMT 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. RUSTEX 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 34

```
\stex_pseudogroup:nn{
  something changing \foo
  something changing \num
}{
  \stex_pseudogroup_restore:N\foo
  \int_set:Nn \num {\int_use:N \num}
}
```

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

```
⟨*cls⟩
%%%%%%%%%%%%   stex.dtx   %%%%%%%%%%%%

\RequirePackage{expl3,l3keys2e}
\ProvidesExplClass{stex}{2023/10/13}{3.4.0}{sTeX document class}
\IfFileExists{stex-expl-compat.sty}{
  \usepackage{stex-expl-compat}
}{}

\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
\ProcessOptions

\RequirePackage{stex}

\LoadClass{article}
⟨/cls⟩
```

Setup code for the package

```
⟨*package⟩
\RequirePackage{expl3,l3keys2e,ltxcmds}
\ProvidesExplPackage{stex}{2023/10/13}{3.4.0}{sTeX package}
\IfFileExists{stex-expl-compat.sty}{
  \usepackage{stex-expl-compat}
}{}
\RequirePackage{stex-logo} % externalized for backwards-compatibility reasons
\RequirePackage{standalone}

\message{^^J*~This~is~sTeX~version~3.4.0~*^^J}
```

Package options:

```
\keys_define:nn { stex / package } {
  debug      .str_set_x:N  = \c_stex_debug_clist ,
  lang       .clist_set:N  = \c_stex_languages_clist ,
  mathhub    .tl_set_x:N   = \mathhub ,
```

```
  usesms      .bool_set:N   = \c_stex_persist_mode_bool ,
  writesms    .bool_set:N   = \c_stex_persist_write_mode_bool ,
  checkterms .bool_set:N    = \c_stex_check_terms_bool ,
  image       .bool_set:N   = \c_tikzinput_image_bool,
  nofrontmatter .bool_set:N = \c_stex_no_frontmatter_bool,
  unknown     .code:n       = {}
}
\exp_args:NNo \clist_set:Nn \c_stex_debug_clist \c_stex_debug_clist
\ProcessKeysOptions { stex / package }
```

Error messages:

```
\input{stex-en.ldf}
```

13.2 Utilities

```
\cs_set_eq:NN \stex_undefined: \undefined
```

13.2.1 Calling kpsewhich and Environment Variables

```
⟨@@=stex_envs⟩
```

\stex_kpsewhich:Nn

```
\cs_new_protected:Nn \stex_kpsewhich:Nn {\group_begin:
  \catcode`\ =12
  \sys_get_shell:nnN { kpsewhich ~ #2 } { } \l_tmpa_tl
  \tl_gset_eq:NN \l_tmpa_tl \l_tmpa_tl
  \group_end:
  \exp_args:NNo\str_set:Nn #1 {\l_tmpa_tl}
  \tl_trim_spaces:N #1
}
```

(*End of definition for* `\stex_kpsewhich:Nn`*. This function is documented on page 142.*)

\stex_get_env:Nn

```
\sys_if_platform_windows:TF{
  \cs_new_protected:Nn \stex_get_env:Nn {\group_begin:
    \escapechar=-1\catcode`\\=12
    \exp_args:NNe \stex_kpsewhich:Nn #1 {-expand-var~\c_percent_str#2\c_percent_str}
    \exp_args:NNx\use:nn\group_end:{
      \str_set:Nn \exp_not:N #1 { #1 }
    }
  }
}{
  \cs_new_protected:Nn \stex_get_env:Nn {
    \stex_kpsewhich:Nn #1 {-var-value~#2}
  }
}
```

(*End of definition for* `\stex_get_env:Nn`*. This function is documented on page 142.*)

13.2.2 Logging

64 ⟨@@=stex_debug⟩

\stex_debug:nn

```
65 \cs_new_protected:Nn \stex_debug:nn {
66   \exp_args:NNo \clist_if_in:NnTF \c_stex_debug_clist { \tl_to_str:n{all} }{
67     \__stex_debug_:nn{#1}{#2}
68   }{
69     \exp_args:NNo \clist_if_in:NnT \c_stex_debug_clist { \tl_to_str:n{#1} }{
70       \__stex_debug_:nn{#1}{#2}
71     }
72   }
73 }
74
75 \cs_new_protected:Nn \__stex_debug_:nn {
76   \msg_set:nnn{stex}{debug / #1}{
77     \\Debug~#1:~#2\\
78   }
79   \msg_none:nn{stex}{debug / #1}
80 }
```

(*End of definition for* `\stex_debug:nn`. *This function is documented on page 120.*)

\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:

```
81 \cs_new_protected:Nn \stex_fatal_error:n {
82   \msg_error:nn{stex}{#1}\input{Fatal~Error!!}
83 }
84 \cs_new_protected:Nn \stex_fatal_error:nnn {
85   \msg_error:nnn{stex}{#1}{#2}{#3}\input{Fatal~Error!!}
86 }
87 \cs_generate_variant:Nn \stex_fatal_error:nnn {nxx}
```

(*End of definition for* `\stex_fatal_error:n` *and* `\stex_fatal_error:nnn`. *These functions are documented on page 142.*)

We check an environment variable for debugging and set things up:

```
88 \stex_get_env:Nn\__stex_debug_env_str{STEX_DEBUG}
89 \str_if_empty:NTF\__stex_debug_env_str {
90   \clist_set_eq:NN \l__stex_debug_cl \c_stex_debug_clist
91 }{
92   \clist_set:No \l__stex_debug_cl {\__stex_debug_env_str}
93 }
94 \clist_clear:N \c_stex_debug_clist
95 \clist_map_inline:Nn \l__stex_debug_cl {
96   \exp_args:NNo \clist_put_right:Nn \c_stex_debug_clist
97   { \tl_to_str:n{#1} }
98 }
99
100 \exp_args:NNo \clist_if_in:NnTF \c_stex_debug_clist {\tl_to_str:n{all}} {
101    \msg_redirect_module:nnn{ stex }{ none }{ warning }
102    \stex_debug:nn{all}{Logging~everything!}
103 }{
104   \clist_map_inline:Nn \c_stex_debug_clist {
105     \msg_redirect_name:nnn{ stex }{ debug / #1 }{ warning }
106     \stex_debug:nn{#1}{Logging~#1}
```

```
  }
}
```

13.2.3 Languages

```
⟨@@=stex_lang⟩
```

\c_stex_languages_prop
\c_stex_language_abbrevs_prop

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

```
\exp_args:NNx \prop_const_from_keyval:Nn \c_stex_languages_prop { \tl_to_str:n {
  en = english ,
  de = ngerman ,
  ar = arabic ,
  bg = bulgarian ,
  ru = russian ,
  fi = finnish ,
  ro = romanian ,
  tr = turkish ,
  fr = french
}}

\exp_args:NNx \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop { \tl_to_str:n {
  english   = en ,
  ngerman   = de ,
  arabic    = ar ,
  bulgarian = bg ,
  russian   = ru ,
  finnish   = fi ,
  romanian  = ro ,
  turkish   = tr ,
  french    = fr
}}
% todo: chinese simplified (zhs)
%       chinese traditional (zht)
```

(*End of definition for* `\c_stex_languages_prop` *and* `\c_stex_language_abbrevs_prop`*. These variables are documented on page 139.*)

\l_stex_current_language_str

```
\str_new:N \l_stex_current_language_str
```

(*End of definition for* `\l_stex_current_language_str`*. This variable is documented on page 139.*)

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

\stex_set_language:n
\stex_set_language:x
\stex_set_language:o

```
\cs_new_protected:Nn \stex_set_language:n {
  \str_set:Nn \l_stex_current_language_str { #1 }
  \prop_if_in:NnTF \c_stex_languages_prop {#1} {
    \tl_set_rescan:Nnx \l__stex_lang_lang_str {}{\prop_item:Nn \c_stex_languages_prop {#1}}
    \cs_if_eq:NNTF\@onlypreamble\@notprerr{
      \ltx@ifpackageloaded{babel}{
        \exp_args:No\selectlanguage\l__stex_lang_lang_str
      }{}
    }{
      \ltx@ifpackageloaded{babel}{}{
        \str_if_eq:nnTF {#1} {tr} {
```

```
            \RequirePackage[turkish,shorthands=:!]{babel}
          }{
            \RequirePackage[\l__stex_lang_lang_str]{babel}
          }
        }
      }
    }{
      \msg_error:nnx{stex}{error/unknownlanguage}{#1}
    }
  }
  \cs_generate_variant:Nn \stex_set_language:n {x,o}
```

(*End of definition for* `\stex_set_language:n`. *This function is documented on page 140.*)

`\stex_language_from_file:`

```
  \cs_new_protected:Nn \stex_language_from_file: {
    \seq_get_right:NN \g_stex_current_file \l_tmpa_str
    \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
    \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % = ".tex/.dtx/.sty"
    \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
      \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
        \exp_args:No \str_if_eq:nnF \l_tmpa_str {ltx} {
          \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
        }
      }
    }
    \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
    \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
      \seq_pop_right:NN \l_tmpa_seq \l__stex_lang_str
      \str_if_eq:NNF \l__stex_lang_str \l_stex_current_language_str {
        \exp_args:NNo \prop_if_in:NnT \c_stex_languages_prop \l__stex_lang_str {
          \stex_set_language:o \l__stex_lang_str
        }
      }
      \stex_debug:nn{lang} {Language~\l_stex_current_language_str~
        inferred~from~file~name}
    }
  }
```

(*End of definition for* `\stex_language_from_file:`. *This function is documented on page 140.*)

Loading babel:

```
  \clist_if_empty:NF \c_stex_languages_clist {
    \bool_set_false:N \l__stex_lang_turkish_bool
    \seq_clear:N \l_tmpa_seq
    \clist_map_inline:Nn \c_stex_languages_clist {
      \str_set:Nx \l_tmpa_str {#1}
      \str_if_eq:nnT {#1}{tr}{
        \bool_set_true:N \l__stex_lang_turkish_bool
      }
      \prop_get:NoNTF \c_stex_languages_prop \l_tmpa_str \l_tmpa_str {
        \tl_set_rescan:Nno \l_tmpa_str {} \l_tmpa_str
        \seq_put_right:No \l_tmpa_seq \l_tmpa_str
      } {
        \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
```

```
    }
  }
  \stex_debug:nn{lang} {Languages:~\seq_use:Nn \l_tmpa_seq {,~} }
  \bool_if:NTF \l__stex_lang_turkish_bool {
    \exp_args:NNe \use:nn \RequirePackage
      {[main=\seq_use:Nn \l_tmpa_seq, ,shorthands=:!]}{babel}
  }{
    \exp_args:NNe \use:nn \RequirePackage
      {[main=\seq_use:Nn \l_tmpa_seq, ]}{babel}
  }
}
```

13.2.4 Group-like Behaviours

```
⟨@@=stex_groups⟩
```

\stex_pseudogroup:nn
\stex_pseudogroup_restore:N

```
\cs_new_protected:Npn \stex_pseudogroup:nn {
  \exp_args:Nne \use:nn
}
\cs_new:Nn \stex_pseudogroup_restore:N {
  \tl_if_exist:NTF #1 {
    \tl_set:Nn \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
  }{
    \cs_undefine:N \exp_not:N #1
  }
}
```

(*End of definition for* `\stex_pseudogroup:nn` *and* `\stex_pseudogroup_restore:N`*. These functions are documented on page 142.*)

\stex_pseudogroup_with:nn

```
\cs_new_protected:Nn \stex_pseudogroup_with:nn {
  \tl_map_inline:nn{#1}{
    \cs_set_eq:cN{__stex_groups_\tl_to_str:n{##1}}##1
  }
  #2
  \tl_map_inline:nn{#1}{
    \cs_set_eq:Nc##1{__stex_groups_\tl_to_str:n{##1}}
    \cs_undefine:c{__stex_groups_\tl_to_str:n{##1}}
  }
}
```

(*End of definition for* `\stex_pseudogroup_with:nn`*. This function is documented on page 142.*)

\stex_metagroup_new:n
\stex_metagroup_new:o

List of all currently existing metagroup identifiers

```
\seq_new:N \l__stex_groups_ids_seq
```

start a new metagroup at the current group level with id #1

```
\cs_new_protected:Nn \stex_metagroup_new:n {
    \str_set:cx{l__stex_groups_#1_int} {\int_use:N\currentgrouplevel}
    \seq_put_right:Nn \l__stex_groups_ids_seq {#1}
}
\cs_generate_variant:Nn \stex_metagroup_new:n {o}
```

(*End of definition for* `\stex_metagroup_new:n`*. This function is documented on page 143.*)

\stex_metagroup_do_in:nn
\stex_metagroup_do_in:nx

```
\prg_new_conditional:Nnn \__stex_groups_exists:n {TF} {
    \str_if_exist:cTF{l__stex_groups_#1_int}
        \prg_return_true: \prg_return_false:
}

\cs_new_protected:Nn \stex_metagroup_do_in:nn {
    \__stex_groups_exists:nTF{#1}{
        \__stex_groups_do_in:nn{#1}{#2}
    }{
      \msg_error:nnn{stex}{error/metagroup/missing}{#1}
    }
}
\cs_generate_variant:Nn \stex_metagroup_do_in:nn {nx}

\cs_new_protected:Nn \__stex_groups_do_in:nn {
  \exp_args:Nnx\stex_debug:nn{metagroup}{adding~to~\detokenize{#1}:^^J\tl_to_str:n{#2}}
  \tl_set:Nn\__stex_groups_tmp{#2}
  \exp_args:Nx \int_compare:nNnF {\use:c{l__stex_groups_#1_int}}
      = \currentgrouplevel {
      \tl_if_exist:cTF{g__stex_groups_#1_\the\currentgrouplevel _content}{
        \exp_args:Nno \tl_gput_right:cn{g__stex_groups_#1_\the\currentgrouplevel _content}
      }{
        \exp_args:Nno \tl_gset:cn{g__stex_groups_#1_\the\currentgrouplevel _content}
      }\__stex_groups_tmp
      \bool_if_exist:cF {l__stex_groups_\the\currentgrouplevel _bool} {
          \group_insert_after:N \__stex_groups_do:
          \bool_set_true:c {l__stex_groups_\the\currentgrouplevel _bool}
      }
  }
  \__stex_groups_tmp
}

\cs_new_protected:Nn \__stex_groups_do: {
  \seq_map_inline:Nn \l__stex_groups_ids_seq {
    \tl_if_exist:cT{g__stex_groups_##1_\int_eval:n{\currentgrouplevel+1}_content}{
      \exp_args:NNno \exp_args:Nno \__stex_groups_do_in:nn{##1}{
        \csname g__stex_groups_##1_\int_eval:n{\currentgrouplevel+1}_content\endcsname
      }
      \cs_undefine:c{g__stex_groups_##1_\int_eval:n{\currentgrouplevel+1}_content}
    }
    \bool_if_exist:cF {l__stex_groups_\int_eval:n\currentgrouplevel _bool} {
      \group_insert_after:N \__stex_groups_do:
      \bool_set_true:c {l__stex_groups_\int_eval:n\currentgrouplevel _bool}
    }
  }
}
```

(*End of definition for* `\stex_metagroup_do_in:nn`. *This function is documented on page* 143.)

13.2.5 HTML Annotations

⟨@@=stex_annotate⟩

\stex_if_do_html:TF Whether to (locally) produce HTML output

```
\bool_new:N \_stex_html_do_output_bool
\bool_set_true:N \_stex_html_do_output_bool

\prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
  \bool_if:nTF \_stex_html_do_output_bool
    \prg_return_true: \prg_return_false:
}
```

(*End of definition for* `\stex_if_do_html:TF`. *This function is documented on page* 140.)

\stex_suppress_html:n Temporarily disable HTML output

```
\cs_new_protected:Nn \stex_suppress_html:n {
  \stex_pseudogroup:nn{
    \bool_set_false:N \_stex_html_do_output_bool
    #1
  }{
    \stex_if_do_html:T {
      \bool_set_true:N \_stex_html_do_output_bool
    }
  }
}
```

(*End of definition for* `\stex_suppress_html:n`. *This function is documented on page* 140.)

We determine the backend:

\stex_if_html_backend_p:
\stex_if_html_backend:TF
\ifstexhtml
\stex@backend

```
\ifcsname if@rustex\endcsname\else
  \expandafter\newif\csname if@rustex\endcsname
  \@rustexfalse
\fi

\stex_get_env:Nn\__stex_annotate_env_str{STEX_FORCE_PDF}
\exp_args:No \str_if_eq:nnTF \__stex_annotate_env_str {true} {
  \def\stex@backend{pdflatex}
}{
  \tl_if_exist:NF\stex@backend{
    \if@rustex
      \def\stex@backend{rustex}
    \else
      \cs_if_exist:NTF\HCode{
        \def\stex@backend{tex4ht}
      }{
        \def\stex@backend{pdflatex}
      }
    \fi
  }
}

\input{stex-backend-\stex@backend.cfg}

\newif\ifstexhtml
\stex_if_html_backend:TF {
  \stexhtmltrue
```

```
  \bool_set_true:N \_stex_html_do_output_bool
}{
  \stexhtmlfalse
  \bool_set_false:N \_stex_html_do_output_bool
}
```

(*End of definition for* `\stex_if_html_backend:TF`, `\ifstexhtml`, *and* `\stex@backend`. *These functions are documented on page 140.*)

_stex_annotate_force_break:n

```
\stex_if_html_backend:TF {
  \cs_new_protected:Nn \_stex_annotate_force_break:n {
    \stex_annotate_invisible:n{~}
    #1
    \stex_annotate_invisible:n{~}
  }
}{
  \cs_new_protected:Nn \_stex_annotate_force_break:n { #1 }
}
```

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

\mmlintent
\mmlarg

```
\stex_if_html_backend:TF {
  \cs_new_protected:Npn \mmlintent #1 #2 {
    \stex_annotate:nn{mml:intent={#1}}{#2}
  }
  \cs_new_protected:Npn \mmlarg #1 #2 {
    \stex_annotate:nn{mml:arg={#1}}{#2}
  }
}{
  \cs_new_protected:Npn \mmlintent #1 #2 { #2 }
  \cs_new_protected:Npn \mmlarg #1 #2 { #2 }
}
```

(*End of definition for* `\mmlintent` *and* `\mmlarg`. *These functions are documented on page* **??**.)

13.2.6 Auxiliary Methods

⟨@@=stex_aux⟩

\stex_deactivate_macro:Nn
\stex_reactivate_macro:N

```
\cs_new_protected:Nn \stex_deactivate_macro:Nn {
  \tl_set_eq:cN{\tl_to_str:n{#1}~~~orig}#1
  \cs_set_protected:Npn#1{
    \exp_args:Nnno\msg_error:nnnn{stex}{error/deactivated-macro}{#1}{#2}
  }
}
\cs_new_protected:Nn \stex_reactivate_macro:N {
  \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1}~~~orig\endcsname
}
```

(*End of definition for* `\stex_deactivate_macro:Nn` *and* `\stex_reactivate_macro:N`. *These functions are documented on page 142.*)

\stex_ignore_spaces_and_pars:

```
\protected\def\stex_ignore_spaces_and_pars:{
  \begingroup\catcode13=10\relax
  \@ifnextchar\par{
    \endgroup\expandafter\stex_ignore_spaces_and_pars:\@gobble
  }{
    \endgroup
  }
}
```

(*End of definition for* `\stex_ignore_spaces_and_pars:`. *This function is documented on page* 142.)

\stex_keys_define:nnnn

```
\cs_new_nopar:Nn \stex_keys_define:nnnn {
  \tl_gset:cn {__stex_aux_keys_#1_pre_tl}{#2}
  \tl_gset:cn {__stex_aux_keys_#1_def_tl}{#3}
  \tl_if_empty:nF{#4}{
    \clist_map_inline:nn{#4}{
      \tl_set_eq:Nc \l__stex_aux_tl {__stex_aux_keys_##1_pre_tl}
      \tl_gput_left:co{__stex_aux_keys_#1_pre_tl} \l__stex_aux_tl
      \tl_set_eq:Nc \l__stex_aux_tl {__stex_aux_keys_##1_def_tl}
      \tl_gput_left:cn{__stex_aux_keys_#1_def_tl} ,
      \tl_gput_left:co{__stex_aux_keys_#1_def_tl} \l__stex_aux_tl
    }
  }
  \tl_set_eq:Nc \l__stex_aux_tl {__stex_aux_keys_#1_def_tl}
  \exp_args:Nno \keys_define:nn {stex / #1} {\l__stex_aux_tl}
}
```

(*End of definition for* `\stex_keys_define:nnnn`. *This function is documented on page* 132.)

\stex_keys_set:nn

```
\cs_new_nopar:Nn \stex_keys_set:nn {
  \use:c{__stex_aux_keys_#1_pre_tl}
  \keys_set:nn {stex / #1} { #2 }
}
```

(*End of definition for* `\stex_keys_set:nn`. *This function is documented on page* 132.)

Some ubiquitous key sets:

```
\stex_keys_define:nnnn{archive file}{
  \str_clear:N \l_stex_key_archive_str
  \str_clear:N \l_stex_key_file_str
}{
  archive .str_set_x:N = \l_stex_key_archive_str ,
  file    .str_set_x:N = \l_stex_key_file_str
}{}

\stex_keys_define:nnnn{id}{
  \str_clear:N \l_stex_key_id_str
}{
  id .str_set_x:N = \l_stex_key_id_str
}{}

\stex_keys_define:nnnn{title}{
```

```
  \tl_clear:N \l_stex_key_title_tl
}{
  title .tl_set:N = \l_stex_key_title_tl
}{}

\stex_keys_define:nnnn{style}{
  \clist_clear:N \l_stex_key_style_clist
}{
  style .clist_set:N = \l_stex_key_style_clist
}{}

\stex_keys_define:nnnn{deprecate}{
  \str_clear:N \l_stex_key_deprecate_str
}{
  deprecate     .str_set_x:N  = \l_stex_key_deprecate_str
}{}

\stex_keys_define:nnnn{uses}{}{
  uses .code:n = {
    \clist_map_inline:nn{#1}{
      \stex_str_if_starts_with:nnTF{##1}[{
        \__stex_aux_split_at_bracket:w ##1 \_stex_end:
      }{
        \usemodule{##1}
      }
    }
  }
}{}

\cs_new_protected:Npn \__stex_aux_split_at_bracket:w [ #1 ] #2 \_stex_end: {
  \usemodule[#1]{#2}
}
```

_stex_do_deprecation:n

```
\cs_new:Nn \_stex_do_deprecation:n {
  \str_if_empty:NF \l_stex_key_deprecate_str {
    \msg_warning:nnxx{stex}{warning/deprecated}{#1}{\l_stex_key_deprecate_str}
  }
}
```

(*End of definition for* `\_stex_do_deprecation:n`. *This function is documented on page* 120.)

_stex_do_id:

```
\cs_new_protected:Nn \_stex_do_id: {
  \stex_if_smsmode:F {
    \str_if_empty:NF \l_stex_key_id_str {
      \exp_args:No \stex_ref_new_doc_target:n \l_stex_key_id_str
    }
  }
}
```

(*End of definition for* `\_stex_do_id:`. *This function is documented on page* 132.)

\stex_new_stylable_env:nnnnnnn
\stex_new_stylable_cmd:nnnn
\stex_style_apply:

```
\cs_new_protected:Nn \stex_new_stylable_cmd:nnnn {
  \exp_after:wN \newcommand \cs:w stexstyle#1 \cs_end:[2][]{
    \__stex_aux_patch:nnn{#1}{##1}{##2}
  }
  \exp_after:wN \NewDocumentCommand\cs:w #1\cs_end:{#2}{
    \cs_set:Npn \stex_style_apply: {
      \__stex_aux_apply_patch:n{#1}
    }
    #3
  }
  \tl_set:cn {__stex_aux_style_#1:} { #4 }
}

\cs_new_protected:Nn \__stex_aux_apply_patch:n {
  \clist_if_empty:NTF \l_stex_key_style_clist {
    \tl_clear:N \thisstyle
    \use:c{__stex_aux_style_#1:}
  }{
    \clist_get:NN \l_stex_key_style_clist \thisstyle
    \tl_if_exist:cTF{__stex_aux_style_#1_\thisstyle :}{
      \use:c{__stex_aux_style_#1_\thisstyle :}
    }{
      \use:c{__stex_aux_style_#1:}
    }
  }
}

\cs_new_protected:Nn \__stex_aux_patch:nnn {
  \str_if_empty:nTF {#2}{
    \tl_set:cn{__stex_aux_style_#1:}{#3}
  }{
    \tl_set:cn{__stex_aux_style_#1_#2:}{#3}
  }
}

\cs_new_protected:Nn \stex_new_stylable_env:nnnnnnn {
  \exp_after:wN \newcommand \cs:w stexstyle#1 \cs_end:[3][]{
    \__stex_aux_patch:nnnn{#1}{##1}{##2}{##3}
  }
  \NewDocumentEnvironment{#7#1}{#2}{
    \cs_set:Npn \stex_style_apply: {
      \__stex_aux_apply_patch_begin:n{#1}
    }
    #3
  }{
    \cs_set:Npn \stex_style_apply: {
      \__stex_aux_apply_patch_end:n{#1}
    }
    #4
  }
  \tl_set:cn {__stex_aux_style_#1_start:} { #5 }
  \tl_set:cn {__stex_aux_style_#1_end:} { #6 }
}
```

```
\cs_new_protected:Nn \__stex_aux_patch:nnnn {
  \str_if_empty:nTF {#2}{
    \tl_set:cn{__stex_aux_style_#1_start:}{#3}
    \tl_set:cn{__stex_aux_style_#1_end:}{#4}
  }{
    \tl_set:cn{__stex_aux_style_#1_#2_start:}{#3}
    \tl_set:cn{__stex_aux_style_#1_#2_end:}{#4}
  }
}

\cs_new_protected:Nn \__stex_aux_apply_patch_begin:n {
  \clist_if_empty:NTF \l_stex_key_style_clist {
    \tl_clear:N \thisstyle
    \use:c{__stex_aux_style_#1_start:}
  }{
    \clist_get:NN \l_stex_key_style_clist \thisstyle
    \stex_debug:nn{styling}{dominant~style:~\thisstyle}
    \tl_if_exist:cTF{__stex_aux_style_#1_\thisstyle _start:}{
      \use:c{__stex_aux_style_#1_\thisstyle _start:}
    }{
      \use:c{__stex_aux_style_#1_start:}
    }
  }
}

\cs_new_protected:Nn \__stex_aux_apply_patch_end:n {
  \tl_if_empty:NTF \thisstyle {
    \use:c{__stex_aux_style_#1_end:}
  }{
    \tl_if_exist:cTF{__stex_aux_style_#1_\thisstyle _end:}{
      \use:c{__stex_aux_style_#1_\thisstyle _end:}
    }{
      \use:c{__stex_aux_style_#1_end:}
    }
  }
}
```

(*End of definition for* `\stex_new_stylable_env:nnnnnnn`, `\stex_new_stylable_cmd:nnnn`, *and* `\stex_style_apply:`. *These functions are documented on page* 133.)

`\stex_str_if_ends_with_p:nn`
`\stex_str_if_ends_with:nnTF`

```
\prg_new_conditional:Nnn \stex_str_if_ends_with:nn {p,T,F,TF} {
  \exp_args:Ne \str_if_eq:nnTF {
    \str_range:nnn{#1}{- \str_count:n{#2}}{-1}
  }{#2}\prg_return_true: \prg_return_false:
}
```

(*End of definition for* `\stex_str_if_ends_with:nnTF`. *This function is documented on page* 136.)

`\stex_str_if_starts_with_p:nn`
`\stex_str_if_starts_with:nnTF`

```
\prg_new_conditional:Nnn \stex_str_if_starts_with:nn {p,T,F,TF} {
  \exp_args:Ne \str_if_eq:nnTF {
    \str_range:nnn{#1}{1}{\str_count:n{#2}}
  }{#2}\prg_return_true: \prg_return_false:
```

```
}
```

(*End of definition for* `\stex_str_if_starts_with:nnTF`. *This function is documented on page* 136.)

`\stex_macro_body:N`

```
\cs_new:Npn \__stex_aux_start:#1\__stex_aux_end: {\exp_not:n{#1}}
\cs_new_protected:Nn \__stex_aux_end: {}
\cs_new:Nn \stex_macro_body:N {
  \exp_args:Nne\use:nn{\exp_after:wN \__stex_aux_start: #1}{
    \__stex_aux_args:e {\cs_parameter_spec:N #1}\__stex_aux_end:
  }
}

\cs_new:Nn \__stex_aux_args:n {
  \tl_if_empty:nF{#1}{
      {##\exp_args:Ne \tl_head:n {\tl_tail:n {#1}}}
      \__stex_aux_args:e {\exp_args:Ne\tl_tail:n{\tl_tail:n{#1}}}
  }
}
\cs_generate_variant:Nn \__stex_aux_args:n {e}
```

(*End of definition for* `\stex_macro_body:N`. *This function is documented on page* 141.)

`\stex_macro_definition:N`

```
\cs_new:Nn \stex_macro_definition:N {
  \__stex_aux_prefix:e {\cs_prefix_spec:N #1}
  \def\exp_not:N #1
  \__stex_aux_params:e {\cs_parameter_spec:N #1}
  {
    \stex_macro_body:N #1
  }
}

\cs_new:Nn \__stex_aux_prefix:n {
  \tl_if_empty:nF{#1}{
    \str_if_eq:eeTF {
      \tl_range:nnn{#1}{1}{10}~
    }{\tl_to_str:n{\protected}}{
      \protected
      \__stex_aux_prefix_long:e {
          \str_range:nnn{#1}{11}{-1}
      }
    }{
      \__stex_aux_prefix_long:n {#1}
    }
  }
}
\cs_generate_variant:Nn \__stex_aux_prefix:n {e}

\cs_new:Nn \__stex_aux_prefix_long:n {
  \tl_if_empty:nF{#1}{
    \str_if_eq:eeT {
      \tl_range:nnn{#1}{1}{10}~
    }{\tl_to_str:n{\long}}{\long}
  }
```

```
}
\cs_generate_variant:Nn \__stex_aux_prefix_long:n {e}

\cs_new:Nn \__stex_aux_params:n {
  \tl_if_empty:nF{#1}{
    \exp_args:NNe \str_if_eq:VnTF \c_hash_str {\tl_head:n{#1}}{
      ####
    }{
      \tl_head:n{#1}
    }
    \__stex_aux_params:e {\tl_tail:n{#1}}
  }
}
\cs_generate_variant:Nn \__stex_aux_params:n {e}
```

(*End of definition for* `\stex_macro_definition:N`. *This function is documented on page* 141.)

13.2.7 Persistence

⟨@@=stex_persist⟩

We check the environment variables:

```
\stex_get_env:Nn\__stex_persist_env_str{STEX_USESMS}
\str_if_empty:NF\__stex_persist_env_str{
  \exp_args:No \str_if_eq:nnF \__stex_persist_env_str{false}{
    \bool_set_true:N \c_stex_persist_mode_bool
  }
}
\stex_get_env:Nn\__stex_persist_env_str{STEX_WRITESMS}
\str_if_empty:NF\__stex_persist_env_str{
  \exp_args:No \str_if_eq:nnF \__stex_persist_env_str{false}{
    \bool_set_true:N \c_stex_persist_write_mode_bool
  }
}
```

`\stex_persist:n`
`\stex_persist:e`

```
\iow_new:N \c__stex_persist_sms_iow

\bool_if:NTF \c_stex_persist_write_mode_bool {
  \stex_if_html_backend:TF{
    \cs_new:Npn \stex_persist:n #1 {}
    \cs_new:Npn \stex_persist:e #1 {}
  }{
    \cs_new_protected:Nn \stex_persist:n {
      \iow_now:Nn \c__stex_persist_sms_iow {#1}
    }
    \cs_generate_variant:Nn \stex_persist:n {e}
  }
}{
  \cs_new:Npn \stex_persist:n #1 {}
  \cs_new:Npn \stex_persist:e #1 {}
}
```

(*End of definition for* `\stex_persist:n`. *This function is documented on page* 141.)

Is called at the end of the `.sty`-file:

```
633
634 \cs_new_protected:Nn \__stex_persist_load_file:n {
635   \file_if_exist:nT{#1}{
636     \group_begin:
637     \cs_set:Npn \stex_persist:n ##1 {}
638     \cs_set:Npn \stex_persist:e ##1 {}
639       \stex_debug:nn{persist}{restoring~from~sms~file}
640       \catcode`\ =10\relax
641       \cs:w @ @ input \cs_end:#1\relax
642     \group_end:
643   }
644 }
645
646 \cs_new_protected:Nn \__stex_persist_write_only: {
647   \iow_open:Nn \c__stex_persist_sms_iow {\jobname.sms}
648   \AtEndDocument{ \iow_close:N \c__stex_persist_sms_iow }
649 }
650
651 \cs_new_protected:Nn \__stex_persist_read_and_write: {
652   \file_if_exist:nTF{\jobname.sms}{
653     \ior_open:Nn \g_tmpa_ior {\jobname.sms}
654     \iow_open:Nn \g_tmpa_iow {\jobname.sms2}
655     \ior_str_map_inline:Nn \g_tmpa_ior {
656       \iow_now:Nn \g_tmpa_iow {##1}
657     }
658     \iow_close:N \g_tmpa_iow
659     \ior_close:N \g_tmpa_ior
660     \__stex_persist_write_only:
661     \ior_open:Nn \g_tmpa_ior {\jobname.sms2}
662     \ior_str_map_inline:Nn \g_tmpa_ior {
663       \iow_now:Nn \c__stex_persist_sms_iow {##1}
664     }
665     \ior_close:N \g_tmpa_ior
666     \__stex_persist_load_file:n{\jobname.sms2}
667   }\__stex_persist_write_only:
668 }
669
670 \cs_new_protected:Nn \_stex_persist_read_now: {
671   \bool_if:NTF \c_stex_persist_mode_bool {
672     \bool_if:NTF \c_stex_persist_write_mode_bool
673       \__stex_persist_read_and_write:
674       {
675         \__stex_persist_load_file:n{\jobname.sms}
676       }
677   }{
678     \bool_if:NT \c_stex_persist_write_mode_bool \__stex_persist_write_only:
679   }
680 }
```

13.2.8 Files, Paths and URIs

681 ⟨@@=stex_path⟩

\stex_file_set:Nn
\stex_file_set:No
\stex_file_set:Nx

```
\cs_new_protected:Nn \stex_file_set:Nn {
  \str_if_empty:nTF {#2} { \seq_clear:N #1 }{
    \exp_args:NNno \seq_set_split:Nnn #1 / { \tl_to_str:n{#2} }
  }
}
\cs_generate_variant:Nn \stex_file_set:Nn {No, Nx}
```

(*End of definition for* `\stex_file_set:Nn`. *This function is documented on page 137.*)

`\stex_file_resolve:Nn`
`\stex_file_resolve:No`
`\stex_file_resolve:Nx`

```
\sys_if_platform_windows:TF{
  \cs_new_protected:Npn \__stex_path_win_take:w #1#2#3 \__stex_path_: {
    \str_set:Nn \l__stex_path_win_drive {#1#2}
    \str_set:Nn \l__stex_path_str{#3}
  }
  \cs_new_protected:Nn \stex_file_resolve:Nn {
    \str_set:Nn \l__stex_path_str {#2}
    \str_clear:N \l__stex_path_win_drive
    \exp_args:NNo \str_replace_all:Nnn \l__stex_path_str \c_backslash_str /
    \exp_args:Nx \str_if_eq:nnT {\str_item:Nn \l__stex_path_str 2} : {
      \exp_after:wN \__stex_path_win_take:w \l__stex_path_str \__stex_path_:
    }
    \stex_file_set:No #1 \l__stex_path_str
    \__stex_path_canonicalize:N #1
    \str_if_empty:NF \l__stex_path_win_drive {
      \seq_pop_left:NN #1 \l__stex_path_str
      \seq_put_left:No #1 \l__stex_path_win_drive
    }
    %\stex_debug:nn{files}{Set~\tl_to_str:n{#1}~to~\stex_file_use:N #1}
  }
}{
  \cs_new_protected:Nn \stex_file_resolve:Nn {
    \str_set:Nn \l__stex_path_str {#2}
    \stex_file_set:No #1 \l__stex_path_str
    \__stex_path_canonicalize:N #1
   % \stex_debug:nn{files}{Set~\tl_to_str:n{#1}~to~\stex_file_use:N #1}
  }
}
\cs_generate_variant:Nn \stex_file_resolve:Nn {No, Nx}

\cs_new_protected:Nn \__stex_path_canonicalize:N {
  \seq_if_empty:NF #1 {
    \seq_pop:NN #1 \l__stex_path_str
    \seq_clear:N \l__stex_path_seq
    \str_if_empty:NTF \l__stex_path_str {
      \seq_map_function:NN #1 \__stex_path_dodots:n
      \seq_put_left:Nn \l__stex_path_seq {}
    }{
      \seq_push:No #1 \l__stex_path_str
      \seq_map_function:NN #1 \__stex_path_dodots:n
    }
    \seq_set_eq:NN #1 \l__stex_path_seq
  }
}
```

```
\cs_new_protected:Nn \__stex_path_dodots:n {
  \str_if_empty:nF{#1}{
    \str_if_eq:nnF {#1} {.} {
      \str_if_eq:nnTF {#1} {..} {
        \seq_if_empty:NF \l__stex_path_seq {
          \seq_pop_right:NN \l__stex_path_seq \l__stex_path_str
        }
      }{
        \seq_put_right:Nn \l__stex_path_seq {#1}
      }
    }
  }
}
```

(*End of definition for* `\stex_file_resolve:Nn`. *This function is documented on page* 137.)

`\stex_if_file_absolute_p:N`
`\stex_if_file_absolute:NTF`

```
\sys_if_platform_windows:TF {
  \prg_new_conditional:Nnn \stex_if_file_absolute:N {p, T, F, TF} {
    \seq_if_empty:NTF #1 \prg_return_false: {
      \tl_set:Nx \l__stex_path_maybewin_str {\seq_item:Nn #1 1}
      \exp_args:No \tl_if_empty:nTF \l__stex_path_maybewin_str \prg_return_true: {
        \exp_args:Nx \str_if_eq:nnTF {\str_item:Nn \l__stex_path_maybewin_str 2} :
          \prg_return_true: \prg_return_false:
      }
    }
  }
}{
  \prg_new_conditional:Nnn \stex_if_file_absolute:N {p, T, F, TF} {
    \seq_if_empty:NTF #1 \prg_return_false: {
      \exp_args:Nx \tl_if_empty:nTF {\seq_item:Nn #1 1}
        \prg_return_true: \prg_return_false:
    }
  }
}
```

(*End of definition for* `\stex_if_file_absolute:NTF`. *This function is documented on page* 137.)

`\stex_file_use:N`

```
\cs_new:Nn \stex_file_use:N {
  \seq_use:Nn #1 /
}
```

(*End of definition for* `\stex_file_use:N`. *This function is documented on page* 137.)

`stex_if_file_starts_with:NNTF`

```
\prg_new_protected_conditional:Nnn \stex_if_file_starts_with:NN {T,F,TF} {
  \seq_set_eq:NN \l__stex_path_a_seq #1
  \seq_set_eq:NN \l__stex_path_b_seq #2
  \tl_clear:N \l__stex_path_return_tl
  \bool_while_do:nn{
    \bool_not_p:n{
      \bool_lazy_any_p:n{
```

```
        {\seq_if_empty_p:N \l__stex_path_a_seq}
        {\seq_if_empty_p:N \l__stex_path_b_seq}
        {\bool_not_p:n{\tl_if_empty_p:N \l__stex_path_return_tl}}
      }
    }
  }{
    \seq_pop_left:NN \l__stex_path_a_seq \l__stex_path_a_tl
    \seq_pop_left:NN \l__stex_path_b_seq \l__stex_path_b_tl
    \str_if_eq:NNF \l__stex_path_a_tl \l__stex_path_b_tl {
      \tl_set:Nn \l__stex_path_return_tl {\prg_return_false:}
    }
  }
  \tl_if_empty:NTF \l__stex_path_return_tl {
    \seq_if_empty:NTF \l__stex_path_b_seq \prg_return_true: \prg_return_false:
  } \l__stex_path_return_tl
}
```

(*End of definition for* `\stex_if_file_starts_with:NNTF`. *This function is documented on page* 137.)

`\stex_file_split_off_ext:NN`
`\stex_file_split_off_lang:NN`

```
\cs_new_protected:Nn \stex_file_split_off_ext:NN {
  \seq_set_eq:NN #1 #2
  \seq_pop_right:NN #1 \l__stex_path_str
  \seq_set_split:NnV \l__stex_path_seq . \l__stex_path_str
  \seq_pop_right:NN \l__stex_path_seq \l__stex_path_str
  \seq_put_right:Nx #1 {\seq_use:Nn \l__stex_path_seq .}
}
\cs_new_protected:Nn \stex_file_split_off_lang:NN {
  \seq_set_eq:NN #1 #2
  \seq_pop_right:NN #1 \l__stex_path_str
  \seq_set_split:NnV \l__stex_path_seq . \l__stex_path_str
  \seq_pop_right:NN \l__stex_path_seq \l__stex_path_str

  \seq_pop_right:NN \l__stex_path_seq \l__stex_path_str
  \exp_args:NNo \prop_if_in:NnF \c_stex_languages_prop \l__stex_path_str {
    \seq_put_right:No \l__stex_path_seq \l__stex_path_str
  }

  \seq_put_right:Nx #1 {\seq_use:Nn \l__stex_path_seq .}
}
```

(*End of definition for* `\stex_file_split_off_ext:NN` *and* `\stex_file_split_off_lang:NN`. *These functions are documented on page* 137.)

URIs:

`\stex_map_uri:Nnnnn`

```
\cs_set_protected:Nn \__stex_path_auth:n {
  \msg_error:nnx{stex}{error/misused-uri}{\tl_to_str:n{#1}}
}
\cs_set_eq:NN \__stex_path_path:n \__stex_path_auth:n
\cs_set_eq:NN \__stex_path_module:n \__stex_path_auth:n
\cs_set_eq:NN \__stex_path_name:n \__stex_path_auth:n

\cs_set_protected:Nn \stex_map_uri:Nnnnn{
  \stex_pseudogroup_with:nn{\__stex_path_auth:n\__stex_path_path:n\__stex_path_module:n\__st
```

```
    \cs_set:Npn \__stex_path_auth:n ##1 {#2}
    \cs_set:Npn \__stex_path_path:n ##1 {#3}
    \cs_set:Npn \__stex_path_module:n ##1 {#4}
    \cs_set:Npn \__stex_path_name:n ##1 {#5}
    #1
  }
}
```

(*End of definition for* `\stex_map_uri:Nnnnn`. *This function is documented on page* 138.)

`\stex_uri_set:Nn`
`\stex_uri_set:No`
`\stex_uri_set:Nx`

```
\str_set:Nx\__stex_path_colonslash{\c_colon_str/}
\cs_new_protected:Nn \__stex_path_uri_set:NnN {
  \str_if_empty:nTF {#2} {
    \msg_error:nnxx{stex}{error/invalid-uri}{\tl_to_str:n{#2}}{empty}
  }{
    \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn #1 \__stex_path_colonslash { \tl_to_str
    \seq_pop_left:NN #1 \l__stex_path_auth_str
    \seq_if_empty:NTF #1 {
      \msg_error:nnxx{stex}{error/invalid-uri}{\tl_to_str:n{#2}}{missing~authority}
    }{
      \exp_args:NNnx \seq_set_split:Nnn #1 ? {\exp_args:NNo \seq_use:Nn #1 \__stex_path_colo
      \seq_pop_left:NN #1 \l__stex_path_path
      #3 \l__stex_path_path \l__stex_path_path
      \seq_if_empty:NTF #1 {
        \exp_args:NNo \__stex_path_uri_set:Nnxnn #1 \l__stex_path_auth_str
          {\stex_file_use:N \l__stex_path_path} {} {}
      }{
        \seq_pop_left:NN #1 \l__stex_path_mod
        \seq_if_empty:NTF #1 {
          \exp_args:NNo \__stex_path_uri_set:Nnxon #1 \l__stex_path_auth_str
            {\stex_file_use:N \l__stex_path_path} \l__stex_path_mod {}
        }{
          \seq_pop_left:NN #1 \l__stex_path_name
          \seq_if_empty:NTF #1 {
            \exp_args:NNo \__stex_path_uri_set:Nnxon #2 \l__stex_path_auth_str
              {\stex_file_use:N \l__stex_path_path} \l__stex_path_mod \l__stex_path_name
          }{
            \msg_error:nnxx{stex}{error/invalid-uri}{\tl_to_str:n{#2}}{too~many~?s}
          }
        }
      }
    }
  }
  \stex_debug:nn{uris}{Set~\tl_to_str:n{#1}~to~\stex_uri_use:N #1}
}

\cs_new_protected:Nn \__stex_path_uri_set:Nnnnn{
  \tl_set:Nn #1 {
    \__stex_path_auth:n{ #2 }
    \__stex_path_path:n{ #3 }
    \__stex_path_module:n{ #4 }
    \__stex_path_name:n{ #5 }
  }
```

```
}
\cs_generate_variant:Nn\__stex_path_uri_set:Nnnnn {Nnxnn,Nnxon,Nnxoo}

\cs_new_protected:Nn \stex_uri_set:Nn {
  \__stex_path_uri_set:NnN #1 {#2} \stex_file_set:No
}
\cs_generate_variant:Nn \stex_uri_set:Nn {No, Nx}
```

(*End of definition for* `\stex_uri_set:Nn`*. This function is documented on page 138.*)

`\stex_uri_resolve:Nn`
`\stex_uri_resolve:No`
`\stex_uri_resolve:Nx`

```
\cs_new_protected:Nn \stex_uri_resolve:Nn {
  \__stex_path_uri_set:NnN #1 {#2} \stex_file_resolve:No
}
\cs_generate_variant:Nn \stex_uri_resolve:Nn {No, Nx}
```

(*End of definition for* `\stex_uri_resolve:Nn`*. This function is documented on page 138.*)

`\stex_uri_use:N`

```
\cs_new:Npn \__stex_path_uri_use:w \__stex_path_auth:n #1 \__stex_path_path:n #2 \__stex_pat
  #1\c_colon_str/ #2 \tl_if_empty:nF { #3 }{ ? #3
    \tl_if_empty:nF { #4 }{ ? #4 } }
}
\cs_new:Nn \stex_uri_use:N {
  \exp_args:Ne \cs_if_eq:NNTF { \tl_head:N #1 } \__stex_path_auth:n {
    \exp_after:wN \__stex_path_uri_use:w #1
  }{
    \msg_error:nnnn{stex}{error/invalid-uri}{#1}{Not~a~URI}
  }
}
```

(*End of definition for* `\stex_uri_use:N`*. This function is documented on page 138.*)

`\stex_uri_from_repo_file:NNNn`
`\stex_uri_from_repo_file_nolang:NNNn`

```
\cs_new_protected:Npn \stex_uri_from_repo_file_nolang:NNNn {
  \__stex_path_from_repo_file:NNNNn \stex_file_split_off_lang:NN
}
\cs_new_protected:Npn \stex_uri_from_repo_file:NNNn {
  \__stex_path_from_repo_file:NNNNn \stex_file_split_off_ext:NN
}

\cs_new_protected:Nn \__stex_path_from_repo_file:NNNNn {
  #1 \l__stex_path_file #4
  \prop_if_exist:NTF #3 {
    \str_clear:N \l__stex_path_uri
    \prop_get:NnNF #3 {#5} \l__stex_path_uri {
      \prop_get:NnNF #3 {ns} \l__stex_path_uri {
        \__stex_path_uri_set:Nnxnn #2 {file}
          {\stex_file_use:N \l__stex_path_file} {} {}
      }
    }
    \str_if_empty:NF \l__stex_path_uri {\__stex_path_relativize:N #2}
  }{
    \exp_args:NNx \__stex_path_uri_set:Nnxnn #2 {\tl_to_str:n{file}}
      {\stex_file_use:N \l__stex_path_file} {} {}
```

```
  }
}

\cs_new_protected:Nn \__stex_path_relativize:N {
  \seq_set_eq:NN \l__stex_path_seq \l__stex_path_file
  \seq_map_inline:Nn \c_stex_mathhub_file { % mathhub path
    \seq_pop_left:NN \l__stex_path_seq \l__stex_path_tl
  }
  \stex_file_set:Nx \l__stex_path_path {\prop_item:Nn \l_stex_current_archive_prop {id} }
  \seq_map_inline:Nn \l__stex_path_path { % id
    \seq_pop_left:NN \l__stex_path_seq \l__stex_path_tl
  }
  \seq_pop_left:NN \l__stex_path_seq \l__stex_path_tl % source

  \stex_uri_set:Nx #1 { \l__stex_path_uri / \stex_file_use:N \l__stex_path_seq }
}
```

(*End of definition for* `\stex_uri_from_repo_file:NNNn` *and* `\stex_uri_from_repo_file_nolang:NNNn`. *These functions are documented on page 138.*)

`\stex_uri_from_current_file:Nn`
`\stex_uri_from_current_file_nolang:Nn`

```
\cs_new_protected:Nn \stex_uri_from_current_file:Nn {
  \stex_debug:nn{docuri}{Here:~\stex_file_use:N \g_stex_current_file}
  \stex_uri_from_repo_file:NNNn #1 \l_stex_current_archive_prop
    \g_stex_current_file {#2}
  \stex_debug:nn{docuri}{resolved:~\stex_uri_use:N #1}
}
\cs_new_protected:Nn \stex_uri_from_current_file_nolang:Nn {
  \stex_uri_from_repo_file_nolang:NNNn #1 \l_stex_current_archive_prop
    \g_stex_current_file {#2}
}
```

(*End of definition for* `\stex_uri_from_current_file:Nn` *and* `\stex_uri_from_current_file_nolang:Nn`. *These functions are documented on page 139.*)

`\stex_uri_add_module:NNn`
`\stex_uri_add_module:NNo`

```
\cs_new_protected:Nn \stex_uri_add_module:NNn {
  \exp_args:Ne \cs_if_eq:NNTF { \tl_head:N #2 } \__stex_path_auth:n {
    \stex_pseudogroup_with:nn
      {\__stex_path_auth:n\__stex_path_path:n\__stex_path_module:n\__stex_path_name:n}
      {
        \cs_set:Npn \__stex_path_module:n ##1 {
          \tl_if_empty:nTF{##1}{
            \exp_not:N \__stex_path_module:n {#3}
          }{
            \msg_error:nnn{stex}{error/invalid-dpath}{#2}
          }
        }
        \cs_set:Npn \__stex_path_name:n ##1 {
          \tl_if_empty:nTF{##1}{
            \exp_not:N \__stex_path_name:n {}
          }{
            \msg_error:nnn{stex}{error/invalid-dpath}{#2}
          }
        }
```

```
        \tl_set:Nx #1 {#2}
      }
  }{
    \msg_error:nnnn{stex}{error/invalid-uri}{#2}{Not~a~URI}
  }
}
\cs_generate_variant:Nn \stex_uri_add_module:NNn {NNo}
```

(*End of definition for* `\stex_uri_add_module:NNn`. *This function is documented on page 139.*)

`\l_stex_current_doc_uri`

```
\tl_new:N \l_stex_current_doc_uri
```

(*End of definition for* `\l_stex_current_doc_uri`. *This variable is documented on page 139.*)

`\stex_set_document_uri:`

```
\cs_new_protected:Nn \stex_set_document_uri: {
  \stex_uri_from_current_file:Nn \l_stex_current_doc_uri {narr}
  %\stex_debug:nn{sref}{Document~URI:~\stex_uri_use:N \l_stex_current_doc_uri}
}
```

(*End of definition for* `\stex_set_document_uri:`. *This function is documented on page 139.*)

`\stex_set_current_namespace:`

```
\cs_new_protected:Nn \stex_set_current_namespace: {
  \stex_uri_from_current_file_nolang:Nn \l_stex_current_ns_uri {source-base}
  %\stex_debug:nn{modules}{Namespace~URI:~\stex_uri_use:N \l_stex_current_ns_uri}
}
```

(*End of definition for* `\stex_set_current_namespace:`. *This function is documented on page 139.*)

We determine the PWD

`\c_stex_pwd_file`
`\c_stex_main_file`

```
\sys_if_platform_windows:TF{
  \stex_get_env:Nn\l__stex_path_str{CD}
}{
  \stex_get_env:Nn\l__stex_path_str{PWD}
}
\stex_file_resolve:No \c_stex_pwd_file \l__stex_path_str
\seq_set_eq:NN \c_stex_main_file \c_stex_pwd_file
\seq_put_right:Nx \c_stex_main_file {\jobname\tl_to_str:n{.tex}}

\stex_debug:nn {files} {PWD:~\stex_file_use:N \c_stex_pwd_file}
```

(*End of definition for* `\c_stex_pwd_file` *and* `\c_stex_main_file`. *These variables are documented on page 138.*)

13.2.9 File Hooks

keeps track of file changes:

```
\seq_gclear_new:N\g__stex_path_stack
\seq_gclear_new:N\g_stex_current_file
```

\stex_filestack_push:n

```
\cs_new_protected:Nn \stex_filestack_push:n {
  \stex_str_if_ends_with:nnTF {#1}{.tex}{
    \stex_file_resolve:No \g_stex_current_file {#1}
  }{
    \stex_file_resolve:No \g_stex_current_file {#1.tex}
  }
  \stex_if_file_absolute:NF \g_stex_current_file {
    \stex_file_resolve:Nx \g_stex_current_file {
      \stex_file_use:N \c_stex_pwd_file / \stex_file_use:N \g_stex_current_file
    }
  }
  \seq_gset_eq:NN \g_stex_current_file \g_stex_current_file
  \exp_args:NNx \seq_gpush:Nn \g__stex_path_stack {\stex_file_use:N \g_stex_current_file}
  \_stex_every_file:
}

\cs_new_protected:Nn \_stex_every_file: {
  \stex_set_document_uri:
  \stex_language_from_file:
  \stex_set_current_namespace:
}
%\AtBeginDocument{\_stex_every_file:}
```

(*End of definition for* `\stex_filestack_push:n`*. This function is documented on page 137.*)

\stex_filestack_pop:

```
\cs_new_protected:Nn \stex_filestack_pop: {
  \seq_if_empty:NF \g__stex_path_stack {
    \seq_gpop:NN \g__stex_path_stack \l__stex_path_str
  }
  \seq_if_empty:NTF \g__stex_path_stack {
    \seq_gset_eq:NN \g_stex_current_file \c_stex_main_file
  }{
    \seq_get:NN \g__stex_path_stack \l__stex_path_str
    \exp_args:NNo \stex_file_set:Nn \g_stex_current_file \l__stex_path_str
    \seq_gset_eq:NN \g_stex_current_file \g_stex_current_file
  }
  \_stex_every_file:
}
```

(*End of definition for* `\stex_filestack_pop:`*. This function is documented on page 137.*)

Hooks for the current file:

```
\cs_new_protected:Nn \stex_input_with_hooks:n {
  \tl_set:Nn \l__stex_path_do_hooks_pre_tl {
    \tl_gset:Nn \l__stex_path_do_hooks_pre_tl {}
    \stex_debug:nn{HERE}{Hook~for~#1^^J\meaning\CurrentFilePath^^J\CurrentFile}
    \tl_if_empty:NTF\CurrentFilePath{
```

```
      \exp_args:No \stex_filestack_push:n {\CurrentFile}
    }{
      \exp_args:Ne \stex_filestack_push:n { \CurrentFilePath / \CurrentFile }
    }
  }
  \input{#1}
  \stex_debug:nn{HERE}{Hook~end~for~#1}
  \stex_filestack_pop:
}
\tl_new:N \l__stex_path_do_hooks_pre_tl {}

\AddToHook{file/before}{
  \l__stex_path_do_hooks_pre_tl
}
%\AddToHook{file/after}{ \stex_filestack_pop: }
```

13.3 Math Archives

⟨@@=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.

```
\sys_if_platform_windows:TF{
  \stex_get_env:Nn \l__stex_mathhub_str {homedrive\c_percent_str\c_percent_str homepath}
}{
  \stex_get_env:Nn \l__stex_mathhub_str {HOME}
}
\stex_file_resolve:No \c_stex_home_file \l__stex_mathhub_str

\str_if_empty:NTF\mathhub{
  \stex_get_env:Nn \l__stex_mathhub_str {MATHHUB}
  \str_if_empty:NTF \l__stex_mathhub_str {
    \ior_open:NnTF \g_tmpa_ior{\stex_file_use:N \c_stex_home_file/.stex/mathhub.path}{
      \group_begin:
        \escapechar=-1\catcode`\\=12
        \ior_str_get:NN \g_tmpa_ior \l__stex_mathhub_str
        \str_gset_eq:NN \l__stex_mathhub_str \l__stex_mathhub_str
      \group_end:
      \ior_close:N \g_tmpa_ior
      \stex_debug:nn{mathhub}{MathHub~directory~determined~from~home~directory}
    }{
      \str_clear:N \l__stex_mathhub_str
    }
  }{
    \stex_debug:nn{mathhub}{MathHub~directory~determined~from~environment~variable}
  }
}{
  \str_set_eq:NN \l__stex_mathhub_str \mathhub
}

\str_if_empty:NTF \l__stex_mathhub_str {
  \msg_warning:nn{stex}{warning/nomathhub}
```

```
  \exp_args:NNe \stex_file_set:Nn \c_stex_mathhub_file {\stex_file_use:N \c_stex_home_file \
  \seq_clear:N \c_stex_mathhub_file
}{
  \stex_file_resolve:No \c_stex_mathhub_file \l__stex_mathhub_str
  \stex_if_file_absolute:NF \c_stex_mathhub_file {
    \exp_args:NNe \stex_file_resolve:Nn \c_stex_mathhub_file {
      \stex_file_use:N \c_stex_main_file / .. / \l__stex_mathhub_str
    }
  }
}

\exp_args:NNe \str_set:Nn \mathhub {\stex_file_use:N \c_stex_mathhub_file}
\stex_debug:nn{mathhub}{MATHHUB:~\mathhub}
```

(*End of 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`

```
\cs_new_protected:Nn \stex_set_current_archive:n {
  \stex_require_archive:n { #1 }
  \stex_debug:nn{mathhub}{switching~to~archive~#1}
  \prop_set_eq:Nc \l_stex_current_archive_prop {
    c_stex_mathhub_#1_manifest_prop
  }
}
```

(*End of definition for* `\l_stex_current_archive` *and* `\stex_set_current_archive:n`. *These variables are documented on page* **??**.)

`\stex_in_archive:nn`

```
\cs_new_protected:Nn \stex_in_archive:nn {
  \cs_if_exist:NF \l__stex_mathhub_cs {
    \cs_set:Npn \l__stex_mathhub_cs ##1 {}
  }
  \stex_pseudogroup:nn{
    \cs_set:Npn \l__stex_mathhub_cs ##1 {#2}
    \tl_if_empty:nTF{#1}{
      \prop_if_exist:NTF \l_stex_current_archive_prop {
        \exp_args:Ne \l__stex_mathhub_cs {\prop_item:Nn \l_stex_current_archive_prop { id }
      }{
        \l__stex_mathhub_cs {}
      }
    }{
      \stex_set_current_archive:n{#1}
      \l__stex_mathhub_cs {#1}
    }
  }{
    \stex_pseudogroup_restore:N \l_stex_current_archive_prop
    \cs_set:Npn \exp_not:N \l__stex_mathhub_cs ##1 {
      \exp_args:No \exp_not:n {\l__stex_mathhub_cs {##1}}
    }
  }
}
```

(*End of definition for* `\stex_in_archive:nn`. *This function is documented on page 134.*)

\stex_require_archive:n
\stex_require_archive:o

```
\cs_new_protected:Nn \stex_require_archive:n {
  \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
    \seq_if_empty:NTF \c_stex_mathhub_file {
      \stex_fatal_error:n{warning/nomathhub}
    }{
      \stex_debug:nn{mathhub}{Opening~archive:~#1}
      \__stex_mathhub_do_manifest:n { #1 }
    }
  }
}
\cs_generate_variant:Nn \stex_require_archive:n {o}
```

(*End of definition for* `\stex_require_archive:n`*. This function is documented on page* 134.)

Code for finding and parsing manifest files:

```
\cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
  \exp_args:Ne \__stex_mathhub_find_manifest:n {\stex_file_use:N \c_stex_mathhub_file / #1}
  \str_if_empty:NT \l__stex_mathhub_manifest_str {
    \stex_fatal_error:nxx{error/noarchive}
      {#1}{\stex_file_use:N \c_stex_mathhub_file}
  }
  \__stex_mathhub_parse_manifest:n {#1}
}

\cs_new_protected:Nn \__stex_mathhub_find_manifest:n {
  \str_clear:N \l__stex_mathhub_manifest_str
  \seq_set_split:Nnn \l__stex_mathhub_seq / {#1}
  \bool_set_true:N \l__stex_mathhub_bool
  \bool_while_do:Nn \l__stex_mathhub_bool {
    \tl_if_eq:NNTF \l__stex_mathhub_seq \c_stex_mathhub_file {
      \bool_set_false:N \l__stex_mathhub_bool
    }{
      \__stex_mathhub_check_manifest:
      \bool_if:NT \l__stex_mathhub_bool {
        \seq_pop_right:NN \l__stex_mathhub_seq \l__stex_mathhub_tl
      }
    }
  }
}
\cs_generate_variant:Nn \__stex_mathhub_find_manifest:n {x}

\cs_new_protected:Nn \__stex_mathhub_check_manifest: {
  \__stex_mathhub_check_manifest:n {MANIFEST.MF}
  \bool_if:NT \l__stex_mathhub_bool {
    \__stex_mathhub_check_manifest:n {META-INF/MANIFEST.MF}
    \bool_if:NT \l__stex_mathhub_bool {
      \__stex_mathhub_check_manifest:n {meta-inf/MANIFEST.MF}
    }
  }
}

\cs_new_protected:Nn \__stex_mathhub_check_manifest:n {
  \stex_debug:nn{mathhub}{Checking~\stex_file_use:N \l__stex_mathhub_seq / #1}
  \file_if_exist:nT {\stex_file_use:N \l__stex_mathhub_seq / #1} {
```

```
    \bool_set_false:N \l__stex_mathhub_bool
    \str_set:Nx \l__stex_mathhub_manifest_str {\stex_file_use:N \l__stex_mathhub_seq / #1}
  }
}

\ior_new:N \c__stex_mathhub_manifest_ior
\cs_new_protected:Nn \__stex_mathhub_parse_manifest:n {
  \ior_open:Nn \c__stex_mathhub_manifest_ior \l__stex_mathhub_manifest_str
  \prop_clear:N \l__stex_mathhub_prop
  \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
    \exp_args:NNNo \exp_args:NNNx
      \seq_set_split:Nnn \l__stex_mathhub_seq \c_colon_str {\tl_to_str:n{##1}}
    \seq_pop_left:NNT \l__stex_mathhub_seq \l__stex_mathhub_key {
      \exp_args:NNo \str_set:Nn \l__stex_mathhub_key \l__stex_mathhub_key
      \str_set:Nx \l__stex_mathhub_val {\seq_use:Nn \l__stex_mathhub_seq :}
      \str_case:Nn \l__stex_mathhub_key {
        {id}              {\prop_put:Nno \l__stex_mathhub_prop { id }      \l__stex_mathhub_
        {narration-base}  {\prop_put:Nno \l__stex_mathhub_prop { narr }    \l__stex_mathhub_
        {url-base}        {\prop_put:Nno \l__stex_mathhub_prop { docurl }  \l__stex_mathhub_
        {source-base}     {\prop_put:Nno \l__stex_mathhub_prop { ns }      \l__stex_mathhub_
        {ns}              {\prop_put:Nno \l__stex_mathhub_prop { ns }      \l__stex_mathhub_
      }
    }
  }
  \ior_close:N \c__stex_mathhub_manifest_ior
  \prop_gset_eq:cN { c_stex_mathhub_#1_manifest_prop } \l__stex_mathhub_prop
  \stex_debug:nn{mathhub}{Result:~\prop_to_keyval:N \l__stex_mathhub_prop}
  \stex_persist:e {
    \prop_gset_from_keyval:cn {c_stex_mathhub_#1_manifest_prop}{
      \prop_to_keyval:N \l__stex_mathhub_prop
    }
  }
}
```

Current MathHub archive

\c_stex_main_archive_prop
\l_stex_current_archive_prop

```
\cs_new_protected:Nn \_stex_main_archive: {
  \stex_if_file_starts_with:NNTF \c_stex_pwd_file \c_stex_mathhub_file {
    \__stex_mathhub_find_manifest:x { \stex_file_use:N \c_stex_pwd_file }
    \str_if_empty:NTF \l__stex_mathhub_manifest_str {
      \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~archive}
    }{
      \__stex_mathhub_parse_manifest:n { main }
      \prop_set_eq:NN \c_stex_main_archive_prop \c_stex_mathhub_main_manifest_prop
      \cs_undefine:N \c_stex_mathhub_main_manifest_prop
      \prop_get:NnN \c_stex_main_archive_prop {id}
        \l__stex_mathhub_str
      \prop_set_eq:cN { c_stex_mathhub_\l__stex_mathhub_str _manifest_prop }
        \c_stex_main_archive_prop
      \exp_args:No \stex_set_current_archive:n { \l__stex_mathhub_str }
      \stex_debug:nn{mathhub}{Current~archive:~
        \prop_item:Nn \l_stex_current_archive_prop {id}
      }
      \bool_if:NT \c_stex_persist_write_mode_bool {
```

```
        \tl_put_right:Nx \_stex_persist_read_now: {
          \stex_persist:n {
            \prop_gset_from_keyval:cn {c_stex_mathhub_\l__stex_mathhub_str _manifest_prop}{
              \prop_to_keyval:N \c_stex_main_archive_prop
            }
            \prop_gset_eq:Nc \exp_not:N \l_stex_current_archive_prop {
              c_stex_mathhub_\l__stex_mathhub_str _manifest_prop
            }
            \prop_gset_eq:Nc \exp_not:N \c_stex_main_archive_prop {
              c_stex_mathhub_\l__stex_mathhub_str _manifest_prop
            }
          }
        }
      }
    }
  }{
    \stex_debug:nn{mathhub}{Not~currently~in~the~MathHub~directory}
  }
}

%\bool_if:NF \c_stex_persist_mode_bool {
  \_stex_main_archive:
%}
```

(*End of definition for* `\c_stex_main_archive_prop` *and* `\l_stex_current_archive_prop`. *These variables are documented on page* 134.)

13.4 Documents

13.4.1 Title

Stores the title, if it exists:

```
⟨@@=stex_doc⟩
\tl_new:N \g__stex_doc_title_tl
```

`\stexdoctitle` Initital definition, will be changed at begin document:

```
\cs_new_protected:Npn \stexdoctitle #1 {
  \tl_gset:Nn \g__stex_doc_title_tl { #1 }
  \global\def\stexdoctitle##1{}
}
```

At begin document, we switch to:

```
\cs_new_protected:Nn \__stex_doc_set_title:n {
  \stex_if_smsmode:F{
    \global\def\stexdoctitle##1{}
    \stex_debug:nn{title}{Setting~title~to:\tl_to_str:n{#1}}
    \tl_gset:Nn \g__stex_doc_title_tl { #1 }
    \__stex_doc_title_html:
  }
}
```

Hooks, changes and HTML:

```
\cs_new_protected:Nn \__stex_doc_title_html: {
  \stex_if_do_html:T{\stex_if_html_backend:T{
```

```
    \stex_annotate_invisible:nn{shtml:doctitle={}}{ \hbox{\g__stex_doc_title_tl} }
  }}
}

\AtBeginDocument {
  \tl_if_empty:NTF \g__stex_doc_title_tl {
    \cs_set_eq:NN \stexdoctitle \__stex_doc_set_title:n
  }{
    \stex_debug:nn{title}{Setting~title~to:\exp_args:No\tl_to_str:n\g__stex_doc_title_tl}
    \global\def\stexdoctitle#1{}
    \__stex_doc_title_html:
  }

  \cs_set_eq:NN \__stex_doc_maketitle: \maketitle
  \global\protected\def\maketitle{
    \tl_if_empty:NF \@title {
      \exp_args:No \stexdoctitle \@title
    }
    \__stex_doc_maketitle:
  }
}
```

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

13.4.2 Sectioning

```
\int_new:N \l_stex_docheader_sect

\tl_set:Nn \stex_current_section_level {document}

\cs_set_protected:Npn \currentsectionlevel {
  \stex_if_do_html:TF{
    \stex_annotate:nn{shtml:currentsectionlevel={},shtml:capitalize=false}{}
  }{
    \stex_current_section_level
  }
  \tl_if_exist:NT\xspace\xspace
}

\cs_set_protected:Npn \Currentsectionlevel {
  \stex_if_do_html:TF{
    \stex_annotate:nn{shtml:currentsectionlevel={},shtml:capitalize=true}{}
  }{
    \exp_args:No \_stex_capitalize:n \stex_current_section_level
  }
  \tl_if_exist:NT\xspace\xspace
}

\stex_if_html_backend:TF {
  \cs_new_protected:Nn \_sfragment_do_level:nn {
    \stexdoctitle{#2}
    \par
    \begin{stex_annotate_env}{shtml:section={\int_use:N \l_stex_docheader_sect}}
      \noindent\stex_annotate:nn{shtml:sectiontitle={}}{
```

```
        \_stex_annotate_force_break:n{#2}
      }\par
  }
  \cs_new_protected:Nn \_sfragment_end: {
    \end{stex_annotate_env}
  }
}{
  \cs_new_protected:Nn \_sfragment_do_level:nn {
    \stexdoctitle{#2}
    \tl_if_empty:NTF \l_stex_key_short_tl {
      \use:c{#1}
    }{
      \exp_args:Nnx \use:nn{\use:c{#1}}{[\exp_args:No \exp_not:n \l_stex_key_short_tl]}
    }{#2}
    \int_incr:N \l_stex_docheader_sect
    \tl_set:Nn \stex_current_section_level{#1}
  }
  \cs_new_protected:Nn \_sfragment_end: {}
}


\cs_new_protected:Npn \__stex_doc_do_section:n {
  \int_case:nnF \l_stex_docheader_sect {
    {0}{\cs_if_exist:NTF \thepart {\_sfragment_do_level:nn{part}}{
      \int_incr:N \l_stex_docheader_sect
      \__stex_doc_do_section:n
    }}
    {1}{\cs_if_exist:NTF \thechapter {\_sfragment_do_level:nn{chapter}}{
      \int_incr:N \l_stex_docheader_sect
      \__stex_doc_do_section:n
    }}
    {2}{\_sfragment_do_level:nn{section}}
    {3}{\_sfragment_do_level:nn{subsection}}
    {4}{\_sfragment_do_level:nn{subsubsection}}
    {5}{\_sfragment_do_level:nn{paragraph}}
  }{\_sfragment_do_level:nn{subparagraph}}
}

\stex_keys_define:nnnn{ sfragment }{
  \tl_clear:N \l_stex_key_short_tl
}{
  short   .tl_set:N   = \l_stex_key_short_tl
}{id}

\NewDocumentEnvironment{sfragment}{ O{} m}{
  \stex_keys_set:nn{sfragment}{#1}
  \__stex_doc_do_section:n{#2}
  \_stex_do_id:
}{
  \_sfragment_end:
}

%\int_incr:N \l_stex_docheader_sect
\NewDocumentEnvironment{blindfragment}{}{
```

```
  \__stex_doc_skip_section:
}{
  \stex_if_html_backend:T{
    \stex_annotate_invisible:n{~}
    \end{stex_annotate_env}
  }
}


\cs_new_protected:Nn \__stex_doc_skip_section_i: {
  \int_case:nn \l_stex_docheader_sect {
    {0}{\cs_if_exist:NF \thepart {
      \int_incr:N \l_stex_docheader_sect \__stex_doc_skip_section_i:
    }}
    {1}{\cs_if_exist:NF \thechapter {
      \int_incr:N \l_stex_docheader_sect \__stex_doc_skip_section_i:
    }}
  }
  \int_incr:N \l_stex_docheader_sect
}

\stex_if_html_backend:TF {
  \cs_new_protected:Nn \__stex_doc_skip_section: {
    \__stex_doc_skip_section_i:
    \begin{stex_annotate_env}{shtml:skipsection={\int_use:N \l_stex_docheader_sect}}
    \stex_annotate_invisible:n{~}
  }
}{
  \cs_set_eq:NN \__stex_doc_skip_section: \__stex_doc_skip_section_i:
}


\cs_new_protected:Nn \__stex_doc_skip_fragment:n {
  \stepcounter{#1}
}

\cs_new_protected:Npn \skipfragment {
  \int_case:nnF \l_stex_docheader_sect {
    {0}{\cs_if_exist:NTF \thepart {\__stex_doc_skip_fragment:n{part}}{
      \int_incr:N \l_stex_docheader_sect
      \skipfragment
    }}
    {1}{\cs_if_exist:NTF \thechapter {\__stex_doc_skip_fragment:n{chapter}}{
      \int_incr:N \l_stex_docheader_sect
      \skipfragment
    }}
    {2}{\__stex_doc_skip_fragment:n{section}}
    {3}{\__stex_doc_skip_fragment:n{subsection}}
    {4}{\__stex_doc_skip_fragment:n{subsubsection}}
    {5}{\__stex_doc_skip_fragment:n{paragraph}}
  }{\__stex_doc_skip_fragment:n{subparagraph}}
}
```

\setsectionlevel

```
\cs_new_protected:Npn \setsectionlevel #1 {
  \str_case:nnF{#1}{
    {part}{\int_set:Nn \l_stex_docheader_sect 0}
    {chapter}{\int_set:Nn \l_stex_docheader_sect 1}
    {section}{\int_set:Nn \l_stex_docheader_sect 2}
    {subsection}{\int_set:Nn \l_stex_docheader_sect 3}
    {subsubsection}{\int_set:Nn \l_stex_docheader_sect 4}
    {paragraph}{\int_set:Nn \l_stex_docheader_sect 5}
  }{
    \int_set:Nn \l_stex_docheader_sect 6
  }
  \cs_if_eq:NNTF\@onlypreamble\@notprerr{
    \stex_annotate_invisible:nn{shtml:sectionlevel={\int_use:N\l_stex_docheader_sect}}{}
  }{}
}

\stex_if_html_backend:T{
  \cs_new_protected:Nn \__stex_doc_check_topsect: {
    \int_case:nnF \l_stex_docheader_sect {
      {0}{\cs_if_exist:NTF \thepart {
        \stex_annotate_invisible:nn{shtml:sectionlevel=0}{}
      }{
        \int_incr:N \l_stex_docheader_sect
        \__stex_doc_check_topsect:
      }}
      {1}{\cs_if_exist:NTF \thechapter {
        \stex_annotate_invisible:nn{shtml:sectionlevel=1}{}
      }{
        \int_incr:N \l_stex_docheader_sect
        \__stex_doc_check_topsect:
      }}
    }{
      \stex_annotate_invisible:nn{shtml:sectionlevel={\int_use:N\l_stex_docheader_sect}}{}
    }
  }
  \AtBeginDocument{\__stex_doc_check_topsect:}
}

\AtBeginDocument{
  \bool_if:NF \c_stex_no_frontmatter_bool {
    \cs_if_exist:NTF\frontmatter{
      \let\__stex_doc_orig_frontmatter\frontmatter
      \let\frontmatter\relax
    }{
      \tl_set:Nn\__stex_doc_orig_frontmatter{
        \clearpage
        %\@mainmatterfalse
        \pagenumbering{roman}
      }
    }
    \cs_if_exist:NTF\backmatter{
      \let\__stex_doc_orig_backmatter\backmatter
      \let\backmatter\relax
    }{
```

```
      \tl_set:Nn\__stex_doc_orig_backmatter{
        \clearpage
        %\@mainmatterfalse
        \pagenumbering{roman}
      }
    }
    \newenvironment{frontmatter}{
      \__stex_doc_orig_frontmatter
    }{
      \cs_if_exist:NTF\mainmatter{
        \mainmatter
      }{
        \clearpage
        %\@mainmattertrue
        \pagenumbering{arabic}
      }
    }
    \newenvironment{backmatter}{
      \__stex_doc_orig_backmatter
    }{
      \cs_if_exist:NTF\mainmatter{
        \mainmatter
      }{
        \clearpage
        %\@mainmattertrue
        \pagenumbering{arabic}
      }
    }
  }
}
```

(*End of definition for* `\setsectionlevel`. *This function is documented on page* 82.)

13.4.3 References

⟨`@@=stex_refs`⟩

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

```
\iow_new:N \c__stex_refs_iow
\AtBeginDocument{\iow_open:Nn \c__stex_refs_iow {\jobname.sref}}
\AtEndDocument{\iow_close:N \c__stex_refs_iow}
```

The following macros are written to the `.aux`-file, and hence use LaTeX2e character code scheme:

```
\cs_new_protected:Npn \STEXInternalSrefRestoreTarget #1#2#3#4#5 {}

\cs_new_protected:Npn \STEXInternalSetSrefSymURL #1 #2 {
  \str_gset:cn{g_stex_sref_sym_\tl_to_str:n{#1}_target}{#2}
}

\stex_ref_new_doc_target:n
                \sreflabel
\seq_new:N \g__stex_refs_files_seq
\int_new:N \l__stex_refs_unnamed_counter_int
```

```
\cs_new_protected:Nn \_stex_ref_new_id:n {
  \str_if_empty:nTF {#1}{
    \int_gincr:N \l__stex_refs_unnamed_counter_int
    \str_set:Nx \l__stex_refs_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
  }{
    \str_set:Nn \l__stex_refs_str {#1}
  }
  \str_set:Nx \l_stex_ref_url_str {\stex_uri_use:N \l_stex_current_doc_uri ? \l__stex_refs_s
}

\cs_new_protected:Nn \stex_ref_new_doc_target:n {
  \_stex_ref_new_id:n{#1}
  %\stex_uri_add_module:NNo \l__stex_refs_uri \l_stex_current_doc_uri \l__stex_refs_str
  %\stex_debug:nn{sref}{New~document~target:~\stex_uri_use:N \l__stex_refs_uri}
  \__stex_refs_add_doc_ref:xo {\stex_uri_use:N \l_stex_current_doc_uri} \l__stex_refs_str
  \stex_if_smsmode:F {
    \iow_now:Nx \c__stex_refs_iow {
      \STEXInternalSrefRestoreTarget
      {\stex_uri_use:N \l_stex_current_doc_uri}
      {\l__stex_refs_str}
      {\@currentcounter}
      {\@currentlabel}
      {
        \tl_if_exist:NT\@currentlabelname{
          \exp_args:No\exp_not:n\@currentlabelname
        }
      }
    }
    \exp_args:Nx \label {sref@\l_stex_ref_url_str}
    \stex_if_do_html:T {
      \pdfdest name "sref@\l_stex_ref_url_str" xyz\relax
    }
  }
}
\NewDocumentCommand \sreflabel {m} {\stex_ref_new_doc_target:n {#1}}

\cs_new_protected:Nn \__stex_refs_add_doc_ref:nn {
  \seq_if_in:NnTF \g__stex_refs_files_seq {#1} {
    \seq_if_in:cnF {g__stex_refs_#1_seq}{#2}{
      \seq_gput_left:cn{g__stex_refs_#1_seq}{#2}
    }
  }{
    \seq_gput_right:Nn \g__stex_refs_files_seq {#1}
    \seq_new:c{g__stex_refs_#1_seq}
    \seq_gput_left:cn{g__stex_refs_#1_seq}{#2}
  }
}
\cs_generate_variant:Nn \__stex_refs_add_doc_ref:nn {xo,xx}
```

(*End of definition for* `\stex_ref_new_doc_target:n` *and* `\sreflabel`. *These functions are documented on page* 121.)

`\sref` `\extref` Optional arguments:

```
\stex_keys_define:nnnn{sref / 1}{}{
  % TODO get rid of this
  fallback  .code:n = {},
  pre       .code:n = {},
  post      .code:n = {}
}{archive file}
\stex_keys_define:nnnn{sref / 2}{}{}{archive file, title}

\str_new:N \l__stex_refs_default_archive_str
\str_new:N \l__stex_refs_default_file_str
\tl_new:N \l__stex_refs_default_title_tl

\cs_set_protected:Nn \__stex_refs_set_keys_b:n {
  \tl_if_empty:nTF{#1}{
    \str_set_eq:NN \l_stex_key_archive_str \l__stex_refs_default_archive_str
    \str_set_eq:NN \l_stex_key_file_str \l__stex_refs_default_file_str
    \tl_set_eq:NN \l_stex_key_title_tl \l__stex_refs_default_title_tl
  }{
    \stex_keys_set:nn{ sref / 2 }{ #1 }
  }
}

\newcommand\srefsetin[3][]{
  \str_set:Nx \l__stex_refs_default_archive_str {#1}
  \str_set:Nx \l__stex_refs_default_file_str {#2}
  \tl_set:Nn \l__stex_refs_default_title_tl {#3}
}

```

Auxiliary methods:

```
\cs_new_protected:Nn \__stex_refs_find_uri:n {
  \str_clear:N \l__stex_refs_uri_str
  \stex_debug:nn{sref}{
    File:~\l_stex_key_file_str^^J
    Repo:\l_stex_key_archive_str
  }
  \str_if_empty:NTF \l_stex_key_file_str {
    \stex_debug:nn{sref}{Empty.~Checking~current~file~for~#1}
    \seq_if_exist:cT{g__stex_refs_\stex_uri_use:N \l_stex_current_doc_uri _seq}{
      \exp_args:Nnx \__stex_refs_find_uri_in_file:nnn{#1}
        {\stex_uri_use:N \l_stex_current_doc_uri}\seq_map_break:
    }
    \str_if_empty:NT \l__stex_refs_uri_str {
      \seq_map_inline:Nn \g__stex_refs_files_seq {
        \__stex_refs_find_uri_in_file:nnn{#1}{##1}{\seq_map_break:n{\seq_map_break:}}
      }
    }
  }{
    \str_if_empty:NTF \l_stex_key_archive_str {
      \prop_if_exist:NTF \l_stex_current_archive_prop {
        \__stex_refs_find_uri_in_prop_file:N \l_stex_current_archive_prop
      }{
        \stex_file_resolve:Nx \l__stex_refs_file
          { \stex_file_use:N \g_stex_current_file / .. / \l_stex_key_file_str }
```

```
        \str_set:Nx \l__stex_refs_uri_str { file:/ \stex_file_use:N \l__stex_refs_file }
      }
    }{
      \stex_require_archive:o \l_stex_key_archive_str
      \prop_set_eq:Nc \l__stex_refs_prop { c_stex_mathhub_\l_stex_key_archive_str _manifest_
      \__stex_refs_find_uri_in_prop_file:N \l__stex_refs_prop
    }
  }
}

\cs_new_protected:Nn \__stex_refs_find_uri_in_prop_file:N {
  \str_set:Nx \l__stex_refs_uri_str {
    \stex_file_use:N \c_stex_mathhub_file /
    \prop_item:Nn #1 {id} /
      source / \l_stex_key_file_str .sref
  }
  \stex_file_resolve:No \l__stex_refs_file \l__stex_refs_uri_str
  \stex_uri_from_repo_file:NNNn \l__stex_refs_uri #1
    \l__stex_refs_file {narr}
  \str_set:Nx \l__stex_refs_uri_str {\stex_uri_use:N \l__stex_refs_uri}
}

\cs_new_protected:Nn \__stex_refs_find_uri_in_file:nnn {
  \stex_debug:nn{sref}{Checking~file~#2}
  \seq_map_inline:cn{g__stex_refs_#2_seq}{
    \str_if_eq:nnT{#1}{##1}{
      \str_set:Nx \l__stex_refs_uri_str {\stex_uri_use:N \l_stex_current_doc_uri}
      \stex_debug:nn{sref}{Found.}
      #3
    }
  }
}
```

Doing the actual referencing:

```
\cs_new_protected:Nn \__stex_refs_do_autoref:n {
  \cs_if_exist:cTF{autoref}{
     \exp_args:Nx\autoref{sref@#1}
   }{
     \exp_args:Nx\ref{sref@#1}
   }
}

\cs_new_protected:Nn \__stex_refs_do_sref:nn {
  \str_if_empty:NTF \l__stex_refs_uri_str {
    \str_if_empty:NTF \l_stex_key_file_str {
      \stex_debug:nn{sref}{autoref~on~\stex_uri_use:N \l_stex_current_doc_uri?#1}
      \exp_args:Ne \__stex_refs_do_autoref:n{\stex_uri_use:N \l_stex_current_doc_uri ? #1}
    }{
      \stex_debug:nn{sref}{srefin~on~#1}
      \__stex_refs_set_keys_b:n{ #2 }
      \__stex_refs_do_sref_in:n{#1}
    }
  }{
    \exp_args:NNo \seq_if_in:NnTF \g__stex_refs_files_seq \l__stex_refs_uri_str {
```

```
      \stex_debug:nn{sref}{Using~ref~file~\l__stex_refs_uri_str}
      \exp_args:Nnx \seq_if_in:cnTF{g_stex_ref_\l__stex_refs_uri_str _seq}{\detokenize{#1}}{
        \stex_debug:nn{sref}{Reference~found~in~ref~files;~autoref~on~\l__stex_refs_uri_str?
        \__stex_refs_do_autoref:n{\l__stex_refs_uri_str?#1}
      }{
        \str_if_empty:NTF \l_stex_key_file_str {
          \stex_debug:nn{sref}{in~empty;~autoref~on~\l__stex_refs_uri_str?#1}
          \__stex_refs_do_autoref:n{\l__stex_refs_uri_str?#1}
        }{
          \stex_debug:nn{sref}{in~non-empty;~srefin~on~\l__stex_refs_uri_str?#1}
          \__stex_refs_set_keys_b:n{ #2 }
          \__stex_refs_do_sref_in:n{#1}
        }
      }
    }{
      \stex_debug:nn{sref}{No~ref~file~found~for~\l__stex_refs_uri_str}
      \str_if_empty:NTF \l_stex_key_file_str {
        \stex_debug:nn{sref}{in~empty;~autoref~on~\l__stex_refs_uri_str?#1}
        \__stex_refs_do_autoref:n{\l__stex_refs_uri_str?#1}
      }{
        \stex_debug:nn{sref}{in~non-empty;~srefin~on~\l__stex_refs_uri_str?#1}
        \__stex_refs_set_keys_b:n{ #2 }
        \__stex_refs_do_sref_in:n{#1}
      }
    }
  }
}

\cs_new_protected:Nn \__stex_refs_do_sref_in:n {
  \stex_debug:nn{sref}{In: \l_stex_key_file_str^^JRepo:\l_stex_key_archive_str}
  \stex_debug:nn{sref}{URI: \l__stex_refs_uri_str?#1}
  \tl_if_exist:cTF{r@sref@\l__stex_refs_uri_str?#1}{
    \__stex_refs_do_autoref:n{\l__stex_refs_uri_str?#1}
  }{
    %\msg_warning:nnn{stex}{warning/smsmissing}{<filename>}
    \group_begin:\catcode13=9\relax\catcode10=9\relax
      \str_if_empty:NTF \l_stex_key_archive_str {
        \prop_if_exist:NTF \l_stex_current_archive_prop {
          \str_set:Nx \l__stex_refs_file_str {
            \stex_file_use:N \c_stex_mathhub_file /
            \prop_item:Nn \l_stex_current_archive_prop { id }
            / source / \l_stex_key_file_str .sref
          }
        }{
          \str_set:Nx \l__stex_refs_file_str {
            \stex_file_use:N \g_stex_current_file / .. / \l_stex_key_file_str . sref
          }
        }
      }{
        \str_set:Nx \l__stex_refs_file_str {
          \stex_file_use:N \c_stex_mathhub_file / \l_stex_key_archive_str
          / source / \l_stex_key_file_str . sref
        }
      }
```

```
      \stex_file_resolve:No \l__stex_refs_file \l__stex_refs_file_str
      \str_set:Nx \l__stex_refs_file_str {\stex_file_use:N \l__stex_refs_file }
      \stex_debug:nn{sref}{File: \l__stex_refs_file_str }
      \exp_args:NNNx \exp_args:No \str_if_eq:nnTF \l__stex_refs_file_str {\stex_file_use:N\c
        \__stex_refs_do_autoref:n{
          \str_if_empty:NF\l__stex_refs_uri_str{\l__stex_refs_uri_str?}#1
        }
      }{
        \exp_args:No \IfFileExists \l__stex_refs_file_str {
          \tl_clear:N \l__stex_refs_return_tl
          \str_set:Nn \l__stex_refs_id_str {#1}
          \let\STEXInternalSrefRestoreTarget\__stex_refs_restore_target:nnnnn
          \use:c{@ @ input}{\l__stex_refs_file_str}
          \exp_args:No \tl_if_empty:nTF \l__stex_refs_return_tl {
            \exp_args:Nnno \msg_warning:nnnn{stex}{warning/smslabelmissing}\l__stex_refs_fil
            \__stex_refs_do_autoref:n{
              \str_if_empty:NF\l__stex_refs_uri_str{\l__stex_refs_uri_str?}#1
            }
          }{
            \l__stex_refs_return_tl
          }
        }{
          \exp_args:Nnno \msg_warning:nnn{stex}{warning/smsmissing}\l__stex_refs_file_str
          \__stex_refs_do_autoref:n{
            \str_if_empty:NF\l__stex_refs_uri_str{\l__stex_refs_uri_str?}#1
          }
        }
      }
    \group_end:
  }
}

\cs_new_protected:Nn \__stex_refs_do_return:nnnn {
  \tl_set:Nn \l__stex_refs_return_tl {
    \stex_annotate:nn{shtml:sref={#4},shtml:srefin={\l__stex_refs_file_str}}{
      \use:c{#3autorefname}~#1\tl_if_empty:nF{#2}{~(#2)}
      \tl_if_empty:NF\l_stex_key_title_tl{
        {}~in~\l_stex_key_title_tl
      }
    }
  }
}

\cs_new_protected:Nn \__stex_refs_restore_target:nnnnn {
  \str_if_empty:NTF \l__stex_refs_uri_str {
    \exp_args:No \str_if_eq:nnT \l__stex_refs_id_str {#2}{
      \__stex_refs_do_return:nnnn{#4}{#5}{#3}{#1?#2}
    }
  }{
    \stex_debug:nn{sref}{\l__stex_refs_uri_str{}~ == ~ #1 ~ ?}
    \exp_args:No \str_if_eq:nnT \l__stex_refs_uri_str {#1}{
      \stex_debug:nn{sref}{\l__stex_refs_id_str~ == ~ #2 ~ ?}
      \exp_args:No \str_if_eq:nnT \l__stex_refs_id_str {#2}{
        \stex_debug:nn{sref}{success!}
```

```
        \__stex_refs_do_return:nnnn{#4}{#5}{#3}{#1?#2}
        \endinput
      }
    }
  }
}
```

The actual macros:

```
\NewDocumentCommand \sref { O{} m O{}}{
  \stex_keys_set:nn { sref / 1 }{ #1 }
  \__stex_refs_find_uri:n { #2 }
  \__stex_refs_do_sref:nn{#2}{#3}
}
\NewDocumentCommand \extref { O{} m m}{
  \stex_keys_set:nn { sref / 1 }{ #1 }
  \__stex_refs_find_uri:n { #2 }
  \__stex_refs_set_keys_b:n{ #3 }
  \str_if_empty:NT \l_stex_key_file_str {
    \msg_error:nn{stex}{error/extrefmissing}
  }
  \__stex_refs_do_sref_in:n{#2}
}
```

(*End of definition for* `\sref` *and* `\extref`*. These functions are documented on page 84.*)

`\stex_ref_new_sym_target:n`

```
\cs_new_protected:Nn \stex_ref_new_symbol:n {
  \cs_if_exist:cF{r@sref@sym@\tl_to_str:n{#1}}{
    \__stex_refs_new_symbol:n{#1}
  }
}

\cs_new_protected:Nn \_stex_sref_do_aux:n {
  #1 \iow_now:Nn \@auxout {#1}
}

\cs_new_protected:Nn \__stex_refs_new_symbol:n {
  \prop_if_exist:NTF \l_stex_current_archive_prop {
    \prop_get:NnNTF \l_stex_current_archive_prop {docurl} \l__stex_refs_str {
      \exp_args:Ne \_stex_sref_do_aux:n {
        \STEXInternalSetSrefSymURL{#1}{\l__stex_refs_str / symbol? #1}
      }
    }{
      \_stex_sref_do_aux:n { \STEXInternalSetSrefSymURL{#1}{} }
    }
  }{
    \_stex_sref_do_aux:n { \STEXInternalSetSrefSymURL{#1}{} }
  }
}

\cs_new_protected:Nn \stex_ref_new_sym_target:n {
  \cs_if_exist:NT \hypertarget{
    \exp_args:Ne \hypertarget{\tl_to_str:n{sref@sym@ #1}}{}
    \str_gset:cx{\tl_to_str:n{r@sref@sym@ #1}}{\tl_to_str:n{sref@sym@ #1}}
  }
```

```
}

\cs_new_protected:Nn \stex_ref_new_sym_target:nn {
  \str_if_eq:nnTF{#1}{#2}{
    \stex_ref_new_sym_target:n{#1}
  }{
    \str_gset:cn{g_stex_sref_sym_ #1 _label}{#2}
  }
}
```

(*End of definition for* `\stex_ref_new_sym_target:n`. *This function is documented on page* 121.)

\srefsym

```
\NewDocumentCommand \srefsym { m m}{
  \stex_get_symbol:n { #1 }
  \exp_args:Ne
  \__stex_refs_sym_aux:nn{\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
}

\cs_new_protected:Npn \__stex_refs_do_internal_link:nn #1 {
  \cs_if_exist:NTF \hyperlink {
    \hyperlink{#1}
  }\use:n
}

\cs_new_protected:Npn \__stex_refs_do_url_link:nn {
  \cs_if_exist:NTF \href \href \use_ii:nn
}

\cs_new_protected:Npn \__stex_refs_sym_aux:nn #1 {
  \cs_if_exist:cTF{\tl_to_str:n{r@sref@sym@#1}}{
    \exp_args:Ne \__stex_refs_do_internal_link:nn{\tl_to_str:n{sref@sym@#1}}
  }{
    \str_if_exist:cTF{g_stex_sref_sym_#1_label}{
      \exp_args:Ne \__stex_refs_sym_aux:nn{\use:c{g_stex_sref_sym_#1_label}}
    }{
      \str_if_empty:cTF{g_stex_sref_sym_#1_target}{
        \exp_args:Ne \__stex_refs_do_internal_link:nn{\tl_to_str:n{sref@sym@#1}}
      }{
        \exp_args:Ne \__stex_refs_do_url_link:nn{\use:c{g_stex_sref_sym_#1_target}}
      }
    }
  }
}
```

(*End of definition for* `\srefsym`. *This function is documented on page* 89.)

\srefsymuri

```
\cs_new_protected:Npn \srefsymuri #1 {
  \__stex_refs_sym_aux:nn{#1}
}
```

(*End of definition for* `\srefsymuri`. *This function is documented on page* 89.)

13.4.4 Inputs

⟨@@=stex_inputs⟩

\stex_resolve_path_pair:Nnn
\stex_resolve_path_pair:Nxx

```
\cs_new_protected:Nn \stex_resolve_path_pair:Nnn {
  \stex_debug:nn{resolving~path}{#3~in~[#2]}
  \str_if_empty:nTF{#2}{
    \prop_if_exist:NTF \l_stex_current_archive_prop {
      \str_set:Nx #1 {\stex_file_use:N \c_stex_mathhub_file /
      \prop_item:Nn \l_stex_current_archive_prop { id } / source /
      #3}
      \stex_debug:nn{resolving~path}{In~current~archive~
        \prop_item:Nn \l_stex_current_archive_prop { id }
        ;~result:~#1}
    }{
      \str_set:Nx #1 {\stex_file_use:N \c_stex_pwd_file / .. / #3 }
      \stex_debug:nn{resolving~path}{No~current~archive;~result:~#1}
    }
  }{
    \stex_require_archive:n { #2 }
    \str_set:Nx #1 {\stex_file_use:N \c_stex_mathhub_file /
    \prop_item:cn {c_stex_mathhub_#2_manifest_prop} { id } / source /
    #3}
    \stex_debug:nn{resolving~path}{result:~#1}
  }
}
\cs_generate_variant:Nn \stex_resolve_path_pair:Nnn {Nxx}
```

(*End of definition for* `\stex_resolve_path_pair:Nnn`. *This function is documented on page* 134.)

\inputref
\mhinput
\ifinputref

```
\newif \ifinputref \inputreffalse

\cs_new_protected:Nn \__stex_inputs_mhinput:nn {
  \stex_in_archive:nn {#1} {
    \ifinputref
      \stex_input_with_hooks:n{ \stex_file_use:N \c_stex_mathhub_file / ##1 / source / #2 }
    \else
      \inputreftrue
        \stex_input_with_hooks:n{ \stex_file_use:N \c_stex_mathhub_file / ##1 / source / #2
      \inputreffalse
    \fi
  }
}

\NewDocumentCommand \mhinput { O{} m}{
  \exp_args:NNx\exp_args:Nnx\__stex_inputs_mhinput:nn{ \tl_to_str:n{#1} }{ \tl_to_str:n{#2}
}

\cs_new_protected:Nn \__stex_inputs_inputref_html:nn {
  \str_clear:N \l_tmpa_str
  \prop_get:NnNF \l_stex_current_archive_prop { narr } \l_tmpa_str {
    \prop_get:NnNF \l_stex_current_archive_prop { ns } \l_tmpa_str {}
  }
```

```
  \tl_if_empty:nTF{ #1 }{
    \IfFileExists{#2}{
      \ifvmode\noindent\fi\stex_annotate_invisible:nn{shtml:inputref={
        \l_tmpa_str / #2
      }}{}
    }{
      \stex_input_with_hooks:n{#2}
    }
  }{
    \IfFileExists{ \stex_file_use:N \c_stex_mathhub_file / #1 / source / #2 }{
      \ifvmode\noindent\fi\stex_annotate_invisible:nn{shtml:inputref={
        \l_tmpa_str / #2
      }}{}
    }{
      \input{ \stex_file_use:N \c_stex_mathhub_file / #1 / source / #2 }
    }
  }
}

\cs_new_protected:Nn \__stex_inputs_inputref_pdf:nn {
  \begingroup
    \inputreftrue
    \tl_if_empty:nTF{ #1 }{
      \stex_input_with_hooks:n{#2}
    }{
      \stex_input_with_hooks:n{ \stex_file_use:N \c_stex_mathhub_file / #1 / source / #2 }
    }
  \endgroup
}

\cs_new_protected:Nn \__stex_inputs_inputref:nn {
  \stex_in_archive:nn {#1} {
    \stex_if_html_backend:TF
      \__stex_inputs_inputref_html:nn
      \__stex_inputs_inputref_pdf:nn
    {##1}{#2}
  }
}

\NewDocumentCommand \inputref { O{} m}{
  \exp_args:NNx \exp_args:Nnx \__stex_inputs_inputref:nn{ \tl_to_str:n{#1} }{ \tl_to_str:n{#
}
```

(*End of definition for* `\inputref`, `\mhinput`, *and* `\ifinputref`. *These functions are documented on page* 83.)

`\addmhbibresource`

```
\cs_new_protected:Nn \__stex_inputs_bibresource:n {
  \__stex_inputs_up_archive:nn{#1}{bib}
  \seq_if_empty:NTF \l__stex_inputs_libinput_files_seq {
    \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\addmhbibresource}{#1.bib}
  }{
    \seq_map_inline:Nn \l__stex_inputs_libinput_files_seq {
      \addbibresource{ ##1 }
```

```
    }
  }
}
\newcommand\addmhbibresource[2][]{
  \tl_if_empty:nTF{#1}{
    \__stex_inputs_bibresource:n{#2}
  }{
    \stex_in_archive:nn{#1}{\__stex_inputs_bibresource:n{#2}}
  }
}
```

(*End of definition for* `\addmhbibresource`. *This function is documented on page 80.*)

`\IfInputref`

```
\stex_if_html_backend:TF{
  \newcommand \IfInputref[2]{
    \stex_annotate:nn{shtml:ifinputref=true}{#1}
    \stex_annotate:nn{shtml:ifinputref=false}{#2}
  }
}{
  \newcommand \IfInputref[2]{
    \ifinputref #1 \else #2 \fi
  }
}
```

(*End of definition for* `\IfInputref`. *This function is documented on page 83.*)

`\libinput`

```
\cs_new_protected:Nn \__stex_inputs_up_archive:nn {
  \prop_if_exist:NF \l_stex_current_archive_prop {
    \msg_error:nnn{stex}{error/notinarchive}\libinput
  }
  \prop_get:NnNF \l_stex_current_archive_prop {id} \l__stex_inputs_id_str {
    \msg_error:nnn{stex}{error/notinarchive}\libinput
  }
  \seq_clear:N \l__stex_inputs_libinput_files_seq
  \seq_set_eq:NN \l__stex_inputs_path_seq \c_stex_mathhub_file
  \seq_set_split:NnV \l__stex_inputs_id_seq / \l__stex_inputs_id_str

  \bool_while_do:nn { ! \seq_if_empty_p:N \l__stex_inputs_id_seq }{
    \str_set:Nx \l__stex_inputs_path_str {\stex_file_use:N \l__stex_inputs_path_seq / meta-i
    \IfFileExists{ \l__stex_inputs_path_str }{
      \exp_args:NNo \seq_if_in:NnF \l__stex_inputs_libinput_files_seq \l__stex_inputs_path_s
        \seq_put_right:No \l__stex_inputs_libinput_files_seq \l__stex_inputs_path_str
      }
    }{}
    \seq_pop_left:NN \l__stex_inputs_id_seq \l__stex_inputs_path_str
    \seq_put_right:No \l__stex_inputs_path_seq \l__stex_inputs_path_str
  }

  \str_set:Nx \l__stex_inputs_path_str {\stex_file_use:N \l__stex_inputs_path_seq / lib / #1
  \IfFileExists{ \l__stex_inputs_path_str }{
    \exp_args:NNo \seq_if_in:NnF \l__stex_inputs_libinput_files_seq \l__stex_inputs_path_str
      \seq_put_right:No \l__stex_inputs_libinput_files_seq \l__stex_inputs_path_str
    }
```

```
  }{}
}

\cs_new_protected:Nn \__stex_inputs_libinput:n {
  \__stex_inputs_up_archive:nn{#1}{tex}
  \seq_if_empty:NTF \l__stex_inputs_libinput_files_seq {
    \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
  }{
    \seq_map_inline:Nn \l__stex_inputs_libinput_files_seq {
      \input{ ##1 }
    }
  }
}

\newcommand \libinput [2][] {
  \tl_if_empty:nTF{#1}{
    \__stex_inputs_libinput:n{#2}
  }{
    \stex_in_archive:nn{#1}{\__stex_inputs_libinput:n{#2}}
  }
}
```

(*End of definition for* `\libinput`. *This function is documented on page 80.*)

`\libusepackage`

```
\newcommand\libusepackage[2][]{
  \__stex_inputs_up_archive:nn{#2}{sty}
  \int_compare:nNnTF {\seq_count:N \l__stex_inputs_libinput_files_seq} = 1 {
    \str_set:Nx \l__stex_inputs_tmp_str {\seq_item:Nn \l__stex_inputs_libinput_files_seq 1}
    \exp_args:Nne \use:n {\usepackage[#1]} {
      \str_range:Nnn\l__stex_inputs_tmp_str 1 {-5}
    }
  }{
    \stex_fatal_error:nnn{error/nofile}{\libusepackage}{#1.sty}
  }
}
```

(*End of definition for* `\libusepackage`. *This function is documented on page 80.*)

`\mhgraphics`
`\cmhgraphics`
`\lstinputmhlisting`
`\clstinputmhlisting`
`\mhtikzinput`
`\cmhtikzinput`

```
\str_new:N \l__stex_inputs_gin_repo_str
\ltx@ifpackageloaded{graphicx}{\use:n}{\AtEndOfPackageFile{graphicx}}{
  \define@key{Gin}{archive}{
    \tl_set:Nx\Gin@mhrepos{\stex_file_use:N \c_stex_mathhub_file / #1 / source /}
  }
  \providecommand\mhgraphics[2][]{
    \tl_set:Nx\Gin@mhrepos{
      \stex_file_use:N \c_stex_mathhub_file / \prop_item:Nn \l_stex_current_archive_prop {id
    }
    \setkeys{Gin}{#1}
    \includegraphics[#1]{ \Gin@mhrepos #2 }
  }
  \providecommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
}
```

```
\ltx@ifpackageloaded{listings}{\use:n}{\AtEndOfPackageFile{listings}}{
  \define@key{lst}{archive}{
    \def\lst@mhrepos{\stex_file_use:N \c_stex_mathhub_file / #1 / source /}
  }
  \newcommand\lstinputmhlisting[2][]{%
    \def\lst@mhrepos{
      \stex_file_use:N \c_stex_mathhub_file / \prop_item:Nn \l_stex_current_archive_prop {id
    }
    \setkeys{lst}{#1}%
    \lstinputlisting[#1]{\lst@mhrepos #2}}
  \newcommand\clstinputmhlisting[2][]{\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
}

\ltx@ifpackageloaded{tikzinput}{\use:n}{\AtEndOfPackageFile{tikzinput}}{
  \define@key{Gin}{archive}{
    \str_set:Nn \l__stex_inputs_gin_repo_str {#1}
    \def\Gin@mhrepos{\stex_file_use:N \c_stex_mathhub_file / #1 / source /}
  }
  \newcommand\mhtikzinput[2][]{%
    \str_clear:N \l__stex_inputs_gin_repo_str
    \def\Gin@mhrepos{
      \stex_file_use:N \c_stex_mathhub_file / \prop_item:Nn \l_stex_current_archive_prop {id
    }
    \setkeys{Gin}{#1}%
    \exp_args:No \stex_in_archive:nn \l__stex_inputs_gin_repo_str {
      \tikzinput[#1]{\Gin@mhrepos #2}
    }
  }
  \newcommand\cmhtikzinput[2][]{\begin{center}\mhtikzinput[#1]{#2}\end{center}}
}
```

(*End of definition for* `\mhgraphics` *and others. These functions are documented on page* 83.)

`\libusetikzlibrary`

```
\cs_new_protected:Nn \__stex_inputs_usetikzlibrary_i:nn {
  \pgfkeys@spdef\pgf@temp{#1}
  \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
  \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgfut
  \expandafter\edef\csname tikz@library@#1@atcode\endcsname{\the\catcode`\@}
  \expandafter\edef\csname tikz@library@#1@barcode\endcsname{\the\catcode`\|}
  \expandafter\edef\csname tikz@library@#1@dollarcode\endcsname{\the\catcode`\$}
  \catcode`\@=11
  \catcode`\|=12
  \catcode`\$=3
  \pgfutil@InputIfFileExists{#2}{}{}
  \catcode`\@=\csname tikz@library@#1@atcode\endcsname
  \catcode`\|=\csname tikz@library@#1@barcode\endcsname
  \catcode`\$=\csname tikz@library@#1@dollarcode\endcsname
}

\cs_new_protected:Nn \__stex_inputs_usetikzlibrary:n{
  \__stex_inputs_up_archive:nn{tikzlibrary#1}{code.tex}
  \int_compare:nNnTF {\seq_count:N \l__stex_inputs_libinput_files_seq} = 1 {
    \exp_args:Nne \__stex_inputs_usetikzlibrary_i:nn{#1}{ \seq_item:Nn \l__stex_inputs_libin
```

```
  }{
    \stex_fatal_error:nnn{error/nofile}{\libusetikzlibrary}{tikzlibrary#1.code.tex}
  }
}

\newcommand \libusetikzlibrary [2][] {
  \cs_if_exist:NF \usetikzlibrary {
    \msg_error:nnx{stex}{error/notikz}{\tl_to_str:n{\libusetikzlibrary}}
  }
  \tl_if_empty:nTF{#1}{
    \__stex_inputs_usetikzlibrary:n{#2}
  }{
    \stex_in_archive:nn{#1}{\__stex_inputs_usetikzlibrary:n{#2}}
  }
}
```

(*End of definition for* `\libusetikzlibrary`. *This function is documented on page* 118.)

13.5 SMS Mode

⟨`@@=stex_smsmode`⟩

Macros and environments allowed in sms mode:

```
\tl_new:N \g__stex_smsmode_allowed_tl
\tl_new:N \g__stex_smsmode_allowed_escape_tl
\seq_new:N \g__stex_smsmode_allowedenvs_seq
```

`\stex_sms_allow:N`
`\stex_sms_allow_escape:N`
`\stex_sms_allow_env:n`

```
\cs_new_protected:Nn \stex_sms_allow:N {
  \tl_gput_right:Nn \g__stex_smsmode_allowed_tl {#1}
}

\cs_new_protected:Nn \stex_sms_allow_escape:N {
  \tl_gput_right:Nn \g__stex_smsmode_allowed_escape_tl {#1}
}

\cs_new_protected:Nn \stex_sms_allow_env:n {
  \exp_args:NNx \seq_gput_right:Nn \g__stex_smsmode_allowedenvs_seq {\tl_to_str:n{#1}}
}
```

(*End of definition for* `\stex_sms_allow:N`, `\stex_sms_allow_escape:N`, *and* `\stex_sms_allow_env:n`. *These functions are documented on page* 135.)

Some initial allowed macros:

```
\stex_sms_allow:N \makeatletter
\stex_sms_allow:N \makeatother
\stex_sms_allow:N \ExplSyntaxOn
\stex_sms_allow:N \ExplSyntaxOff
\stex_sms_allow:N \rustexBREAK
```

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

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

(*End of definition for* `\stex_if_smsmode:TF`. *This function is documented on page 135.*)

`\stex_sms_allow_import:Nn`
`\stex_sms_allow_import_env:nn`

```
\tl_new:N \g__stex_smsmode_allowed_import_tl
\seq_new:N \g__stex_smsmode_allowed_import_env_seq
\cs_new_protected:Nn \stex_sms_allow_import:Nn {
  \tl_gput_right:Nn \g__stex_smsmode_allowed_import_tl {#1}
  \tl_gset:cn{\tl_to_str:n{#1}~-~smsmode} {#2}
}
\cs_new_protected:Nn \stex_sms_allow_import_env:nn {
  \exp_args:NNx \seq_gput_right:Nn \g__stex_smsmode_allowed_import_env_seq {\tl_to_str:n{#1}
  \tl_gset:cn{\tl_to_str:n{#1}~-~env~-~smsmode} {#2}
}

\tl_new:N \g_stex_sms_import_code
```

(*End of definition for* `\stex_sms_allow_import:Nn` *and* `\stex_sms_allow_import_env:nn`. *These functions are documented on page 136.*)

`\stex_file_in_smsmode:nn`
`\stex_file_in_smsmode:on`

```
\cs_new_protected:Nn \__stex_smsmode_in_smsmode:n { \stex_suppress_html:n {
  \vbox_set:Nn \l_tmpa_box {
    \bool_set_true:N \g__stex_smsmode_bool
    \bool_set_false:N \_stex_html_do_output_bool
    #1
  }
  %\box_clear:N \l_tmpa_box
} }

\quark_new:N \q__stex_smsmode_break

\cs_new_protected:Nn \__stex_smsmode_start_smsmode:n {
  \everyeof{\q__stex_smsmode_break\exp_not:N}
  \let\stex_smsmode_do:\__stex_smsmode_smsmode_do:
  \exp_after:wN \exp_after:wN \exp_after:wN
  \stex_smsmode_do:
  \cs:w @ @ input\cs_end: "#1" \relax
}

\cs_new_protected:Nn \stex_file_in_smsmode:nn {
  \seq_gclear:N \l__stex_smsmode_importmodules_seq
  \seq_gclear:N \l__stex_smsmode_sigmodules_seq
  \tl_clear:N \g_stex_sms_import_code
  \group_begin:
    \let \l_stex_metatheory_uri \c_stex_default_metatheory
    \cs_set:Npn \stex_check_term:n ##1 {}
    \seq_clear:N \l_stex_all_modules_seq
    \str_clear:N \l_stex_current_module_str
    #2
    \stex_filestack_push:n{#1}
    \__stex_smsmode_in_smsmode:n {
      \let \__stex_smsmode_do_aux_curr:N \__stex_smsmode_do_aux_imports:N
      \tl_map_inline:Nn \g__stex_smsmode_allowed_import_tl {
        \use:c{\tl_to_str:n{##1}~-~smsmode}
```

```
      }
      \seq_map_inline:Nn \g__stex_smsmode_allowed_import_env_seq {
        \use:c{\tl_to_str:n{##1}~~~env~~~smsmode}
      }
      \__stex_smsmode_start_smsmode:n{#1}
    }
    \__stex_smsmode_in_smsmode:n \g_stex_sms_import_code
    \__stex_smsmode_in_smsmode:n {
      \let \__stex_smsmode_do_aux_curr:N \__stex_smsmode_do_aux_normal:N
      \__stex_smsmode_start_smsmode:n{#1}
    }
    \stex_filestack_pop:
  \group_end:
}
\cs_generate_variant:Nn \stex_file_in_smsmode:nn {on}
```

(*End of definition for* `\stex_file_in_smsmode:nn`. *This function is documented on page* 135.)

`\stex_smsmode_do:`

```
\cs_new_protected:Nn \__stex_smsmode_smsmode_do: {
 %\stex_if_smsmode:T {
    \__stex_smsmode_do:w
  %}
}
\let\stex_smsmode_do:\relax


\cs_new:Nn \__stex_smsmode_check_cs:NNn {
  \exp_after:wN\if\exp_after:wN\relax\exp_not:N#3
  \exp_after:wN#1\exp_after:wN#3\else
  \exp_after:wN#2\fi
}

\cs_new_protected:Npn \__stex_smsmode_do:w #1 {
  \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 }}{
    \__stex_smsmode_check_cs:NNn \__stex_smsmode_do_aux:N \__stex_smsmode_do:w { #1 }
  }{
    \__stex_smsmode_do:w
  }
}

\cs_new_protected:Nn \__stex_smsmode_do_aux:N {
  \cs_if_eq:NNF #1 \q__stex_smsmode_break {
    \__stex_smsmode_do_aux_curr:N #1
  }
}

\cs_new_protected:Nn \__stex_smsmode_do_aux_imports:N {
 % \stex_debug:nn{sms}{Checking~\tl_to_str:n{#1}~in~import}
  \tl_if_in:NnTF \g__stex_smsmode_allowed_import_tl {#1} {
    \stex_debug:nn{sms}{Executing~\tl_to_str:n{#1}~in~import}
    #1
  }{
    \cs_if_eq:NNTF \begin #1 {
```

```
      \__stex_smsmode_check_begin:Nn \g__stex_smsmode_allowed_import_env_seq
    }{
      \cs_if_eq:NNTF \end #1 {
        \__stex_smsmode_check_end:Nn \g__stex_smsmode_allowed_import_env_seq
      }{
        \__stex_smsmode_do:w
      }
    }
  }
}

\cs_new_protected:Nn \__stex_smsmode_do_aux_normal:N {
 % \stex_debug:nn{sms}{Checking~\tl_to_str:n{#1}~in~sms~mode}
  \tl_if_in:NnTF \g__stex_smsmode_allowed_tl {#1} {
    \stex_debug:nn{sms}{Executing~\tl_to_str:n{#1}}
    #1\__stex_smsmode_do:w
  }{
    \tl_if_in:NnTF \g__stex_smsmode_allowed_escape_tl {#1} {
      \stex_debug:nn{sms}{Executing~escaped~\tl_to_str:n{#1}}
      #1
    }{
      \cs_if_eq:NNTF \begin #1 {
        \__stex_smsmode_check_begin:Nn \g__stex_smsmode_allowedenvs_seq
      }{
        \cs_if_eq:NNTF \end #1 {
          \__stex_smsmode_check_end:Nn \g__stex_smsmode_allowedenvs_seq
        }{
          \__stex_smsmode_do:w
        }
      }
    }
  }
}

\cs_new_protected:Nn \__stex_smsmode_check_begin:Nn {
 % \stex_debug:nn{sms}{Checking~environment~#2}
  \seq_if_in:NxTF #1 { \tl_to_str:n{#2} }{
    \stex_debug:nn{sms}{Environment~#2}
    \begin{#2}
  }{
    \__stex_smsmode_do:w
  }
}
\cs_new_protected:Nn \__stex_smsmode_check_end:Nn {
%  \stex_debug:nn{sms}{Checking~end~environment~#2}
  \seq_if_in:NxTF #1 { \tl_to_str:n{#2} }{
    \stex_debug:nn{sms}{End~Environment~#2}
    \end{#2}\__stex_smsmode_do:w
  }{
    %\str_if_eq:nnTF{#2}{document} \endinput
    \__stex_smsmode_do:w
  }
}
```

(*End of definition for* `\stex_smsmode_do:`. *This function is documented on page* 136.)

13.6 Modules

13.6.1 The smodule-environment

⟨@@=stex_modules⟩

\l_stex_current_module_str

The current module:

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

(*End of definition for* `\l_stex_current_module_str`. *This variable is documented on page* 122.)

\l_stex_all_modules_seq

Stores all modules currently in scope

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

(*End of definition for* `\l_stex_all_modules_seq`. *This variable is documented on page* 122.)

\stex_every_module:n

```
⟨@@=stex_module_setup⟩
\tl_clear:N \g_stex_every_module_tl {
}
\cs_new_protected:Nn \stex_every_module:n {
  \tl_gput_right:Nn \g_stex_every_module_tl { #1 }
}
```

(*End of definition for* `\stex_every_module:n`. *This function is documented on page* 122.)

\stex_module_setup:n

Sets up a new module:

```
\cs_new_protected:Npn \stex_module_setup:n {
  \stex_if_in_module:TF \__stex_module_setup_setup_nested:n \__stex_module_setup_setup_top:n
}

\cs_new_protected:Nn \__stex_module_setup_setup_top:n {
  \__stex_module_setup_get_uri_str:n{#1}
  \stex_debug:nn{module}{Module~URI:~\l__stex_module_setup_ns_str?#1}
  \str_if_empty:NTF \l_stex_key_sig_str
  \_stex_module_setup_top_nosig:n \__stex_module_setup_setup_top_sig:n {\l__stex_module_setu
  \stex_metagroup_new:o \l_stex_current_module_str
  \g_stex_every_module_tl
  \stex_execute_in_module:x {
    \_stex_do_deprecation:n{#1}
  }
  \__stex_module_setup_load_meta:
}

\cs_new_protected:Nn \_stex_module_setup_top_nosig:n {
  \stex_if_module_exists:nTF{#1}{
    \stex_debug:nn{modules}{(already exists)}
  }{
    \tl_gclear:c{c_stex_module_ #1 _code}
    \prop_gclear:c{c_stex_module_ #1 _morphisms_prop }
    \prop_gclear:c{c_stex_module_ #1 _symbols_prop }
    \prop_gclear:c{c_stex_module_ #1 _notations_prop }
  }
  \str_set:Nx \l_stex_current_module_str {#1}
  \seq_put_right:No \l_stex_all_modules_seq \l_stex_current_module_str
```

```
}

\cs_new_protected:Nn \__stex_module_setup_setup_top_sig:n {
  \stex_if_module_exists:nTF{#1}{
    \stex_debug:nn{modules}{(already exists)}
  }{
    \stex_debug:nn{modules}{(needs loading)}
    \__stex_module_setup_load_sig:
  }
  %\stex_if_smsmode:F { % WHY?
    \stex_activate_module:x {
      #1
    }
  %}
  \str_set:Nx\l_stex_current_module_str{#1}
}

\cs_new_protected:Nn \__stex_module_setup_load_sig: {
  \stex_file_split_off_ext:NN \l__stex_module_setup_sigfile \g_stex_current_file
  \stex_file_split_off_lang:NN \l__stex_module_setup_sigfile \l__stex_module_setup_sigfile
  \exp_args:Ne \stex_file_in_smsmode:nn {
    \stex_file_use:N \l__stex_module_setup_sigfile . \l_stex_key_sig_str . tex
  }{}
}

\cs_new_protected:Nn \__stex_module_setup_setup_nested:n {
  \exp_after:wN
    \__stex_module_setup_split_module:n \l_stex_current_module_str \__stex_module_setup_end:
  \stex_debug:nn{module}{Nested~Module~URI:~\l_stex_current_module_str}
  \seq_put_right:No \l_stex_all_modules_seq \l_stex_current_module_str
  \stex_metagroup_new:o \l_stex_current_module_str
}


\cs_new_protected:Nn \__stex_module_setup_get_uri_str:n {
  \str_clear:N \l__stex_module_setup_ns_str
  \stex_map_uri:Nnnnn \l_stex_current_ns_uri {
    \str_set:Nx \l__stex_module_setup_ns_str{##1\c_colon_str/}
  }{
    \seq_set_split:Nnn \l__stex_module_setup_seq / {##1}
    \seq_pop_right:NN \l__stex_module_setup_seq \l__stex_module_setup_seg
    \exp_args:No \str_if_eq:nnF \l__stex_module_setup_seg {#1} {
      \seq_put_right:No \l__stex_module_setup_seq \l__stex_module_setup_seg
    }
    \tl_put_right:Nx \l__stex_module_setup_ns_str {\seq_use:Nn \l__stex_module_setup_seq /}
  }{}{}
}

\cs_new_protected:Npn \__stex_module_setup_split_module:n #1?#2 \__stex_module_setup_end: #3
  \_stex_module_setup_top_nosig:n { #1 ? #2 / #3}
}

\bool_new:N \l_stex_in_meta_bool
\bool_set_false:N \l_stex_in_meta_bool
```

```
\cs_new_protected:Nn \__stex_module_setup_load_meta: {
  \tl_if_empty:NF \l_stex_metatheory_uri {
    \stex_execute_in_module:x{
      \stex_pseudogroup_with:nn{\l_stex_in_meta_bool}{
        \stex_activate_module:n {\stex_uri_use:N \l_stex_metatheory_uri }
      }
    }
  }
}

⟨@@=stex_modules⟩
```

(*End of definition for* `\stex_module_setup:n`*. This function is documented on page* 122.)

`\stex_close_module:`

```
\cs_new:Nn \stex_close_module: {
  \bool_if:NT \c_stex_persist_write_mode_bool \__stex_modules_persist_module:
  \stex_debug:nn{module}{
    Closing~module~\l_stex_current_module_str^^J
    Code:~\expandafter\meaning\csname c_stex_module_\l_stex_current_module_str _code\endcsna
    Imports:\exp_after:wN \prop_to_keyval:N \cs:w c_stex_module_\l_stex_current_module_str _
    Declarations:\exp_after:wN \prop_to_keyval:N \cs:w c_stex_module_\l_stex_current_module_
    Notations:\exp_after:wN \prop_to_keyval:N \cs:w c_stex_module_\l_stex_current_module_str
  }
}

\cs_new_protected:Nn \__stex_modules_persist_module: {
  \stex_persist:e {
    \__stex_modules_restore_module:nnnn {\l_stex_current_module_str}{
      \exp_after:wN \prop_to_keyval:N \cs:w
        c_stex_module_\l_stex_current_module_str _morphisms_prop
      \cs_end:
    }{
      \exp_after:wN \prop_to_keyval:N \cs:w
        c_stex_module_\l_stex_current_module_str _symbols_prop
      \cs_end:
    }{
      \exp_after:wN \exp_after:wN \exp_after:wN \exp_not:n
      \exp_after:wN \exp_after:wN \exp_after:wN
      { \cs:w c_stex_module_\l_stex_current_module_str _code \cs_end: }
    }{}
    \prop_map_function:cN{c_stex_module_\l_stex_current_module_str _notations_prop}
      \__stex_modules_persist_nots_i:nn
    \exp_not:N \STEXRestoreNotsEnd {}
  }
}

\cs_new_protected:Nn \__stex_modules_restore_module:nnnn {
  \prop_gset_from_keyval:cn{c_stex_module_\tl_to_str:n{#1}_morphisms_prop}{#2}
  \cs_set:Npn \__stex_modules_tl {#3}
  \exp_args:Nno \prop_gset_from_keyval:cn{c_stex_module_\tl_to_str:n{#1}_symbols_prop}\__ste
  \prop_map_inline:cn{c_stex_module_\tl_to_str:n{#1}_symbols_prop}{
    \stex_ref_new_symbol:n{#1?##1}
```

```
  }
  \cs_gset:cpn{c_stex_module_\tl_to_str:n{#1}_code}{#4}
  \prop_gclear:c{c_stex_module_\tl_to_str:n{#1} _notations_prop}
  \str_set:Nn \l__stex_modules_restore_mod_str {#1}
  \group_begin:
    \catcode`_=8\relax
    \catcode`:=12\relax
    \__stex_modules_restore_nots:n
}

\cs_new:Nn \__stex_modules_persist_nots_i:nn {
  \exp_not:n{#2}
}

\quark_new:N \STEXRestoreNotsEnd

\cs_new_protected:Nn \__stex_modules_restore_nots:n {
  \__stex_modules_restore_nots_i:n
}

\cs_new_protected:Nn \__stex_modules_restore_nots_i:n {
  \tl_if_eq:nnTF{#1}{\STEXRestoreNotsEnd}{
    \group_end:
  }{
    \__stex_modules_restore_nots_ii:nnnnn {#1}
  }
}

\cs_new_protected:Nn \__stex_modules_restore_nots_ii:nnnnn {
  \cs_set:Npn \l__stex_modules_tl {{#4}{#5}}
  \exp_args:NNe\use:nn\prop_gput:cnn{
    {c_stex_module_\l__stex_modules_restore_mod_str _notations_prop}
    {\tl_to_str:n{#1!#2}}{
      {\tl_to_str:n{#1}}{\tl_to_str:n{#2}}{#3}
      \exp_args:No \exp_not:n \l__stex_modules_tl
    }
  }
  \__stex_modules_restore_nots_i:n
}
```

(*End of definition for* `\stex_close_module:`. *This function is documented on page* 122.)

`\l_stex_metatheory_uri`

```
\tl_new:N \l_stex_metatheory_uri
```

(*End of definition for* `\l_stex_metatheory_uri`. *This variable is documented on page* **??**.)

`\setmetatheory`

```
\cs_new_protected:Nn \__stex_modules_set_metatheory:nn {
  \group_begin:
    \stex_debug:nn{metatheory}{Setting~metatheory~[#1]#2}
    \stex_import_module_uri:nn { #1 } { #2 }
    \stex_debug:nn{metatheory}{Here:^^J
      \l_stex_import_archive_str^^J
      \l_stex_import_path_str^^J
```

```
      \l_stex_import_name_str^^J
    }
    \stex_import_require_module:ooo
      \l_stex_import_archive_str
      \l_stex_import_path_str
      \l_stex_import_name_str
    \stex_debug:nn{metatheory}{Found:~\l_stex_import_ns_str}
    \exp_args:Nne \use:nn {
      \group_end: \stex_uri_resolve:Nn \l_stex_metatheory_uri
    }{{\l_stex_import_ns_str}}
}

\NewDocumentCommand \setmetatheory {O{} m}{
  \__stex_modules_set_metatheory:nn { #1 }{ #2 }
  \stex_smsmode_do:
}
\stex_sms_allow_escape:N \setmetatheory
```

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

Keys and key handling:

```
\stex_keys_define:nnnn{smodule}{
  \str_clear:N \l_stex_key_sig_str
}{
  meta          .code:n       = {
    \str_if_empty:nTF {#1}{
      \tl_clear:N \l_stex_metatheory_uri
    }{
      \stex_uri_resolve:Nx \l_stex_metatheory_uri { #1 }
    }
  },
  ns            .code:n       = {
    \stex_uri_resolve:Nx \l_stex_current_ns_uri { #1 }
  } ,
  lang          .code:n       = {
    \stex_set_language:n { #1 }
  } ,
  sig           .str_set_x:N  = \l_stex_key_sig_str ,
  creators      .code:n       = {} , % todo ?
  contributors  .code:n       = {} , % todo ?
  srccite       .code:n       = {}   % todo ?
}{id, title, style, deprecate}
```

`smodule` (*env.*)

```
\stex_new_stylable_env:nnnnnnn {module} {O{} m}{
  \stex_keys_set:nn { smodule }{ #1 }
  \tl_set_eq:NN \thistitle \l_stex_key_title_tl
  \tl_if_empty:NF \thistitle {
    \exp_args:No \stexdoctitle \thistitle
  }
  \exp_args:Nx \stex_module_setup:n { \tl_to_str:n{ #2 } }

  \stex_if_do_html:T {
    \exp_args:Nne \begin{stex_annotate_env} {
      shtml:theory={\l_stex_current_module_str},
```

```
      shtml:language={ \l_stex_current_language_str},
      shtml:signature={\l_stex_key_sig_str}
      \tl_if_empty:NF \l_stex_metatheory_uri {,
        shtml:metatheory={\stex_uri_use:N \l_stex_metatheory_uri}
      }
    }
    \stex_annotate_invisible:n{}
  }
  \stex_if_smsmode:F {
    \str_set_eq:NN \thismoduleuri \l_stex_current_module_str
    \tl_set:Nn \thismodulename {#2}
    \stex_style_apply:
  }
  \stex_smsmode_do:
}{
  \stex_close_module:
  \stex_if_smsmode:F \stex_style_apply:
  \stex_if_do_html:T{ \end{stex_annotate_env} }
}{}{}{s}

\stex_sms_allow_env:n{smodule}
```

`\stex_if_in_module_p:`
`\stex_if_in_module:TF`

Are we currently in a module?

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

(*End of definition for* `\stex_if_in_module:TF`*. This function is documented on page 122.*)

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

Does a module with this URI exist?

```
\prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
  \tl_if_exist:cTF { c_stex_module_#1_code }
    \prg_return_true: \prg_return_false:
}
```

(*End of definition for* `\stex_if_module_exists:nTF`*. This function is documented on page 122.*)

`\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)

```
\cs_new_protected:Nn \stex_do_up_to_module:n {
  \exp_args:No \stex_metagroup_do_in:nn \l_stex_current_module_str {#1}
}
\cs_generate_variant:Nn \stex_do_up_to_module:n {x}
```

(*End of definition for* `\stex_do_up_to_module:n`*. This function is documented on page 123.*)

`\stex_module_add_code:n`
`\stex_module_add_code:x`

```
\cs_new_protected:Nn \stex_module_add_code:n {
  \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
}
\cs_generate_variant:Nn \stex_module_add_code:n {x}
```

(*End of definition for* `\stex_module_add_code:n`*. This function is documented on page 123.*)

\stex_execute_in_module:n
\stex_execute_in_module:x

```
\cs_new_protected:Nn \stex_execute_in_module:n { \stex_if_in_module:TF {
  \stex_module_add_code:n { #1 }
  \stex_do_up_to_module:n { #1 }
}{ #1 }}
\cs_generate_variant:Nn \stex_execute_in_module:n {x}
```

(*End of definition for* `\stex_execute_in_module:n`. *This function is documented on page* 122.)

\STEXexport

```
\NewDocumentCommand \STEXexport {} {
  \ExplSyntaxOn
  \__stex_modules_export:n
}
\cs_new_protected:Nn \__stex_modules_export:n {
  \stex_ignore_spaces_and_pars:#1\ExplSyntaxOff
  \stex_module_add_code:n { \stex_ignore_spaces_and_pars:#1}
  \stex_smsmode_do:
}
```

Only allowed in modules, and allowed (escaped) in sms mode:

```
\stex_deactivate_macro:Nn \STEXexport {module~environments}
\stex_sms_allow_escape:N \STEXexport
\stex_every_module:n {\stex_reactivate_macro:N \STEXexport}
```

(*End of definition for* `\STEXexport`. *This function is documented on page* 86.)

\stex_module_add_morphism:nnnn
\stex_module_add_morphism:nonn
\stex_module_add_morphism:ooox

```
\cs_new_protected:Nn \stex_module_add_morphism:nnnn {
  \exp_args:Nne \prop_gput:cnn{c_stex_module_\l_stex_current_module_str _morphisms_prop}{
    \tl_if_empty:nTF{#1}{[#2]}{#1}
  }{{#1}{#2}{#3}{#4}}
}
\cs_generate_variant:Nn \stex_module_add_morphism:nnnn {nonn,ooox}
```

(*End of definition for* `\stex_module_add_morphism:nnnn`. *This function is documented on page* 130.)

\stex_module_add_symbol:nnnnnnN

#1 : {⟨*Macro name*⟩}
#2 : {⟨*Name*⟩}
#3 : {⟨*arity*⟩}
#4 : {({⟨*Arg num*⟩}{⟨*Arg str*⟩})*}
#5 : Definiens
#6 : type
#7 : Return
#8 : Command

```
\cs_new_protected:Nn \stex_module_add_symbol:nnnnnnnN {
  \stex_debug:nn{declaration}{New~declaration:~\l_stex_current_module_str?#2^^J
    Macro:#1^^JArity:#3~(#4)^^J
    Def:~\tl_to_str:n{#5}^^J
    Type:~\tl_to_str:n{#6}^^J
    Returns:~\tl_to_str:n{#7}
  }
  %\prop_gput:cnx{c_stex_module_\l_stex_current_module_str _symbols_prop}
  %{#2}{\exp_not:n{{#1}{#2}{#3}{#4}{#5}{#6}}{#7}\exp_not:n{#8}}
```

```
  \prop_gput:cnn{c_stex_module_\l_stex_current_module_str _symbols_prop}
  {#2}{{#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}}
  \tl_if_empty:nF{#1}{
    \stex_execute_in_module:n {
      \__stex_modules_activate_sym:n {#2}
    }
  }
}

\cs_new_protected:Nn \__stex_modules_activate_sym:n {
  \prop_map_inline:cn{c_stex_module_\l_stex_current_module_str _symbols_prop}{
    \str_if_eq:nnT{#1}{##1}{
      \__stex_modules_activate_i:nnnnnnnn ##2
    }
  }
}
\cs_new_protected:Nn \__stex_modules_activate_i:nnnnnnnn {
  \stex_debug:nn{activating}{#1:\l_stex_current_module_str^^J
    \tl_to_str:n{{#2}{#3}{#4}{#5}{#6}{#7}#8}
  }
  \cs_set:cpx{#1} {
    \_stex_invoke_symbol:nnnnnnnN
    {\l_stex_current_module_str}
    \exp_not:n{{#2}{#3}{#4}{#5}{#6}{#7}{#8}}
  }
  \stex_debug:nn{activating}{done}
  \prop_map_break:
}
```

(*End of definition for* `\stex_module_add_symbol:nnnnnnN`. *This function is documented on page* **??**.)

`\stex_module_add_notation:nnnnn`
`\stex_module_add_notation:eoeoo`
`\stex_set_notation_macro:nnnnn`

#1 : URI
#2 : variant
#3 : arity
#4 : macro body
#5 : op

```
\cs_new_protected:Nn \stex_module_add_notation:nnnnn {
  \stex_debug:nn{notations}{Adding~notation:^^J
    #1~\c_hash_str#2~#3^^J
    to~\l_stex_current_module_str
  }
  \prop_gput:cnn{c_stex_module_\l_stex_current_module_str _notations_prop}
  {#1!#2}{{#1}{#2}{#3}{#4}{#5}}
  \stex_execute_in_module:n {
    \__stex_modules_activate_not:nn{#1}{#2}
  }
}
\cs_generate_variant:Nn \stex_module_add_notation:nnnnn {eoeoo}


\cs_new_protected:Nn \__stex_modules_activate_not:nn {
  \prop_map_inline:cn{c_stex_module_\l_stex_current_module_str _notations_prop}{
    \str_if_eq:nnT{#1!#2}{##1}{
      \prop_map_break:n{\stex_set_notation_macro:nnnnn ##2 }
```

```
    }
  }
}
```

(*End of definition for* `\stex_module_add_notation:nnnnn` *and* `\stex_set_notation_macro:nnnnn`. *These functions are documented on page 125.*)

`\stex_set_notation_macro:nnnnn`
`\stex_set_notation_macro:eoexo`

```
\cs_new_protected:Nn \stex_set_notation_macro:nnnnn {
  \tl_set:cn {l_stex_notation_#1_#2_cs}{#4}
  \cs_if_exist:cF{l_stex_notation_#1__cs}{
    \tl_set:cn {l_stex_notation_#1__cs}{#4}
  }
  \tl_if_empty:nF{#5}{
    \tl_set:cn{l_stex_notation_#1_op_#2_cs}{#5}
    \cs_if_exist:cF{l_stex_notation_#1_op__cs}{
      \cs_set_eq:cc {l_stex_notation_#1_op__cs}{l_stex_notation_#1_op_#2_cs}
    }
  }
}
\cs_generate_variant:Nn \stex_set_notation_macro:nnnnn {eoexo}
```

(*End of definition for* `\stex_set_notation_macro:nnnnn`. *This function is documented on page 126.*)

`\stex_activate_module:n`
`\stex_activate_module:o`
`\stex_activate_module:x`

```
\cs_new_protected:Nn \stex_activate_module:n {
  \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
    \stex_debug:nn{modules}{Activating~module~#1^^J\expandafter\meaning\csname c_stex_module
    \seq_put_right:Nn \l_stex_all_modules_seq { #1 }
    \stex_pseudogroup:nn{
      \str_set:Nn \l_stex_current_module_str {#1}
      \use:c{ c_stex_module_#1_code }
    }{
      \stex_pseudogroup_restore:N \l_stex_current_module_str
    }
  }
}
\cs_generate_variant:Nn \stex_activate_module:n {o,x}
```

(*End of definition for* `\stex_activate_module:n`. *This function is documented on page 122.*)

Iterating:

```
⟨@@=stex_iterate⟩
```

`\stex_iterate_symbols:n`

```
\cs_new_protected:Nn \stex_iterate_symbols:n {
  \stex_pseudogroup_with:nn{\__stex_iterate_sym_cs:nnnnnnnnN\stex_iterate_break:\stex_iterat
    \cs_set:Npn \__stex_iterate_sym_cs:nnnnnnnnN
    ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 ##9 { #1 }
    \cs_set:Npn \stex_iterate_break: {
      \prop_map_break:n{\seq_map_break:}
    }
    \cs_set:Npn \stex_iterate_break:n ##1 {
      \prop_map_break:n{\seq_map_break:n{##1}}
    }
```

```
    \seq_map_inline:Nn \l_stex_all_modules_seq {
      \prop_map_inline:cn{c_stex_module_##1_symbols_prop}{
        \__stex_iterate_sym_cs:nnnnnnnnN {##1} ####2
      }
    }
  }
}
```

(*End of definition for* `\stex_iterate_symbols:n`. *This function is documented on page 123.*)

`\stex_iterate_symbols:nn`

```
\cs_new_protected:Nn \stex_iterate_symbols:nn {
  \seq_clear:N \l__stex_iterate_mods_seq
  \stex_pseudogroup_with:nn{\__stex_iterate_sym_cs:nnnnnnnnN}{
    \cs_set:Npn \__stex_iterate_sym_cs:nnnnnnnnN
    ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 ##9 { #2 }
    \clist_map_function:nN {#1} \__stex_iterate_it_decl_i:n
  }
}

\cs_new_protected:Nn \__stex_iterate_it_decl_i:n {
  \seq_if_in:NnF \l__stex_iterate_mods_seq {#1} {
    \seq_put_left:Nn \l__stex_iterate_mods_seq {#1}
    \prop_map_inline:cn{c_stex_module_#1_morphisms_prop}{
      \__stex_iterate_it_decl_check:nnnn ##2
    }
    \prop_map_inline:cn{c_stex_module_#1_symbols_prop}{
      \__stex_iterate_sym_cs:nnnnnnnnN {#1} ##2
    }
  }
}
\cs_new_protected:Nn \__stex_iterate_it_decl_check:nnnn {
  \tl_if_empty:nT{#1}{
    \__stex_iterate_it_decl_i:n {#2}
  }
}
```

(*End of definition for* `\stex_iterate_symbols:nn`. *This function is documented on page 123.*)

`\stex_iterate_notations:nn`

```
\cs_new_protected:Nn \stex_iterate_notations:nn {
  \seq_clear:N \l__stex_iterate_mods_seq
  \stex_pseudogroup_with:nn{\__stex_iterate_not_cs:nnnnn}{
    \cs_set:Npn \__stex_iterate_not_cs:nnnnn
    ##1 ##2 ##3 ##4 ##5 { #2 }
    \clist_map_function:nN {#1} \__stex_iterate_it_not_i:n
  }
}

\cs_new_protected:Nn \__stex_iterate_it_not_i:n {
  \seq_if_in:NnF \l__stex_iterate_mods_seq {#1} {
    \seq_put_left:Nn \l__stex_iterate_mods_seq {#1}
    \prop_map_inline:cn{c_stex_module_#1_notations_prop}{
      \__stex_iterate_not_cs:nnnnn ##2
    }
```

```
    \prop_map_inline:cn{c_stex_module_#1_morphisms_prop}{
      \__stex_iterate_it_not_check:nnnn ##2
    }
  }
}
\cs_new_protected:Nn \__stex_iterate_it_not_check:nnnn {
  \tl_if_empty:nT{#1}{
    \__stex_iterate_it_not_i:n {#2}
  }
}
```

(*End of definition for* `\stex_iterate_notations:nn`. *This function is documented on page* 126.)

`\stex_iterate_morphisms:nn`

```
\cs_new_protected:Nn \stex_iterate_morphisms:nn {
  \seq_clear:N \l__stex_iterate_mods_seq
  \bool_set_true:N \l__stex_iterate_continue_bool
  \cs_set:Npn \__stex_iterate_morphism_cs:nnnn ##1 ##2 ##3 ##4 ##5 {
    #2
    \bool_if:NT \l__stex_iterate_continue_bool {
      \str_if_eq:nnTF{##1}{[##2]}{
        \tl_put_right:Nn \l__stex_iterate_todo_tl {
          \__stex_iterate_iterate_morphism:nn{##5}{##2}
        }
      }{
        \tl_put_right:Nn \l__stex_iterate_todo_tl {
          \__stex_iterate_iterate_morphism:nn{##5 / ##1}{##2}
        }
      }
    }
  }
  \cs_set:Npn \stex_iterate_break:n ##1 {
    \bool_set_false:N \l__stex_iterate_continue_bool
    \prop_map_break:n{##1}
  }
  \__stex_iterate_iterate_morphism:nn{}{#1}
}

\cs_new_protected:Nn \__stex_iterate_iterate_morphism:nn {
  \tl_clear:N \l__stex_iterate_todo_tl
  \seq_if_in:NnF \l__stex_iterate_mods_seq {#1 #2}{
    \seq_put_right:Nn \l__stex_iterate_mods_seq {#1 #2}
    \prop_map_inline:cn{c_stex_module_#2_morphisms_prop}{
      \__stex_iterate_morphism_cs:nnnn ##2 {#1}
      % TODO
      % ##1: name or [mpath]
      % ##2 = {####1}{####2}{####3}{####4}
      % ####1 = name
      % ####2 = mpath
      % ####3 = type
      % ####4 = {origname}{newname}*
    }
    \bool_if:NT \l__stex_iterate_continue_bool \l__stex_iterate_todo_tl
  }
}
```

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

13.6.2 Structural Features

⟨@@=stex_features⟩

`\stex_structural_feature_module:nn`
`\stex_structural_feature_module_end:`

```
\cs_new_protected:Nn \stex_structural_feature_module:nn {
  \stex_if_do_html:TF {
    \exp_args:Nne \begin{stex_annotate_env} {
      shtml:feature-#2={
       \l_stex_current_module_str/#1}
      \str_if_empty:NF \l_stex_macroname_str {,
        shtml:macroname={\l_stex_macroname_str}
      }
    }
    \stex_annotate_invisible:n{}
  }\group_begin:
  \stex_module_setup:n {#1-module}
}

\cs_new_protected:Nn \stex_structural_feature_module_end: {
  \tl_gset_eq:NN \g_stex_last_feature_str \l_stex_current_module_str
  \stex_close_module:
  \stex_if_do_html:TF{
    \end{stex_annotate_env}
  }\group_end:
}
```

(*End of definition for* `\stex_structural_feature_module:nn` *and* `\stex_structural_feature_module_end:`. *These functions are documented on page* 128.)

`\stex_structural_feature_morphism:nnn`
`\stex_structural_feature_morphism_end:`

```
\bool_new:N \l__stex_features_implicit_bool
\cs_new_protected:Nn \stex_structural_feature_morphism:nnnnn {
  \str_clear:N \l_stex_current_domain_str
  \tl_if_empty:nT{#3}{
    \_stex_get_mathstructure:n{#4}
    \str_set_eq:NN \l_stex_current_domain_str \l_stex_get_structure_module_str
  }
  \str_if_empty:NT \l_stex_current_domain_str {
    \stex_import_module_uri:nn { #3 }{ #4 }
    \group_begin:
    \stex_import_require_module:ooo
      \l_stex_import_archive_str
      \l_stex_import_path_str
      \l_stex_import_name_str
    \exp_args:Nnx \use:nn \group_end: {
      \str_set:Nn \exp_not:N\l_stex_current_domain_str {\l_stex_import_ns_str}
    }
  }
  \tl_if_empty:nTF{#1}{
    \bool_set_true:N \l__stex_features_implicit_bool
    \str_set:Nx \l_tmpa_str {[\l_stex_current_domain_str]}
  }{
```

```
    \bool_set_false:N \l__stex_features_implicit_bool
    \str_set:Nn \l_tmpa_str {#1}
  }

  \stex_if_do_html:TF {
    \begin{stex_annotate_env} {
      shtml:feature-#2={\l_stex_current_module_str?\l_tmpa_str},
      shtml:domain={\l_stex_current_domain_str}
      #5
    }
    \stex_annotate_invisible:n{}
  }\group_begin:
  \str_set:Nn \l__stex_features_feature_str {#2}
  \str_set_eq:NN \l_stex_feature_name_str \l_tmpa_str
  \__stex_features_setup:
  \__stex_features_reactivate:
  %^^A\stex_activate_module:o \l_stex_current_domain_str
  \exp_args:Ne \stex_metagroup_new:n {\l_stex_current_module_str / \l_stex_feature_name_str}
}

\cs_new_protected:Nn \__stex_features_do_for_list: {
  \seq_clear:N \l_stex_fors_seq
  \clist_map_inline:Nn \l_stex_key_for_clist {
    \exp_args:Ne\stex_get_in_morphism:n{\tl_to_str:n{##1}}
    \seq_put_right:Nx \l_stex_fors_seq
      {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
  }
}

\cs_new_protected:Nn \__stex_features_add_definiens:nn {
  \__stex_features_set_definiens_macros: #1\__stex_features_break:
  \_stex_assign_do:n{#2}
  #2
}
\cs_new_protected:Npn \__stex_features_set_definiens_macros: #1?#2?#3\__stex_features_break:
  \str_set:Nn \l_stex_get_symbol_mod_str {#1?#2}
  \str_set:Nn \l_stex_get_symbol_name_str {#3}
  \exp_args:Nne\use:nn{\__stex_features_set_definiens_macros_i:nnnnnnn}{
    \prop_item:Nn \l_stex_morphism_symbols_prop {[#1?#2]/[#3]}
  }
}
\cs_new_protected:Nn \__stex_features_set_definiens_macros_i:nnnnnnn {
  \tl_set:Nn \l_stex_get_symbol_def_tl{#4}
}

\cs_new_protected:Nn \stex_structural_feature_morphism_end: {
  \str_gset_eq:NN \l_stex_feature_name_str \l_stex_feature_name_str
  \str_gset_eq:NN \l_stex_current_domain_str \l_stex_current_domain_str
  \seq_gset_eq:NN \l_stex_morphism_symbols_prop \l_stex_morphism_symbols_prop
  \seq_gset_eq:NN \l_stex_morphism_renames_prop \l_stex_morphism_renames_prop
  \seq_gset_eq:NN \l_stex_morphism_morphisms_seq \l_stex_morphism_morphisms_seq
  \__stex_features_do_elaboration:
  \stex_if_do_html:TF{
    \end{stex_annotate_env}
```

```
  }\group_end:
}

\cs_new_protected:Nn \__stex_features_setup: {
  \prop_clear:N \l_stex_morphism_symbols_prop
  \prop_clear:N \l_stex_morphism_renames_prop
  \seq_clear:N \l_stex_morphism_morphisms_seq
  \__stex_features_do_decls:
  \exp_args:No \__stex_features_do_morphisms:n \l_stex_current_domain_str
}

\cs_new_protected:Nn \__stex_features_rename_all: {

}

\cs_new:Npn \__stex_features_clean:nnw [#1] / #2 \_stex_end: {
  [#1]/[#2]
}

\cs_new_protected:Nn \__stex_features_do_decls: {
  \exp_args:No \stex_iterate_symbols:nn \l_stex_current_domain_str {
    \stex_str_if_starts_with:nnTF{##3}[{
      \exp_args:NNe \prop_put:Nnn \l_stex_morphism_symbols_prop {
        \__stex_features_clean:nnw ##3 \_stex_end:
      }
    }{
    \prop_put:Nnn \l_stex_morphism_symbols_prop
      {[##1]/[##3]}
    }{
      {##2}{##4}{##5}{##6}{##7}{##8}##9
    }
  }
}

\cs_new_protected:Nn \stex_structural_feature_morphism_check_total: {
  \prop_map_inline:Nn \l_stex_morphism_symbols_prop {
    \__stex_features_total_check: ##1 ##2
  }
}

\cs_new_protected:Npn \__stex_features_total_check: [#1]/[#2] #3 #4 #5 #6 #7 #8 #9 {
  \tl_if_empty:nT{#6}{
    \msg_error:nnxx{stex}{error/needsdefiniens}{#1?#2}{total~morphism}
  }
}

\cs_new:Npn \__stex_features_split_qm:w #1 ? #2 ? #3 { #3 }
\cs_new_protected:Nn \__stex_features_do_elaboration: {
  \stex_debug:nn{morphisms}{
    Elaborating:^^J\prop_to_keyval:N \l_stex_morphism_symbols_prop
    ^^J
    Renamings:^^J
    \prop_to_keyval:N \l_stex_morphism_renames_prop
  }
```

```
  \prop_map_inline:Nn \l_stex_morphism_symbols_prop {
    \__stex_features_elab_check: ##1 ##2
  }
  \exp_args:No\stex_iterate_notations:nn\l_stex_current_domain_str{
    \prop_get:NnNTF \l_stex_morphism_renames_prop {##1}\l__stex_features_tmp {
      \exp_args:Ne \stex_module_add_notation:nnnnn
      {\l_stex_current_module_str ? \exp_after:wN \use_ii:nn \l__stex_features_tmp}
    }{
      \exp_args:Ne \stex_module_add_notation:nnnnn
      {\l_stex_current_module_str ? \l_stex_feature_name_str
       / \__stex_features_split_qm:w ##1}
    }{##2}{##3}{##4}{##5}
  }
  \stex_module_add_morphism:ooox
    \l_stex_feature_name_str
    \l_stex_current_domain_str
    \l__stex_features_feature_str
    {\prop_map_function:NN \l_stex_morphism_renames_prop \__stex_features_rename:nn}
}

\cs_new:Nn \__stex_features_rename:nn{
  {#1}{\use_ii:nn#2}
}

\cs_new_protected:Npn \__stex_features_elab_check: [#1]/[#2] #3 {
  \prop_get:NnNTF \l_stex_morphism_renames_prop {#1?#2} \l__stex_features_tmp {
    \stex_debug:nn{morphisms}{Generating~\l__stex_features_tmp}
    \exp_after:wN \stex_module_add_symbol:nnnnnnnN \l__stex_features_tmp
  }{
    \bool_if:NTF \l__stex_features_implicit_bool {
      \stex_debug:nn{morphisms}{Generating~#3:~\l_stex_feature_name_str / #2}
      \exp_args:Nno \stex_module_add_symbol:nnnnnnnN {#3}{\l_stex_feature_name_str / #2}
    }{
      \stex_debug:nn{morphisms}{Generating~\l_stex_feature_name_str / #2}
      \exp_args:Nno \stex_module_add_symbol:nnnnnnnN {}{\l_stex_feature_name_str / #2}
    }
  }
}

\cs_new_protected:Nn \__stex_features_do_morphisms:n {
    \prop_map_inline:cn {c_stex_module_#1_morphisms_prop}{
      \__stex_features_do_morph:nnnn ##2
    }
}

\cs_new_protected:Nn \__stex_features_do_morph:nnnn {
    \tl_if_empty:nF{#3}{
      \seq_put_right:Nn \l_stex_morphism_morphisms_seq {{#1}{#2}{#3}}
    }
    \__stex_features_do_morphisms:n{#2}
}

\cs_new_protected:Npn \__stex_features_reactivate: {
  \stex_deactivate_macro:Nn \symdecl {module~environments}
```

```
  \stex_deactivate_macro:Nn \textsymdecl {module~environments}
  \stex_deactivate_macro:Nn \symdef {module~environments}
  \stex_deactivate_macro:Nn \notation {module~environments}
  \stex_deactivate_macro:Nn \importmodule {module~environments}
  \stex_deactivate_macro:Nn \requiremodule {module~environments}
  \stex_deactivate_macro:Nn \smodule {outside~of~morphisms}
  \stex_reactivate_macro:N \assign
  \stex_reactivate_macro:N \assignMorphism
  \stex_reactivate_macro:N \renamedecl
  \cs_set_eq:NN \_stex_do_for_list: \__stex_features_do_for_list:
  \cs_set_eq:NN \_stex_add_definiens:nn \__stex_features_add_definiens:nn
}
```

(*End of definition for* `\stex_structural_feature_morphism:nnn` *and* `\stex_structural_feature_morphism_end:`*. These functions are documented on page* **??**.)

`\stex_get_in_morphism:n`

```
\cs_new_protected:Nn \stex_get_in_morphism:n {
  \str_clear:N \l_stex_get_symbol_name_str
  \prop_map_inline:Nn \l_stex_morphism_symbols_prop {
    \exp_args:Nx\__stex_features_get_check:nnnn{\tl_to_str:n{#1}}##1##2
  }
  \str_if_empty:NT \l_stex_get_symbol_name_str {
    \prop_map_inline:Nn \l_stex_morphism_renames_prop {
      \__stex_features_renamed_check:nnnnn{#1}##1=##2
    }
    \str_if_empty:NT \l_stex_get_symbol_name_str {
      \msg_error:nnxx{stex}{error/unknownsymbolin}{#1}{
        morphism~\l_stex_feature_name_str
      }
    }
  }
}

\cs_new_protected:Npn \__stex_features_renamed_check:nnnnn #1#2?#3?#4=#5#6 {
  \str_if_eq:nnTF{#1}{#5}{
    \exp_args:Nnx \use:nn{\__stex_features_check_break:nnnnnnnnn{#2?#3}{#4}}{
      \prop_item:Nn \l_stex_morphism_symbols_prop {[#2?#3]/[#4]}
    }
  }{
    \str_if_eq:nnT{#1}{#6}{
      \exp_args:Nnx \use:nn{\__stex_features_check_break:nnnnnnnnn{#2?#3}{#4}}{
        \prop_item:Nn \l_stex_morphism_symbols_prop {[#2?#3]/[#4]}
      }
    }
  }
}

\cs_new_protected:Npn \__stex_features_get_check:nnnn #1[#2]/[#3]#4 {
  \str_if_eq:nnTF{#1}{#3}{
    \__stex_features_check_break:nnnnnnnnn{#2}{#3}{#4}
  }{
    \str_if_eq:nnTF{#1}{#4}{
      \__stex_features_check_break:nnnnnnnnn{#2}{#3}{#4}
```

```
    }{
      \use_none:nnnnnn
    }
  }
}

\cs_new_protected:Nn \__stex_features_check_break:nnnnnnnnn {
  \prop_map_break:n{
    \str_set:Nn \l_stex_get_symbol_mod_str{#1}
    \str_set:Nn \l_stex_get_symbol_name_str{#2}
    \str_set:Nn \l_stex_get_symbol_macro_str{#3}
    \int_set:Nn \l_stex_get_symbol_arity_int {#4}
    \tl_set:Nn \l_stex_get_symbol_args_tl {#5}
    \tl_set:Nn \l_stex_get_symbol_def_tl {#6}
    \tl_set:Nn \l_stex_get_symbol_type_tl {#7}
    \tl_set:Nn \l_stex_get_symbol_return_tl {#8}
    \tl_set:Nn \l_stex_get_symbol_invoke_cs {#9}
  }
}
```

(*End of definition for* `\stex_get_in_morphism:n`. *This function is documented on page* 131.)

13.7 Inheritance

13.7.1 \importmodule/\usemodule

⟨@@=stex_importmodule⟩

\usemodule

```
\stex_new_stylable_cmd:nnnn {usemodule} { O{} m } {
  \stex_import_module_uri:nn { #1 }{ #2 }
  \stex_import_require_module:ooo
    \l_stex_import_archive_str
    \l_stex_import_path_str
    \l_stex_import_name_str
  \stex_if_do_html:T {
    \hbox{\stex_annotate_invisible:nn
      {shtml:usemodule=\l_stex_import_ns_str} {}}
  }
  \stex_if_smsmode:F{
    \group_begin:
    \tl_set_eq:NN \thismoduleuri \l_stex_import_ns_str
    \tl_set_eq:NN \thismodulename \l_stex_import_name_str
    \tl_clear:N \thisstyle
    \stex_style_apply:
    \group_end:
  }
}{}
```

(*End of definition for* `\usemodule`. *This function is documented on page* 93.)

\stex_import_module_uri:nn

```
\cs_new_protected:Nn \stex_import_module_uri:nn {
  \stex_debug:nn{importmodule}{URI:~>#1<~>#2<}
```

```
  \exp_args:NNnx \seq_set_split:Nnn \__stex_importmodule_seq ? { \tl_to_str:n{ #2 } }
  \seq_pop_right:NN \__stex_importmodule_seq \l_stex_import_name_str
  \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \__stex_importmodule_seq ? }
  \tl_if_empty:nTF { #1 } {
    \stex_debug:nn{importmodule}{No~archive}
    \prop_if_exist:NTF \l_stex_current_archive_prop {
      \stex_debug:nn{importmodule}{Picking~current~archive}
      \str_set:Nx \l_stex_import_archive_str {
        \prop_item:Nn \l_stex_current_archive_prop { id }
      }
    }{
      \str_clear:N \l_stex_import_archive_str
      \str_set:Nn \l_stex_import_uri_str {file:}
      \str_if_empty:NTF \l_stex_import_path_str {
        \stex_debug:nn{importmodule}{Empty~Path}
        \stex_file_split_off_ext:NN \l__stex_importmodule_path_seq \g_stex_current_file
        \stex_file_split_off_lang:NN \l__stex_importmodule_path_seq \l__stex_importmodule_pa
        \str_set:Nx \l_stex_import_path_str {
          \stex_file_use:N \l__stex_importmodule_path_seq
        }
      }{
        \stex_debug:nn{importmodule}{Resolving~path~\l_stex_import_path_str~relative~to~\ste
        \stex_file_resolve:Nx \l__stex_importmodule_seq { \stex_file_use:N \g_stex_current_f
        \str_set:Nx \l_stex_import_path_str {
          \stex_file_use:N \l__stex_importmodule_seq
        }
        \stex_debug:nn{importmodule}{...yields~\l_stex_import_path_str}
      }
    }
  } {
    \stex_debug:nn{importmodule}{Archive~#1}
    \str_set:Nx \l_stex_import_archive_str { #1 }
    \stex_require_archive:o \l_stex_import_archive_str
    \str_set:Nx \l_stex_import_uri_str { \prop_item:cn{ c_stex_mathhub_ \l_stex_import_archi
  }
}
```

(*End of definition for* `\stex_import_module_uri:nn`. *This function is documented on page* 130.)

`\stex_import_require_module:nnn`
`\stex_import_require_module:ooo`

```
\cs_new_protected:Npn \stex_import_require_module:nnn #1 {
  \tl_if_empty:nTF { #1 } {
    \str_clear:N \l__stex_importmodule_archive_str
    \str_set:Nn \l_stex_import_uri_str {file:}
    \__stex_importmodule_get_module:nnn {}
  }{
    \stex_require_archive:n { #1 }
    \str_set:Nx \l__stex_importmodule_archive_str {#1}
    \str_set:Nx \l_stex_import_uri_str { \prop_item:cn{ c_stex_mathhub_ #1 _manifest_prop}{
    \str_set:Nx \l__stex_importmodule_str { \stex_file_use:N \c_stex_mathhub_file / #1 / sou
    \exp_args:No \__stex_importmodule_get_module:nnn \l__stex_importmodule_str
  }
}
\cs_generate_variant:Nn \stex_import_require_module:nnn {ooo}
```

```
\cs_new_protected:Npn \stex_import_require_module_safe:nnn #1 {
  \tl_if_empty:nTF { #1 } {
    \str_clear:N \l__stex_importmodule_archive_str
    \str_set:Nn \l_stex_import_uri_str {file:}
    \__stex_importmodule_get_module_safe:nnn {}
  }{
    \stex_require_archive:n { #1 }
    \str_set:Nx \l__stex_importmodule_archive_str {#1}
    \str_set:Nx \l_stex_import_uri_str { \prop_item:cn{ c_stex_mathhub_ #1 _manifest_prop}{
    \str_set:Nx \l__stex_importmodule_str { \stex_file_use:N \c_stex_mathhub_file / #1 / sou
    \exp_args:No \__stex_importmodule_get_module_safe:nnn \l__stex_importmodule_str
  }
}

\cs_new_protected:Nn \__stex_importmodule_get_module_uri:nnn {
  \tl_if_empty:nF {#2}{
    \str_set:Nx \l_stex_import_uri_str {\l_stex_import_uri_str / #2}
  }
  \stex_debug:nn{importmodule}{~>#1<^^J>#2<^^J>#3<^^J>\l_stex_import_uri_str<^^J
    Current~file:\stex_file_use:N \g_stex_current_file^^J
    Current~namespace:\stex_uri_use:N \l_stex_current_ns_uri
  }
  \stex_if_module_exists:nTF {\l_stex_import_uri_str?#3} {
    \str_set:Nx \l_stex_import_ns_str {\l_stex_import_uri_str?#3}
  }{
    \stex_if_module_exists:nTF{\stex_uri_use:N \l_stex_current_ns_uri ? #3}{
      \str_set:Nx \l_stex_import_ns_str {\stex_uri_use:N \l_stex_current_ns_uri ? #3}
    }{
      \__stex_importmodule_get_from_file:nnn{#1}{#2}{#3}
      \str_set:Nx \l_stex_import_ns_str {\l_stex_import_uri_str?#3}
    }
  }
}
\cs_new_protected:Nn \__stex_importmodule_get_module_uri_safe:nnn {
  \tl_if_empty:nF {#2}{
    \str_set:Nx \l_stex_import_uri_str {\l_stex_import_uri_str / #2}
  }
  \stex_debug:nn{importmodule}{~>#1<^^J>#2<^^J>#3<^^J>\l_stex_import_uri_str<^^J
    Current~file:\stex_file_use:N \g_stex_current_file^^J
    Current~namespace:\stex_uri_use:N \l_stex_current_ns_uri
  }
  \stex_if_module_exists:nTF {\l_stex_import_uri_str?#3} {
    \str_set:Nx \l_stex_import_ns_str {\l_stex_import_uri_str?#3}
  }{
    \stex_if_module_exists:nTF{\stex_uri_use:N \l_stex_current_ns_uri ? #3}{
      \str_set:Nx \l_stex_import_ns_str {\stex_uri_use:N \l_stex_current_ns_uri ? #3}
    }{
      \__stex_importmodule_get_from_file_safe:nnn{#1}{#2}{#3}
      \str_set:Nx \l_stex_import_ns_str {\l_stex_import_uri_str?#3}
    }
  }
}
```

```
\cs_new_protected:Nn \__stex_importmodule_get_module:nnn {
  \stex_debug:nn{importmodule}{Requiring~>[#1]#2?#3<}
  \__stex_importmodule_get_module_uri:nnn{#1}{#2}{#3}
  \stex_activate_module:o \l_stex_import_ns_str
}

\cs_new_protected:Nn \__stex_importmodule_get_module_safe:nnn {
  \stex_debug:nn{importmodule}{Requiring~>[#1]#2?#3<}
  \__stex_importmodule_get_module_uri_safe:nnn{#1}{#2}{#3}
}

\cs_new_protected:Nn \__stex_importmodule_get_from_file:nnn {
  \stex_file_resolve:Nx \l__stex_importmodule_seq { \tl_if_empty:nF{ #1 }{ #1 / } #2 }
  \str_set:Nx \l__stex_importmodule_str {\stex_file_use:N \l__stex_importmodule_seq}
  \stex_debug:nn{imports}{Looking~for~\l_stex_import_uri_str?#3...}
  \__stex_importmodule_check_file:nn{ /#3.tex }{
    \__stex_importmodule_check_file:nn{/#3.\l_stex_current_language_str .tex}{
      \__stex_importmodule_check_file:nn{/#3.en.tex}{
        \__stex_importmodule_check_file:nn{.tex}{
          \__stex_importmodule_check_file:nn{.\l_stex_current_language_str .tex}{
            \__stex_importmodule_check_file:nn{.en.tex}{
              \msg_error:nnx{stex}{error/unknownmodule}{\l_stex_import_uri_str?#3}
            }
          }
        }
      }
    }
  }
  \stex_if_smsmode:TF{
    \exp_args:NNo \exp_args:Nnx \str_if_eq:nnTF{\l__stex_importmodule_str}{\stex_file_use:N
      \stex_debug:nn{imports}{Skipping~current~file}
    }{
      \__stex_importmodule_load_file:n{#3}
    }
  }{
    \__stex_importmodule_load_file:n{#3}
  }
}
\cs_new_protected:Nn \__stex_importmodule_get_from_file_safe:nnn {
  \stex_file_resolve:Nx \l__stex_importmodule_seq { \tl_if_empty:nF{ #1 }{ #1 / } #2 }
  \str_set:Nx \l__stex_importmodule_str {\stex_file_use:N \l__stex_importmodule_seq}
  \stex_debug:nn{imports}{Looking~for~\l_stex_import_uri_str?#3...}
  \__stex_importmodule_check_file:nn{ /#3.tex }{
    \__stex_importmodule_check_file:nn{/#3.\l_stex_current_language_str .tex}{
      \__stex_importmodule_check_file:nn{/#3.en.tex}{
        \__stex_importmodule_check_file:nn{.tex}{
          \__stex_importmodule_check_file:nn{.\l_stex_current_language_str .tex}{
            \__stex_importmodule_check_file:nn{.en.tex}{
            }
          }
        }
      }
    }
  }
```

```
  \stex_if_smsmode:TF{
    \exp_args:NNo \exp_args:Nnx \str_if_eq:nnTF{\l__stex_importmodule_str}{\stex_file_use:N
      \stex_debug:nn{imports}{Skipping~current~file}
    }{
      \IfFileExists{ \l__stex_importmodule_str }{
      \__stex_importmodule_load_file:n{#3}
      }{}
    }
  }{
    \IfFileExists{ \l__stex_importmodule_str }{
    \__stex_importmodule_load_file:n{#3}
    }{}
  }
}

\cs_new_protected:Nn \__stex_importmodule_load_file:n {
  \stex_file_in_smsmode:on \l__stex_importmodule_str {
    \str_if_empty:NF \l__stex_importmodule_archive_str {
      \stex_set_current_archive:n \l__stex_importmodule_archive_str
    }
    \stex_debug:nn{modules}{Loading~\l__stex_importmodule_str}
  }
  \stex_if_module_exists:nF {\l_stex_import_uri_str?#1} {
    \msg_error:nnx{stex}{error/unknownmodule}{\l_stex_import_uri_str?#1}
  }
}

\cs_new_protected:Npn \__stex_importmodule_check_file:nn #1 {
  \stex_debug:nn{imports}{Checking~ \l__stex_importmodule_str #1}
  \IfFileExists{ \l__stex_importmodule_str #1 }{
    \stex_debug:nn{imports}{Success}
    \str_set:Nx \l__stex_importmodule_str { \l__stex_importmodule_str #1 }
  }
}
```

(*End of definition for* `\stex_import_require_module:nnn`. *This function is documented on page* 130.)

`\importmodule`

```
\stex_new_stylable_cmd:nnnn{importmodule} { O{} m } {
  \__stex_importmodule_import_module:nn {#1}{#2}
  \stex_smsmode_do:
}{}
\stex_deactivate_macro:Nn \importmodule {module~environments}

\cs_new_protected:Nn \__stex_importmodule_import_module:nn {
  \stex_import_module_uri:nn { #1 }{ #2 }
  \stex_import_require_module:ooo
    \l_stex_import_archive_str
    \l_stex_import_path_str
    \l_stex_import_name_str
  \stex_execute_in_module:x{
    \stex_activate_module:n{\l_stex_import_ns_str}
  }
  \stex_module_add_morphism:nonn
```

```
    {}{\l_stex_import_ns_str}{import}{}
  \stex_if_do_html:T {
    \stex_annotate_invisible:nn
      {shtml:import=\l_stex_import_ns_str} {}
  }
  \stex_if_smsmode:F{
    \group_begin:
    \tl_set:Nn \thisarchive {#1}
    \tl_set_eq:NN \thismoduleuri \l_stex_import_ns_str
    \tl_set_eq:NN \thismodulename \l_stex_import_name_str
    \tl_clear:N \thisstyle
    \stex_style_apply:
    \group_end:
  }
}

\cs_new_protected:Nn \__stex_importmodule_import_module_presms:nn {
  \stex_import_module_uri:nn { #1 }{ #2 }
  \tl_gput_right:Nx \g_stex_sms_import_code {
    \stex_import_require_module_safe:nnn
      {\l_stex_import_archive_str}
      {\l_stex_import_path_str}
      {\l_stex_import_name_str}
  }
}

\stex_sms_allow_escape:N \importmodule
\stex_every_module:n {\stex_reactivate_macro:N \importmodule}
\stex_sms_allow_import:Nn \importmodule {
  \stex_reactivate_macro:N \importmodule
  \let \__stex_importmodule_import_module:nn \__stex_importmodule_import_module_presms:nn
}

\stex_new_stylable_cmd:nnnn{requiremodule} { O{} m } {
  \stex_import_module_uri:nn { #1 }{ #2 }
  \stex_import_require_module:ooo
    \l_stex_import_archive_str
    \l_stex_import_path_str
    \l_stex_import_name_str
  \stex_do_up_to_module:x{
    \stex_activate_module:n{\l_stex_import_ns_str}
  }
  \stex_if_do_html:T {
    \stex_annotate_invisible:nn
      {shtml:import=\l_stex_import_ns_str} {}
  }
  \stex_if_smsmode:F{
    \group_begin:
    \tl_set:Nn \thisarchive {#1}
    \tl_set_eq:NN \thismoduleuri \l_stex_import_ns_str
    \tl_set_eq:NN \thismodulename \l_stex_import_name_str
    \tl_clear:N \thisstyle
    \stex_style_apply:
    \group_end:
```

```
  }
  \stex_smsmode_do:
}{}
\stex_deactivate_macro:Nn \requiremodule {module~environments}
```

(*End of definition for* `\importmodule`. *This function is documented on page 93.*)

13.7.2 Theory Morphisms

⟨@@=stex_morphisms⟩

```
\stex_new_stylable_cmd:nnnn {assign} { m m }{
  \stex_get_in_morphism:n{#1}
  \_stex_assign_do:n{#2}
  \stex_smsmode_do:
}{}
\stex_sms_allow_escape:N\assign

\cs_new_protected:Nn \_stex_assign_do:n{
  \stex_debug:nn{assign}{Assigning~\l_stex_get_symbol_name_str~to~\tl_to_str:n{#1}}
  \tl_if_empty:NF \l_stex_get_symbol_def_tl {
    %\msg_error:nnxx{stex}{error/symbolalreadydefined}{\l_stex_get_symbol_name_str}{
    %  morphism~\l_stex_feature_name_str
    %}
  }
  \stex_check_term:n{#1}
  \stex_debug:nn{HERE!}{
    \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str^^J
    \tl_to_str:n{#1}
  }
  \stex_if_do_html:T{
    \stex_annotate_invisible:nn{shtml:assign={\l_stex_get_symbol_mod_str?\l_stex_get_symbol_
      \_stex_annotate_force_break:n{
        \mode_if_math:T\hbox{$\stex_annotate:nn{shtml:definiens={}}{#1}$}
      }
    }
  }
  \exp_args:Ne \stex_metagroup_do_in:nx{
    \l_stex_current_module_str / \l_stex_feature_name_str
  }{
    \prop_put:Nnn \exp_not:N \l_stex_morphism_symbols_prop
    {[\l_stex_get_symbol_mod_str]/[\l_stex_get_symbol_name_str]}
    {
      {\l_stex_get_symbol_macro_str}
      {\int_use:N \l_stex_get_symbol_arity_int}
      {\l_stex_get_symbol_args_tl}
      {\exp_not:n{#1}}
      {\exp_args:No\exp_not:n\l_stex_get_symbol_type_tl}
      {\exp_args:No\exp_not:n\l_stex_get_symbol_return_tl}
      {\l_stex_get_symbol_invoke_cs}
    }
  }
}
```

```
\stex_new_stylable_cmd:nnnn {renamedecl} { m O{} m }{
  \stex_get_in_morphism:n{#1}
  \_stex_renamedecl_do:nn{#2}{#3}
  \stex_smsmode_do:
}{}
\stex_sms_allow_escape:N\renamedecl

\cs_new_protected:Nn \_stex_renamedecl_do:nn {
  \stex_debug:nn{renamedecl}{Renaming~\l_stex_get_symbol_name_str~to~[#1]{#2}}
  \stex_if_do_html:T{
    \exp_args:Ne \stex_annotate_invisible:nn{
      shtml:rename={\l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str},
      shtml:macroname={#2}
      \str_if_empty:nF{#1}{ ,shtml:to={#1} }
    }{}
  }
  \exp_args:Ne \stex_metagroup_do_in:nx{
    \l_stex_current_module_str / \l_stex_feature_name_str
  }{
    \prop_put:Nnn \exp_not:N \l_stex_morphism_renames_prop
    {\l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str}{{#2}{
      \tl_if_empty:nTF{#1}{\l_stex_feature_name_str/\l_stex_get_symbol_name_str}{#1}
    }}
  }
}

\stex_new_stylable_cmd:nnnn {assignMorphism} { m m }{
  \str_clear:N \l__stex_morphisms_morphism_dom_str
  \stex_iterate_morphisms:nn\l_stex_current_domain_str{
    \stex_debug:nn{assignMorphism}{
      Checking:~#1~vs:^^J##1^^J##2^^J##3^^J##4
    }
    \str_if_eq:nnTF{#1}{##1}{
      \__stex_morphisms_do_morph_assign:nnn{##1}{##2}{##4}
    }{
      \stex_str_if_ends_with:nnT{##2}{#1}{
        \__stex_morphisms_do_morph_assign:nnn{##1}{##2}{##4}
      }
    }
  }
  \str_if_empty:NT \l__stex_morphisms_morphism_dom_str {
    \msg_error:nnn{stex}{error/nomorphism}{#1}
  }
  \bool_set_false:N \l_tmpa_bool
  \stex_iterate_morphisms:nn \l_stex_current_module_str {
    \stex_debug:nn{assignMorphism}{
      Checking:~#2~vs:^^J##1^^J##2^^J##3^^J##4
    }
    \str_if_eq:nnTF{#2}{##1}{
      \stex_debug:nn{assignMorphism}{match!}
      \stex_iterate_break:n{
        \stex_annotate_invisible:nn{
          shtml:assignMorphismFrom={\l__stex_morphisms_morphism_dom_str}
          ahtml:assignMorphismTo={\l_stex_current_module_str?##1}
```

```
        }{}
        \bool_set_true:N \l_tmpa_bool
      }
    }{
      \stex_str_if_ends_with:nnT{##2}{#2}{
        \stex_debug:nn{assignMorphism}{match!}
        \stex_iterate_break:n{
          \stex_annotate_invisible:nn{
            shtml:assignMorphismFrom={\l__stex_morphisms_morphism_dom_str},
            shtml:assignMorphismTo={\l_stex_current_module_str?##1}
          }{}
          \bool_set_true:N \l_tmpa_bool
        }
      }
    }
  }
  \bool_if:NF \l_tmpa_bool {
    \msg_error:nnn{stex}{error/nomorphism}{#2}
  }
}{}
\cs_new_protected:Nn \__stex_morphisms_do_morph_assign:nnn {
  \stex_iterate_break:n{
    \str_set:Nx \l__stex_morphisms_morphism_dom_str { \l_stex_current_domain_str ? #1 }
    \stex_debug:nn{assignMorphism}{match!}
    \stex_iterate_symbols:nn{#2}{
      \stex_debug:nn{assignMorphism}{removing~##1?##3}
      % TODO: non-trivial assignments
      \prop_remove:Nn \l_stex_morphism_symbols_prop {
        [##1]/[##3]
      }
    }
  }
}

\stex_deactivate_macro:Nn \assign {morphism~environments}
\stex_deactivate_macro:Nn \renamedecl {morphism~environments}
\stex_deactivate_macro:Nn \assignMorphism {morphism~environments}

\stex_new_stylable_env:nnnnnnn {copymodule}{m O{} m}{

  \stex_structural_feature_morphism:nnnnn{#1}{morphism}{#2}{#3}{,shtml:total=false}

  \stex_if_smsmode:F {
    \tl_set:Nn \thiscopyname { #1 }
    \tl_set_eq:NN \thismoduleuri \l_stex_current_domain_str
    \stex_style_apply:
  }
  \stex_smsmode_do:
}{
  \stex_if_smsmode:F {
    \stex_style_apply:
  }
  \stex_structural_feature_morphism_end:
}{}{}{}
\stex_deactivate_macro:Nn \copymodule {module~environments}
```

```
\stex_every_module:n {
  \stex_reactivate_macro:N \copymodule
}
\stex_sms_allow_env:n{copymodule}

\stex_new_stylable_env:nnnnnnn {interpretmodule}{m O{} m}{
  \stex_structural_feature_morphism:nnnnn{#1}{morphism}{#2}{#3}{,shtml:total=true}
  \stex_if_smsmode:F {
    \tl_set:Nn \thiscopyname { #1 }
    \tl_set_eq:NN \thismoduleuri \l_stex_current_domain_str
    \stex_style_apply:
  }
  \stex_smsmode_do:
}{
  \stex_structural_feature_morphism_check_total:
  \stex_if_smsmode:F {
    \stex_style_apply:
  }
  \stex_structural_feature_morphism_end:
}{}{}{}
\stex_deactivate_macro:Nn \interpretmodule {module~environments}
\stex_every_module:n {
  \stex_reactivate_macro:N \interpretmodule
}
\stex_sms_allow_env:n{interpretmodule}

\stex_new_stylable_env:nnnnnnn {realization}{O{} m}{

  \stex_structural_feature_morphism:nnnnn{}{morphism}{#1}{#2}{,shtml:total=true}
  %\stex_execute_in_module:x{
  %  \stex_activate_module:n{\l_stex_current_domain_str}
  %}
  \stex_if_smsmode:F {
    \tl_set:Nn \thiscopyname { #2 }
    \tl_set_eq:NN \thismoduleuri \l_stex_current_domain_str
    \stex_style_apply:
  }
  \stex_smsmode_do:
}{
  \stex_structural_feature_morphism_check_total:
  \stex_if_smsmode:F {
    \stex_style_apply:
  }
  \stex_structural_feature_morphism_end:
}{}{}{}
\stex_deactivate_macro:Nn \realization {module~environments}
\stex_every_module:n {
  \stex_reactivate_macro:N \realization
}
\stex_sms_allow_env:n{realization}

\cs_new_protected:Nn \__stex_morphisms_parse_assign:n {
  \str_clear:N \l__stex_morphisms_name_str
  \str_clear:N \l__stex_morphisms_newname_str
  \tl_clear:N \l__stex_morphisms_ass_tl
```

```
  \stex_debug:nn{morphisms}{Parsing~#1}
  \exp_args:NNe \seq_set_split:Nnn \l__stex_morphisms_seq {\tl_to_str:n{@}} {#1}
  \int_compare:nNnTF {\seq_count:N \l__stex_morphisms_seq} = 1 {
    \stex_debug:nn{morphisms}{No~@}
    \seq_pop_left:NN \l__stex_morphisms_seq \l__stex_morphisms_next_tl
  }{
    \seq_pop_left:NN \l__stex_morphisms_seq \l__stex_morphisms_name_str
    \stex_debug:nn{morphisms}{Name:~\l__stex_morphisms_name_str}
    \exp_args:NNo \str_set:Nn \l__stex_morphisms_name_str \l__stex_morphisms_name_str
    \tl_set:Nx \l__stex_morphisms_next_tl {\seq_use:Nn \l__stex_morphisms_seq @}
  }
  \exp_args:NNNo \seq_set_split:Nnn \l__stex_morphisms_seq = \l__stex_morphisms_next_tl
  \str_if_empty:NTF \l__stex_morphisms_name_str {
    \seq_pop_left:NN \l__stex_morphisms_seq \l__stex_morphisms_name_str
    \exp_args:NNo \str_set:Nn \l__stex_morphisms_name_str \l__stex_morphisms_name_str
    \tl_set:Nx \l__stex_morphisms_ass_tl {\seq_use:Nn \l__stex_morphisms_seq =}
  }{
    \seq_pop_left:NN \l__stex_morphisms_seq \l__stex_morphisms_newname_str
    \exp_args:NNo \str_set:Nn \l__stex_morphisms_newname_str \l__stex_morphisms_newname_str
    \tl_set:Nx \l__stex_morphisms_ass_tl {\seq_use:Nn \l__stex_morphisms_seq =}
  }
  \__stex_morphisms_do_parsed_assign:
}

\cs_new_protected:Nn \__stex_morphisms_do_parsed_assign: {
  \exp_args:No \stex_get_in_morphism:n \l__stex_morphisms_name_str
  \str_if_empty:NF \l__stex_morphisms_newname_str {
    \exp_after:wN \__stex_morphisms_do_parsed_newname: \l__stex_morphisms_newname_str \__ste
  }
  \tl_if_empty:NF \l__stex_morphisms_ass_tl {
    \exp_args:No \_stex_assign_do:n \l__stex_morphisms_ass_tl
  }
}

\cs_new_protected:Nn \__stex_morphisms_do_parsed_newname: {
  \peek_charcode:NTF [ {
    \__stex_morphisms_do_parsed_newname:w
  }{
    \__stex_morphisms_do_parsed_newname:w []
  }
}

\cs_new_protected:Npn \__stex_morphisms_do_parsed_newname:w [#1] #2 \__stex_morphisms_end: {
  \_stex_renamedecl_do:nn{#1}{#2}
}

\stex_new_stylable_cmd:nnnn{copymod}{m O{} m m}{
  \stex_structural_feature_morphism:nnnnn{#1}{morphism}{#2}{#3}{,shtml:total=false}

  \clist_map_function:nN{#4}\__stex_morphisms_parse_assign:n

  \stex_if_smsmode:F {
    \tl_set:Nn \thiscopyname { #1 }
    \tl_set_eq:NN \thismoduleuri \l_stex_current_domain_str
```

```
    \stex_style_apply:
  }
  \stex_structural_feature_morphism_end:
  \stex_smsmode_do:
}{}
\stex_deactivate_macro:Nn \copymod {module~environments}
\stex_every_module:n {
  \stex_reactivate_macro:N \copymod
}
\stex_sms_allow_escape:N\copymod


\stex_new_stylable_cmd:nnnn{interpretmod}{m O{} m m}{
  \stex_structural_feature_morphism:nnnnn{#1}{morphism}{#2}{#3}{,shtml:total=true}

  \clist_map_function:nN{#4}\__stex_morphisms_parse_assign:n

  \stex_if_smsmode:F {
    \tl_set:Nn \thiscopyname { #1 }
    \tl_set_eq:NN \thismoduleuri \l_stex_current_domain_str
    \stex_style_apply:
  }
  \stex_structural_feature_morphism_check_total:
  \stex_structural_feature_morphism_end:
  \stex_smsmode_do:
}{}
\stex_deactivate_macro:Nn \interpretmod {module~environments}
\stex_every_module:n {
  \stex_reactivate_macro:N \interpretmod
}
\stex_sms_allow_escape:N\interpretmod


\stex_new_stylable_cmd:nnnn{realize}{O{} m m}{
  \stex_structural_feature_morphism:nnnnn{}{morphism}{#1}{#2}{,shtml:total=true}

  \clist_map_function:nN{#3}\__stex_morphisms_parse_assign:n

  \stex_if_smsmode:F {
    \tl_set:Nn \thiscopyname { #1 }
    \tl_set_eq:NN \thismoduleuri \l_stex_current_domain_str
    \stex_style_apply:
  }
  \stex_structural_feature_morphism_check_total:
  \stex_structural_feature_morphism_end:
  \stex_smsmode_do:
}{}
\stex_deactivate_macro:Nn \realize {module~environments}
\stex_every_module:n {
  \stex_reactivate_macro:N \realize
}
\stex_sms_allow_escape:N\realize
```

13.8 Symbols

13.8.1 Declarations

⟨@@=stex_symdecl⟩

Some setup:

\stex_if_check_terms_p:
\stex_if_check_terms:*TF*

```
\stex_if_html_backend:TF {
  \prg_new_conditional:Nnn \stex_if_check_terms: {p, T, F, TF} {
    \prg_return_false:
  }
}{
  \stex_get_env:Nn\__stex_symdecl_env_str{STEX_CHECKTERMS}
  \str_if_empty:NF\__stex_symdecl_env_str{
    \exp_args:No \str_if_eq:nnF \__stex_symdecl_env_str{false}{
      \bool_set_true:N \c_stex_check_terms_bool
    }
  }
  \bool_if:NTF \c_stex_check_terms_bool {
    \prg_new_conditional:Nnn \stex_if_check_terms: {p, T, F, TF} {
      \prg_return_true:
    }
  }{
    \prg_new_conditional:Nnn \stex_if_check_terms: {p, T, F, TF} {
      \prg_return_false:
    }
  }
}
```

(*End of definition for* `\stex_if_check_terms:TF`*. This function is documented on page 124.*)

\stex_check_term:n

```
\stex_if_check_terms:TF{
  \cs_new_protected:Nn \stex_check_term:n {
    \hbox_set:Nn \l_tmpa_box {
      \group_begin:
        $#1$
      \group_end:
    }
  }
}{
  \cs_new_protected:Nn \stex_check_term:n {}
}
```

(*End of definition for* `\stex_check_term:n`*. This function is documented on page 124.*)

`symdecl` arguments:

```
\stex_keys_define:nnnn{symargs}{
  \str_clear:N \l_stex_key_args_str
  \str_clear:N \l_stex_key_role_str
  \str_clear:N \l_stex_key_reorder_str
  \str_clear:N \l_stex_key_assoc_str
}{
  args      .str_set:N  = \l_stex_key_args_str ,
  reorder   .str_set:N  = \l_stex_key_reorder_str ,
```

```
  assoc      .choices:nn   = {bin,binl,binr,pre,conj,pwconj}
    {\str_set:Nx \l_stex_key_assoc_str \l_keys_choice_tl},
  role       .str_set:N     = \l_stex_key_role_str
}{}

\stex_keys_define:nnnn{decl}{
  \str_clear:N \l_stex_key_name_str
  \str_clear:N \l_stex_key_args_str
  \tl_clear:N \l_stex_key_type_tl
  \tl_clear:N \l_stex_key_def_tl
  \tl_clear:N \l_stex_key_return_tl
  \str_clear:N \l_stex_key_wikidata_str
  \clist_clear:N \l_stex_key_argtypes_clist
}{
  name       .str_set:N  = \l_stex_key_name_str ,

  return     .tl_set:N     = \l_stex_key_return_tl ,
  argtypes   .clist_set:N  = \l_stex_key_argtypes_clist ,
  wikidata   .str_set:N    = \l_stex_key_wikidata_str ,

  type       .tl_set:N     = \l_stex_key_type_tl  ,
  def        .tl_set:N     = \l_stex_key_def_tl   ,

  align      .code:n       = {},
  gfc        .code:n       = {}
}{style,deprecate,symargs}
% \_stex_do_deprecation:n{#2}
```

\symdecl

```
\str_new:N \l_stex_macroname_str
\stex_new_stylable_cmd:nnnn {symdecl} { s m O{}} {
  \stex_keys_set:nn{decl}{#3}
  \IfBooleanTF #1 {
    \str_clear:N \l_stex_macroname_str
  }{
    \str_set:Nx \l_stex_macroname_str { #2 }
  }
  \stex_symdecl_top:n{#2}

  \stex_if_smsmode:F{
    \group_begin:
    \tl_set:Nx \thisdecluri {\l_stex_current_module_str ? \l_stex_key_name_str}
    \tl_set_eq:NN \thisdeclname \l_stex_key_name_str
    \tl_set_eq:NN \thistype \l_stex_key_type_tl
    \tl_set_eq:NN \thisdefiniens \l_stex_key_def_tl
    \tl_set_eq:NN \thisargs \l_stex_key_args_str
    \tl_clear:N \thisstyle
    \stex_style_apply:
    \group_end:
  }
  \stex_smsmode_do:
}{}
\stex_deactivate_macro:Nn \symdecl {module~environments}
\stex_every_module:n {\stex_reactivate_macro:N \symdecl}
```

```
\stex_sms_allow_escape:N \symdecl
```

(*End of definition for* `\symdecl`. *This function is documented on page* 87.)

`\stex_symdecl_top:n`

```
\cs_new_protected:Nn \stex_symdecl_top:n {
  \str_if_empty:NT \l_stex_key_name_str {
    \str_set:Nx \l_stex_key_name_str { #1 }
  }
  \stex_symdecl_do:
  \_stex_symdecl_check_terms:
  \__stex_symdecl_add_decl:
  \stex_if_do_html:T {
    \_stex_symdecl_html:
  }
}

\cs_new_protected:Nn \__stex_symdecl_add_decl: {
  \exp_args:Nnx \use:nn {\stex_module_add_symbol:nnnnnnnN}{
    {\l_stex_macroname_str}
    {\l_stex_key_name_str}
    {\int_use:N \l_stex_get_symbol_arity_int}
    {\l_stex_get_symbol_args_tl}
    {\tl_if_empty:NF \l_stex_key_def_tl{DEFED} }%{\exp_args:No \exp_not:n \l_stex_key_def_tl
    {}%{\exp_args:No \exp_not:n \l_stex_key_type_tl}
    {\exp_args:No \exp_not:n \l_stex_key_return_tl}
    \stex_invoke_symbol:
  }
  \exp_args:Ne \stex_ref_new_symbol:n
    {\l_stex_current_module_str?\l_stex_key_name_str}
}

\cs_new:Nn \_stex_return_args:nn {
  {\svar{ARGUMENT_#1}\_stex_eat_exclamation_point:}
}

\cs_new_protected:Nn \_stex_symdecl_html: {
  \exp_args:Ne \stex_annotate_invisible:nn {
    shtml:symdecl = {\l_stex_current_module_str ? \l_stex_key_name_str},
    shtml:args = {\l_stex_key_args_str}
    \str_if_empty:NF \l_stex_macroname_str {,
      shtml:macroname={\l_stex_macroname_str}
    }
    \str_if_empty:NF \l_stex_key_wikidata_str {,
      shtml:wikidata={\l_stex_key_wikidata_str}
    }
    \str_if_empty:NF \l_stex_key_assoc_str {,
      shtml:assoctype={\l_stex_key_assoc_str}
    }
    \str_if_empty:NF \l_stex_key_reorder_str {,
      shtml:reorderargs={\l_stex_key_reorder_str}
    }
    \str_if_empty:NF \l_stex_key_role_str {,
      shtml:role={\l_stex_key_role_str}
```

```
    }
  }{\hbox\bgroup\_stex_annotate_force_break:n{
      \bool_set_true:N \stex_in_invisible_html_bool
      \tl_if_empty:NF \l_stex_key_type_tl {
        $\stex_annotate:nn{shtml:type={}}{\l_stex_key_type_tl}$
      }
      \tl_if_empty:NF \l_stex_key_def_tl {
        $\stex_annotate:nn{shtml:definiens={}}{\l_stex_key_def_tl}$
      }
      \tl_if_empty:NF \l_stex_key_return_tl{
        \exp_args:Nno \use:n{
        \cs_generate_from_arg_count:NNnn \l__stex_symdecl_cs
        \cs_set:Npn \l_stex_get_symbol_arity_int} \l_stex_key_return_tl
        \tl_set:Nx \l__stex_symdecl_args_tl {\_stex_map_args:N \_stex_return_args:nn}
        $\stex_annotate:nn{shtml:returntype={}}{
          \exp_after:wN \l__stex_symdecl_cs \l__stex_symdecl_args_tl!
        }$
      }
      \clist_if_empty:NF \l_stex_key_argtypes_clist {
        \stex_annotate:nn{shtml:argtypes={}}{\_stex_annotate_force_break:n{
          \clist_map_inline:Nn \l_stex_key_argtypes_clist {
              $\stex_annotate:nn{shtml:type={}}{##1}$
          }
        }}
      }
  }\egroup}
}
```

(*End of definition for* `\stex_symdecl_top:n`. *This function is documented on page 125.*)

`\stex_symdecl_do:` Requires the above keys and `\l_stex_macroname_str` to be set first

```
\cs_new_protected:Nn \stex_symdecl_do: {
  \_stex_do_deprecation:n \l_stex_key_name_str
  \__stex_symdecl_parse_arity:
  \__stex_symdecl_do_args:
}

\int_new:N \l_stex_assoc_args_count

\cs_new_protected:Nn \__stex_symdecl_parse_arity: {
  \int_zero:N \l_stex_get_symbol_arity_int
  \int_zero:N \l_stex_assoc_args_count
  \str_map_inline:Nn \l_stex_key_args_str {
    \str_case:nnF ##1 {
      0 { \str_map_break: }
      1 { \str_map_break:n{
        \int_set:Nn \l_stex_get_symbol_arity_int {1}
        \str_set:Nn \l_stex_key_args_str {i}
      } }
      2 { \str_map_break:n{
        \int_set:Nn \l_stex_get_symbol_arity_int {2}
        \str_set:Nn \l_stex_key_args_str {ii}
      } }
      3 { \str_map_break:n{
```

```
        \int_set:Nn \l_stex_get_symbol_arity_int {3}
        \str_set:Nn \l_stex_key_args_str {iii}
      } }
      4 { \str_map_break:n{
        \int_set:Nn \l_stex_get_symbol_arity_int {4}
        \str_set:Nn \l_stex_key_args_str {iiii}
      } }
      5 { \str_map_break:n{
        \int_set:Nn \l_stex_get_symbol_arity_int {5}
        \str_set:Nn \l_stex_key_args_str {iiiii}
      } }
      6 { \str_map_break:n{
        \int_set:Nn \l_stex_get_symbol_arity_int {6}
        \str_set:Nn \l_stex_key_args_str {iiiiii}
      } }
      7 { \str_map_break:n{
        \int_set:Nn \l_stex_get_symbol_arity_int {7}
        \str_set:Nn \l_stex_key_args_str {iiiiiii}
      } }
      8 { \str_map_break:n{
        \int_set:Nn \l_stex_get_symbol_arity_int {8}
        \str_set:Nn \l_stex_key_args_str {iiiiiiii}
      } }
      9 { \str_map_break:n{
        \int_set:Nn \l_stex_get_symbol_arity_int {9}
        \str_set:Nn \l_stex_key_args_str {iiiiiiiii}
      } }
      i {\int_incr:N \l_stex_get_symbol_arity_int}
      b {\int_incr:N \l_stex_get_symbol_arity_int}
      a {\int_incr:N \l_stex_get_symbol_arity_int \int_incr:N \l_stex_assoc_args_count}
      B {\int_incr:N \l_stex_get_symbol_arity_int \int_incr:N \l_stex_assoc_args_count}
    }{
      \msg_error:nnxx{stex}{error/wrongargs}{
        \l_stex_current_module_str ? \l_stex_key_name_str
      }{##1}
    }
  }
}

\cs_new_protected:Nn \__stex_symdecl_do_args: {
  \tl_clear:N \l_stex_get_symbol_args_tl
  \int_step_inline:nn \l_stex_get_symbol_arity_int {
    \tl_put_right:Nn \l_stex_get_symbol_args_tl {##1}
    \tl_put_right:Nx \l_stex_get_symbol_args_tl {
      \str_item:Nn \l_stex_key_args_str {##1}
    }
  }
}
```

(*End of definition for* `\stex_symdecl_do:`. *This function is documented on page* 124.)

`\_stex_symdecl_check_terms:`

```
\cs_new_protected:Nn \_stex_symdecl_check_terms: {
  \stex_check_term:n{
```

```
    \stex_debug:nn{check_terms}{Checking~type...}
    \group_begin:\l_stex_key_type_tl\group_end:
    \stex_debug:nn{check_terms}{Checking~definiens...}
    \group_begin:\l_stex_key_def_tl\group_end:
    \stex_debug:nn{check_terms}{Checking~return...}
    \group_begin:\l_stex_key_return_tl!\group_end:
    \stex_debug:nn{check_terms}{Checking~argument~types...}
    \group_begin:\l_stex_key_argtypes_clist\group_end:
  }
}
```

(*End of definition for* `\_stex_symdecl_check_terms:`. *This function is documented on page* 125.)

`\textsymdecl`

```

\stex_keys_define:nnnn{textsymdecl}{
  \str_clear:N \l_stex_key_name_str
  \tl_clear:N \l_stex_key_type_tl
  \tl_clear:N \l_stex_key_def_tl
}{
  name      .str_set:N  = \l_stex_key_name_str ,
  type      .tl_set:N     = \l_stex_key_type_tl ,
  def       .tl_set:N     = \l_stex_key_def_tl
}{style,deprecate}

\stex_new_stylable_cmd:nnnn {textsymdecl} {m O{} m} {
  \stex_keys_set:nn{symdef}{}
  \stex_keys_set:nn{textsymdecl}{#2}
  \str_set:Nx \l_stex_macroname_str { #1 }
  \str_if_empty:NT \l_stex_key_name_str {
    \str_set:Nn \l_stex_key_name_str {#1}
  }%{
  %  \str_set:Nx \l_stex_key_name_str {\l_stex_key_name_str-sym}
  %}
  \str_set:Nn \l_stex_key_role_str {textsymdecl}

  \stex_symdecl_do:
  \_stex_symdecl_check_terms:
  \exp_args:Nnx \use:nn {\stex_module_add_symbol:nnnnnnnN}{
    {\l_stex_macroname_str}
    {\l_stex_key_name_str}
    {0}{}
    {\tl_if_empty:NF \l_stex_key_def_tl{DEFED} }
    {}% type
    {\use:c{#1name_nospace}}% return
    \stex_invoke_text_symbol:
  }
  \exp_args:Ne \stex_ref_new_symbol:n
    {\l_stex_current_module_str?\l_stex_key_name_str}
  \stex_if_do_html:T {
    \_stex_symdecl_html:
  }

  \int_set:Nn \l_stex_get_symbol_arity_int 0
```

```
  \tl_clear:N \l_stex_key_op_tl
  \str_clear:N \l_stex_key_intent_str
  \str_clear:N \l_stex_key_prec_str
  \str_set_eq:NN \l_stex_get_symbol_mod_str \l_stex_current_module_str
  \str_set_eq:NN \l_stex_get_symbol_name_str \l_stex_key_name_str
  \stex_notation_parse:n{\hbox{#3}}
  \_stex_notation_add:
  \stex_if_do_html:T {
    \def\comp{\_comp}
    \_stex_notation_do_html:n{\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
  }
  \stex_execute_in_module:x{
    \__stex_symdecl_set_textsymdecl_macro:nnn{#1}{\l_stex_current_module_str?\l_stex_key_nam
    \exp_not:n{{#3}}
  }

  \stex_if_smsmode:F{
    \group_begin:
    \tl_set:Nx \thisdecluri {\l_stex_current_module_str ? \l_stex_key_name_str}
    \tl_set_eq:NN \thisdeclname \l_stex_key_name_str
    \tl_clear:N \thisstyle
    \stex_style_apply:
    \group_end:
  }
  \stex_smsmode_do:
}{}
\stex_deactivate_macro:Nn \textsymdecl {module~environments}
\stex_every_module:n {\stex_reactivate_macro:N \textsymdecl}
\stex_sms_allow_escape:N \textsymdecl

\cs_new_protected:Nn \__stex_symdecl_set_textsymdecl_macro:nnn {
  \cs_set_protected:cpn{#1name_nospace}{#3}
  \cs_set_protected:cpn{#1name}{
    \mode_if_vertical:T{\hbox_unpack:N\c_empty_box}
    \mode_if_math:T\hbox{\let\xspace\relax #3}
    \mode_if_math:F{\cs_if_exist:NT\xspace\xspace}
  }
}

\cs_new_protected:Nn \stex_invoke_text_symbol: {
  \mode_if_vertical:T{\hbox_unpack:N\c_empty_box}
  \_stex_term_oms_or_omv:nnn{}{}{\maincomp{\let\xspace\relax\l_stex_current_return_tl}}
  \group_end:\mode_if_math:F{\cs_if_exist:NT\xspace\xspace}
}
```

(*End of definition for* `\textsymdecl`. *This function is documented on page* 88.)

`\stex_get_symbol:n`

```
\cs_new_protected:Nn \stex_get_symbol:n {
  \_stex_get_symbol:n{ #1 }
  \str_if_empty:NT \l_stex_get_symbol_name_str {
    \msg_error:nnn{stex}{error/unknownsymbol}{#1}
  }
}
```

```
3996
3997 \cs_new_protected:Nn \_stex_get_symbol:n {
3998   \str_clear:N \l_stex_get_symbol_mod_str
3999   \str_clear:N \l_stex_get_symbol_name_str
4000   \cs_if_exist:cTF { #1 }{
4001     \cs_set_eq:Nc \l__stex_symdecl_cs { #1 }
4002     % command name
4003     \exp_args:Ne \tl_if_empty:nTF { \cs_argument_spec:N \l__stex_symdecl_cs }{
4004       % ...that takes no arguments
4005       \exp_args:Ne \cs_if_eq:NNTF {\tl_head:N \l__stex_symdecl_cs}
4006         \_stex_invoke_symbol:nnnnnnnN
4007         \__stex_symdecl_get_symbol_from_cs:
4008         {\__stex_symdecl_get_symbol_from_string:n { #1 }}
4009     }{
4010       \__stex_symdecl_get_symbol_from_string:n { #1 }
4011     }
4012   }{
4013     \__stex_symdecl_get_symbol_from_string:n { #1 }
4014   }
4015 }
4016
4017 \int_new:N \l_stex_get_symbol_arity_int
4018 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
4019   \stex_debug:nn{symbols}{Getting~from~cs...}
4020   \stex_pseudogroup_with:nn{\_stex_invoke_symbol:nnnnnnnN}{
4021     \cs_set:Npn \_stex_invoke_symbol:nnnnnnnN ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 {
4022       \str_set:Nn \l_stex_get_symbol_mod_str {##1}
4023       \str_set:Nn \l_stex_get_symbol_name_str {##2}
4024       \int_set:Nn \l_stex_get_symbol_arity_int {##3}
4025       \tl_set:Nn \l_stex_get_symbol_args_tl {##4}
4026       \tl_set:Nn \l_stex_get_symbol_def_tl {##5}
4027       \tl_set:Nn \l_stex_get_symbol_type_tl {##6}
4028       \tl_set:Nn \l_stex_get_symbol_return_tl {##7}
4029       \tl_set:Nn \l_stex_get_symbol_invoke_cs {##8}
4030     }
4031     \l__stex_symdecl_cs
4032   }
4033 }
4034
4035 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
4036   \stex_debug:nn{symbols}{Getting~from~string~#1...}
4037   \seq_set_split:Nnn \l__stex_symdecl_seq ? {#1}
4038   \seq_pop_right:NN \l__stex_symdecl_seq \l__stex_symdecl_name
4039   \seq_if_empty:NTF \l__stex_symdecl_seq {
4040     \exp_args:No \__stex_symdecl_get_from_one_string:n {#1}
4041   }{
4042     \exp_args:NNe \exp_args:Nno \__stex_symdecl_get_symbol_from_modules:nn {
4043       \seq_use:Nn \l__stex_symdecl_seq ?
4044     } \l__stex_symdecl_name
4045   }
4046 }
4047
4048 \cs_new_protected:Nn \__stex_symdecl_sym_from_str_i:nnnn {
4049   \bool_lazy_any:nTF{
```

```
    {\str_if_eq_p:nn{#2}{#3}}
    {\str_if_eq_p:nn{#2}{#4}}
    {\stex_str_if_ends_with_p:nn{#4}{/#2}}
    }{
      \__stex_symdecl_sym_i_finish:nnnnnnnN{#1}{#4}
    }{
      \__stex_symdecl_sym_i_gobble:nnnnnn
    }
}
\cs_new_protected:Nn \__stex_symdecl_sym_i_gobble:nnnnnn {}

\cs_new_protected:Nn \__stex_symdecl_sym_i_finish:nnnnnnnN {
  \prop_map_break:n{\seq_map_break:n{
    \str_set:Nn \l_stex_get_symbol_mod_str {#1}
    \str_set:Nn \l_stex_get_symbol_name_str {#2}
    \int_set:Nn \l_stex_get_symbol_arity_int {#3}
    \tl_set:Nn \l_stex_get_symbol_args_tl {#4}
    \tl_set:Nn \l_stex_get_symbol_def_tl {#5}
    \tl_set:Nn \l_stex_get_symbol_type_tl {#6}
    \tl_set:Nn \l_stex_get_symbol_return_tl {#7}
    \tl_set:Nn \l_stex_get_symbol_invoke_cs {#8}
  }}
}

\cs_new_protected:Nn \__stex_symdecl_get_symbol_from_modules:nn {
  \stex_debug:nn{symbols}{Getting~#2~in~#1...}
  \seq_map_inline:Nn \l_stex_all_modules_seq {
    \stex_str_if_ends_with:nnT{##1}{#1}{
      \prop_map_inline:cn{c_stex_module_##1_symbols_prop}{
        \__stex_symdecl_sym_from_str_i:nnnn{##1}{#2} ####2
      }
    }
  }
}

\cs_new_protected:Nn \__stex_symdecl_get_from_one_string:n {
  \stex_debug:nn{symbols}{Getting~#1~anywhere...}
  \stex_iterate_symbols:n{
    %\stex_debug:nn{symbols}{>#1==##2~|~#1==##3<...}
    \bool_lazy_any:nT{
      {\str_if_eq_p:nn{#1}{##2}}
      {\str_if_eq_p:nn{#1}{##3}}
      {\stex_str_if_ends_with_p:nn{##3}{/#1}}
      }{
        \stex_iterate_break:n{
          \str_set:Nn \l_stex_get_symbol_mod_str {##1}
          \str_set:Nn \l_stex_get_symbol_name_str {##3}
          \int_set:Nn \l_stex_get_symbol_arity_int {##4}
          \tl_set:Nn \l_stex_get_symbol_args_tl {##5}
          \tl_set:Nn \l_stex_get_symbol_def_tl {##6}
          \tl_set:Nn \l_stex_get_symbol_type_tl {##7}
          \tl_set:Nn \l_stex_get_symbol_return_tl {##8}
          \tl_set:Nn \l_stex_get_symbol_invoke_cs {##9}
        }
```

```
    }
  }
}
```

(*End of definition for* `\stex_get_symbol:n`. *This function is documented on page 124.*)

13.8.2 Notations

⟨@@=stex_notations⟩

`\_stex_map_args:N`
`\_stex_map_notation_args:N`

```
\cs_new:Nn \_stex_map_args:N {
  \tl_if_empty:NF \l_stex_get_symbol_args_tl {
    \exp_after:wN \__stex_notations_map_args_i:w \exp_after:wN
    #1 \l_stex_get_symbol_args_tl \__stex_notations_args_end:
  }
}
\cs_new:Npn \__stex_notations_map_args_i:w #1 #2 #3 #4 \__stex_notations_args_end: {
  #1 #2 #3
  \tl_if_empty:nF{#4}{
    \__stex_notations_map_args_i:w #1 #4 \__stex_notations_args_end:
  }
}

\cs_new:Nn \_stex_map_notation_args:N {
  \tl_if_empty:NF \l_stex_notation_args_tl {
    \exp_after:wN \__stex_notations_map_args_ii:w \exp_after:wN
    #1 \l_stex_get_symbol_args_tl \__stex_notations_args_end:
  }
}
\cs_new:Npn \__stex_notations_map_args_ii:w #1 #2 #3 #4 #5 #6 \__stex_notations_args_end: {
  #1 #2 #3 #4 #5
  \tl_if_empty:nF{#6}{
    \__stex_notations_map_args_ii:w #1 #6 \__stex_notations_args_end:
  }
}
```

(*End of definition for* `\_stex_map_args:N` *and* `\_stex_map_notation_args:N`. *These functions are documented on page* **??**.)

`notation` arguments:

```
\stex_keys_define:nnnn{notation}{
  \str_clear:N \l_stex_key_variant_str
  \str_clear:N \l_stex_key_prec_str
  \str_clear:N \l_stex_key_op_tl
  \str_clear:N \l_stex_key_intent_str
  \clist_clear:N \l_stex_key_intent_args_clist
}{
  variant    .str_set_x:N  = \l_stex_key_variant_str ,
  prec       .str_set_x:N  = \l_stex_key_prec_str ,
  op         .tl_set:N     = \l_stex_key_op_tl ,
  intent     .str_set:N    = \l_stex_key_intent_str ,
  argnames   .clist_set:N  = \l_stex_key_intent_args_clist ,
  unknown    .code:n       = {
    \str_if_empty:NTF \l_keys_key_str {
        \str_set:Nx \l_stex_key_variant_str {\l_keys_key_tl}
```

```
    }{
        \str_set_eq:NN \l_stex_key_variant_str \l_keys_key_str
    }
  }
}{style}
```

\notation

```
\stex_new_stylable_cmd:nnnn {notation} { s m O{} m} {
  \stex_keys_set:nn{notation}{#3}
  \stex_get_symbol:n{#2}
  \stex_notation_parse:n{#4}
  \stex_if_check_terms:T{ \_stex_notation_check: }
  \_stex_notation_add:
  \stex_if_do_html:T {
    \def\comp{\_comp}
    \_stex_notation_do_html:n{\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
  }
  \IfBooleanTF#1{
    \_stex_notation_set_default:n{
      \l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str
    }
  }{}
  \stex_if_smsmode:F{
    \group_begin:
    \__stex_notations_styledefs:
    \stex_style_apply:
    \group_end:
  }
  \stex_smsmode_do:
}{}

\cs_new_protected:Nn \__stex_notations_styledefs: {
  \str_set_eq:NN\thisnotationvariant\l_stex_key_variant_str
  \str_set:Nn \thisdeclname \l_stex_get_symbol_name_str
  \tl_set:Nx \thisdecluri {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
  \def\thisnotation{
    $
    \tl_set_eq:NN \l_stex_current_symbol_str\thisdecluri
    \exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs}{
      \_stex_notation_make_args:
    }$
  }
}

\stex_deactivate_macro:Nn \notation {module~environments}
\stex_every_module:n {\stex_reactivate_macro:N \notation}
\stex_sms_allow_escape:N \notation
```

(*End of definition for* `\notation`. *This function is documented on page* 90.)

`\stex_notation_parse:n` requires the above keys, `\l_stex_get_symbol_arity_int`, and `\l_stex_get_symbol_-args_tl`

```
\cs_new_protected:Nn \stex_notation_parse:n {
  \tl_if_empty:NF \l_stex_key_op_tl {
```

```
    \tl_set:Nx \l_stex_key_op_tl { \exp_not:N\maincomp {
      \exp_args:No \exp_not:n \l_stex_key_op_tl
    } }
  }
  \seq_clear:N \l__stex_notations_precs_seq
  \tl_clear:N \l_stex_notation_args_tl
  \int_compare:nNnTF \l_stex_get_symbol_arity_int = 0 {
    \__stex_notations_const_precs:
    \tl_if_empty:NT \l_stex_key_op_tl {
      \tl_set:Nn \l_stex_key_op_tl { \maincomp{#1} }
    }
  }{
    \__stex_notations_fun_precs:
    \str_set:Nn \l__stex_notations_missing_str {#1}
    \tl_clear:N \l__stex_notations_missing_tl
    \_stex_map_args:N \__stex_notations_add_missing_args:nn
    \tl_if_empty:NT \l_stex_key_op_tl {
      \hbox_set:Nn \l_tmpa_box {
        \str_set:Nn \l_stex_current_symbol_str {}
        \cs_set:Npn \l_tmpa_cs ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 ##9 { #1 }
        \cs_set:Npn \maincomp ##1 {
          \tl_gset:Nn \l_stex_key_op_tl { \maincomp{##1} }
          ##1
        }
        \cs_set:Npn \argsep ##1 ##2 {##1 ##2}
        \cs_set:Npn \argmap ##1 ##2 ##3 {##1 ##3}
        \cs_set:Npn \argarraymap ##1 ##2 ##3 ##4 {
          ##1 ##2
        }
        \stex_suppress_html:n{$\l_tmpa_cs abcdefghj$}
      }
    }
  }
  \exp_args:NNe
  \tl_set:Nn \l_stex_notation_macrocode_cs {
    \STEXInternalNotation
      { \l_stex_key_variant_str }
      { \l__stex_notations_opprec_tl }
      { \l_stex_key_intent_str }
      { \l_stex_notation_args_tl }
      {
        \int_compare:nNnTF \l_stex_get_symbol_arity_int = 0
        { \exp_not:n { \maincomp{ #1 } } }
        { \exp_not:n { #1 } \l__stex_notations_missing_tl }
      }
  }
  \stex_debug:nn{notation}{Notation:~\meaning\l_stex_notation_macrocode_cs}
}

\cs_new_protected:Nn \__stex_notations_const_precs: {
  \str_if_empty:NTF \l_stex_key_prec_str {
    \tl_set:No \l__stex_notations_opprec_tl { \neginfprec }
  }{
    \str_if_eq:onTF \l_stex_key_prec_str {nobrackets}{
```

```
      \tl_set:No \l__stex_notations_opprec_tl { \neginfprec }
    }{
      \tl_set_eq:NN \l__stex_notations_opprec_tl \l_stex_key_prec_str
    }
  }
}

\cs_new_protected:Nn \__stex_notations_fun_precs: {
  \str_if_empty:NTF \l_stex_key_prec_str {
    \tl_set:No \l__stex_notations_opprec_tl { \neginfprec }
  }{
    \str_if_eq:onTF \l_stex_key_prec_str {nobrackets}{
      \tl_set:No \l__stex_notations_opprec_tl { \neginfprec }
    }{
      \tl_set_eq:NN \l__stex_notations_opprec_tl \l_stex_key_prec_str
    }
  }
  \str_if_empty:NTF \l_stex_key_prec_str {
    \tl_set:Nn \l__stex_notations_opprec_tl { 0 }
    \int_step_inline:nn \l_stex_get_symbol_arity_int {
      \seq_put_right:Nn \l__stex_notations_precs_seq {0}
    }
  }{
    \str_if_eq:onTF \l_stex_key_prec_str {nobrackets}{
      \stex_debug:nn{notation}{No~brackets}
      \tl_set:No \l__stex_notations_opprec_tl { \neginfprec }
      \int_step_inline:nn \l_stex_get_symbol_arity_int {
        \exp_args:NNo \seq_put_right:Nn \l__stex_notations_precs_seq \infprec
      }
    }\__stex_notations_parse_precs:
  }
  \__stex_notations_do_argnames:
}

\cs_new_protected:Nn \__stex_notations_parse_precs: {
  \stex_debug:nn{notation}{parsing~precedence~\l_stex_key_prec_str}
  \seq_set_split:NnV \l__stex_notations_seq ; \l_stex_key_prec_str
  \seq_pop_left:NNTF \l__stex_notations_seq \l__stex_notations_str {
    \tl_set_eq:NN \l__stex_notations_opprec_tl \l__stex_notations_str
    \seq_pop_left:NNT \l__stex_notations_seq \l__stex_notations_str {
      \exp_args:NNo \seq_set_split:NnV \l__stex_notations_seq
        {\tl_to_str:n{x}} \l__stex_notations_str
    }
  }{
    \tl_set:No \l__stex_notations_opprec_tl { 0 }
  }
  \int_step_inline:nn \l_stex_get_symbol_arity_int {
    \seq_pop_left:NNTF \l__stex_notations_seq \l__stex_notations_str {
      \seq_put_right:No \l__stex_notations_precs_seq \l__stex_notations_str
    }{
      \seq_put_right:No \l__stex_notations_precs_seq \l__stex_notations_opprec_tl
    }
  }
}
```

```
\cs_new_protected:Nn \__stex_notations_do_argnames: {
  \tl_clear:N \l_stex_notation_args_tl
  \_stex_map_args:N \__stex_notations_do_argname:nn
}

\cs_new_protected:Nn \__stex_notations_do_argname:nn {
  \clist_if_empty:NTF \l_stex_key_intent_args_clist {
    \tl_put_right:Nx \l_stex_notation_args_tl {
      #1#2{\seq_item:Nn \l__stex_notations_precs_seq #1}{
        \str_if_empty:NF \l_stex_key_intent_str {#1}
      }
    }
  }{
    \tl_put_right:Nx \l_stex_notation_args_tl {
      #1#2{\seq_item:Nn \l__stex_notations_precs_seq #1}
      {\c_dollar_str\clist_item:Nn \l_stex_key_intent_args_clist 1}
    }
    \clist_pop:NN \l_stex_key_intent_args_clist \l_tmpa_tl
  }
}

\cs_new:Nn \__stex_notations_add_missing_args:nn {
  \exp_args:NNe \str_if_in:NnF \l__stex_notations_missing_str {\c_hash_str\c_hash_str#1}{
    \tl_put_right:Nn \l__stex_notations_missing_tl{\STEXinvisible{## #1}}
  }
}
```

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

`\_stex_notation_check:`
`\_stex_notation_add:`
`\_stex_notation_do_html:n`
`\_stex_notation_make_args:`

```
\cs_new_protected:Nn \_stex_notation_check: {
  \stex_check_term:n{
    \str_set:Nn \l_stex_current_symbol_str {test}
    \cs_set:Npn \comp ##1 {##1}
    \stex_debug:nn{check_terms}{Checking~notation...}
    \exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs{}}{
      \_stex_notation_make_args:
    }
  }
}

\cs_new_protected:Nn \_stex_notation_add: {
  \stex_module_add_notation:eoeoo{
    \l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str
    }\l_stex_key_variant_str
    {\int_use:N \l_stex_get_symbol_arity_int}
    \l_stex_notation_macrocode_cs
    \l_stex_key_op_tl
}

\cs_new_protected:Nn \_stex_notation_do_html:n {
  \hbox{\stex_annotate_invisible:nn {
    shtml:notation={#1},
```

```
    shtml:notationfragment={\l_stex_key_variant_str},
    shtml:precedence={\l__stex_notations_opprec_tl},
    shtml:argprecs={\seq_use:Nn \l__stex_notations_precs_seq ,}
  }{
    \cs_set_protected:Npn \argsep ##1 ##2 {
      \stex_annotate:nn{shtml:argsep={}}{
        ##1 ##2
      }
    }
    \cs_set_protected:Npn \argmap ##1 ##2 ##3 {
      \cs_set:Npn \__stex_notations_map_cs: ####1 { ##2 }
      \stex_annotate:nn{shtml:argmap={}}{
        \__stex_notations_map_cs:{##1} ##3
      }
    }
    \cs_set_protected:Npn \maincomp {
      \_do_comp:nNn {maincomp}\compemph@uri
    }
    $
      \str_set:Nx \l_stex_current_symbol_str {#1}
      \stex_annotate:nn{shtml:notationcomp={}}{
        \exp_args:Nne \use:nn {
          \l_stex_notation_macrocode_cs {}
        }{
          \_stex_map_args:N \__stex_notations_make_arg_html:nn
        }
      }
    $
    \tl_if_empty:NF \l_stex_key_op_tl {
      $
      \str_set:Nx \l_stex_current_symbol_str {#1}
      \stex_annotate:nn{shtml:notationopcomp={}}{
        \_stex_term_oms:nnn{\l_stex_key_variant_str}{}{\l_stex_key_op_tl}
      }
      $
    }
  }}
}

\cs_new:Nn \__stex_notations_make_arg_html:nn {
%  \str_case:nnF #2 {
%    a {{
%      \stex_annotate:nn{shtml:argnum=#1a}{x},
%      \stex_annotate:nn{shtml:argnum=#1b}{x}
%    }}
%    B {{
%      \stex_annotate:nn{shtml:argnum=#1a}{x},
%      \stex_annotate:nn{shtml:argnum=#1b}{x}
%    }}
%  }{
    {
      \stex_annotate:nn{shtml:argnum=#1}{x}
    }
%  }
```

```
}

\cs_new:Nn \_stex_notation_make_args: {
  \_stex_map_notation_args:N \__stex_notations_make_arg:nnnn
}


\cs_new:Nn \__stex_notations_make_arg:nnnn  {
  \str_case:nnF #2 {
    a {{
        a\c_math_subscript_token{#1,1},
        a\c_math_subscript_token{#1,2}
      }}
    B {{
        B\c_math_subscript_token{#1,1},
        B\c_math_subscript_token{#1,2}
      }}
  }{
    \_stex_term_arg:nnnnn{#1}{#2}{#3}{#4}
      {{#2}\c_math_subscript_token{#1}}
  }
}
```

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

\setnotation
_stex_notation_set_default:n

```
\cs_new_protected:Npn \setnotation #1 #2 {
  \stex_get_symbol:n{#1}
  \cs_if_exist:cTF{l_stex_notation_
    \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
    _#2_cs
  }{
    \tl_set_eq:Nc \l_stex_notation_macrocode_cs {l_stex_notation_
      \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
      _#2_cs
    }
    \cs_if_exist:cTF{l_stex_notation_
      \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
      _op_#2_cs
    }{
      \tl_set_eq:Nc \l_stex_key_op_tl {l_stex_notation_
        \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
        _op_#2_cs
      }
    }{
      \tl_clear:N \l_stex_key_op_tl
    }
    \_stex_notation_set_default:n{
      \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
    }
  }{
    \msg_error:nnxx{stex}{unknownnotation}{#2}{
      \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
```

```
    }
  }
}

\cs_new_protected:Nn \_stex_notation_set_default:n{
  \stex_if_in_module:TF{
    \stex_module_add_notation:eoeoo{#1}{}
      {\int_use:N \l_stex_get_symbol_arity_int}
      \l_stex_notation_macrocode_cs
      \l_stex_key_op_tl
  }{
    \cs_set_eq:cN {l_stex_notation_
      \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
      __cs}\l_stex_notation_macrocode_cs
    \tl_if_empty:NF \l_stex_key_op_tl {
      \cs_set_eq:cN{l_stex_notation_
      \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
      _op__cs}\l_stex_key_op_tl
    }
  }
}
```

(*End of definition for* `\setnotation` *and* `\_stex_notation_set_default:n`*. These functions are documented on page 91.*)

`\varnotation`

```
\stex_new_stylable_cmd:nnnn {varnotation} { s m O{} m} {
  \stex_keys_set:nn{notation}{#3}
  \stex_get_var:n{#2}
  \str_set_eq:NN \l_stex_key_name_str \l_stex_get_symbol_name_str
  \stex_notation_parse:n{#4}
  \stex_if_check_terms:T{ \_stex_notation_check: }
  \_stex_vardecl_notation_macro:
  \IfBooleanTF#1{
    \_stex_notation_set_default:n{\l_stex_get_symbol_name_str}
  }{}
  \group_begin:
  \tl_set_eq:NN \thisvarname \l_stex_get_symbol_name_str
  \tl_clear:N \thisstyle
  \str_set_eq:NN\thisnotationvariant\l_stex_key_variant_str
  \def\thisnotation{
    $\let\l_stex_current_symbol_str\thisvarname
      \def\comp{\_varcomp}\exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs{}}{
      \_stex_notation_make_args:
    }$
  }
  \stex_style_apply:
  \group_end:
}{}
```

(*End of definition for* `\varnotation`*. This function is documented on page 95.*)

`\symdef`

```
\stex_keys_define:nnnn{symdef}{}{}{decl,notation}
```

```
\cs_new_protected:Nn \_stex_symdef_styledefs: {
  \tl_set:Nx \thisdecluri {\l_stex_current_module_str ? \l_stex_key_name_str}
  \tl_set_eq:NN \thisdeclname \l_stex_key_name_str
  \tl_set_eq:NN \thistype \l_stex_key_type_tl
  \tl_set_eq:NN \thisdefiniens \l_stex_key_def_tl
  \tl_set_eq:NN \thisargs \l_stex_key_args_str
  \tl_clear:N \thisstyle
  \str_set_eq:NN\thisnotationvariant\l_stex_key_variant_str
  \def\thisnotation{
    $\let\l_stex_current_symbol_str\thisdecluri
      \def\comp{\_comp}\exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs}{
      \_stex_notation_make_args:
    }$
  }
}

\stex_new_stylable_cmd:nnnn {symdef} { m O{} m} {
  \stex_keys_set:nn{symdef}{#2}
  \str_set:Nx \l_stex_macroname_str { #1 }
  \stex_symdecl_top:n{#1}
  \stex_debug:nn{symdef}{Doing~\l_stex_current_module_str ? \l_stex_key_name_str}
  \str_set_eq:NN \l_stex_get_symbol_mod_str \l_stex_current_module_str
  \str_set_eq:NN \l_stex_get_symbol_name_str \l_stex_key_name_str
  \stex_notation_parse:n{#3}
  \stex_debug:nn{Here!}{\meaning\l_stex_notation_args_tl}
  \_stex_notation_check:
  \_stex_notation_add:
  \stex_if_do_html:T{
    \_stex_notation_do_html:n{\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
  }
  \stex_if_smsmode:F{
    \group_begin:
    \_stex_symdef_styledefs:
    \stex_style_apply:
    \group_end:
  }
  \stex_smsmode_do:
}{}

\stex_deactivate_macro:Nn \symdef {module~environments}
\stex_every_module:n {\stex_reactivate_macro:N \symdef}
\stex_sms_allow_escape:N \symdef
```

(*End of definition for* `\symdef`. *This function is documented on page* 88.)

`\stex_do_default_notation_op:`

```
\cs_new_protected:Nn \stex_do_default_notation: {
  \stex_do_default_notation_op:
  \tl_if_empty:NTF \l_stex_current_args_tl {
    \tl_clear:N \l__stex_notations_args_tl
  }{
    \__stex_notations_make_name:
    \tl_set_eq:NN \l_stex_get_symbol_args_tl \l_stex_current_args_tl
    \tl_set:Nx \l__stex_notations_args_tl {
```

```
      \_stex_map_args:N \__stex_notations_augment_arg:nn
    }
    \tl_put_right:Nn \l_stex_default_notation {\comp(}
    \seq_clear:N \l_tmpa_seq
    \int_step_inline:nn \l_stex_current_arity_str {
      \seq_put_right:Nn \l_tmpa_seq {#### ##1}
    }
    \tl_put_right:Nx \l_stex_default_notation {
      \seq_use:Nn \l_tmpa_seq {\mathpunct{\comp{,}}}
    }
    \tl_put_right:Nn \l_stex_default_notation {\comp)}
  }
  \tl_set:Nx \l_stex_default_notation {\STEXInternalNotation{}{0}{}{\l__stex_notations_args_
    \exp_args:No \exp_not:n \l_stex_default_notation
  }}
}

\cs_new:Nn \__stex_notations_augment_arg:nn {
  #1#2{0}{}
}

\cs_new_protected:Nn \__stex_notations_make_name: {
  \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq ? \l_stex_current_symbol_str
  \seq_pop_right:NN \l_tmpa_seq \l__stex_notations_name_str
  \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq / \l__stex_notations_name_str
  \seq_pop_right:NN \l_tmpa_seq \l__stex_notations_name_str
}

\cs_new_protected:Nn \stex_do_default_notation_op: {
  \__stex_notations_make_name:
  \tl_set:Nx \l_stex_default_notation {\exp_not:N \maincomp{ \exp_not:N \mathrm {\l__stex_no
}
```

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

`\STEXInternalNotation`

```
% 1: variant 2: operator precedence 3: intent 4: arguments 5: code 6: next

\cs_new_protected:Npn \STEXInternalNotation #1 #2 #3 #4 #5 #6 {
  \__stex_notations_process_notation:nnnnnn{#1}{#2}{#3}{#4}{#5}{
    \l__stex_notations_code_tl
    #6
  }
}

\cs_new_protected:Npn \__stex_notations_process_notation:nnnnnn #1 #2 #3 #4 {
  \tl_if_empty:nTF{#4}{
    \__stex_notations_simple:nnnnn{#1}{#2}{#3}
  }{
    \__stex_notations_complex:nnnnnn{#1}{#2}{#3}{#4}
  }
}

\cs_new_protected:Nn \__stex_notations_simple:nnnnn {
```

```
  \stex_debug:nn{Notation~code}{\tl_to_str:n{#4}}
  \tl_set:Nn \l__stex_notations_code_tl {
    \cs_set:Npn \l__stex_notations_cs {
      \_stex_maybe_brackets:nn{#2}{
        \_stex_term_oms_or_omv:nnn{#1}{#3}{#4}
      }
    }
    \l__stex_notations_cs
  }
  \stex_debug:nn{Here:Notation}{\meaning \l__stex_notations_code_tl }
  #5
}

\cs_new_protected:Nn \__stex_notations_complex:nnnnnn {
  \stex_debug:nn{Notation~code}{\tl_to_str:n{#5}}
  \int_zero:N \l_tmpa_int
  \tl_set:Nn \l__stex_notations_pre_tl {\cs_set_eq:NN \_stex_term_oma_or_omb:nnn \_stex_term
  \tl_set:Nn \l__stex_notations_code_tl {
    \cs_generate_from_arg_count:NNnn \l__stex_notations_cs \cs_set:Npn \l_tmpa_int
    {
      \_stex_maybe_brackets:nn{#2}{
        \_stex_term_oma_or_omb:nnn{#1}{#3}{
          \bool_set_false:N \l_stex_brackets_dones_bool
          #5
        }
      }
    }
    \l__stex_notations_cs
  }
  \tl_set:Nn \l__stex_notations_after_tl{
    \exp_args:NNo
    \tl_put_left:Nn \l__stex_notations_code_tl \l__stex_notations_pre_tl
    \tl_put_left:Nx \l__stex_notations_code_tl {
      \int_set:Nn \l_tmpa_int {\int_use:N \l_tmpa_int}
    }
    \stex_debug:nn{Here:Notation}{\meaning \l__stex_notations_code_tl }
    #6
  }
  \__stex_notations_parse_notation_args:nnnnw #4 \__stex_notations_args_end:
}

\cs_new_protected:Npn \__stex_notations_parse_notation_args:nnnnw #1 #2 #3 #4 #5 \__stex_not
  \tl_if_empty:nTF{#5}{
    \__stex_notations_add_last:nnnnn{#1}{#2}{#3}{#4}
  }{
    \__stex_notations_add_next:nnnnnn{#1}{#2}{#3}{#4}{#5}
  }
}

\cs_new_protected:Nn \__stex_notations_add_next:nnnnnn {
  \__stex_notations_add:nnnnn{#1}{#2}{#3}{#4}{#6}
  \__stex_notations_parse_notation_args:nnnnw #5 \__stex_notations_args_end:
}
```

```
\cs_new_protected:Nn \__stex_notations_add_last:nnnnn {
  \__stex_notations_add:nnnnn{#1}{#2}{#3}{#4}{#5}
  \l__stex_notations_after_tl
}

\cs_new_protected:Nn \__stex_notations_add:nnnnn {
  \int_incr:N \l_tmpa_int
  \str_case:nn{#2}{
    i {
      \tl_put_right:Nn \l__stex_notations_code_tl {
        {\_stex_term_arg:nnnnn{#1}{#2}{#3}{#4}{#5}}
      }
    }
    b {
      \tl_set:Nn \l__stex_notations_pre_tl {
        \cs_set_eq:NN \_stex_term_oma_or_omb:nnn \_stex_term_omb:nnn
      }
      \tl_put_right:Nn \l__stex_notations_code_tl {
        {\_stex_term_arg:nnnnn{#1}{#2}{#3}{#4}{#5}}
      }
    }
    a {
      \tl_put_right:Nn \l__stex_notations_code_tl {
        {\_stex_term_arg_aB:nnnnn{#1}{#2}{#3}{#4}{#5}}
      }
    }
    B  {
      \tl_set:Nn \l__stex_notations_pre_tl {
        \cs_set_eq:NN \_stex_term_oma_or_omb:nnn \_stex_term_omb:nnn
      }
      \tl_put_right:Nn \l__stex_notations_code_tl {
        {\_stex_term_arg_aB:nnnnn{#1}{#2}{#3}{#4}{#5}}
      }
    }
  }
}
```

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

a/B-mode argument handling

\argsep

```
\cs_new_protected:Nn \__stex_notations_check_aB_arg:Nn {
  \exp_args:Ne \cs_if_eq:NNF {\tl_head:n{#2}}
    \_stex_term_arg_aB:nnnnn {
    \msg_error:nnx{stex}{error/assocarg}{\tl_to_str:n{#1}}
  }
}

\cs_new_protected:Npn \argsep #1 #2 {
  \__stex_notations_check_aB_arg:Nn\argsep{#1}
  \stex_pseudogroup_with:nn{\_stex_term_do_aB_clist:}{
    \tl_set:Nn \_stex_term_do_aB_clist: {
      \seq_use:Nn \l_stex_aB_args_seq {#2}
```

```
    }
    #1
  }
}
```

(*End of definition for* `\argsep`. *This function is documented on page 92.*)

`\argmap`

```
\cs_new_protected:Npn \argmap #1 #2 #3 {
  \__stex_notations_check_aB_arg:Nn\argmap{#1}
  \stex_pseudogroup_with:nn{
    \_stex_term_do_aB_clist:
    \__stex_notations_map_cs:
  }{
    \cs_set:Npn \__stex_notations_map_cs: ##1 { #2 }
    \tl_set:Nn \_stex_term_do_aB_clist: {
      \seq_clear:N \l_tmpa_seq
      \seq_map_inline:Nn \l_stex_aB_args_seq {
        \tl_if_eq:nnTF{##1}{\ellipses}{
          \seq_put_right:Nn \l_tmpa_seq \ellipses
        }{
          \seq_put_right:Nx \l_tmpa_seq {
            \exp_after:wN \exp_not:n \exp_after:wN { \__stex_notations_map_cs: {##1} }
          }
        }
      }
      \seq_set_eq:NN \l_stex_aB_args_seq \l_tmpa_seq
      \seq_use:Nn \l_stex_aB_args_seq {#3}
    }
    #1
  }
}
```

(*End of definition for* `\argmap`. *This function is documented on page 92.*)

`\argarraymap`

```
\int_new:N \l__stex_notations_clist_count_int
\cs_new_protected:Npn \argarraymap #1 #2 #3 #4 {
  \__stex_notations_check_aB_arg:Nn\argarraymap{#1}
  \stex_pseudogroup_with:nn{
    \_stex_term_do_aB_clist:
    \__stex_notations_map_cs:
  }{
    \cs_set:Npn \__stex_notations_map_cs: ##1 { #3 }
    \int_set:Nn \l__stex_notations_clist_count_int {\exp_args:No\clist_count:n{\tl_to_str:n{
    \tl_set:Nn \_stex_term_do_aB_clist: {
      \tl_clear:N \l_tmpa_tl
      \int_zero:N \l_tmpa_int
      \seq_map_inline:Nn \l_stex_aB_args_seq {
        \int_incr:N \l_tmpa_int
        \int_compare:nNnT \l_tmpa_int > \l__stex_notations_clist_count_int {
          \int_set:Nn \l_tmpa_int 1
        }
        \tl_put_right:Nx \l_tmpa_tl {
          \exp_after:wN \exp_not:n \exp_after:wN { \__stex_notations_map_cs: {##1} }
```

```
            \clist_item:nn{#4}\l_tmpa_int
          }
        }
        \seq_set_eq:NN \l_stex_aB_args_seq \l_tmpa_seq
        \begin{array}{#2}
            \l_tmpa_tl
        \end{array}
      }
      #1
    }
}
```

(*End of definition for* `\argarraymap`. *This function is documented on page 92.*)

13.8.3 Variables

⟨@@=stex_vars⟩

`\vardef`

```
\tl_new:N \l_stex_variables_prop
\bool_new:N \l__stex_vars_bind_bool
\cs_new_protected:Nn \_stex_variable:nnnnnnnN {}
\stex_keys_define:nnnn{vardef}{
  \bool_set_false:N \l__stex_vars_bind_bool
}{
  bind  .bool_set:N  = \l__stex_vars_bind_bool
}{symdef}

\stex_new_stylable_cmd:nnnn {vardef} { m O{} m} {
  \stex_keys_set:nn{vardef}{#2}
  \str_set:Nx \l_stex_macroname_str { #1 }
  \str_if_empty:NT \l_stex_key_name_str {
    \str_set:Nx \l_stex_key_name_str { #1 }
  }

  \stex_symdecl_do:
  \_stex_symdecl_check_terms:
  \__stex_vars_add:
  \__stex_vars_macro:
  \stex_if_do_html:T \__stex_vars_html:

  \int_set:Nn \l_stex_get_symbol_arity_int {\l_stex_get_symbol_arity_int}
  \stex_debug:nn{vardef}{Doing~\l_stex_key_name_str}
  \tl_set_eq:NN \l_stex_get_symbol_return_tl \l_stex_key_return_tl
  \stex_notation_parse:n{#3}
  \stex_if_check_terms:T{ \_stex_notation_check: }
  \_stex_vardecl_notation_macro:
  \stex_if_do_html:T {
    \def\comp{\_varcomp}
    \_stex_notation_do_html:n \l_stex_key_name_str
  }
  \group_begin:
  \tl_set_eq:NN \thisvarname \l_stex_key_name_str
  \tl_clear:N \thisstyle
  \str_set_eq:NN\thisnotationvariant\l_stex_key_variant_str
```

```
  \def\thisnotation{
    $\let\l_stex_current_symbol_str\thisvarname
      \def\comp{\_varcomp}\exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs{}}{
      \_stex_notation_make_args:
    }$
  }
  \stex_style_apply:
  \group_end:\ignorespaces
}{}

\cs_new_protected:Nn \__stex_vars_add: {
  \exp_args:NNNo \exp_args:NNnx
  \prop_put:Nnn \l_stex_variables_prop \l_stex_key_name_str {
    {\l_stex_macroname_str}
    {\l_stex_key_name_str}
    {\int_use:N \l_stex_get_symbol_arity_int}
    {\l_stex_get_symbol_args_tl}
    {\exp_args:No \exp_not:n \l_stex_key_def_tl}
    {\exp_args:No \exp_not:n \l_stex_key_type_tl}
    {\exp_args:No \exp_not:n \l_stex_key_return_tl}
    \stex_invoke_symbol:
  }
}

\cs_new_protected:Nn \__stex_vars_macro: {
  \tl_set:cx{\l_stex_macroname_str}{
    \_stex_invoke_variable:nnnnnnN
      {\l_stex_key_name_str}
      {\int_use:N \l_stex_get_symbol_arity_int}
      {\l_stex_get_symbol_args_tl}
      {\exp_args:No \exp_not:n \l_stex_key_def_tl}
      {\exp_args:No \exp_not:n \l_stex_key_type_tl}
      {\exp_args:No \exp_not:n \l_stex_key_return_tl}
      \stex_invoke_symbol:
  }
}

\cs_new_protected:Nn \__stex_vars_html: {
  \stex_if_do_html:T {
    \hbox\bgroup\exp_args:Ne \stex_annotate_invisible:nn {
      shtml:vardef = {\l_stex_key_name_str},
      shtml:args = {\l_stex_key_args_str}
      \str_if_empty:NF \l_stex_macroname_str {,
        shtml:macroname={\l_stex_macroname_str}
      }
      \str_if_empty:NF \l_stex_key_assoc_str {,
        shtml:assoctype={\l_stex_key_assoc_str}
      }
      \str_if_empty:NF \l_stex_key_role_str {,
        shtml:role={\l_stex_key_role_str}
      }
      \str_if_empty:NF \l_stex_key_reorder_str {,
        shtml:reorderargs={\l_stex_key_reorder_str}
      }
```

```
      \bool_if:NT \l__stex_vars_bind_bool {,
        shtml:bind={}
      }
    }{
      \_stex_annotate_force_break:n{
        \bool_set_true:N \stex_in_invisible_html_bool
        \tl_if_empty:NF \l_stex_key_type_tl {
          \stex_annotate:nn{shtml:type={}}{$\l_stex_key_type_tl$}
        }
        \tl_if_empty:NF \l_stex_key_def_tl {
          \stex_annotate:nn{shtml:definiens={}}{$\l_stex_key_def_tl$}
        }
        \tl_if_empty:NF \l_stex_key_return_tl{
          \exp_args:Nno \use:n{
          \cs_generate_from_arg_count:NNnn \l__stex_vars_cs
          \cs_set:Npn \l_stex_get_symbol_arity_int} \l_stex_key_return_tl
          \tl_set:Nx \l__stex_vars_args_tl {\_stex_map_args:N \_stex_return_args:nn}
          $\stex_annotate:nn{shtml:returntype={}}{
            \exp_after:wN \l__stex_vars_cs \l__stex_vars_args_tl!}$
        }
        \tl_if_empty:NF \l_stex_key_argtypes_clist {
          \stex_annotate:nn{shtml:argtypes={}}{
            \_stex_annotate_force_break:n{
              \clist_map_inline:Nn \l_stex_key_argtypes_clist {
                $\stex_annotate:nn{shtml:type={}}{##1}$
              }
            }
          }
        }
      }
    }\egroup
  }
}
```

(*End of definition for* `\vardef`. *This function is documented on page 95.*)

`\_stex_vardecl_notation_macro:`

```
\cs_new_protected:Nn \_stex_vardecl_notation_macro: {
  \tl_set_eq:cN {l_stex_notation_
    \l_stex_key_name_str _
    \l_stex_key_variant_str _cs
  }\l_stex_notation_macrocode_cs
  \cs_if_exist:cF {l_stex_notation_\l_stex_key_name_str __cs}{
    \tl_set_eq:cN{l_stex_notation_\l_stex_key_name_str __cs}
      \l_stex_notation_macrocode_cs
  }
  \tl_if_empty:NF \l_stex_key_op_tl {
    \tl_set_eq:cN {l_stex_notation_\l_stex_key_name_str _op_
    \l_stex_key_variant_str _cs}\l_stex_key_op_tl
    \cs_if_exist:cF {l_stex_notation_\l_stex_key_name_str _op__cs}{
      \cs_set_eq:cN{l_stex_notation_\l_stex_key_name_str _op__cs}
      \l_stex_key_op_tl
    }
  }
}
```

(*End of definition for* `\_stex_vardecl_notation_macro:`. *This function is documented on page* 125.)

`\stex_get_symbol_or_var:n`
`\stex_get_var:n`

```
\cs_new_protected:Nn \__stex_vars_set_vars:nnnnnnN {
  \stex_debug:nn{symbols}{Variable~#1~found}
  \cs_set:Npn \_stex_variable:nnnnnnnN ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 {}
  \str_clear:N \l_stex_get_symbol_mod_str
  \str_set:Nn \l_stex_get_symbol_name_str {#1}
  \int_set:Nn \l_stex_get_symbol_arity_int {#2}
  \tl_set:Nn \l_stex_get_symbol_args_tl {#3}
  \tl_set:Nn \l_stex_get_symbol_def_tl {#4}
  \tl_set:Nn \l_stex_get_symbol_type_tl {#5}
  \tl_set:Nn \l_stex_get_symbol_return_tl {#6}
  \tl_set:Nn \l_stex_get_symbol_invoke_cs {#7}
}

\cs_new_protected:Nn \__stex_vars_get_var:n {
  \prop_map_inline:Nn \l_stex_variables_prop {
    \__stex_vars_check_var:nnnnnnnnN {#1} ##2
  }
}

\cs_new_protected:Nn \__stex_vars_check_var:nnnnnnnnN {
  \str_if_eq:nnTF{#1}{#2}{
    \prop_map_break:n{\__stex_vars_set_vars:nnnnnnN {#3}{#4}{#5}{#6}{#7}{#8}#9}
  }{
    \str_if_eq:nnT{#1}{#3}{
      \prop_map_break:n{\__stex_vars_set_vars:nnnnnnN {#3}{#4}{#5}{#6}{#7}{#8}#9}
    }
  }
}

\cs_new_protected:Nn \stex_get_var:n {
  \str_clear:N \l_stex_get_symbol_name_str
  \__stex_vars_get_var:n{#1}
  \str_if_empty:NT \l_stex_get_symbol_name_str {
    \msg_error:nnn{stex}{error/unknownsymbol}{#1}
  }
}

\cs_new_protected:Nn \stex_get_symbol_or_var:n {
  \str_clear:N \l_stex_get_symbol_name_str
  \__stex_vars_get_var:n{#1}
  \str_if_empty:NT \l_stex_get_symbol_name_str {
    \stex_debug:nn{symbols}{No~variable~#1~found}
    \stex_get_symbol:n{#1}
  }
}
```

(*End of definition for* `\stex_get_symbol_or_var:n` *and* `\stex_get_var:n`. *These functions are documented on page* 125.)

`\svar`

```
\NewDocumentCommand \svar {O{} m}{
```

```
  \group_begin:
    \tl_if_empty:nTF{#1}{
      \str_set:Nn \l_stex_current_symbol_str {#2}
    }{
      \str_set:Nn \l_stex_current_symbol_str {#1}
    }
    \bool_if:NTF \l_stex_allow_semantic_bool{
        \tl_clear:N \l_stex_current_term_tl
        \_stex_term_omv:nnn{}{}{\_varcomp{#2}}
    }{
      \msg_error:nnxx{stex}{error/notallowed}{Variable}{\l_stex_current_symbol_str}
    }
  \group_end:
}
```

(*End of definition for* `\svar`. *This function is documented on page* 95.)

13.8.4 Sequences

⟨@@=stex_seqs⟩

`\varseq`

```
\stex_new_stylable_cmd:nnnn {varseq}{m O{} m m} {
  \stex_keys_set:nn{symdef}{#2}
  \str_set:Nx \l_stex_macroname_str { #1 }
  \str_if_empty:NT \l_stex_key_name_str {
    \str_set:Nx \l_stex_key_name_str { #1 }
  }
  \str_if_empty:NT \l_stex_key_args_str {
    \str_set:Nn \l_stex_key_args_str {1}
  }
  \stex_symdecl_do:

  \tl_set_eq:NN \l_stex_get_symbol_return_tl \l_stex_key_return_tl
  \clist_set:Nn \l__stex_seqs_range_clist {#3}
  \tl_if_empty:NTF \l_stex_key_op_tl {
    \stex_notation_parse:n{#4}
    \tl_clear:N \l_stex_key_op_tl
  }{
    \stex_notation_parse:n{#4}
  }
  \stex_if_do_html:T \__stex_seqs_html:
  \stex_if_check_terms:T \__stex_seqs_check_terms:
  \__stex_seqs_add:
  \__stex_seqs_macro:
  \stex_if_check_terms:T \_stex_notation_check:
  \_stex_vardecl_notation_macro:
  \group_begin:
  \tl_set_eq:NN \thisvarname \l_stex_key_name_str
  \tl_clear:N \thisstyle
  \str_set_eq:NN\thisnotationvariant\l_stex_key_variant_str
  \def\thisnotation{
    $\let\l_stex_current_symbol_str\thisvarname
      \def\comp{\_varcomp}\exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs{}}{
      \__stex_seqs_make_args:
```

```
    }$
  }
  \stex_style_apply:
  \group_end:\ignorespaces
}{}

\cs_new_protected:Nn \__stex_seqs_add: {
  \exp_args:NNNo \exp_args:NNnx
  \prop_put:Nnn \l_stex_variables_prop \l_stex_key_name_str {
    {\l_stex_macroname_str}
    {\l_stex_key_name_str}
    {\int_use:N \l_stex_get_symbol_arity_int}
    {\l_stex_get_symbol_args_tl}
    {\exp_args:No \exp_not:n \l_stex_key_def_tl}
    {\exp_args:No \exp_not:n \l__stex_seqs_range_clist}
    {\exp_args:No \exp_not:n \l_stex_key_return_tl}
    \stex_invoke_sequence:
  }
}

\cs_new_protected:Nn \__stex_seqs_macro: {
  \tl_set:cx{\l_stex_macroname_str}{
    \_stex_invoke_variable:nnnnnnN
      {\l_stex_key_name_str}
      {\int_use:N \l_stex_get_symbol_arity_int}
      {\l_stex_get_symbol_args_tl}
      {\exp_args:No \exp_not:n \l_stex_key_def_tl}
      {\exp_args:No \exp_not:n \l__stex_seqs_range_clist}
      {\exp_args:No \exp_not:n \l_stex_key_return_tl}
      \stex_invoke_sequence:
  }
}

\cs_new_protected:Nn \__stex_seqs_make_args: { \TODO }
\cs_new_protected:Nn \__stex_seqs_check_terms: { \TODO }

\cs_new_protected:Nn \__stex_seqs_html: {
    \exp_args:Ne \stex_annotate_invisible:nn {
      shtml:varseq = {\l_stex_key_name_str},
      shtml:args = {\l_stex_key_args_str}
      \str_if_empty:NF \l_stex_macroname_str {,
        shtml:macroname={\l_stex_macroname_str}
      }
      \str_if_empty:NF \l_stex_key_assoc_str {,
        shtml:assoctype={\l_stex_key_assoc_str}
      }
      \str_if_empty:NF \l_stex_key_role_str {,
        shtml:role={\l_stex_key_role_str}
      }
      \str_if_empty:NF \l_stex_key_reorder_str {,
        shtml:reorderargs={\l_stex_key_reorder_str}
      }
    }{\hbox\bgroup
      \_stex_annotate_force_break:n{
```

```
        \tl_if_empty:NF \l_stex_key_type_tl {
          \stex_annotate:nn{shtml:type={}}{$\l_stex_key_type_tl$}
        }
        \tl_if_empty:NF \l_stex_key_def_tl {
          \stex_annotate:nn{shtml:definiens={}}{$\l_stex_key_def_tl$}
        }
        \tl_if_empty:NF \l_stex_key_return_tl{
          \exp_args:Nno \use:n{
          \cs_generate_from_arg_count:NNnn \l__stex_seqs_cs
          \cs_set:Npn \l_stex_get_symbol_arity_int} \l_stex_key_return_tl
          \tl_set:Nx \l__stex_seqs_args_tl {\_stex_map_args:N \_stex_return_args:nn}
          \stex_annotate:nn{shtml:returntype={}}{
            $\exp_after:wN \l__stex_seqs_cs \l__stex_seqs_args_tl!$}
        }
        \tl_if_empty:NF \l_stex_key_argtypes_clist {
          \stex_annotate:nn{shtml:argtypes={}}{
            \_stex_annotate_force_break:n{
              \clist_map_inline:Nn \l_stex_key_argtypes_clist {
                \stex_annotate:nn{shtml:type={}}{$##1$}
              }
            }
          }
        }
      }
    \egroup}
}
```

(*End of definition for* \varseq. *This function is documented on page 95.*)

\stex_invoke_sequence:

```
\cs_new_protected:Nn \stex_invoke_sequence: {
  \peek_charcode_remove:NTF ! {
    \peek_charcode:NTF [ \__stex_seqs_do_op:w { \__stex_seqs_do_op:w [] }
  }\__stex_seqs_do_first:
}

\cs_new_protected:Npn \__stex_seqs_do_op:w [#1] {
  \cs_if_exist:cTF {l_stex_notation_\l_stex_current_symbol_str _op_#1_cs}{
    \_stex_maybe_brackets:nn{\neginfprec}{
      \_stex_term_oms_or_omv:nnn{#1}{}
      {\use:c{l_stex_notation_\l_stex_current_symbol_str _op_#1_cs}}
    }
    \group_end:
  }{
    \__stex_seqs_get_index_notation:n{#1}
    \peek_charcode:NTF [ \__stex_seqs_doop_range:w { \__stex_seqs_doop_range:w[] }
  }
}

\cs_new_protected:Npn \__stex_seqs_doop_range:w [#1] {
  \bool_set_true:N \l_stex_allow_semantic_bool
  \clist_clear:N \l__stex_seqs_clist
  \clist_map_function:NN \l_stex_current_type_tl \__stex_seqs_doop_arg:n
    \stex_annotate:nn{
```

```
      shtml:term=OMV,
      shtml:head={\l_stex_current_symbol_str},
      shtml:notationid={}
    }{
      \l__stex_seqs_clist
    }
    \group_end:
}

\cs_new_protected:Nn \__stex_seqs_doop_arg:n {
  \tl_if_eq:nnTF{#1}{\ellipses}{
    \clist_put_right:Nn \l__stex_seqs_clist {
       \ellipses
    }
  }{
    \clist_put_right:Nn \l__stex_seqs_clist {
      \exp_args:No \str_if_eq:nnTF \l_stex_current_arity_str {1}{
        \group_begin:
          \l__stex_seqs_cs \group_end: {#1}

      }{
        \group_begin:
          \l__stex_seqs_cs \group_end: #1
      }
    }
  }
}

\cs_new_protected:Nn \__stex_seqs_get_index_notation:n {
  \cs_if_exist:cTF {l_stex_notation_\l_stex_current_symbol_str _#1_cs}{
    \cs_set_eq:Nc \l__stex_seqs_cs {l_stex_notation_\l_stex_current_symbol_str _#1_cs}
  }{
    \stex_do_default_notation:
    \cs_set_eq:NN \l__stex_seqs_cs \l_stex_default_notation
  }
}


\cs_new:Nn \__stex_seqs_do_first_arg:n {{\exp_not:n{## #1}}}

\cs_new_protected:Nn \__stex_seqs_do_first: {
  \exp_args:Nnx \use:nn{
  \cs_generate_from_arg_count:NNnn \l__stex_seqs_cs \cs_set:Npn
  \l_stex_current_arity_str} {{
    \tl_set:Nn \exp_not:N \l__stex_seqs_first_args_tl {
      \int_step_function:nN \l_stex_current_arity_str \__stex_seqs_do_first_arg:n
    }
    \exp_not:N \__stex_seqs_do_first_next:
  }}
  \l__stex_seqs_cs
}

\cs_new_protected:Nn \__stex_seqs_do_first_next: {
  \peek_charcode_remove:NTF ! {
```

```
    \peek_charcode:NTF [ \__stex_seqs_do_one:w {\__stex_seqs_do_one:w []}
  }{
    \peek_charcode:NTF [ \__stex_seqs_do_all:w {\__stex_seqs_do_all:w []}
  }
}

\cs_new_protected:Npn \__stex_seqs_do_one:w [#1] {
  \__stex_seqs_get_index_notation:n{#1}
  \stex_debug:nn{HERE~seq~one}{\meaning\l__stex_seqs_cs^^J\meaning\l__stex_seqs_first_args_t
  \exp_args:Nno\use:nn{\l__stex_seqs_cs\group_end:}\l__stex_seqs_first_args_tl
}

\cs_new_protected:Npn \__stex_seqs_do_all:w [#1] {
  \stex_debug:nn{HERE~seq~all}{\meaning\l__stex_seqs_first_args_tl}
  \exp_args:Nno\use:nn{\_stex_invoke_notation:w [#1]}\l__stex_seqs_first_args_tl
}
```

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

`\seqmap`

```
\cs_new_protected:Npn \seqmap #1 #2 {
  \symuse{Metatheory?sequence~expression}{\seqmap{#1}{#2}}%\l_tmpa_tl {#2}
}
```

(*End of definition for* `\seqmap`. *This function is documented on page* 96.)

13.8.5 Expressions

⟨@@=stex_expr⟩

Various variables:

```
\bool_new:N \l_stex_allow_semantic_bool
\bool_set_true:N \l_stex_allow_semantic_bool
\tl_new:N \l_stex_current_term_tl
\tl_set:Nn \l_stex_every_symbol_tl {
  \bool_set_false:N \l_stex_allow_semantic_bool
}
```

`\_stex_next_symbol:n`

```
\tl_new:N \l__stex_expr_reset_tl
\cs_new_protected:Nn \_stex_next_symbol:n {
  \tl_set:Nx \l_stex_every_symbol_tl {
    \tl_gset:Nn \exp_not:N \l__stex_expr_reset_tl {
      \tl_set:Nn \exp_not:N \l_stex_every_symbol_tl  {
        \exp_args:No \exp_not:n \l_stex_every_symbol_tl
      }
      \tl_gset:Nn \exp_not:N \l__stex_expr_reset_tl {
        \exp_args:No \exp_not:n \l__stex_expr_reset_tl
      }
    }
    \tl_set:Nn \exp_not:N \l_stex_every_symbol_tl  {
      \exp_args:No \exp_not:n \l_stex_every_symbol_tl
    }
    \exp_not:n{ \aftergroup \l__stex_expr_reset_tl }
    \exp_not:N \l_stex_every_symbol_tl
```

```
    \exp_not:n{ #1 }
  }
}
\cs_generate_variant:Nn \_stex_next_symbol:n {e}
```

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

`\STEXinvisible`

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

(*End of definition for* `\STEXinvisible`. *This function is documented on page 81.*)

Invoking Semantic Macros

`\_stex_invoke_symbol:nnnnnnnN`

```
\cs_new_protected:Nn \_stex_invoke_symbol:nnnnnnnN {
  \bool_if:NTF \l_stex_allow_semantic_bool{
    \stex_if_html_backend:T{\ifvmode\indent\fi}
    \__stex_expr_setup:nnnnnn{\_comp}{#1?#2}{#3}{#4}{#7}{#6}
    \cs_set_eq:NN \_stex_term_oms_or_omv:nnn \_stex_term_oms:nnn
    \tl_put_right:Nn \l_stex_current_redo_tl{
      \cs_set_eq:NN \_stex_term_oms_or_omv:nnn \_stex_term_oms:nnn
    }
    #8
  }{
    \msg_error:nnxx{stex}{error/notallowed}{#1?#2}{\l_stex_current_symbol_str}
  }
}
\cs_generate_variant:Nn \_stex_invoke_symbol:nnnnnnnN {ooxooooN}

\cs_new_protected:Nn \__stex_expr_setup:nnnnnn {
  \group_begin:
  \tl_clear:N \l_stex_return_notation_tl
  \tl_set:Nn \l_stex_current_redo_tl {
    \let \this \stex_current_this:
    \def\comp{#1}
    \def\maincomp{\comp}
    \str_set:Nn \l_stex_current_symbol_str {#2}
    \str_set:Nn \l_stex_current_arity_str{ #3 }
    \tl_set:Nn \l_stex_current_args_tl{ #4 }
    \tl_set:Nn \l_stex_current_return_tl{ #5 }
    \tl_set:Nn \l_stex_current_type_tl{ #6 }
    \tl_clear:N \l_stex_current_term_tl
  }
  \tl_put_right:Nx \l_stex_current_redo_tl {
    \exp_args:No \exp_not:n \l_stex_every_symbol_tl
  }
  \l_stex_current_redo_tl
}
```

(*End of definition for* `\_stex_invoke_symbol:nnnnnnnN`. *This function is documented on page* **??**.)

_stex_invoke_variable:nnnnnn

```
\cs_new_protected:Nn \_stex_invoke_variable:nnnnnnN {
  \bool_if:NTF \l_stex_allow_semantic_bool{
    \stex_if_html_backend:T{\ifvmode\indent\fi}
    \__stex_expr_setup:nnnnnn{\_varcomp}{#1}{#2}{#3}{#6}{#5}
    \cs_set_eq:NN \_stex_term_oms_or_omv:nnn \_stex_term_omv:nnn
    \tl_put_right:Nn \l_stex_current_redo_tl {
      \cs_set_eq:NN \_stex_term_oms_or_omv:nnn \_stex_term_omv:nnn
    }
    #7
  }{
    \msg_error:nnxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
  }
}
```

(*End of definition for* `\_stex_invoke_variable:nnnnnn`. *This function is documented on page* **??**.)

\symuse

```
\cs_new_protected:Npn \symuse #1 {
  \stex_get_symbol:n{#1}
  \exp_args:Nno \use:n {\_stex_invoke_symbol:ooxooooN
  \l_stex_get_symbol_mod_str
  \l_stex_get_symbol_name_str
  {\int_use:N \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
}
```

(*End of definition for* `\symuse`. *This function is documented on page 89.*)

\stex_invoke_symbol: Top-Level: Check whether text/math mode or custom notation, whether delimited by !, return code etc.

```
\cs_new_protected:Nn \stex_invoke_symbol: {
  \stex_debug:nn{expressions}{Invoking~\l_stex_current_symbol_str}
  \mode_if_math:TF \__stex_expr_invoke_math: \__stex_expr_invoke_text:
}

\cs_new_protected:Nn \__stex_expr_invoke_text: {
  \stex_debug:nn{expressions}{text~mode}
  \peek_charcode_remove:NTF ! \__stex_expr_invoke_op_custom:n \__stex_expr_invoke_custom:n
}

\cs_new_protected:Nn \__stex_expr_invoke_math: {
  \stex_debug:nn{expressions}{math~mode}
  \peek_charcode_remove:NTF ! {
    % operator
    \peek_charcode_remove:NTF * \__stex_expr_invoke_op_custom:n {
      % op notation
      \peek_charcode:NTF [ \__stex_expr_invoke_op_notation:w {
        \__stex_expr_invoke_op_notation:w []
      }
```

```
    }
  }{
    \peek_charcode_remove:NTF * \__stex_expr_invoke_custom:n {
      % normal
      \peek_charcode:NTF [ \_stex_invoke_notation:w {
        \_stex_invoke_notation:w []
      }
    }
  }
}
```

Notations:

```
\cs_new_protected:Npn \_stex_invoke_notation:w [#1] {
  \stex_debug:nn{expressions}{using~notation~#1~for~\l_stex_current_symbol_str}
  \cs_if_exist:cTF{l_stex_notation_\l_stex_current_symbol_str _#1_cs}{
    \tl_if_empty:NTF \l_stex_current_return_tl {
      \stex_debug:nn{expressions}{return~empty}
      \use:c{l_stex_notation_\l_stex_current_symbol_str _#1_cs}{\group_end:\_stex_eat_exclam
    }{
      \stex_debug:nn{expressions}{return?}
      \exp_after:wN\exp_after:wN\exp_after:wN
      \__stex_expr_invoke_return_maybe:n
      \exp_after:wN\exp_after:wN\exp_after:wN
      {\cs:w l_stex_notation_\l_stex_current_symbol_str _#1_cs \cs_end: {}}
    }
  }{
    \stex_do_default_notation:
    \tl_if_empty:NTF \l_stex_current_return_tl {
      \l_stex_default_notation{\group_end:\_stex_eat_exclamation_point:}
    }{
      \exp_after:wN
      \__stex_expr_invoke_return_maybe:n
      \exp_after:wN
      {\l_stex_default_notation {}}
    }
  }
}

\cs_new_protected:Npn \__stex_expr_invoke_op_notation:w [#1] {
  \stex_debug:nn{expressions}{op~notation~for~\l_stex_current_symbol_str}
  \cs_if_exist:cTF{l_stex_notation_\l_stex_current_symbol_str _op_#1_cs}{
    \_stex_maybe_brackets:nn{\neginfprec}{
      \_stex_term_oms_or_omv:nnn{#1}{}
      {\use:c{l_stex_notation_\l_stex_current_symbol_str _op_#1_cs}}
    }
    \group_end:
  }{
    \int_compare:nNnTF \l_stex_current_arity_str = 0 {
      \tl_clear:N \l_stex_current_return_tl
      \_stex_invoke_notation:w [#1]
    }{
      \stex_do_default_notation_op:
      \_stex_maybe_brackets:nn{\neginfprec}{
        \_stex_term_oms_or_omv:nnn{#1}{}
```

```
        {\l_stex_default_notation}
      }
      \group_end:
    }
  }
}
```

Return:

```
\cs_new_protected:Nn \__stex_expr_invoke_return_maybe:n {
  \tl_clear:N \l__stex_expr_return_args_tl
  \tl_set:Nn \l__stex_expr_return_this_tl {#1}
  \exp_args:Nnx \use:n {
  \cs_generate_from_arg_count:NNnn \__stex_expr_ret_cs
    \cs_set:Npn \l_stex_current_arity_str } {
      \int_step_function:nN \l_stex_current_arity_str \__stex_expr_return_arg:n
      \__stex_expr_invoke_return_next:
    }
  \__stex_expr_ret_cs
}

\cs_new:Nn \__stex_expr_return_arg:n {
  \tl_put_right:Nn \exp_not:N \l__stex_expr_return_args_tl {{#### #1}}
}

\cs_new_protected:Nn \__stex_expr_invoke_return_next: {
  \peek_charcode_remove:NTF ! {
    \exp_after:wN \l__stex_expr_return_this_tl \l__stex_expr_return_args_tl \group_end:
  }\__stex_expr_invoke_return:
}

\cs_new_protected:Nn \__stex_expr_invoke_return: {
  \tl_set:Nx \l__stex_expr_return_this_tl {
    \__stex_expr_return_notation:n {
      \exp_after:wN \exp_after:wN \exp_after:wN
      \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
        \exp_after:wN \l__stex_expr_return_this_tl \l__stex_expr_return_args_tl
      }
    }
  }
  \stex_debug:nn{return}{Notation:~\meaning\l__stex_expr_return_this_tl}
  \tl_put_left:Nx \l__stex_expr_return_this_tl {
    \group_begin:\exp_args:No \exp_not:n \l_stex_current_redo_tl
  }
  \exp_args:Nnx \use:n {
  \cs_generate_from_arg_count:NNnn \__stex_expr_ret_cs
    \cs_set:Npn \l_stex_current_arity_str } {
      \exp_args:No \exp_not:n \l_stex_current_return_tl
    }
  \stex_debug:nn{return}{
    \meaning\__stex_expr_ret_cs^^J
    \meaning\l__stex_expr_return_this_tl^^J
    \exp_args:No \exp_not:n \l__stex_expr_return_args_tl^^J
  }
  \exp_args:Nnx \use:nn {
```

```
    \exp_after:wN \group_end: \__stex_expr_ret_cs
  }{
    \exp_args:No \exp_not:n \l__stex_expr_return_args_tl
    {
      \exp_args:No \exp_not:n \l__stex_expr_return_this_tl
      \group_end:
    }
  }
}


\cs_new_protected:Nn \__stex_expr_return_notation:n {
  \tl_if_empty:NTF \l_stex_return_notation_tl { #1 }{
    \l_stex_return_notation_tl
  }
}
```

Custom Notations:

```
\cs_new_protected:Nn \__stex_expr_invoke_op_custom:n {
  \stex_debug:nn{expressions}{custom~op}
  \bool_set_true:N \l_stex_allow_semantic_bool
  \_stex_term_oms_or_omv:nnn{}{}{\maincomp{#1}}
  \group_end:
}

\int_new:N \l__stex_expr_arg_counter_int
\cs_new_protected:Nn \__stex_expr_invoke_custom:n {
  \stex_debug:nn{custom}{custom~notation~for~\l_stex_current_symbol_str}
  \stex_pseudogroup:nn{
    \bool_set_true:N \l_stex_allow_semantic_bool
    \prop_gclear:N \l__stex_expr_customs_prop
    \seq_gclear:N \l__stex_expr_customs_seq
    \int_gzero:N \l__stex_expr_arg_counter_int
    \tl_if_empty:NF \l_stex_current_args_tl {
      \exp_after:wN \__stex_expr_add_prop_arg:nnw \l_stex_current_args_tl \_stex_args_end:
      \cs_set_eq:NN \arg \__stex_expr_arg:n
    }
    \tl_set_eq:NN \l_stex_get_symbol_args_tl \l_stex_current_args_tl
    \cs_set_eq:NN \__stex_expr_do_ab_next:nnn \_stex_term_oma:nnn
    \_stex_map_args:N \__stex_expr_check_b:nn
    \__stex_expr_do_ab_next:nnn{}{}{#1}
  }{
    \prop_if_exist:NT \l__stex_expr_customs_prop {
      \prop_gset_from_keyval:Nn \exp_not:N \l__stex_expr_customs_prop {
        \prop_to_keyval:N \l__stex_expr_customs_prop
      }
    }
    \int_gset:Nn \l__stex_expr_arg_counter_int { \int_use:N \l__stex_expr_arg_counter_int}
    \seq_if_exist:NT \l__stex_expr_customs_seq {
      \seq_gset_split:Nnn \exp_not:N \l__stex_expr_customs_seq , {
        \seq_use:Nn \l__stex_expr_customs_seq ,
      }
    }
```

```
  }
  % TODO check that all arguments are present
  \group_end:
}

\cs_new_protected:Npn \__stex_expr_add_prop_arg:nnw #1 #2 #3\_stex_args_end: {
  \prop_gput:Nnn \l__stex_expr_customs_prop {#1} {}
  \seq_gput_right:Nn \l__stex_expr_customs_seq {#2}
  \tl_if_empty:nF{#3}{\__stex_expr_add_prop_arg:nnw #3 \_stex_args_end:}
}

\cs_new:Nn \__stex_expr_check_b:nn {
  \str_case:nn #2 {
    b {\cs_set_eq:NN \__stex_expr_do_ab_next:nnn \_stex_term_omb:nnn}
    B {\cs_set_eq:NN \__stex_expr_do_ab_next:nnn \_stex_term_omb:nnn}
  }
}

\NewDocumentCommand \__stex_expr_arg:n {s O{} m} {
  \IfBooleanTF #1 {
    \stex_annotate_invisible:n{
      \__stex_expr_arg_inner:nn{#2}{#3}
    }
  }{
    \__stex_expr_arg_inner:nn{#2}{#3}
  }
}

\cs_new_protected:Nn \__stex_expr_arg_inner:nn {
  \tl_if_empty:nTF{#1}{
    \int_gincr:N \l__stex_expr_arg_counter_int
    \exp_args:Ne \__stex_expr_check:nTF{ \int_use:N \l__stex_expr_arg_counter_int }{
      \__stex_expr_arg_do:oon \l_tmpa_tl \l_tmpb_tl
    }{
      \__stex_expr_arg_inner:nn{}
    }{ #2 }
  }{
    \__stex_expr_check:nTF {#1}{
      \__stex_expr_arg_do:oon \l_tmpa_tl \l_tmpb_tl { #2 }
    }{
      \exp_args:No \str_case:nnTF \l_tmpb_tl {
        {a}{
          \exp_args:NNnx \prop_gput:Nnn \l__stex_expr_customs_prop {#1}{
            \l_tmpa_tl X
          }
          \tl_set:Nx \l_tmpa_tl { #1 \int_eval:n {\tl_count:N \l_tmpa_tl + 1} }
        }
        {B}{
          \exp_args:NNnx \prop_gput:Nnn \l__stex_expr_customs_prop {#1}{
            \l_tmpa_tl X
          }
          \tl_set:Nx \l_tmpa_tl { #1 \int_eval:n {\tl_count:N \l_tmpa_tl + 1} }
        }
      }{
```

```
        \__stex_expr_arg_do:oon \l_tmpa_tl \l_tmpb_tl { #2 }
      }{
        \msg_error:nnxx{stex}{error/invalidarg}{#1}{\l_stex_current_symbol_str}
      }
    }
  }
}

\prg_new_conditional:Nnn \__stex_expr_check:n {TF} {
  \exp_args:NNe \prop_get:NnNTF \l__stex_expr_customs_prop {#1} \l_tmpa_tl {
    \tl_set:Nx \l_tmpb_tl {\seq_item:Nn \l__stex_expr_customs_seq {#1} }
    \tl_if_empty:NTF \l_tmpa_tl {
      \exp_args:NNe \prop_gput:Nnn \l__stex_expr_customs_prop
        { #1 }{X}
      \exp_args:No \str_case:nnF \l_tmpb_tl {
        {a}{
          \tl_set:Nx \l_tmpa_tl{ #1 1 }
        }
        {B}{
          \tl_set:Nx \l_tmpa_tl{ #1 1 }
        }
      }{
        \tl_set:Nx \l_tmpa_tl{ #1 }
      }
      \prg_return_true:
    }{
      \prg_return_false:
    }
  }{
    \msg_error:nnxx{stex}{error/invalidarg}{#1}{\l_stex_current_symbol_str}
    \prg_return_false:
  }
}

% #1 argnum #2 argmode #3 code
\cs_new_protected:Nn \__stex_expr_arg_do:nnn {
  \stex_debug:nn{custom}{Doing~argument~#1~of~mode~#2:~\tl_to_str:n{#3}}
  \group_begin:
    \bool_set_true:N \l_stex_allow_semantic_bool
    \_stex_term_arg:nnn {#2}{#1}{#3}
  \group_end:
}
\cs_generate_variant:Nn \__stex_expr_arg_do:nnn {oon}
```

(*End of definition for* `\stex_invoke_symbol:`. *This function is documented on page* 131.)

Argument Handling and Annotating

`\_stex_term_arg:nnnnn`
`\_stex_term_arg:nnn`

```
% 1: argnum 2: argmode 3: precedence 4: argname 5: code
\cs_new_protected:Nn \_stex_term_arg:nnnnn {
  \group_begin:
    \str_clear:N \l_stex_current_symbol_str
    \tl_clear:N \l_stex_current_term_tl
```

```
    \int_set:Nn \l_stex_notation_downprec { #3 }
    \bool_set_true:N \l_stex_allow_semantic_bool
    \_stex_term_arg:nnn {#2}{#1}{
      \tl_if_empty:nTF{#4}{
        #5
      }{
        \stex_annotate:nn{mml:arg={#4}}{#5}
      }
    }
  \group_end:
}

\cs_new_protected:Nn \_stex_term_arg:nnn {
  \stex_annotate:nn{ shtml:arg={#2}, shtml:argmode={#1}}{
    \_stex_annotate_force_break:n{ #3 }
  }
}
```

(*End of definition for* `\_stex_term_arg:nnnnn` *and* `\_stex_term_arg:nnn`*. These functions are documented on page* ??.)

`\_stex_term_arg_aB:nnnnn`

```
\tl_set:Nn \__stex_expr_do_aB_clist: {
  \seq_use:Nn \l_stex_aB_args_seq {
    \mathpunct{\comp{,}}
  }
}
\tl_set_eq:NN \_stex_term_do_aB_clist: \__stex_expr_do_aB_clist:

\int_new:N \l__stex_expr_count_int
\cs_new_protected:Nn \_stex_term_arg_aB:nnnnn {
  \tl_if_empty:nTF{#5}{
    \_stex_term_arg:nnnnn{#1}{#2}{#3}{#4}{}
  }{
    \seq_clear:N \l_stex_aB_args_seq
    \int_zero:N \l__stex_expr_count_int
    \clist_map_inline:nn{#5}{
      \__stex_expr_aB_arg:nnnnn{##1}{#1}{#2}{#3}{#4}
    }
    \_stex_term_do_aB_clist:
  }
}

% 1: code 2: argnum 3: argmode 4: precedence 5: argname
\cs_new_protected:Npn \__stex_expr_aB_arg:nnnnn #1 {
  \int_incr:N \l__stex_expr_count_int
  \__stex_expr_is_varseq:nTF{#1}{
    \exp_after:wN \exp_after:wN \exp_after:wN
    \__stex_expr_assoc_seq:nnnnnnn
    \exp_after:wN
    \__stex_expr_gobble:nnnnnnnn #1 \__stex_expr_end:
  }{
    \__stex_expr_is_seqmap:nTF{#1}{
      \exp_args:NNe \use:nn \__stex_expr_do_seqmap:nnnnnn {\tl_tail:n{#1}}
```

```
    }{
      \__stex_expr_aB_simple_arg:nnnnn{#1}
    }
  }
}

\cs_new:Npn \__stex_expr_gobble:nnnnnnnn #1 #2 #3 #4 #5 #6 #7 #8 #9 \__stex_expr_end: {
  {#2} #3 {#6}
}

\cs_new_protected:Nn \__stex_expr_aB_simple_arg:nnnnn{
  \seq_put_right:Nx \l_stex_aB_args_seq {
    \_stex_term_arg:nnnnn{#2\int_use:N\l__stex_expr_count_int}{#3}{#4}{#5}{
      \exp_not:n{
        \tl_set_eq:NN \_stex_term_do_aB_clist: \__stex_expr_do_aB_clist:
        #1
      }
    }
  }
}

\cs_new_protected:Nn \_stex_is_sequentialized:n {
  \group_begin: #1 \group_end:
}
```

Conditionals: Is the argument a sequence variable or a `\seqmap`?

```
\prg_new_conditional:Nnn \__stex_expr_is_varseq:n {TF} {
  \int_compare:nNnTF {\tl_count:n{#1}} = 1 {
    \exp_args:Ne \cs_if_eq:NNTF {\exp_args:No \tl_head:n{#1}}
    \_stex_invoke_variable:nnnnnnN {
      \exp_args:Ne \cs_if_eq:NNTF {\exp_args:No\tl_item:nn{#1}{8}}
        \stex_invoke_sequence:
          \prg_return_true:\prg_return_false:
    }\prg_return_false:
  }\prg_return_false:
}

\prg_new_conditional:Nnn \__stex_expr_is_seqmap:n {TF} {
  \int_compare:nNnTF {\tl_count:n{#1}} = 3 {
    \exp_args:Ne \tl_if_eq:nnTF {\tl_head:n{#1}} {\seqmap}
      \prg_return_true:\prg_return_false:
  }\prg_return_false:
}
```

Sequence variable:

```
% 1: name 2: arity 3: clist 4: argnum 5: argmode 6: precedence 7: argname
\cs_new_protected:Nn \__stex_expr_assoc_seq:nnnnnnn {
  \group_begin:
    \seq_clear:N \l_stex_aB_args_seq
    \__stex_expr_assoc_make_seq:nnn{#1}{#3}{#2}
  \exp_args:NNe \use:nn \group_end: {
    \seq_put_right:Nn \exp_not:N \l_stex_aB_args_seq {
      \_stex_is_sequentialized:n{
        \_stex_term_arg:nnnnn{#4\int_use:N\l__stex_expr_count_int}{#5}{#6}{#7}{
```

```
          \bool_set_true:N \l_stex_allow_semantic_bool
          \str_set:Nn \exp_not:N \l_stex_current_symbol_str
            {\l_stex_current_symbol_str}
          \tl_if_empty:NF \l_stex_current_term_tl {
            \tl_set:Nn \exp_not:N \l_stex_current_term_tl {
              \exp_args:No \exp_not:n \l_stex_current_term_tl
            }
          }
          \stex_annotate:nn{
            shtml:term=OMV,
            shtml:head={#1},
            shtml:notationid={}
          }{
            \_stex_annotate_force_break:n{
              \_stex_term_do_aB_clist:
            }
          }
        }
      }
    }
  }
}

% #1: name, #2: clist, #3:arity
\cs_new_protected:Nn \__stex_expr_assoc_make_seq:nnn {
  \cs_if_exist:cTF{l_stex_notation_#1__cs}{
    \cs_set_eq:Nc \l__stex_expr_cs {l_stex_notation_#1__cs}
  }{
    \stex_do_default_notation:
    \cs_set_eq:NN \l__stex_expr_cs \l_stex_default_notation
  }
  \clist_map_inline:nn{#2}{
    \tl_if_eq:nnTF{##1}{\ellipses}{
      \seq_put_right:Nn \l_stex_aB_args_seq { ##1 }
    }{
      \int_compare:nNnTF {#3} = 1 {
        \tl_set:Nn \l__stex_expr_iarg_tl { {##1} }
      }{
        \tl_set:Nn \l__stex_expr_iarg_tl { ##1 }
      }
      \seq_put_right:Nx \l_stex_aB_args_seq {
        \group_begin:
        \exp_not:n {
          \tl_set_eq:NN \_stex_term_do_aB_clist: \__stex_expr_do_aB_clist:
          \def\comp{\_varcomp}
          \str_set:Nn \l_stex_current_symbol_str{#1}
        }
        \exp_after:wN \exp_after:wN \exp_after:wN \exp_not:n
        \exp_after:wN \exp_after:wN \exp_after:wN {
          \exp_after:wN \l__stex_expr_cs \exp_after:wN \group_end: \l__stex_expr_iarg_tl
        }
      }
    }
  }
```

```
}
```

\seqmap:

```
% 1: fun 2: clist 3: argnum 4: argmode 5: precedence 6: argname
\cs_new_protected:Nn \__stex_expr_do_seqmap:nnnnnn {
  \group_begin:
    \cs_set:Npn \l_tmpa_cs ##1 {#1}
    \seq_clear:N \l_stex_aB_args_seq
    \clist_map_inline:nn{#2}{
      \__stex_expr_is_varseq:nTF{##1}{
        \exp_after:wN
        \__stex_expr_varseq_in_map:nnnnnnnn ##1
      }{
        \seq_put_right:Nn \l_stex_aB_args_seq {##1}
      }
    }
    \seq_clear:N \l__stex_expr_old_seq
    \seq_map_inline:Nn \l_stex_aB_args_seq {
      \tl_if_eq:nnTF{##1}{\ellipses}{
        \seq_put_right:Nn \l__stex_expr_old_seq {##1}
      }
      % TODO \_stex_is_sequentialized:n
      {
        \exp_args:NNo \seq_put_right:Nn \l__stex_expr_old_seq {
          \l_tmpa_cs {##1}
        }
      }
    }
    \seq_set_eq:NN \l_stex_aB_args_seq \l__stex_expr_old_seq
  \exp_args:NNe \use:nn \group_end: {
    \seq_put_right:Nn \exp_not:N \l_stex_aB_args_seq {
      \_stex_is_sequentialized:n{
        \_stex_term_arg:nnnnn{#4\int_use:N\l__stex_expr_count_int}{#4}{#5}{#6}{
          \bool_set_true:N \l_stex_allow_semantic_bool
          \str_set:Nn \exp_not:N \l_stex_current_symbol_str
            {\l_stex_current_symbol_str}
          \tl_set:Nn \exp_not:N \l_stex_current_term_tl {
            \symuse{Metatheory?sequence~map}
            {\exp_args:No \exp_not:n \l_tmpa_tl}
            {\_stex_term_do_aB_clist:}
          }
          \__stex_expr_do_headterm:nn{}{
            \_stex_term_do_aB_clist:
          }
        }
      }
    }
  }
}

\cs_new_protected:Nn \__stex_expr_varseq_in_map:nnnnnnnn {
  \__stex_expr_assoc_make_seq:nnn{#2}{#6}{#3}
}
```

(*End of 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

```
\cs_new_protected:Nn \_stex_eat_exclamation_point: {
  \peek_charcode_remove:NT ! {
    \_stex_eat_exclamation_point:
  }
}

\bool_new:N \stex_in_invisible_html_bool
\bool_set_false:N \stex_in_invisible_html_bool
\stex_if_html_backend:TF {
  % 1: variant 2: intent 3: code
  \cs_new_protected:Nn \_stex_term_oms:nnn {
    \tl_if_empty:NTF \l_stex_current_term_tl {
      \stex_annotate:nn{
        shtml:term=OMID,
        shtml:head={\l_stex_current_symbol_str},
        shtml:notationid={#1},
      }{
        \_stex_annotate_force_break:n{#3}
      }
    }{
      \__stex_expr_do_headterm:nn{#1}{#3}
    }
  }
  \cs_new_protected:Nn \_stex_term_omv:nnn {
    \tl_if_empty:NTF \l_stex_current_term_tl {
      \stex_annotate:nn{
        shtml:term=OMV,
        shtml:head={\l_stex_current_symbol_str},
        shtml:notationid={#1}
      }{
        \_stex_annotate_force_break:n{#3}
      }
    }{
      \__stex_expr_do_headterm:nn{#1}{#3}
    }
  }
  \cs_new_protected:Nn \__stex_expr_do_headterm:nn {
    \bool_if:NTF \stex_in_invisible_html_bool {
      {\bool_set_true:N \l_stex_allow_semantic_bool
        \ensuremath{\l_stex_current_term_tl}
      }
    }{
      \stex_annotate:nn{
        shtml:term=complex,
        shtml:head={\l_stex_current_symbol_str},
        shtml:notationid={#1}
      }{
        \_stex_annotate_force_break:n{
          \stex_annotate_invisible:nn{shtml:headterm={}}{
            {\bool_set_true:N \l_stex_allow_semantic_bool
              \ensuremath{\l_stex_current_term_tl}
```

```
            }
          }
        }
        #2
      }
    }
  }
}{
  \cs_new_protected:Nn \_stex_term_oms:nnn {#3}
  \cs_new_protected:Nn \_stex_term_omv:nnn {#3}
  \cs_new_protected:Nn \__stex_expr_do_headterm:nn { #2 }
}
\cs_set_eq:NN \_stex_term_oms_or_omv:nnn \_stex_term_oms:nnn
```

(*End of 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`

```
\stex_if_html_backend:TF {
  \cs_new_protected:Nn \_stex_term_oma:nnn {
    \tl_if_empty:NTF \l_stex_current_term_tl {
      \stex_annotate:nn{
        shtml:term=OMA,
        shtml:head={\l_stex_current_symbol_str},
        shtml:notationid={#1}
      }{
        \_stex_annotate_force_break:n{#3}
      }
    }{
      \stex_annotate:nn{
        shtml:term=OMA,
        shtml:head={\l_stex_current_symbol_str},
        shtml:notationid={#1}
      }{
        \_stex_annotate_force_break:n{
          \stex_annotate_invisible:nn{shtml:headterm={}}{{
            \bool_set_true:N \l_stex_allow_semantic_bool
            \l_stex_current_term_tl
          }}
          #3
        }
      }
    }
  }
}{
  \cs_new_protected:Nn \_stex_term_oma:nnn {#3}
}
```

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

`\_stex_term_omb:nnn`

```
\stex_if_html_backend:TF {
  \cs_new_protected:Nn \_stex_term_omb:nnn {
    \tl_if_empty:NTF \l_stex_current_term_tl {
      \stex_annotate:nn{
```

```
        shtml:term=OMBIND,
        shtml:head={\l_stex_current_symbol_str},
        shtml:notationid={#1}
      }{
        \_stex_annotate_force_break:n{#3}
      }
    }{
      \stex_annotate:nn{
        shtml:term=OMBIND,
        shtml:head={\l_stex_current_symbol_str},
        shtml:notationid={#1}
      }{
        \_stex_annotate_force_break:n{
          \stex_annotate_invisible:nn{shtml:headterm={}}{{
            \bool_set_true:N \l_stex_allow_semantic_bool
            \l_stex_current_term_tl
          }}
          #3
        }
      }
    }
  }
}{
  \cs_new_protected:Nn \_stex_term_omb:nnn { #3 }
}
```

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

Automated Bracketing

`\infprec`
`\neginfprec`

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

(*End of definition for* `\infprec` *and* `\neginfprec`. *These functions are documented on page* 91.)

`\dobrackets`

```
\int_new:N \l_stex_notation_downprec
\int_set:Nn \l_stex_notation_downprec \infprec
\tl_set:Nn \l__stex_expr_left_bracket_str (
\tl_set:Nn \l__stex_expr_right_bracket_str )
\bool_new:N \l_stex_brackets_dones_bool

\cs_new_protected:Nn \_stex_maybe_brackets:nn {
  \bool_if:NTF \l_stex_brackets_dones_bool {
    \bool_set_false:N \l_stex_brackets_dones_bool
    #2
  } {
    \stex_debug:nn{brackets}{#1>\int_eval:n \l_stex_notation_downprec?}
    \int_compare:nNnTF { #1 } > \l_stex_notation_downprec {
      %\bool_if:NTF \l_stex_inparray_bool { #2 }{
        \dobrackets {
          #2
        }
```

```
      %}
    }{
      #2
    }
  }
}

%\RequirePackage{scalerel}
\cs_new_protected:Npn \dobrackets #1 {
  %\ThisStyle{\if D\m@switch
  %    \exp_args:Nnx \use:nn
  %    { \exp_after:wN \left\l__stex_expr_left_bracket_str #1 }
  %    { \exp_not:N\right\l__stex_expr_right_bracket_str }
  %  \else
  \group_begin:
  %\stex_pseudogroup_with:nn{\l_stex_brackets_dones_bool\l_stex_notation_downprec}{
    \bool_set_true:N \l_stex_brackets_dones_bool
    %\int_set:Nn \l_stex_notation_downprec \infprec
    \mathopen{\cs_if_exist:NT\l_stex_current_symbol_str\comp
      \l__stex_expr_left_bracket_str
    }
    #1
  \group_end:%}
  \mathclose{\cs_if_exist:NT\l_stex_current_symbol_str\comp
    \l__stex_expr_right_bracket_str
  }
  %\fi}
}
```

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

`\withbrackets`

```
\cs_new_protected:Npn \withbrackets #1 #2 #3 {
  \stex_pseudogroup_with:nn{\l__stex_expr_left_bracket_str\l__stex_expr_right_bracket_str}{
    \tl_set:Nn \l__stex_expr_left_bracket_str { #1 }
    \tl_set:Nn \l__stex_expr_right_bracket_str { #2 }
    #3
  }
}
```

(*End of definition for* `\withbrackets`. *This function is documented on page 92.*)

`\dowithbrackets`

```
\cs_new_protected:Npn \dowithbrackets #1 #2 #3 {
  \withbrackets{#1}{#2}{\dobrackets{#3}}
}
```

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

Symname and Variants

`\symname` `\sn` `\sns` `\Symname` `\Sn` `\Sns` `\symref` `\sr` `\varref` `\varname` `\Varname`

```
\def\maincomp{\comp}

\stex_keys_define:nnnn{symname}{
```

```
  \tl_clear:N \l_stex_key_pre_tl
  \tl_clear:N \l_stex_key_post_tl
  %\tl_clear:N \l_stex_key_proot_tl
}{
  pre    .tl_set:N   = \l_stex_key_pre_tl ,
  post   .tl_set:N   = \l_stex_key_post_tl ,
  root   .code:n     = {}%.tl_set:N   = \l_stex_key_root_tl
}{}

\NewDocumentCommand \symref { O{} m m} {
  \group_begin:
  \stex_keys_set:nn{symname}{#1}
  \stex_get_symbol:n{#2}
  \__stex_expr_do_ref:nNn{#3}\symrefemph@uri\_stex_term_oms:nnn
}
\let\sr\symref

\NewDocumentCommand \symname { O{} m} {
  \group_begin:
  \stex_keys_set:nn{symname}{#1}
  \stex_get_symbol:n{#2}
  \__stex_expr_do_ref:nNn{
    \l_stex_key_pre_tl\l_stex_get_symbol_name_str\l_stex_key_post_tl
  }\symrefemph@uri\_stex_term_oms:nnn
}
\let\sn\symname
\protected\def\sns{\symname[post=s]}

\NewDocumentCommand \Symname { O{} m} {
  \group_begin:
  \stex_keys_set:nn{symname}{#1}
  \stex_get_symbol:n{#2}
  \__stex_expr_do_ref:nNn{
    \l_stex_key_pre_tl\exp_after:wN\_stex_capitalize:n\l_stex_get_symbol_name_str\l_stex_key
  }\symrefemph@uri\_stex_term_oms:nnn
}
\cs_new_protected:Nn \_stex_capitalize:n {
  \uppercase{#1}
}
\let\Sn\Symname
\protected\def\Sns{\Symname[post=s]}

\cs_new:Npn \_stex_split_slash: #1/#2/#3\_stex_args_end: {
 \tl_if_empty:nTF{#3}{
  #2
 }{
  \_stex_split_slash: #2 / #3 \_stex_args_end:
 }
}

\NewDocumentCommand \varref { O{} m m} {
  \group_begin:
  \stex_keys_set:nn{symname}{#1}
  \stex_get_var:n{#2}
```

```
  \__stex_expr_do_ref:nNn{#3}\varemph@uri{
    \str_set_eq:NN \l_stex_current_symbol_str\l_stex_get_symbol_name_str
    \def\comp{\_varcomp}
    \_stex_term_omv:nnn
  }
}

\NewDocumentCommand \varname { O{} m} {
  \group_begin:
  \stex_keys_set:nn{symname}{#1}
  \stex_get_var:n{#2}
  \__stex_expr_do_ref:nNn{
    \l_stex_key_pre_tl\l_stex_get_symbol_name_str\l_stex_key_post_tl
  }\varemph@uri{
    \str_set_eq:NN \l_stex_current_symbol_str\l_stex_get_symbol_name_str
    \def\comp{\_varcomp}
    \_stex_term_omv:nnn
  }
}

\NewDocumentCommand \Varname { O{} m} {
  \group_begin:
  \stex_keys_set:nn{symname}{#1}
  \stex_get_var:n{#2}
  \__stex_expr_do_ref:nNn{
    \l_stex_key_pre_tl\exp_after:wN\_stex_capitalize:n\l_stex_get_symbol_name_str\l_stex_key
  }\varemph@uri{
    \str_set_eq:NN \l_stex_current_symbol_str\l_stex_get_symbol_name_str
    \def\comp{\_varcomp}
    \_stex_term_omv:nnn
  }
}


\cs_new_protected:Nn \__stex_expr_do_ref:nNn {
  \stex_if_html_backend:T{\ifvmode\indent\fi}
  \bool_if:NTF \l_stex_allow_semantic_bool{
    \str_set:Nx\l_stex_current_symbol_str
      {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
    \str_if_in:NnT \l_stex_get_symbol_name_str / {
      \str_set:Nx \l_stex_get_symbol_name_str {
        \exp_after:wN \_stex_split_slash: \l_stex_get_symbol_name_str
        /\_stex_args_end:
      }
    }
    \tl_clear:N \l_stex_current_term_tl
    \def\comp{\_comp}
    \let\compemph@uri#2
    #3{}{}{\comp{#1}}
  }{
    \msg_error:nnxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
  }
  \group_end:
}
```

(*End of definition for* `\symname` *and others. These functions are documented on page 89.*)

Highlighting

```
⟨@@=stex_notationcomps⟩
```

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

```
\cs_new_protected:Nn \_do_comp:nNn {
  \stex_pseudogroup_with:nn{\comp}{
    \def\comp##1{##1}
    \str_if_empty:NTF \l_stex_current_symbol_str {
      #3
    }{
      \stex_if_html_backend:TF {
        \stex_annotate:nn { shtml:#1 = \l_stex_current_symbol_str}{ #3 }
      }{
        \exp_args:Nno #2 { #3 } \l_stex_current_symbol_str
      }
    }
  }
}

\cs_new_protected:Npn \_comp {
  \_do_comp:nNn {comp}\compemph@uri
}

\cs_new_protected:Npn \_varcomp {
  \_do_comp:nNn {varcomp}\varemph@uri
}

\cs_new_protected:Npn \_defcomp {
  \_do_comp:nNn {definiendum}\defemph@uri
}

\cs_set_protected:Npn \comp {}

\cs_new_protected:Npn \compemph@uri #1 #2 {
    \compemph{ #1 }
}

\cs_new_protected:Npn \compemph #1 {
    #1
}

\cs_new_protected:Npn \defemph@uri #1 #2 {
    \defemph{#1}
}

\cs_new_protected:Npn \defemph #1 {
    \ifmmode\else\expandafter\textbf\fi{#1}
}

\cs_new_protected:Npn \symrefemph@uri #1 #2 {
    \symrefemph{#1}
```

```
}

\cs_new_protected:Npn \symrefemph #1 {
    \emph{#1}
}

\cs_new_protected:Npn \varemph@uri #1 #2 {
    \varemph{#1}
}

\cs_new_protected:Npn \varemph #1 {
    #1
}
```

(*End of definition for* `\comp` *and others. These functions are documented on page 90.*)

13.9 Mathematical Structures

```
⟨@@=stex_structures⟩
```

`\this`

```
\cs_new_protected:Npn \stex_current_this: {
  { \bool_set_true:N \l_stex_allow_semantic_bool
    \tl_if_empty:NTF \l_stex_current_this_tl {{}}{
      \str_set:Nx \l_stex_current_symbol_str {
        \l_stex_current_module_str ? \l__stex_structures_name_str
      }
      \maincomp{\l_stex_current_this_tl}
    }
  }
}
\let \this \stex_current_this:
```

(*End of definition for* `\this`*. This function is documented on page 96.*)

`mathstructure` (*env.*)

```
\stex_new_stylable_env:nnnnnnn {mathstructure}{m O{}}{
  \__stex_structures_begin:nn{#1}{#2}
  \stex_smsmode_do:
}{
  \stex_structural_feature_module_end:
  \__stex_structures_do_externals:
}{}{}{}

\stex_keys_define:nnnn{mathstructure}{
  \tl_clear:N \l_stex_current_this_tl
  \str_clear:N \l__stex_structures_name_str
}{
  this    .tl_set:N = \l_stex_current_this_tl ,
  unknown .code:n   = {
    \str_if_empty:NTF \l_keys_key_str {
      \str_set:Nx \l__stex_structures_name_str {\l_keys_key_tl}
    }{
      \str_set_eq:NN \l__stex_structures_name_str \l_keys_key_str
```

```
    }
  }
}{}

\cs_new_protected:Nn \__stex_structures_begin:nn {
  \stex_keys_set:nn {mathstructure}{#2}
  \str_if_empty:NT \l__stex_structures_name_str {
    \str_set:Nn \l__stex_structures_name_str {#1}
  }
  \def\comp{\_comp}

  \exp_args:Nne \use:nn { \stex_module_add_symbol:nnnnnnnN }
    { {#1}{\l__stex_structures_name_str}{0}{}{defed}{
      \l_stex_current_module_str / \l__stex_structures_name_str-module
    }}
    {}\stex_invoke_structure:
  \str_set:Nx \l_stex_macroname_str {#1}
  \stex_execute_in_module:x{
    \seq_clear:c{l_stex_structure_macros_\l_stex_current_module_str / \l__stex_structures_na
    \seq_put_right:cn{l_stex_structure_macros_\l_stex_current_module_str / \l__stex_structur
  }
  \exp_args:No \stex_structural_feature_module:nn
    {\l__stex_structures_name_str}{structure}
}

\stex_sms_allow_env:n{mathstructure}
\stex_deactivate_macro:Nn \mathstructure {module~environments}
\stex_every_module:n {\stex_reactivate_macro:N \mathstructure}

\cs_new_protected:Nn \__stex_structures_do_externals: {
  \tl_set:Nn \l__stex_structures_replace_this_tl {####1}
  \exp_args:No \stex_iterate_symbols:nn{\g_stex_last_feature_str}{
    \__stex_structures_external_decl:nnnn{##5}{##4}{##3}{##8}
  }
}

\cs_new_protected:Nn \__stex_structures_external_decl:nnnn {
  %\stex_debug:nn{structure}{
  %  Generating~external~declaration~\l__stex_structures_name_str/#3~in~
  %  \l_stex_current_module_str^^J
  %  \tl_to_str:n{#1}^^J\tl_to_str:n{#2}^^J\tl_to_str:n{#4}
  %}
  %\tl_set:Nn \l_stex_get_symbol_args_tl {#1}
  %\exp_args:Nnx \use:nn {\stex_module_add_symbol:nnnnnnnN} {
  %  {}{\l__stex_structures_name_str/#3}{\int_eval:n{#2 + 1}}
  %  {1i\tl_if_empty:nF{#1}{\_stex_map_args:N \__stex_structures_shift_argls:nn}}
  %  {defed}{typed}
  %}{#4}\stex_invoke_outer_field:
}

\cs_new:Nn \__stex_structures_shift_argls:nn {
  \int_eval:n{#1+1}#2
}
```

\stex_get_mathstructure:n

```
\cs_new_protected:Nn \stex_get_mathstructure:n {
  \_stex_get_mathstructure:n{#1}
  \str_if_empty:NT \l_stex_get_structure_module_str {
    \msg_error:nnn{stex}{error/unknownstructure}{#1}
  }
}
\cs_new_protected:Nn \_stex_get_mathstructure:n {
  \str_clear:N \l_stex_get_structure_module_str
  \_stex_get_symbol:n{#1}
  \str_if_empty:NF \l_stex_get_symbol_name_str {
    \exp_args:No \tl_if_eq:NNT \l_stex_get_symbol_invoke_cs \stex_invoke_structure: {
      \str_set_eq:NN \l_stex_get_structure_module_str \l_stex_get_symbol_type_tl
    }
  }
}
```

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

extstructure (*env.*)

```
\stex_new_stylable_env:nnnnnnn {extstructure}{m O{} m}{
  \seq_clear:N \l__stex_structures_imports_seq
  \clist_map_inline:nn{#3}{
    \stex_get_mathstructure:n{##1}
    \clist_map_inline:Nn \l_stex_get_symbol_type_tl {
      \exp_args:Ne \stex_if_module_exists:nT{\tl_to_str:n{####1}}{
        \seq_put_right:Nn \l__stex_structures_imports_seq{####1}
      }
    }
    \stex_execute_in_module:x{
      \seq_put_right:cn{l_stex_structure_macros_\l_stex_get_structure_module_str _seq}{#1}
    }
  }
  \__stex_structures_begin:nn{#1}{#2}
  \seq_map_inline:Nn\l__stex_structures_imports_seq{
    \stex_if_do_html:T {
      \hbox{\stex_annotate_invisible:nn
        {shtml:import={##1}} {}}
    }
    \stex_module_add_morphism:nonn
      {}{##1}{import}{}
    \stex_execute_in_module:x{
      \stex_activate_module:n{##1}
    }
  }
  \stex_smsmode_do:
}{
  \stex_structural_feature_module_end:
  \__stex_structures_do_externals:
}{}{}{}

\stex_sms_allow_env:n{extstructure}
\stex_deactivate_macro:Nn \extstructure {module~environments}
\stex_every_module:n {
```

```
  \stex_reactivate_macro:N \extstructure
}

\cs_new:Nn \__stex_structures_extend_structure_i:NnnnnnnnN {
  \exp_not:n{#1{#2}{#3}{#4}{#5}{defed}}{\l__stex_structures_extmod_str,#7}\exp_not:n{{#8}#9}
}
\cs_new_protected:Nn \__stex_structures_extend_structure:nn {
  \stex_debug:nn{ext}{Extending~#1~by~#2}
  \str_set:Nn \l__stex_structures_extmod_str{#2}
  \tl_set:cx{#1}{
    \exp_after:wN \exp_after:wN \exp_after:wN
    \__stex_structures_extend_structure_i:NnnnnnnnN \cs:w #1 \cs_end:
  }
}

\stex_new_stylable_env:nnnnnnn {extstructure*}{m}{
  \__stex_structures_new_extstruct_name:
  \seq_clear:N \l__stex_structures_imports_seq
  \stex_get_mathstructure:n{#1}

  \clist_map_inline:Nn \l_stex_get_symbol_type_tl {
    \exp_args:Ne \stex_if_module_exists:nT{\tl_to_str:n{##1}}{
      \seq_put_right:Nn \l__stex_structures_imports_seq{##1}
    }
  }

  \stex_execute_in_module:x{
    \seq_map_inline:cn{l_stex_structure_macros_\l_stex_get_structure_module_str _seq}{
      \exp_not:N \tl_if_exist:cT{####1}{
        \__stex_structures_extend_structure:nn{####1}{\l_stex_current_module_str/\l__stex_st
      }
    }
  }

  \exp_args:No \__stex_structures_begin:nn\l__stex_structures_exstruct_name_str{}

  \seq_map_inline:Nn \l__stex_structures_imports_seq {
    \stex_if_do_html:T {
      \stex_annotate_invisible:nn
        {shtml:import= {##1}} {}
    }
    \stex_module_add_morphism:nonn
      {}{##1}{import}{}
    \stex_execute_in_module:x{
      \stex_activate_module:n{##1}
    }
  }

  \stex_smsmode_do:
}{
  \prop_map_inline:cn{
    c_stex_module_ \l_stex_current_module_str _symbols_prop
  }{
    \__stex_structures_check_def:nnnnnnnn ##2
```

```
  }
  \stex_structural_feature_module_end:
  \__stex_structures_do_externals:
}{}{}{}

\stex_sms_allow_env:n{extstructure*}
\exp_after:wN \stex_deactivate_macro:Nn
  \cs:w extstructure*\cs_end: {module~environments}
\stex_every_module:n {
  \exp_after:wN \stex_reactivate_macro:N \cs:w extstructure*\cs_end:
}

\cs_new_protected:Nn \__stex_structures_check_def:nnnnnnnn {
  \tl_if_empty:nT{#5}{
    \msg_error:nnnn{stex}{error/needsdefiniens}{#2}{extstructure*}
  }
}

\stex_every_module:n{ \str_set:Nn \l__stex_structures_extname_count 0}

\cs_new_protected:Nn \__stex_structures_new_extstruct_name: {
  \stex_do_up_to_module:n {
    \str_set:Nx \l__stex_structures_extname_count {\int_eval:n{\l__stex_structures_extname_c
  }
  \str_set:Nx \l__stex_structures_exstruct_name_str {EXTSTRUCT_\l__stex_structures_extname_c
}
```

Invoking structures:

```
\cs_new_protected:Nn \stex_invoke_structure: {
  \tl_set:Nn \l__stex_structures_set_comp_tl {\__stex_structures_set_thiscomp:}
  \__stex_structures_invoke_top:n {}
}

\cs_new_protected:Nn \__stex_structures_invoke_top:n {
  \stex_debug:nn{structure}{
    invoking~structure~{\l_stex_current_type_tl}<\tl_to_str:n{#1}>
  }
  \peek_charcode:NTF [ {
    \__stex_structures_merge:nw{#1}
  }{
    \__stex_structures_invokation_type:n {#1}
    \tl_set:Nn \l__stex_structures_this_tl {}
    \peek_charcode_remove:NTF ! {
      \peek_charcode:NTF [ {
        \__stex_structures_maybe_notation:w
      }{
        \__stex_structures_maybe_notation:w []
      }
    }{
      \__stex_structures_invoke_this:n
    }
  }
}
```

```
\cs_new_protected:Npn \__stex_structures_merge:nw #1 [ #2 ] {
  \exp_args:Ne \stex_str_if_starts_with:nnTF {\tl_to_str:n{#2}}{comp=}{
    \__stex_structures_set_customcomp: #2 \__stex_structures_end:
    \__stex_structures_invoke_top:n{#1}
  }{
    \exp_args:Ne \stex_str_if_starts_with:nnTF {\tl_to_str:n{#2}}{this=}{
      \__stex_structures_set_thisnotation: #2 \__stex_structures_end:
      \__stex_structures_invoke_top:n{#1}
    }{
      \tl_if_empty:nTF{#1}{
        \__stex_structures_invoke_top:n{#2}
      }{
        \tl_if_empty:nTF{#2}{
          \__stex_structures_invoke_top:n{#1}
        }{
          \__stex_structures_invoke_top:n{#1,#2}
        }
      }
    }
  }
}

\cs_new_protected:Npn \__stex_structures_set_thisnotation: this= #1 \__stex_structures_end:
  \tl_set:Nn \l_stex_return_notation_tl { \comp{#1} }
  \tl_set:Nn \l__stex_structures_set_comp_tl {}
}

\cs_new_protected:Npn \__stex_structures_set_customcomp: comp= #1 \__stex_structures_end: {
  \tl_set:Nn \l__stex_structures_set_comp_tl {
    \__stex_structures_set_custom_comp:n{#1}
  }
  \tl_set:Nn \l_stex_return_notation_tl { \comp{} }
}
```

The structure type:

```
\cs_new_protected:Nn \__stex_structures_invokation_type:n {
  \__stex_structures_do_assign_list:n{#1}
  \clist_if_empty:NTF \l__stex_structures_fields_clist {
    \int_compare:nNnTF {\clist_count:N \l_stex_current_type_tl}
      = 1 {
        \tl_set:Nx \l__stex_structures_current_type_tl {
          \exp_args:No \exp_not:n \l_stex_current_redo_tl
          \_stex_term_oms_or_omv:nnn{}{}{}
        }
      }{
        \exp_args:No \__stex_structures_make_type:n \l_stex_current_type_tl
    }
  }{
    \int_compare:nNnTF {\clist_count:N \l_stex_current_type_tl}
      = 1 {
        \__stex_structures_make_type:n {}
      }{
        \exp_args:No \__stex_structures_make_type:n \l_stex_current_type_tl
    }
```

```
  }
}

\cs_new_protected:Nn \__stex_structures_do_assign_list:n {
  \clist_clear:N \l__stex_structures_fields_clist
  \tl_if_empty:nF {#1} {
    \keyval_parse:NNn\TODO\__stex_structures_do_assign:nn{#1}
  }
}

\cs_new_protected:Nn \__stex_structures_do_assign:nn {
  \clist_put_right:Nn \l__stex_structures_fields_clist {{#1}{#2}}
}

\cs_new_protected:Nn \__stex_structures_make_type:n {
  \tl_if_empty:nTF{#1}{
    \seq_clear:N \l_tmpa_seq
  }{
    \seq_set_split:Nnn \l_tmpa_seq ,{#1}
    \seq_pop_right:NN \l_tmpa_seq \l_tmpa_tl
    \seq_reverse:N \l_tmpa_seq
  }
  \tl_set:Nx \l__stex_structures_current_type_tl {
    \symuse{Metatheory?module~type~merge}{
      {
        \exp_args:No \exp_not:n \l_stex_current_redo_tl
        \_stex_term_oms_or_omv:nnn{}{}{}
      }
      \seq_map_function:NN \l_tmpa_seq \__stex_structures_make_mod:n
      \clist_if_empty:NF \l__stex_structures_fields_clist {
        ,\symuse{Metatheory?anonymous~record}{
          \exp_args:Ne \tl_tail:n{
            \clist_map_function:NN \l__stex_structures_fields_clist \__stex_structures_make_
          }
        }
      }
    }
  }
}

\cs_new:Nn \__stex_structures_make_mod:n {
  ,\symuse{Metatheory?module~type}{
    \stex_annotate:nn{shtml:term=OMMOD,shtml:head={#1}}{}
  }
}

\cs_new:Nn \__stex_structures_make_oml:n {
  \__stex_structures_make_oml:nn #1
}
\cs_new:Nn \__stex_structures_make_oml:nn {
  ,\stex_annotate:nn{
    shtml:term=OML,
    shtml:head={#1}
  }{
```

```
    \_stex_annotate_force_break:n{
      \stex_annotate:nn{shtml:definiens={}}{\exp_not:n{#2!}}
    }
  }
}
```

Insert the structure type as a term:

```
\cs_new:Nn \__stex_structures_current_type: {
  %\exp_args:No \exp_not:n \l_stex_current_redo_tl
  \tl_set:Nn \exp_not:N \l_stex_current_term_tl {
    \exp_args:No\exp_not:n\l__stex_structures_current_type_tl
  }
  \_stex_term_oms_or_omv:nnn{}{}{}
}
```

The structure type itself:

```
\cs_new_protected:Npn \__stex_structures_maybe_notation:w [ #1 ] {
  \tl_set_eq:NN \l_stex_current_term_tl \l__stex_structures_current_type_tl
  \cs_if_exist:cTF{l_stex_notation_\l_stex_current_symbol_str _#1_cs}{
    \use:c{l_stex_notation_\l_stex_current_symbol_str _#1_cs}\group_end:
  }{
    \__stex_structures_make_prop:
    \__stex_structures_make_prop_assign:
    \__stex_structures_present_i:w [#1]
  }
}

\cs_new_protected:Nn \__stex_structures_present: {
  \peek_charcode:NTF [ {
    \__stex_structures_present_i:w
  }{
    \__stex_structures_present:nn{}{}
  }
}

\cs_new_protected:Npn \__stex_structures_present_i:w [#1] {
  \int_compare:nNnTF{\clist_count:n{#1}} = 1 {
    \__stex_structures_present:nn{}{#1}
  }{
    \peek_charcode:NTF [ {
      \__stex_structures_present_ii:nw{#1}
    }{
      \__stex_structures_present:nn{#1}{}
    }
  }
}

%First: clist, second:notation-id
\cs_new_protected:Npn \__stex_structures_present_ii:nw #1 [#2] {
  \__stex_structures_present:nn{#1}{#2}
}

\cs_new_protected:Nn \__stex_structures_present:nn {
  \clist_clear:N \l__stex_structures_clist
  \tl_if_empty:nTF{#1}{
```

```
    \cs_set:Npn \l__stex_structures_cs ##1 ##2 ##3 {
      \tl_if_empty:nF{##2}{
        \__stex_structures_present_entry:nn {##1}{##3}
      }
    }
  }{
    \cs_set:Npn \l__stex_structures_cs ##1 ##2 ##3 {
      \exp_args:Ne \clist_if_in:nnT{\tl_to_str:n{#1}}{##1}{
        \__stex_structures_present_entry:nn {##1}{##3}
      }
    }
  }
  \prop_map_inline:Nn \l__stex_structures_prop {
    \l__stex_structures_cs {##1} ##2
  }
  \_stex_term_oms_or_omv:nnn{}{}{
    \exp_args:Nno \use:n{
      \bool_set_true:N \l_stex_allow_semantic_bool
      \symuse{Metatheory?mathematical~structure}[#2]
    }{\l__stex_structures_clist}
  }\group_end:
}

\cs_new_protected:Nn \__stex_structures_present_entry:nn {
  \seq_if_in:NnTF \l__stex_structures_assigned_seq {#1}{
    \clist_put_right:Nn \l__stex_structures_clist {#2!}
  }{
    \exp_args:NNe \clist_put_right:Nn \l__stex_structures_clist {
      \_stex_next_symbol:n {
        \exp_args:No \exp_not:n \l__stex_structures_set_comp_tl
        \tl_set:Nn \exp_not:N \l__stex_structures_this_tl {
          \exp_args:No \exp_not:n \l__stex_structures_this_tl
        }
        \exp_not:n {
          \tl_set_eq:NN \this \l__stex_structures_this_tl
        }
        \tl_set:Nn \exp_not:N \l_stex_return_notation_tl {
          \exp_args:No \exp_not:n \l_stex_return_notation_tl
        }
      }
      \exp_not:n{#2!}
    }
  }
}

\cs_new_protected:Npn \_thiscomp #1 #2 {
  {\tl_set:cn{this}{{}}#1{#2}\c_math_subscript_token{
    \group_begin:
    \bool_set_true:N \l_stex_allow_semantic_bool
    \l__stex_structures_this_tl
    \group_end:
  }
  }
}
```

```
\cs_new_protected:Nn \__stex_structures_set_thiscomp: {
  \exp_args:Ne \tl_if_eq:NNF {\tl_head:N \maincomp} \_thiscomp {
    \edef\maincomp {\_thiscomp{\comp}}
  }
}

\cs_new_protected:Nn \__stex_structures_set_custom_comp:n {
  \exp_args:Ne \tl_if_eq:NNF {\tl_head:N \maincomp} \_customthiscomp {
    \cs_set_protected:Npx \_customthiscomp ##1 {
      \group_begin:
        \bool_set_true:N \l_stex_allow_semantic_bool
        \exp_not:n{
          \cs_set:Npn \l__stex_structures_comp_cs ##1  {
            #1
          }
          \def\maincomp
        }{\comp}
        \exp_not:N\l__stex_structures_comp_cs{\comp{##1}}
      \group_end:
    }
    \def\maincomp {\_customthiscomp}
  }
}

```

`this` (of type `structure`):

```
\cs_new_protected:Nn \__stex_structures_invoke_this:n {
  \peek_charcode_remove:NTF ! {
    \exp_args:Nne\use:nn{
      \group_end:\symuse{Metatheory?of~type}[invisible]{#1}
    }{
      {\__stex_structures_current_type:}
    }
  }{
    \__stex_structures_invoke_maybe_field:nn{#1}
  }
}

\cs_new_protected:Nn \__stex_structures_invoke_maybe_field:nn {
  \__stex_structures_make_prop:
  \__stex_structures_set_this:n{#1}
  \tl_if_empty:nTF{#2}{
    \__stex_structures_make_prop_assign:
    \__stex_structures_present:
  }{
    \__stex_structures_invoke_field:n{#2}
  }
}

\cs_new_protected:Nn \__stex_structures_set_this:n {
  \tl_if_empty:nTF{#1}{
    %\tl_put_right:Nn \l_stex_current_redo_tl {
    %  \tl_clear:N \l__stex_structures_this_tl
```

```
    %}
  }{
    \tl_set:Nx \l__stex_structures_this_tl {{
      \bool_set_true:N \l_stex_allow_semantic_bool
      \tl_set:Nn \exp_not:N \this {
        \exp_args:No \exp_not:n \this
      }
      \exp_not:n{#1}
    }}
    \tl_set_eq:NN \this \l__stex_structures_this_tl
    % \l_stex_return_notation_tl
  }
}

\cs_new_protected:Nn \__stex_structures_get_field_name:n {
  \str_set:Nx \l__stex_structures_field_name_str {
    \exp_args:Nne \use:n {\exp_after:wN \use_i:nn \use:n}
    {\prop_item:Nn \l__stex_structures_prop {#1}}
  }
  \str_if_empty:NT \l__stex_structures_field_name_str {
    \str_set:Nn \l__stex_structures_field_name_str {#1}
  }
}

\cs_new_protected:Nn \__stex_structures_invoke_field:n {
  \prop_if_in:NnTF \l__stex_structures_prop {#1}{
    \__stex_structures_get_field_name:n{#1}
    \tl_clear:N \l__stex_structures_more_nextsymbol_tl
    %\exp_args:NNe \seq_if_in:NnF \l__stex_structures_assigned_seq {\tl_to_str:n{#1}}{
      \tl_set:Nx \l__stex_structures_more_nextsymbol_tl {
        \tl_set:Nn \exp_not:N \l__stex_structures_this_tl {
          \exp_args:No \exp_not:n \l__stex_structures_this_tl
        }
        \exp_not:n {
          \tl_set_eq:NN \this \l__stex_structures_this_tl
        }
        \tl_set:Nn \exp_not:N \l_stex_return_notation_tl {
          \exp_args:No \exp_not:n \l_stex_return_notation_tl
        }
        \exp_args:No \exp_not:n \l__stex_structures_set_comp_tl
      }
    %}
    \exp_args:NNx \use:nn \group_end: {
      \_stex_next_symbol:n {
        \exp_args:No \exp_not:n \l__stex_structures_redo_tl
        \tl_set:Nn \exp_not:N \l_stex_current_term_tl {
          \symuse{Metatheory?record~field}{
            \symuse{Metatheory?of~type}{
              \exp_args:No \exp_not:n \l__stex_structures_this_tl
            }{  \__stex_structures_current_type: }
          }{
            \stex_annotate:nn{shtml:term=OML,shtml:head={\l__stex_structures_field_name_str}
          }
        }
```

```
        \exp_args:No \exp_not:n \l__stex_structures_more_nextsymbol_tl
      }
      \exp_not:N \use_ii:nn
      \prop_item:Nn \l__stex_structures_prop {#1}
    }
  }{
    \msg_error:nnn{stex}{error/unknownfield}{#1}
  }
}

\cs_new_protected:Nn \__stex_structures_make_prop: {
  \prop_clear:N \l__stex_structures_prop
  \seq_clear:N \l__stex_structures_seq
  \seq_clear:N \l__stex_structures_assigned_seq
  \tl_clear:N \l__stex_structures_redo_tl
  \__stex_structures_prop_do_decls:
  \__stex_structures_prop_do_notations:
}

\cs_new_protected:Nn \__stex_structures_make_prop_assign: {
  \clist_if_empty:NF \l__stex_structures_fields_clist {
    \clist_map_inline:Nn \l__stex_structures_fields_clist {
      \__stex_structures_make_prop_assign:nn ##1
    }
  }
}

\cs_new_protected:Nn \__stex_structures_make_prop_assign:nn {
  \prop_if_in:NnTF \l__stex_structures_prop {#1}{
    \exp_args:NNe \seq_put_right:Nn \l__stex_structures_assigned_seq {\tl_to_str:n{#1}}
    \exp_args:Nne \use:nn {\__stex_structures_make_prop_assign_replace:nnnn {#1}{#2}}
    {\prop_item:Nn \l__stex_structures_prop {#1}}
  }{
    \msg_error:nnn{stex}{error/unknownfieldass}{#1}
  }
}
\cs_new_protected:Nn \__stex_structures_make_prop_assign_replace:nnnn {
  \prop_put:Nnn \l__stex_structures_prop {#1}{{#3}{#2}}
  \tl_if_empty:nF{#3}{
    \tl_set:cn{#1}{ #2 }
    \tl_put_right:Nn \l__stex_structures_redo_tl {
      \tl_set:cn{#1}{ #2 }
    }
  }
}

\cs_new_protected:Nn \__stex_structures_prop_do_decls: {
  \exp_args:No \stex_iterate_symbols:nn \l_stex_current_type_tl {
    \tl_if_empty:nTF{##2}{
      \__stex_structures_do_decl_nomacro:nnnnnnnnn{##3}
    }{
      \__stex_structures_do_decl:nnnnnnnnn{##2}
    }
    {##1}{##3}{##4}{##5}{##6}{##7}{##8}{##9}
```

```
  }
}

\cs_new_protected:Nn \__stex_structures_do_decl_nomacro:nnnnnnnnn {
  \prop_if_in:NnF \l__stex_structures_prop {#1} {
    \seq_put_left:Nx \l__stex_structures_seq {\tl_to_str:n{#2?#3}}
    \prop_put:Nnn \l__stex_structures_prop {#1}{
      {}{
        \_stex_invoke_symbol:nnnnnnnN
        {#2}
        {#3}
        {#4}{#5}{#6}{#7}{#8}#9
      }
    }
  }
}

\cs_new_protected:Nn \__stex_structures_do_decl:nnnnnnnnn {
  \prop_if_in:NnF \l__stex_structures_prop {#1} {
    \seq_put_left:Nx \l__stex_structures_seq {\tl_to_str:n{#2?#3}}
    \prop_put:Nnn \l__stex_structures_prop {#1}{
      {#3}{
        \_stex_invoke_symbol:nnnnnnnN
        {#2}
        {#3}
        {#4}{#5}{#6}{#7}{#8}#9
      }
    }
  }
  %\tl_set:cn{#1}{
  %  \_stex_invoke_symbol:nnnnnnnN
  %  {#2}{#3}{#4}{#5}{#6}{#7}{#8}#9
  %}
  %\tl_put_right:Nn \l__stex_structures_redo_tl {
  %  \tl_set:cn{#1}{
  %    \_stex_invoke_symbol:nnnnnnnN
  %    {#2}{#3}{#4}{#5}{#6}{#7}{#8}#9
  %  }
  %}
}

\cs_new_protected:Nn \__stex_structures_prop_do_notations: {
  \exp_args:No \stex_iterate_notations:nn\l_stex_current_type_tl{
    \exp_args:NNe \seq_if_in:NnT \l__stex_structures_seq {\tl_to_str:n{##1}}{
      \tl_put_right:Nn \l__stex_structures_redo_tl {
        \cs_if_exist:cF{l_stex_notation_##1 _##2_cs}{
          \tl_set:cn{l_stex_notation_##1 _##2_cs}{##4}
        }
        \cs_if_exist:cF{l_stex_notation_##1 __cs}{
          \tl_set:cn{l_stex_notation_##1 __cs}{##4}
        }
      }
      \cs_if_exist:cF{l_stex_notation_##1 _##2_cs}{
        \tl_set:cn{l_stex_notation_##1 _##2_cs}{##4}
```

```
      }
      \cs_if_exist:cF{l_stex_notation_##1 __cs}{
        \tl_set:cn{l_stex_notation_##1 __cs}{##4}
      }
      \tl_if_empty:nF{##5}{
        \tl_put_right:Nn \l__stex_structures_redo_tl {
          \cs_if_exist:cF{l_stex_notation_##1 _op_##2_cs}{
            \tl_set:cn{l_stex_notation_##1 _op_##2_cs}{##5}
          }
          \cs_if_exist:cF{l_stex_notation_##1 _op__cs}{
            \tl_set:cn{l_stex_notation_##1 _op__cs}{##5}
          }
        }
        \cs_if_exist:cF{l_stex_notation_##1 _op_##2_cs}{
          \tl_set:cn{l_stex_notation_##1 _op_##2_cs}{##5}
        }
        \cs_if_exist:cF{l_stex_notation_##1 _op__cs}{
          \tl_set:cn{l_stex_notation_##1 _op__cs}{##5}
        }
      }
    }
  }
}
```

\usestructure

```
\cs_new_protected:Npn \usestructure #1 {
  \stex_get_mathstructure:n{ #1 }
  \seq_clear:N \l__stex_structures_imports_seq
  \clist_map_inline:Nn \l_stex_get_symbol_type_tl {
    \exp_args:Ne \stex_if_module_exists:nT{\tl_to_str:n{##1}}{
      \seq_put_right:Nn \l__stex_structures_imports_seq{##1}
    }
  }
  \seq_map_inline:Nn \l__stex_structures_imports_seq {
    \stex_if_do_html:T {
      \hbox{\stex_annotate_invisible:nn
        {shtml:usemodule=##1} {}}
    }
    \stex_activate_module:n {##1}
  }
}
```

(*End of definition for* \usestructure. *This function is documented on page* 96.)

13.10 Statements

```
⟨@@=stex_statements⟩

\stex_keys_define:nnnn{statement}{
  \str_clear:N \l_stex_key_name_str
  \str_clear:N \l_stex_key_macroname_str
  \clist_clear:N \l_stex_key_for_clist
  \str_clear:N \l_stex_key_args_str
```

```
  \tl_clear:N \l_stex_key_type_tl
  \tl_clear:N \l_stex_key_def_tl
  \tl_clear:N \l_stex_key_return_tl
  \clist_clear:N \l_stex_key_argtypes_clist
}{
  name    .str_set:N = \l_stex_key_name_str ,
  for     .clist_set:N = \l_stex_key_for_clist ,
  macro   .str_set:N = \l_stex_key_macroname_str ,
  % start   .str_set:N = \l_stex_key_title_str , % TODO remove
  type  .tl_set:N = \l_stex_key_type_tl ,
  judgment .code:n = {},
  from .code:n= {}, % TODO remove
  to .code:n={} % TODO remove
}{id,title,style,symargs}
```

\stex_new_statement:nn

```
\cs_new_protected:Npn \_stex_do_for_list: {
  \seq_clear:N \l_stex_fors_seq
  \clist_map_inline:Nn \l_stex_key_for_clist {
    \exp_args:Ne\stex_get_symbol:n{\tl_to_str:n{##1}}
    \seq_put_right:Nx \l_stex_fors_seq
      {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
  }
}

\cs_new_protected:Nn \__stex_statements_setup:nn {
  \str_if_empty:NF \l_stex_key_macroname_str {
    \str_if_empty:NT \l_stex_key_name_str {
      \str_set_eq:NN \l_stex_key_name_str \l_stex_key_macroname_str
    }
  }
  \_stex_do_for_list:
  \str_if_empty:NF \l_stex_key_name_str {
    \__stex_statements_force_id:
    \seq_put_right:Nx \l_stex_fors_seq {
      \l_stex_current_module_str ? \l_stex_key_name_str
    }
    \str_set_eq:NN \l_stex_macroname_str \l_stex_key_macroname_str
    \str_set:Nn \l_stex_key_role_str {#2}
    \stex_symdecl_do:
    \exp_args:Nnx \use:nn {\stex_module_add_symbol:nnnnnnnN}{
      {\l_stex_key_macroname_str}{\l_stex_key_name_str}
      {\int_use:N \l_stex_get_symbol_arity_int}
      {\l_stex_get_symbol_args_tl}
      {#1}{}{}\stex_invoke_symbol:
    }
    \stex_if_do_html:T \_stex_symdecl_html:
  }
  \str_clear:N \l__stex_statements_uri_str
  \str_if_empty:NTF \l_stex_key_name_str {
    \stex_debug:nn{statement}{no~name}
    \int_compare:nNnTF {\seq_count:N \l_stex_fors_seq} = 1 {
      \str_set:Nx \l__stex_statements_uri_str {\seq_item:Nn \l_stex_fors_seq 1}
      \stex_debug:nn{statement}{for:~\l__stex_statements_uri_str}
```

```
    }{
      \stex_debug:nn{statement}{no~for}
    }
  }{
    \str_set:Nx \l__stex_statements_uri_str {\l_stex_current_module_str ? \l_stex_key_name_s
    \stex_debug:nn{statement}{name:~\l__stex_statements_uri_str}
  }
}

\cs_new:Nn \__stex_statements_html_keyvals:nn {
  shtml:#1={},
  shtml:inline={#2},
  \seq_if_empty:NF \l_stex_fors_seq {,
    shtml:fors={\seq_use:Nn \l_stex_fors_seq ,}
  }
  \str_if_empty:NF \l_stex_key_id_str {,
    shtml:id={\stex_uri_use:N \l_stex_current_doc_uri ? \l_stex_key_id_str}
  }
  \clist_if_empty:NF \l_stex_key_style_clist {,
    shtml:styles={\l_stex_key_style_clist}
  }
}

\cs_new_protected:Nn \stex_new_statement:nnn {
  \stex_new_stylable_env:nnnnnnn {#1}{O{}}{
    \stex_keys_set:nn{statement}{##1}
    #3

    \stex_if_smsmode:F {
      \exp_args:Nne \begin{stex_annotate_env}{
        \__stex_statements_html_keyvals:nn{#1}{false}
      }
      \tl_set_eq:NN \thistitle \l_stex_key_title_tl
      \str_set_eq:NN \thisname \l_stex_key_name_str
      \clist_set_eq:NN \thisfor \l_stex_key_for_str
      \stex_if_html_backend:TF {
        \noindent
        \stex_annotate:nn{shtml:statementtitle={}}{\_stex_annotate_force_break:n\l_stex_key_
      }
      \stex_style_apply:
    }
    \_stex_do_id:
    \stex_smsmode_do:
  }{
    \stex_if_smsmode:F {
      \stex_if_html_backend:F \stex_style_apply:
      \end{stex_annotate_env}
    }
  }{}{}{s}
  \stex_sms_allow_env:n{s#1}

  \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
  \stex_reactivate_macro:N \defnotation
  \stex_reactivate_macro:N \definame
```

```
  \stex_reactivate_macro:N \Definame
}
\stex_new_statement:nnn{example}{ex}{\__stex_statements_setup:nn{}{example}}
\stex_new_statement:nnn{paragraph}{}{
  \clist_if_in:NnTF \l_stex_key_style_clist {symdoc}{
    \__stex_statements_force_id:
    \__stex_statements_setup:nn{}{}
    \__stex_statements_setup_def:
  }{
    \__stex_statements_setup:nn{}{}
  }
}
```

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

definiendum

```
\cs_new_protected:Nn \__stex_statements_do_defref:nn {
  \stex_if_html_backend:T{\ifvmode\indent\fi}
  \group_begin:
  \stex_get_symbol:n{#1}
  \bool_if:NTF \l_stex_allow_semantic_bool{
    \str_set:Nx\l_stex_current_symbol_str
      {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
    \str_if_in:NnT \l_stex_get_symbol_name_str / {
      \str_set:Nx \l_stex_get_symbol_name_str {
        \exp_after:wN \_stex_split_slash: \l_stex_get_symbol_name_str
        /\_stex_args_end:
      }
    }
    \exp_args:No \stex_ref_new_sym_target:n \l_stex_current_symbol_str
    \def\comp{\_defcomp}
    \stex_annotate:nn{shtml:definiendum=\l_stex_current_symbol_str}{\comp{#2}}
  }{
    \msg_error:nnxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
  }
  \group_end:
}

\NewDocumentCommand \defnotation{ m } {
  \_stex_next_symbol:n { \def\comp{\_defcomp}}#1
}
\stex_deactivate_macro:Nn \defnotation {definition~environments}

\NewDocumentCommand \definiendum { O{} m m} {
  \stex_keys_set:nn{symname}{ #1 }
  \__stex_statements_do_defref:nn{#2}{#3}
}
\stex_deactivate_macro:Nn \definiendum {definition~environments}

\NewDocumentCommand \definame { O{} m } {
  \stex_keys_set:nn{symname}{#1}
  \__stex_statements_do_defref:nn{#2}{
    \l_stex_key_pre_tl\l_stex_get_symbol_name_str\l_stex_key_post_tl
  }
```

```
}
\stex_deactivate_macro:Nn \definame {definition~environments}

\NewDocumentCommand \Definame { O{} m } {
  \stex_keys_set:nn{symname}{#1}
  \__stex_statements_do_defref:nn{#2}{
    \l_stex_key_pre_tl\exp_after:wN\_stex_capitalize:n\l_stex_get_symbol_name_str\l_stex_key
  }
}
\stex_deactivate_macro:Nn \Definame {definition~environments}


\NewDocumentCommand \definiens { O{} m }{
  \group_begin:
  \str_clear:N \l_stex_get_symbol_name_str
  \tl_if_empty:nF {#1} {
    \stex_get_symbol:n { #1 }
    \str_set:Nx \l__stex_statements_uri_str
      {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
  }
  \str_if_empty:NT \l__stex_statements_uri_str {
    \msg_error:nn{stex}{error/definiensfor}
  }
  \stex_debug:nn{definiens}{Checking~\l__stex_statements_uri_str}

  \exp_args:No \_stex_add_definiens:nn \l__stex_statements_uri_str{#2}

  \group_end:
  \stex_smsmode_do:
}
\stex_deactivate_macro:Nn \definiens {definition~environments}
\stex_sms_allow_escape:N \definiens

\cs_new_protected:Nn \_stex_add_definiens:nn {
  \exp_args:Nno \stex_str_if_starts_with:nnT{#1} \l_stex_current_module_str {
    \prop_map_inline:cn{c_stex_module_\l_stex_current_module_str _symbols_prop}{
      \stex_debug:nn{definiens}{#1 == \l_stex_current_module_str?##1}
      \str_if_eq:noT {#1} {\l_stex_current_module_str?##1}{
        \prop_map_break:n{\_stex_add_definiens_inner:nnnnnnnn ##2}
      }
    }
  }
  \stex_if_smsmode:F{
    \stex_annotate:nn{ shtml:definiens={#1}}{
      #2 %\_stex_annotate_force_break:n{ #2 }
    }
  }
}

\cs_new_protected:Nn \_stex_add_definiens_inner:nnnnnnnn {
  \stex_debug:nn{definiens}{Adding~definiens~to~\l_stex_current_module_str?#2}
  \prop_gput:cnn{c_stex_module_\l_stex_current_module_str _symbols_prop}
    {#2}{{#1}{#2}{#3}{#4}{defed}{#6}{#7}{#8}}
}
```

```
\NewDocumentCommand \varbind {m} {
  \clist_map_inline:nn {#1} {
    \stex_get_var:n {##1}
    \stex_if_do_html:T {
      \stex_annotate_invisible:nn {shtml:bind=\l_stex_get_symbol_name_str}{}
    }
  }
}
\stex_deactivate_macro:Nn \varbind {definition~or~assertion~environments}

\NewDocumentCommand \conclusion { O{} m} {
  \group_begin:
  \str_clear:N \l_stex_get_symbol_name_str
  \tl_if_empty:nF {#1} {
    \stex_get_symbol:n { #1 }
    \str_set:Nx \l__stex_statements_uri_str
      {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
  }
  \str_if_empty:NT \l__stex_statements_uri_str {
    \msg_error:nn{stex}{error/conclusionfor}
  }
  \stex_annotate:nn{ shtml:conclusion=\l__stex_statements_uri_str}{
    #2 %\_stex_annotate_force_break:n{ #2 }
  }
  \group_end:
}
\stex_deactivate_macro:Nn \conclusion {assertion~environments}

\NewDocumentCommand \premise {O{} m} {
  \tl_if_empty:nF {#1} {
    \stex_debug:nn{Here:}{Variable~#1}
    \exp_args:Nne\use:nn{\vardef}{{v#1}[name=#1]{#1}}
  }
  \stex_annotate:nn{shtml:premise={#1}}{#2}
}
\stex_deactivate_macro:Nn \premise {assertion~environments}
```

(*End of definition for* `definiendum`. *This function is documented on page 100.*)

13.11 Proofs

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

```
⟨@@=stex_proof⟩
\stex_keys_define:nnnn{ spf }{
\tl_clear:N \l_stex_key_for_clist
\tl_clear:N \l_stex_key_from_tl
\tl_set_eq:NN \l_stex_key_proofend_tl \__stex_proof_proof_box_tl
\tl_clear:N \l_stex_key_continues_tl
\tl_clear:N \l_stex_key_term_tl
\tl_clear:N \l_stex_key_functions_tl
\tl_clear:N \l_stex_key_method_tl
  \bool_set_false:N \l_stex_key_hide_bool
```

```
}{
  for         .clist_set:N  = \l_stex_key_for_clist ,
  from        .tl_set:N     = \l_stex_key_from_tl ,
  proofend    .tl_set:N     = \l_stex_key_proofend_tl,
  continues   .tl_set:N     = \l_stex_key_continues_tl,
  functions   .tl_set:N     = \l_stex_key_functions_tl,
  term        .tl_set:N     = \l_stex_key_term_tl,
  method      .tl_set:N     = \l_stex_key_method_tl,
  hide        .bool_set:N   = \l_stex_key_hide_bool
}{id,style,title}

\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.

```
\intarray_new:Nn\l__stex_proof_counter_intarray{50}
\cs_new_protected:Npn \__stex_proof_insert_number: {
  \int_set:Nn \l_tmpa_int {1}
  \bool_while_do:nn {
    \int_compare_p:nNn {
      \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
    } > 0
  }{
    \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int .
    \int_incr:N \l_tmpa_int
  }
}
\cs_new_protected:Nn \__stex_proof_number_as_string:N {
  \str_clear:N #1
  \int_set:Nn \l_tmpa_int {1}
  \bool_while_do:nn {
    \int_compare_p:nNn {
      \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
    } > 0
  }{
    \str_put_right:Nx #1 {\intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int .}
    \int_incr:N \l_tmpa_int
  }
}

\cs_new_protected:Npn \__stex_proof_inc_counter: {
  \int_set:Nn \l_tmpa_int {1}
  \bool_while_do:nn {
    \int_compare_p:nNn {
      \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
    } > 0
  }{
    \int_incr:N \l_tmpa_int
```

```
  }
  \int_compare:nNnF \l_tmpa_int = 1 {
    \int_decr:N \l_tmpa_int
  }
  \intarray_gset:Nnn \l__stex_proof_counter_intarray \l_tmpa_int {
    \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int + 1
  }
}

\cs_new_protected:Npn \__stex_proof_add_counter: {
  \int_set:Nn \l_tmpa_int {1}
  \bool_while_do:nn {
    \int_compare_p:nNn {
      \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
    } > 0
  }{
    \int_incr:N \l_tmpa_int
  }
  \intarray_gset:Nnn \l__stex_proof_counter_intarray \l_tmpa_int { 1 }
}

\cs_new_protected:Npn \__stex_proof_remove_counter: {
  \int_set:Nn \l_tmpa_int {1}
  \bool_while_do:nn {
    \int_compare_p:nNn {
      \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
    } > 0
  }{
    \int_incr:N \l_tmpa_int
  }
  \int_decr:N \l_tmpa_int
  \intarray_gset:Nnn \l__stex_proof_counter_intarray \l_tmpa_int { 0 }
}
```

spfsketch

```
\newenvironment{spfsketchenv}{}{}
\stex_new_stylable_cmd:nnnn{spfsketch}{O{} m}{\par
  \begin{spfsketchenv}
  \stex_keys_set:nn{spf}{#1}
  \_stex_do_for_list:
  \_stex_do_id:
  \exp_args:Ne \stex_annotate:nn{
    shtml:proofsketch={
      \seq_if_empty:NF \l_stex_fors_seq {
        \seq_use:Nn \l_stex_fors_seq ,
      }
    }
  }{
    \stex_style_apply:
    #2
  }
  \end{spfsketchenv}
}{
  \noindent\emph{\spfsketchenvautorefname :}~
```

```
}
```

(*End of 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

```
\tl_set:Nn \__stex_proof_proof_box_tl {
  \ltx@ifpackageloaded{amssymb}{$\square$}{
    \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
  }
}

\tl_set:Nn \sproofend {
  \tl_if_empty:NF \l_stex_key_proofend_tl {
    \hfil\null\nobreak\hfill\l_stex_key_proofend_tl\par\smallskip
  }
}
```

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

`\stexcommentfont`

```
\cs_new_protected:Npn \stexcommentfont {
  \small\itshape
}
```

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

`sproof` (*env.*)

```
\cs_new_protected:Nn \__stex_proof_start_list:n {
  \begin{list}{}{
    \setlength\topsep{0pt}
    \setlength\parsep{0pt}
    \setlength\rightmargin{0pt}
  }\item[#1]
}
\cs_new_protected:Nn \__stex_proof_end_list: {
  \end{list}
}

\cs_new_protected:Nn \__stex_proof_html: {
  \stex_annotate_invisible:n{\hbox{
    \tl_if_empty:NF \l_stex_key_term_tl {
      $\stex_annotate:nn{shtml:proofterm={}}{\l_stex_key_term_tl}$
    }
    \tl_if_empty:NF \l_stex_key_method_tl {
      \stex_annotate:nn{shtml:proofmethod={}}{\l_stex_key_method_tl}
    }
  }}
}

\cs_new_protected:Nn \__stex_proof_html_env:n {
  \exp_args:Nne \begin{stex_annotate_env}{
    shtml:#1={
      \seq_if_empty:NF \l_stex_fors_seq {
        \seq_use:Nn \l_stex_fors_seq ,
```

```
      }
    }
    \bool_if:NT \l_stex_key_hide_bool {,
      shtml:proofhide=true
    }
  }
  \__stex_proof_html:
}

\bool_set_false:N \l__stex_proof_in_spfblock_bool
\cs_new_protected:Nn \__stex_proof_begin_proof:nn {\par
  \intarray_gzero:N \l__stex_proof_counter_intarray
  \intarray_gset:Nnn \l__stex_proof_counter_intarray 1 1
  \stex_keys_set:nn{spfsteps}{#1}
  \_stex_do_for_list:
  \stex_if_do_html:T {
    \__stex_proof_html_env:n{proof}
  }
  \seq_map_inline:Nn \l_stex_fors_seq {
    \stex_debug:nn{definiens}{Adding~definiens~to~##1}
    \_stex_add_definiens:nn {##1}{\STEXinvisible{proven}}
  }
  \stex_style_apply:
  \_stex_do_id:
  \stex_reactivate_macro:N \subproof
  \stex_reactivate_macro:N \spfstep
  \stex_reactivate_macro:N \conclude
  \stex_reactivate_macro:N \assumption
  \stex_reactivate_macro:N \eqstep
  \stex_reactivate_macro:N \yield
  \stex_reactivate_macro:N \spfblock
  \stex_reactivate_macro:N \spfjust
  \stex_annotate:nn{shtml:prooftitle={}}{#2}
  \stex_if_do_html:T{
    \begin{stex_annotate_env}{shtml:proofbody={}}
  }
}
\stex_new_stylable_env:nnnnnnn{proof}{O{} m}{
  \__stex_proof_begin_proof:nn{#1}{#2}
  \bool_set_true:N\l__stex_proof_in_spfblock_bool\__stex_proof_start_list:n{}
  \group_begin:\stexcommentfont
}{
  \stex_style_apply:
  \stex_if_do_html:T{\end{stex_annotate_env}\end{stex_annotate_env}}
}{
  \emph{\sproofautorefname :}~
}{
  \sproofend
}{s}
\AddToHook{env/sproof/end}{
  \bool_if:NT\l__stex_proof_in_spfblock_bool {
    \group_end:\__stex_proof_end_list:
  }
}
```

```
\stex_new_stylable_env:nnnnnnn{proof*}{O{}}{
  \__stex_proof_begin_proof:nn{#1}{}
  \bool_set_false:N\l__stex_proof_in_spfblock_bool
}{
  \stex_style_apply:
  \stex_if_do_html:T{\end{stex_annotate_env}\end{stex_annotate_env}}
}{
  \emph{Proof:}~
}{
  \sproofend
}{s}
```

subproof (*env.*)

```
\str_set_eq:NN \subproofautorefname \spfstepautorefname
\stex_new_stylable_env:nnnnnnn{subproof}{s O{} m}{\par
  \stex_keys_set:nn{spf}{#2}
  \_stex_do_for_list:
  \stex_if_do_html:T {
    \__stex_proof_html_env:n{subproof}
  }
  \seq_map_inline:Nn \l_stex_fors_seq {
    \stex_debug:nn{definiens}{Adding~definiens~to~##1}
    \_stex_add_definiens:nn {##1}{\STEXinvisible{proven}}
  }

  \IfBooleanTF #1 {
    \stex_style_apply:
    \str_if_empty:NF \l_stex_key_id_str {
      \__stex_proof_number_as_string:N \@currentlabel
      \str_set:Nx \@currentHref{subproof.\@currentlabel}
      \_stex_do_id:
    }
    \bool_set_false:N \l__stex_proof_in_spfblock_bool
    \stex_annotate:nn{shtml:prooftitle={}}{#3}
  }{
    \bool_if:NTF \l__stex_proof_in_spfblock_bool {
      \str_if_empty:NF \l_stex_key_id_str {
        \__stex_proof_number_as_string:N \@currentlabel
        \str_set:Nx \@currentHref{subproof.\@currentlabel}
        \_stex_do_id:
      }
      \__stex_proof_start_list:n\__stex_proof_insert_number:
        \stex_annotate:nn{shtml:prooftitle={}}{#3}
        \__stex_proof_add_counter:
        \stex_style_apply:
    }{
      \stex_annotate:nn{shtml:prooftitle={}}{#3}
      \stex_style_apply:
      \_stex_do_id:
    }
  }
  \stex_if_do_html:T{
    \begin{stex_annotate_env}{shtml:proofbody={}}
```

```
  }
  \bool_if:NT \l__stex_proof_in_spfblock_bool {\group_begin:\stexcommentfont}
}{
  \stex_style_apply:
  \bool_if:NT \l__stex_proof_in_spfblock_bool \__stex_proof_inc_counter:
  \stex_if_do_html:T{\end{stex_annotate_env}}
  \bool_if:NT\l__stex_proof_in_spfblock_bool \__stex_proof_end_list:
  \stex_if_do_html:T{\end{stex_annotate_env}}
  \aftergroup \__stex_proof_inblock_restore:
}{}{}{}
\AddToHook{env/subproof/before}{
  \bool_if:NT \l__stex_proof_in_spfblock_bool \group_end:
}
\AddToHook{env/subproof/end}{
  \bool_if:NT\l__stex_proof_in_spfblock_bool {
    \group_end:\__stex_proof_remove_counter:
    %\__stex_proof_end_list:
  }
}
\stex_deactivate_macro:Nn \subproof {sproof~environments}

\cs_new_protected:Nn \__stex_proof_inblock_restore: {
  \bool_if:NT\l__stex_proof_in_spfblock_bool {
    \group_begin:\stexcommentfont
  }
}
```

\spfstep
\conclude
\assumption
\have
\eqstep

```

\stex_keys_define:nnnn { spfsteps } {
\clist_clear:N \l_stex_key_for_clist
  \str_clear:N \l_stex_key_name_str
\tl_clear:N \l_stex_key_method_tl
\tl_clear:N \l_stex_key_term_tl
}{
  for         .clist_set:N  = \l_stex_key_for_clist ,
  method      .tl_set:N     = \l_stex_key_method_tl,
  term        .tl_set:N     = \l_stex_key_term_tl,
  name        .str_set_x:N  = \l_stex_key_name_str
  % todo: style=inline
}{id,style,title}

\newenvironment{spfstepenv}{
  \str_set_eq:NN \spfstepenvautorefname \spfstepautorefname
}{}

\cs_new_protected:Nn \__stex_proof_step_html:nn {
  \stex_if_do_html:TF{
    \exp_args:Ne \stex_annotate:nn{
      shtml:spf#1={
        \seq_if_empty:NF \l_stex_fors_seq {
          \seq_use:Nn \l_stex_fors_seq ,
        }
      }
```

```
      \str_if_empty:NF \l_stex_key_name_str {,
        shtml:stepname={\l_stex_key_name_str}
      }
    }{
      \__stex_proof_html:
      #2
    }
  }{ #2 }
}

\cs_new_protected:Nn \__stex_proof_make_step_macro:Nnnnn {
  \NewDocumentCommand #1 {s O{} +m} {
    \bool_if:NT \l__stex_proof_in_spfblock_bool \group_end:
    \stex_keys_set:nn{spfsteps}{##2}
    \str_if_empty:NF \l_stex_key_name_str {
      \stex_debug:nn{Here:}{Variable~\l_stex_key_name_str}
      \exp_args:Nne\use:nn{\vardef}{{v\l_stex_key_name_str}[name=\l_stex_key_name_str]{\l_st
    }

    \begin{spfstepenv}
      \str_if_empty:NF \l_stex_key_id_str {
        \__stex_proof_number_as_string:N \@currentlabel
        \str_set:Nx \@currentHref{spfstep.\@currentlabel}
        \_stex_do_id:
      }

    \bool_if:NTF \l__stex_proof_in_spfblock_bool {
      \IfBooleanTF ##1 {
        \__stex_proof_step_html:nn{#2}{##3}
      }{
        \__stex_proof_step_html:nn{#2}{\__stex_proof_start_list:n{#3} ##3 \__stex_proof_end_
        #5
      }
      \end{spfstepenv}
      \group_begin:\stexcommentfont
    }{
      \__stex_proof_step_html:nn{#2}{##3}
      \end{spfstepenv}
    }
  }
  \stex_deactivate_macro:Nn #1 {sproof~environments}
}

\__stex_proof_make_step_macro:Nnnnn \assumption {assumption} \__stex_proof_insert_number: {}
\__stex_proof_make_step_macro:Nnnnn \conclude {conclusion} {$\Rightarrow$} {} {}
\__stex_proof_make_step_macro:Nnnnn \spfstep {step} \__stex_proof_insert_number: {} \__stex_

\NewDocumentCommand \eqstep {s m}{
  \bool_if:NTF \l__stex_proof_in_spfblock_bool {
    \group_end:
    \IfBooleanTF #1 {
      \__stex_proof_step_html:nn{eqstep}{$= #2$}
    }{
      \__stex_proof_step_html:nn{eqstep}{\__stex_proof_start_list:n{$=$} $#2$ \__stex_proof_
```

```
    }
    \group_begin:\stexcommentfont
  }{
    \__stex_proof_step_html:nn{eqstep}{$= #2$}
  }
}
\stex_deactivate_macro:Nn \eqstep {sproof~environments}

\NewDocumentCommand \yield {+m}{
  \stex_annotate:nn{shtml:proofterm={}}{ #1 }
}
\stex_deactivate_macro:Nn \yield {sproof~environments}

\NewDocumentEnvironment{spfblock}{}{
  \bool_set_false:N \l__stex_proof_in_spfblock_bool
}{
  \aftergroup\__stex_proof_inblock_restore:
}
\stex_deactivate_macro:Nn \spfblock {sproof~environments}
\AddToHook{env/spfblock/before}{
  \bool_if:NT \l__stex_proof_in_spfblock_bool \group_end:
}


\newcommand\spfjust[1]{
  \stex_annotate:nn{spfjust={}}{ #1 }
}
\stex_deactivate_macro:Nn \spfjust {sproof~environments}
```

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

13.12 Metatheory

```
⟨@@=stex_meta⟩
\group_begin:
  \cs_set:Npn \__stex_modules_persist_module: {}
  \cs_set:Npn \stex_check_term:n #1 {}
  \cs_set:Npn \_stex_sref_do_aux:n #1 { #1 }
  \bool_set_false:N \_stex_html_do_output_bool
  \bool_set_false:N \c_stex_check_terms_bool
  \stex_uri_resolve:Nn \l_stex_current_ns_uri {http://mathhub.info/sTeX/meta}
  \stex_module_setup:n{Metatheory}

  \symdef{of~type}[args=ii,invisible]{#1}
  \notation{of~type}[colon]{#1 \mathbin{\comp{:}} #2}

  \symdef{apply}[args=ia,prec=0;\infprec x\infprec]{#1\mathopen{\comp(} #2 \mathclose{\comp)
  \notation{apply}[lambda]{#1\; \argsep{#2}{\;}}
  \notation{apply}[infixop]{\argsep{#2}{\mathbin{#1}}}
  \notation{apply}[infixrel]{\argsep{#2}{\mathrel{#1}}}

  % structures
  \symdef{module~type}[args=i,op=\mathtt{MOD}]
  {\mathopen{\comp{\mathtt{MOD}(}}#1\mathclose{\comp{)}}}
```

```
  \symdef{module~type~merge}[args=a,op=\oplus]
    {\argsep{#1}{\mathbin{\comp{\oplus}}}}
  \symdef{anonymous~record}[args=a]
    {\mathopen{\comp{[[}}#1\mathclose{\comp{]]}}}
  \symdef{record~field}[args=2]{#1\comp{.}#2}
  \symdecl*{record~type}

  \symdecl{mathstruct}[name=mathematical~structure,args=a] % TODO
  \notation{mathstruct}[angle,prec=nobrackets]
    {\mathopen{\comp\langle} #1 \mathclose{\comp\rangle}}
  \notation{mathstruct}[parens,prec=nobrackets]
    {\mathopen{\comp(} #1 \mathclose{\comp)}}

  % sequences
  \symdef{ellipses}[ldots]{\ldots}
  \symdef{sequence~expression}[comma,args=a]{#1}
  \symdef{sequence~type}[args=1]{#1^{\comp\ast}}
  \symdef{sequence~map}[args=ia]{
    \comp{\mathrm{map}}\mathopen{\comp{(}}#1\mathpunct{\comp{,}}
    #2\mathclose{\comp{)}}}
  }
\iffalse
  % binder (\forall, \Pi, \lambda etc.)
  \symdef{pibind}[name=dependent function type,prec=nobrackets,
    op=(\cdot)\;\to\;\cdot,args=Bi,assoc=pre]
    {\argmap{#1}{
      \mathopen{\comp(} ##1 \mathclose{\comp{)}}}
      }{\mathbin{\comp{\to}}} \mathbin{\comp{\to}} #2}
  \notation{pibind}[forall]{\comp\forall #1\mathpunct{\comp.} #2}
  \notation{pibind}[Pi]{\mathop{\comp\prod}\c_math_subscript_token{#1}#2}

  \symdef{mapbind}[name=lambda,mapsto,prec=nobrackets,op=\mapsto,args=Bi,assoc=pre]
    { #1 \mathrel{\comp\mapsto} #2}
  \notation{mapbind}[lambda,prec=nobrackets,op=\lambda]
    {\comp\lambda #1 \mathpunct{\comp.} #2}
\fi
  \symdecl{bind}[args=Bi,assoc=pre]
  \notation{bind}[depfun,prec=nobrackets,op=(\cdot)\;\to\;\cdot]
    {\mathopen{\comp(} #1 \mathclose{\comp{)}}\mathbin{\comp{\to}}} #2}
  \notation{bind}[forall]{\comp\forall #1.\;#2}
  \notation{bind}[Pi]{\mathop{\comp\prod}\c_math_subscript_token{#1}#2}

  \symdef{implicit~bind}[args=Bi,assoc=pre]{\mathopen{\comp\{} #1 \mathclose{\comp{\}}\c_math

  \symdecl*{integer~literal}
  \notation{integer~literal}{\mathbb Z}

  \symdecl*{ordinal}
  \notation{ordinal}{\mathtt{Ord}}

  % propositions
  \symdef{prop}[name=proposition]{\mathtt{Prop}}
  \symdef{judgment~holds}[args=i,role=judgment]{\comp\vdash\;#1}

```

```
  % any object
  \symdef{object}{\mathtt{Obj}}

  % TODO DELETE
  \symdef{aseqdots}[args=a,prec=nobrackets]
    {#1\comp{,\ldots}}%{##1\comp,##2}
  \symdef{aseqfromto}[args=ai,prec=nobrackets]
    {#1\comp{,\ldots,}#2}%{##1\comp,##2}
  \symdef{aseqfromtovia}[args=aii,prec=nobrackets]
    {#1\comp{,\ldots,}#2\comp{,\ldots,}#3}%{##1\comp,##2}


  \stex_close_module:
  \stex_uri_add_module:NNn \l_stex_metatheory_uri \l_stex_current_ns_uri {Metatheory}
  \global \let \l_stex_metatheory_uri \l_stex_metatheory_uri
  \global \let \c_stex_default_metatheory \l_stex_metatheory_uri
\group_end:
```

13.13 MMT Interfaces

```
⟨@@=todo⟩
\cs_new_protected:Npn \MSC #1 {}
```

\MMTinclude

```
\stex_new_stylable_cmd:nnnn{MMTinclude}{m}{
  \stex_annotate_invisible:nn{shtml:import={#1}}{}
}{}
\stex_deactivate_macro:Nn \MMTinclude {module~environments}
\stex_every_module:n {\stex_reactivate_macro:N \MMTinclude}
```

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

\MMTrule

```
\NewDocumentCommand \MMTrule {m m}{
  \tl_if_empty:nTF{#2}{\seq_clear:N \l_tmpa_seq}{
    \seq_set_split:Nnn \l_tmpa_seq , {#2}
  }
  \int_zero:N \l_tmpa_int
  \stex_annotate_invisible:n{
    $
    \stex_annotate:nn{shtml:rule={scala://#1}}{
      \_stex_annotate_force_break:n{
        \seq_if_empty:NF \l_tmpa_seq {
          \seq_map_inline:Nn \l_tmpa_seq {
            \int_incr:N \l_tmpa_int
            \stex_annotate:nn{
              shtml:argmode=i,
              shtml:arg={\int_use:N \l_tmpa_int}
            }{ ##1 }
          }
        }
      }
    }$
  }
}
```

```
\stex_deactivate_macro:Nn \MMTrule {module~environments}
\stex_every_module:n{\stex_reactivate_macro:N \MMTrule}
```

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

`mmtinterface` (*env.*)

```
\NewDocumentEnvironment { mmtinterface } {  O{} m m } {
  \_stex_module_setup_top_nosig:n { #3 }
  \str_set_eq:NN \l__todo_mmt_module_str \l_stex_current_module_str
  \str_clear:N \l_stex_current_module_str
  \stex_keys_set:nn { smodule }{ #1 }
  \stex_module_setup:n{ #2 }
  \str_set_eq:NN \l__todo_stex_module_str \l_stex_current_module_str
  \stex_debug:nn{mmt}{Interface~\l__todo_stex_module_str^^Jfor~\l__todo_mmt_module_str}

  \stex_if_do_html:T {
    \exp_args:Nne \begin{stex_annotate_env} {
      shtml:theory={\l_stex_current_module_str},
      shtml:language={ \l_stex_current_language_str},
      shtml:signature={}
      \tl_if_empty:NF \l_stex_metatheory_uri {,
        shtml:metatheory={\stex_uri_use:N \l_stex_metatheory_uri}
      }
    }
    \stex_annotate_invisible:n{}
    \stex_annotate_invisible:nn
      {shtml:import=\l__todo_mmt_module_str} {}
  }
  \stex_module_add_code:x{
    \stex_activate_module:n{ \l__todo_mmt_module_str }
  }
  \stex_module_add_morphism:nonn
    {}{\l__todo_mmt_module_str}{import}{}
  \stex_reactivate_macro:N \mmtdef
  \stex_smsmode_do:
}{
    \str_set_eq:NN \l_stex_current_module_str \l__todo_mmt_module_str
    \stex_close_module:
    \str_set_eq:NN \l_stex_current_module_str \l__todo_stex_module_str
    \stex_close_module:
    \stex_if_do_html:T { \end{stex_annotate_env} }
}
\stex_sms_allow_env:n{mmtinterface}
```

`\mmtdef`

```
\NewDocumentCommand \mmtdef {m O{} m} {
  \stex_keys_set:nn{symdef}{#2}
  \str_set:Nx \l_stex_macroname_str { #1 }
  \str_if_empty:NT \l_stex_key_name_str {
    \str_set:Nx \l_stex_key_name_str { #1 }
  }
  \stex_symdecl_do:

  \str_set_eq:NN \l_stex_current_module_str \l__todo_mmt_module_str
```

```
  \cs_set_eq:NN \l__todo_old_metagroup_cd \stex_metagroup_do_in:nn
  \cs_set_protected:Npn \stex_metagroup_do_in:nn ##1 ##2 {##2}
  \exp_args:Nnx \use:nn {\stex_module_add_symbol:nnnnnnnN}{
    {\l_stex_macroname_str}
    {\l_stex_key_name_str}
    {\int_use:N \l_stex_get_symbol_arity_int}
    {\l_stex_get_symbol_args_tl}
    {}
    {}
    {}
    \stex_invoke_symbol:
  }
  \cs_set_eq:NN \stex_metagroup_do_in:nn \l__todo_old_metagroup_cd
  \str_set_eq:NN \l_stex_current_module_str \l__todo_stex_module_str

  \str_set_eq:NN \l_stex_get_symbol_mod_str \l__todo_mmt_module_str
  \str_set_eq:NN \l_stex_get_symbol_name_str \l_stex_key_name_str
  \stex_notation_parse:n{#3}
  \_stex_notation_check:
  \_stex_notation_add:
  \stex_if_do_html:T{
    \_stex_notation_do_html:n{\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
  }
  \stex_smsmode_do:
}
\stex_deactivate_macro:Nn \mmtdef {mmtinterface~environments}
\stex_sms_allow_escape:N \mmtdef
```

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

VoLL-KI Annotations

```
\newcommand\precondition[2]{
    \str_clear:N \l_stex_get_symbol_name_str
    \stex_get_symbol:n{#2}
    \str_if_empty:NTF \l_stex_get_symbol_name_str{
        \errmessage{Unknown~symbol~#2}
    }{
  \str_case:nnTF {#1}{
      {remember}{}
      {understand}{}
      {analyze}{}
      {evaluate}{}
      {apply}{}
      {create}{}
  }{
   \stex_annotate_invisible:nn{
   shtml:preconditionsymbol={\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str},
   shtml:preconditiondimension={#1}
   }{}
  }{\errmessage{Unknown~cognitive~dimension~#1}}
    }
}
\newcommand\objective[2]{
    \str_clear:N \l_stex_get_symbol_name_str
    \stex_get_symbol:n{#2}
```

```
    \str_if_empty:NTF \l_stex_get_symbol_name_str{
        \errmessage{Unknown~symbol~#2}
    }{
  \str_case:nnTF {#1}{
{remember}{}
{understand}{}
{analyze}{}
{evaluate}{}
{apply}{}
{create}{}
  }{
   \stex_annotate_invisible:nn{
   shtml:objectivesymbol={\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str},
   shtml:objectivedimension={#1}
   }{}
  }{\errmessage{Unknown~cognitive~dimension~#1}}
    }
}
\seq_if_empty:NT \g_stex_current_file {
  \seq_gset_eq:NN \g_stex_current_file \c_stex_main_file
}
\_stex_persist_read_now:
\_stex_every_file:
\cs_new_protected:Nn \__todo_newlabel:n {
  \exp_args:Ne\__todo_old_newlabel:{\tl_to_str:n{#1}}
}
\AtBeginDocument{
  \iow_now:Nn \@auxout {
    \ExplSyntaxOn
    \let\__todo_old_newlabel:\newlabel
    \let\newlabel\__todo_newlabel:n
    \ExplSyntaxOff
  }
}
⟨/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.

```
⟨*cls⟩
⟨@@=notesslides⟩
\ProvidesExplClass{notesslides}{2023/10/13}{3.4.0}{notesslides Class}
\RequirePackage{l3keys2e}

\str_const:Nn \c__notesslides_class_str {article}

\keys_define:nn{notesslides / cls}{
  class   .str_set_x:N = \c__notesslides_class_str,
  notes   .bool_set:N  = \c_notesslides_notes_bool ,
  slides  .code:n      = { \bool_set_false:N \c_notesslides_notes_bool },
  %docopt  .str_set_x:N = \c__notesslides_docopt_str,
  unknown .code:n      = {
    \PassOptionsToClass{\CurrentOption}{beamer}
    \PassOptionsToClass{\CurrentOption}{\c__notesslides_class_str}
    \PassOptionsToPackage{\CurrentOption}{notesslides}
    \PassOptionsToPackage{\CurrentOption}{stex}
  }
}
\ProcessKeysOptions{ notesslides / cls }

\RequirePackage{stex}
\stex_if_html_backend:T {
  \bool_set_true:N\c_notesslides_notes_bool
}

\bool_if:NTF \c_notesslides_notes_bool {
  \PassOptionsToPackage{notes=true}{notesslides}
  \message{notesslides.cls:~Formatting~document~in~notes~mode}
```

```
}{
  \PassOptionsToPackage{notes=false}{notesslides}
  \message{notesslides.cls:~Formatting~document~in~slides~mode}
}

\bool_if:NTF \c_notesslides_notes_bool {
  \LoadClass{\c__notesslides_class_str}
}{
  \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
  %\newcounter{Item}
  %\newcounter{paragraph}
  %\newcounter{subparagraph}
  %\newcounter{Hfootnote}
}
\RequirePackage{notesslides}
```
⟨/cls⟩

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

⟨*package⟩
```
\ProvidesExplPackage{notesslides}{2023/10/13}{3.4.0}{notesslides Package}
\RequirePackage{l3keys2e}

\keys_define:nn{notesslides / pkg}{
  notes           .bool_set:N   = \c_notesslides_notes_bool ,
  slides          .code:n       = { \bool_set_false:N \c_notesslides_notes_bool },
  sectocframes    .bool_set:N   = \c_notesslides_sectocframes_bool ,
  topsect         .str_set_x:N  = \c_notesslides_topsect_str,
  unknown         .code:n       = {
    \PassOptionsToPackage{\CurrentOption}{stex}
    \PassOptionsToPackage{\CurrentOption}{tikzinput}
  }
}
\ProcessKeysOptions{ notesslides / pkg }

\RequirePackage{stex}
\stex_if_html_backend:T {
  \bool_set_true:N\c_notesslides_notes_bool
}

\cs_set:Npn \sectiontitleemph #1 {
  \textbf{\Large #1}
}

\newif\ifnotes
\bool_if:NTF \c_notesslides_notes_bool {
  \notestrue
  \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
  \RequirePackage[noamsthm,hyperref]{beamerarticle}
  \RequirePackage{mdframed}
  \str_if_empty:NTF \c_notesslides_topsect_str{
    %\setsectionlevel{section}
  } {
    \exp_args:No \setsectionlevel \c_notesslides_topsect_str
  }
```

```
}{
  \notesfalse

  \cs_new_protected:Nn \__notesslides_do_sectocframes: {
    \cs_set_protected:Nn \__notesslides_do_label:n {
      \str_case:nnF{##1}{
        {part}    {
          \tl_set:Nx\l__notesslides_num{\thepart}
          \tl_set:cx{@ @ label}{
            \cs_if_exist:NTF\parttitlename{\exp_not:N\parttitlename}{\exp_not:N\partname}{}~
        }
        {chapter} {
          \tl_set:Nx\l__notesslides_num{\thechapter}
          \tl_set:cx{@ @ label}{
            \cs_if_exist:NTF\chaptertitlename{\exp_not:N\chaptertitlename}{\exp_not:N\chapte
        }
        {section} {
          \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\thechapter.}\thesection}
          \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
        }
        {subsection} {
          \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\thechapter.}\thesection.
          \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
        }
        {subsubsection} {
          \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\thechapter.}\thesection.
          \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
        }
        {paragraph} {
          \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\thechapter.}\thesection.
          \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
        }
      }{
        \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\thechapter.}\thesection.\t
        \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
      }
    }
    \cs_set_protected:Nn \_sfragment_do_level:nn {
      \tl_if_exist:cT{c@##1}{\stepcounter{##1}}
      \addcontentsline{toc}{##1}{\protect\numberline{\use:c{the##1}}##2}
      \__notesslides_do_label:n{##1}
      \pdfbookmark[\int_use:N \l_stex_docheader_sect]{\l__notesslides_num\ ##2}{##1.\l__note
      \begin{frame}[noframenumbering]
        \vfill\centering
        \sectiontitleemph{
          \use:c{@ @ label} ##2
        }
      \end{frame}
      \int_incr:N \l_stex_docheader_sect
      \tl_set:Nn \stex_current_section_level{##1}
    }
  }

  \AtBeginDocument{
```

```
    \str_if_empty:NTF \c_notesslides_topsect_str {
      \setsectionlevel{section}
    } {
      \exp_args:No \setsectionlevel \c_notesslides_topsect_str
      \exp_args:No \str_if_eq:nnTF \c_notesslides_topsect_str {chapter} {
        \__notesslides_define_chapter:
      }{
        \exp_args:No \str_if_eq:nnT \c_notesslides_topsect_str {part} {
          \__notesslides_define_chapter:
          \__notesslides_define_part:
        }
      }
    }
  }

  \bool_if:NT \c_notesslides_sectocframes_bool {
    \__notesslides_do_sectocframes:
  }
}

\cs_new_protected:Nn \__notesslides_define_chapter: {
  \cs_if_exist:NF \chaptername {
    \cs_set_protected:Npn \chaptername {Chapter}
  }
  \cs_if_exist:NF \chapter {
    \cs_set_protected:Npn \chapter {INVALID}
  }
  \cs_if_exist:NF \c@chapter {
    \newcounter{chapter}\counterwithin*{section}{chapter}
  }
}

\cs_new_protected:Nn \__notesslides_define_part: {
  \cs_if_exist:NF \partname {
    \cs_set_protected:Npn \partname {Part}
  }
  \cs_if_exist:NF \part {
    \cs_set_protected:Npn \part {INVALID}
  }
  \cs_if_exist:NF \c@part {
    \newcounter{part}\counterwithin*{chapter}{part}
  }
}
```

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

```
\def \c__notesslides_document_str{document}
\newcommand\afterprematurestop{}
\def\prematurestop@endsfragment{
  \unless\ifx\@currenvir\c__notesslides_document_str
    \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafte
    \expandafter\prematurestop@endsfragment
  \fi
}
```

```
\providecommand\prematurestop{
  \stex_if_html_backend:F{
  \message{Stopping~sTeX~processing~prematurely}
  \prematurestop@endsfragment
  \afterprematurestop
  \end{document}
  }
}
```

(*End of definition for* `\prematurestop`. *This function is documented on page* 108.)

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.

```
\bool_if:NT \c_notesslides_notes_bool {
  \renewcommand\usetheme[2][]{\usepackage[#1]{beamertheme#2}}
}
\NewDocumentCommand \libusetheme {O{} m} {
  \libusepackage[#1]{beamertheme#2}
}
```

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

```
\newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
\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.

```
\ifnotes

\newlength{\slideframewidth}
\setlength{\slideframewidth}{1.5pt}
```

`frame` (*env.*) We first define the keys.

```
\cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
  \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
    \bool_set_true:N #1
  }{
    \bool_set_false:N #1
  }
}

\stex_keys_define:nnnn{notesslides / frame}{
  \str_clear:N \l__notesslides_frame_label_str
  \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
  \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
  \bool_set_true:N \l__notesslides_frame_fragile_bool
  \bool_set_true:N \l__notesslides_frame_shrink_bool
  \bool_set_true:N \l__notesslides_frame_squeeze_bool
  \bool_set_true:N \l__notesslides_frame_t_bool
}{
  label               .str_set_x:N  = \l__notesslides_frame_label_str,
  allowframebreaks    .code:n       = {
```

```
    \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
  },
  allowdisplaybreaks  .code:n       = {
    \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
  },
  fragile             .code:n       = {
    \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
  },
  shrink              .code:n       = {
    \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
  },
  squeeze             .code:n       = {
    \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
  },
  t                   .code:n       = {
    \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
  },
  unknown   .code:n       = {}
}{}
```

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

```
\cs_new_protected:Nn \__notesslides_setup_itemize: {
  \def\itemize@level{outer}
  \def\itemize@outer{outer}
  \def\itemize@inner{inner}
  %\newcommand\metakeys@show@keys[2]{\marginnote{{\scriptsize ##2}}}
  \renewenvironment{itemize}{
    \ifx\itemize@level\itemize@outer
      \def\itemize@label{$\rhd$}
    \fi
    \ifx\itemize@level\itemize@inner
      \def\itemize@label{$\scriptstyle\rhd$}
    \fi
    \begin{list}
    {\itemize@label}
    {\setlength{\labelsep}{.3em}
      \setlength{\labelwidth}{.5em}
      \setlength{\leftmargin}{1.5em}
    }
    \edef\itemize@level{\itemize@inner}
  }{
    \end{list}
  }
}
```

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

```
\stex_if_html_backend:TF {
  \cs_new_protected:Nn \__notesslides_frame_box_begin: {
    \vbox\bgroup
    \begin{stex_annotate_env}{shtml:frame={}}
      \mdf@patchamsthm\notesslidesfont
  }
  \cs_new_protected:Nn \__notesslides_frame_box_end: {
    %^^A \notesslides@slidelabel
```

```
    \medskip\par\noindent\tiny\notesslidesfooter
    \end{stex_annotate_env}\egroup
  }
}{
  \cs_new_protected:Nn \__notesslides_frame_box_begin: {
    \begin{mdframed}[
      linewidth=\slideframewidth,
      skipabove=1ex,
      skipbelow=1ex,
      userdefinedwidth=\slidewidth,
      align=center
    ]\notesslidesfont
  }
  \cs_new_protected:Nn \__notesslides_frame_box_end: {
    \medskip\par\noindent\tiny\notesslidesfooter%^^A\notesslides@slidelabel
    \end{mdframed}
  }
}
```

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

```
\renewenvironment{frame}[1][]{
  \stex_keys_set:nn{notesslides / frame}{#1}
  \stepcounter{framenumber}
  \renewcommand\newpage{\addtocounter{framenumber}{1}}
  \def\@currentlabel{\theframenumber}
  \str_if_empty:NF \l__notesslides_frame_label_str {
    \label{\l__notesslides_frame_label_str}
  }
  \__notesslides_setup_itemize:
  \__notesslides_frame_box_begin:
}{
  \__notesslides_frame_box_end:
}
```

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

`\frametitle`

```
\renewcommand{\frametitle}[1]{
  \stexdoctitle { #1 }
  \notesslidestitleemph{#1}\medskip
}
```

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

`\pause`

```
\newcommand\pause{}
```

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

We redefine the `columns` and `column` environments:

```
\renewenvironment{columns}[1][]{
  \par\noindent
  \begin{minipage}
  \slidewidth\centering\leavevmode
%  \stex_if_html_backend:T{
```

```
%    \cs_if_exist:NT \rustex_if:T {
%      \rustex_if:T {\par
%        \rustex_direct_HTML:n{<table><tr><td>}
%      }
%    }
%  }
}{
%  \stex_if_html_backend:T{
%    \cs_if_exist:NT \rustex_if:T {
%      \rustex_if:T {\par
%        \rustex_direct_HTML:n{</td></tr></table>}
%      }
%    }
%  }
  \end{minipage}\par\noindent
}
\newsavebox\columnbox
\renewenvironment<>{column}[2][]{
  \begin{lrbox}{\columnbox}
%    \stex_if_html_backend:T{
%      \cs_if_exist:NT \rustex_if:T {
%        \rustex_if:T {\par
%          \rustex_direct_HTML:n{</td><td>}
%        }
%      }
%    }
    \begin{minipage}{#2}
}{
  \end{minipage}
%  \stex_if_html_backend:T{
%    \cs_if_exist:NT \rustex_if:T {
%      \rustex_if:T {\par
%        \rustex_direct_HTML:n{</td><td>}
%      }
%    }
%  }
  \end{lrbox}\usebox\columnbox
}

\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.

```
\bool_if:NTF \c_notesslides_notes_bool {
  \renewenvironment{note}{\ignorespaces}{}
}{
  \renewenvironment{note}{\setbox \l_tmpa_box\vbox\bgroup}{\egroup}
}
```

For other environments we introduce variants prefixed with `n`, which are excluded in `slides` mode.

```
\cs_new_protected:Nn \__notesslides_notes_env:nnnn {
  \bool_if:NTF \c_notesslides_notes_bool {
    \newenvironment{#1}#2{#3}{#4}
  }{
    \newenvironment{#1}#2{
      \cs_set:Npn \__notesslides_eat: ####1 \end ####2 {
        \str_if_eq:nnTF{#1}{####2}{
          \end{#1}
        }{
          \__notesslides_eat:
        }
      }
      \__notesslides_eat:
      %\setbox\l_tmpa_box\vbox\bgroup#3
    }{
      %#4\egroup
    }
  }
}

\__notesslides_notes_env:nnnn{nparagraph}{[1][]}{\begin{sparagraph}[#1]}{\end{sparagraph}}
\__notesslides_notes_env:nnnn{nfragment}{[2][]}{\begin{sfragment}[#1]{#2}}{\end{sfragment}}
\__notesslides_notes_env:nnnn{ndefinition}{[1][]}{\begin{sdefinition}[#1]}{\end{sdefinition}
\__notesslides_notes_env:nnnn{nassertion}{[1][]}{\begin{sassertion}[#1]}{\end{sassertion}}
\__notesslides_notes_env:nnnn{nproof}{[2][]}{\begin{sproof}[#1]{#2}}{\end{sproof}}
\__notesslides_notes_env:nnnn{nexample}{[1][]}{\begin{sexample}[#1]}{\end{sexample}}

\RequirePackage{graphicx}

\NewDocumentCommand\frameimage{s O{} m}{
  \IfBooleanTF #1 {
    \begin{frame}[plain]
  }{
    \begin{frame}
  }
    \bool_if:NTF \c_notesslides_notes_bool {
      \slidewidth=\dimexpr\slidewidth-(2\slideframewidth)\relax
    }{
      \slidewidth=\textwidth\relax
    }
    \def\Gin@ewidth{}\setkeys{Gin}{#2}
    \tl_if_empty:NTF \Gin@ewidth {
      \mhgraphics[width=\slidewidth,#2]{#3}
    }{
      \mhgraphics[#2]{#3}
    }
  \end{frame}
}
```

hacking inputref:

\inputref*

```
\cs_set_eq:NN\__notesslides_inputref:\inputref
\cs_set_protected:Npn\inputref{\@ifstar\ninputref\__notesslides_inputref:}
```

```
\bool_if:NTF \c_notesslides_notes_bool {
  \newcommand\ninputref[2][]{
    \__notesslides_inputref:[#1]{#2}
  }
}{
  \newcommand\ninputref[2][]{}
}
```

(*End of definition for* `\inputref*`. *This function is documented on page* 107.)

14.1.4 Styling Across Notes/Slides

```
\def\notesslidestitleemph#1{
  {\Large\bf\sf#1}
  \vskip0.1\baselineskip
  \leaders\vrule width \textwidth
  \vskip0.4pt%
  \nointerlineskip
}

\def\notesslidesfooter{}

\let\notesslidesfont\sffamily
```

14.1.5 Beamer Compatibility

All of this should be removed and made part of a template

```

\bool_if:NT \c_notesslides_notes_bool {
  \def\author{\@dblarg\ns@author}
  \long\def\ns@author[#1]#2{%
    \tl_if_empty:nTF{#1}{
      \def\beamer@shortauthor{#2}
    }{
      \def\beamer@shortauthor{#1}
    }
    \def\@author{#2}
  }
  \def\title{\@dblarg\ns@title}
  \long\def\ns@title[#1]#2{%
    \tl_if_empty:nTF{#1}{
      \def\beamer@shorttitle{#2}
    }{
      \def\beamer@shorttitle{#1}
    }
    \def\@title{#2}
    \stexdoctitle{#2}
  }
  \def\insertshortauthor{
    \hbox\bgroup\def\\{}\cs_if_exist:NT\beamer@shortauthor\beamer@shortauthor\egroup
  }
  \def\insertshorttitle{
    \hbox\bgroup\def\\{}\cs_if_exist:NT\beamer@shorttitle\beamer@shorttitle\egroup
  }
  \stex_if_html_backend:TF{
```

```
    \def\insertframenumber{\stex_annotate:nn{shtml:framenumber={}}{}}
  }{
    \def\insertframenumber{\@arabic\c@framenumber}
  }
  \def\insertshortdate{\today}
}
```

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.

```
\gdef\printexcursions{}
\newcommand\excursionref[2]{% label, text
  \bool_if:NT \c_notesslides_notes_bool {
    \begin{sparagraph}[title=Excursion]
      #2 \sref[fallback=the appendix]{#1}.
    \end{sparagraph}
  }
}
\newcommand\activate@excursion[2][]{
  \tl_gput_right:Nn\printexcursions{\inputref[#1]{#2}}
}
\newcommand\excursion[4][]{% repos, label, path, text
  \bool_if:NT \c_notesslides_notes_bool {
    \activate@excursion[#1]{#3}
    \excursionref{#2}{#4}
  }
}
```

(*End of definition for* \excursion. *This function is documented on page* 109.)

\excursiongroup

```
\keys_define:nn{notesslides / excursiongroup }{
  id        .str_set_x:N  = \l__notesslides_excursion_id_str,
  intro     .tl_set:N     = \l__notesslides_excursion_intro_tl,
  archive   .str_set_x:N  = \l__notesslides_excursion_mhrepos_str
}
\cs_new_protected:Nn \__notesslides_excursion_args:n {
  \tl_clear:N \l__notesslides_excursion_intro_tl
  \str_clear:N \l__notesslides_excursion_id_str
  \str_clear:N \l__notesslides_excursion_mhrepos_str
  \keys_set:nn {notesslides / excursiongroup }{ #1 }
}
\newcommand\excursiongroup[1][]{
  \__notesslides_excursion_args:n{ #1 }
  \tl_if_empty:NF\printexcursions
  {\IfInputref{}{\begin{note}
    \begin{sfragment}{Excursions}% TODO pass on id
      \ifdefempty\l__notesslides_excursion_intro_tl{}{
        \exp_args:NNe \use:nn \inputref{[\l__notesslides_excursion_mhrepos_str]{
          \l__notesslides_excursion_intro_tl
        }}
      }
```

```
      \printexcursions%
    \end{sfragment}
  \end{note}}}
}
\ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
```

(*End of definition for* `\excursiongroup`. *This function is documented on page* 109.)

```
\prop_new:N \g__notesslides_variables_prop
\cs_set_protected:Npn \setSGvar #1 #2 {
  \prop_gput:Nnn \g__notesslides_variables_prop {#1}{#2}
}
\cs_set_protected:Npn \useSGvar #1 {
  \prop_item:Nn \g__notesslides_variables_prop {#1}
}
\cs_set_protected:Npn \ifSGvar #1 #2 #3 {
  \prop_get:NnNF \g__notesslides_variables_prop {#1} \l__notesslides_tmp {
    \PackageError{document-structure}
    {The sTeX Global variable #1 is undefined}
    {set it with \protect\setSGvar}\TODO better error
  }
  \tl_if_eq:NnT \l__notesslides_tmp {#2}{ #3 }
}


⟨/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.

```
⟨*package⟩
⟨@@=problems⟩
\ProvidesExplPackage{problem}{2023/10/13}{3.4.0}{Semantic Markup for Problems}
\RequirePackage{l3keys2e}

\keys_define:nn { problem / pkg }{
  notes     .default:n    = { true },
  notes     .bool_set:N   = \c__problems_notes_bool,
  gnotes    .default:n    = { true },
  gnotes    .bool_set:N   = \c__problems_gnotes_bool,
  hints     .default:n    = { true },
  hints     .bool_set:N   = \c__problems_hints_bool,
  solutions .default:n    = { true },
  solutions .bool_set:N   = \c__problems_solutions_bool,
  pts       .default:n    = { true },
  pts       .bool_set:N   = \c__problems_pts_bool,
  min       .default:n    = { true },
  min       .bool_set:N   = \c__problems_min_bool,
  %boxed     .default:n    = { true },
  %boxed     .bool_set:N   = \c__problems_boxed_bool,
```

```
  test     .default:n    = { true },
  test     .bool_set:N   = \c__problems_test_bool,
  unknown     .code:n       = {
    \PassOptionsToPackage{\CurrentOption}{stex}
  }
}
\newif\ifsolutions

\ProcessKeysOptions{ problem / pkg }
\bool_if:NTF \c__problems_solutions_bool {
  \solutionstrue
}{
  \solutionsfalse
}
\newif\ifintest
\bool_if:NTF \c__problems_test_bool {
  \intesttrue
}{
  \intestfalse
}

\RequirePackage{stex}
```

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

```
\AddToHook{begindocument}{
  \ExplSyntaxOn\makeatletter
  \input{problem-english.ldf}
  \ltx@ifpackageloaded{babel}{
\clist_set:Nx \l_tmpa_clist {\exp_args:No \tl_to_str:n \bbl@loaded}
      \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
        \input{problem-ngerman.ldf}
      }
      \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
        \input{problem-finnish.ldf}
      }
      \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
        \input{problem-french.ldf}
      }
      \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
        \input{problem-russian.ldf}
      }
  }{}
  \makeatother\ExplSyntaxOff
}
```

(*End of 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.

```
\bool_new:N \l_stex_key_autogradable_bool
```

```
\stex_keys_define:nnnn{ problem }{
  \tl_set:Nn \l_stex_key_pts_tl 0
  \tl_set:Nn \l_stex_key_min_tl 0
  \str_clear:N \l_stex_key_name_str
  \str_clear:N \l_stex_key_mhrepos_str
  \bool_set_false:N \l_stex_key_autogradable_bool
}{
  pts     .tl_set:N     = \l_stex_key_pts_tl,
  min     .tl_set:N     = \l_stex_key_min_tl,
  name    .str_set:N    = \l_stex_key_name_str,
  autogradable .bool_set:N = \l_stex_key_autogradable_bool,
  archive  .code:n = {},
  %archive .str_set:N    = \l_stex_key_mhrepos_str,
  creators .code:n = {}
  %imports .tl_set:N     = \l__problems_prob_imports_tl,
  %refnum  .int_set:N    = \l__problems_prob_refnum_int,
}{id,title,style,uses}
```

Then we set up a counter for problems.

\numberproblemsin

```
\newcounter{sproblem}[section]
\newcommand\numberproblemsin[1]{
  \@addtoreset{sproblem}{#1}
  \def\thesproblem{\arabic{#1}.\arabic{sproblem}}
}
\numberproblemsin{section}
%\def\theplainsproblem{\arabic{sproblem}}
%\def\thesproblem{\thesection.\theplainsproblem}
```

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

sproblem (*env.*)

```
\cs_new:Nn \__problems_activate_macros: {
  \stex_reactivate_macro:N \solution
  \stex_reactivate_macro:N \mcb
  \stex_reactivate_macro:N \scb
  \stex_reactivate_macro:N \fillinsol
  \stex_reactivate_macro:N \hint
  \stex_reactivate_macro:N \exnote
  \stex_reactivate_macro:N \gnote
}

\newcounter{pts}
\newcounter{min}
\bool_new:N \l__problems_in_problem_bool
\bool_new:N \l__problems_has_pts_bool
\bool_new:N \l__problems_has_min_bool
\bool_set_false:N \l__problems_in_problem_bool
\stex_new_stylable_env:nnnnnnn {problem} {O{}}{
  \bool_if:NT \l__problems_in_problem_bool {
    \msg_error:nn{stex}{error/nestedproblem}
  }
  \cs_if_exist:NTF \l_problem_inputproblem_keys_tl {
    \tl_put_left:Nn \l_problem_inputproblem_keys_tl {#1,}
```

```
    \exp_args:Nno \stex_keys_set:nn{problem}{
      \l_problem_inputproblem_keys_tl
    }
  }{
    \stex_keys_set:nn{problem}{#1}
  }
  \refstepcounter{sproblem}

  \stex_if_do_html:T {
    \str_if_empty:NT \l_stex_key_name_str {
      \stex_file_split_off_lang:NN \l__problems_path_seq \g_stex_current_file
      \seq_get_right:NN \l__problems_path_seq \l_stex_key_name_str
    }
    \exp_args:Nne \begin{stex_annotate_env} {
      shtml:problem={\l_stex_key_name_str},
      shtml:language={ \l_stex_current_language_str},
      shtml:autogradable={\bool_if:NTF \l_stex_key_autogradable_bool {true}{false}}
    }
    \tl_if_empty:NF \l_stex_key_title_tl {
      \exp_args:No \stexdoctitle \l_stex_key_title_tl
    }
    \stex_annotate_invisible:nn{
      shtml:problempoints={\l_stex_key_pts_tl}
    }{ \l_stex_key_pts_tl }
  }

  \tl_set_eq:NN \thistitle \l_stex_key_title_tl

  \bool_set_true:N \l__problems_in_problem_bool
  \tl_set_eq:NN \l__problems_pts_tl \l_stex_key_pts_tl
  \tl_set_eq:NN \l__problems_min_tl \l_stex_key_min_tl
  \tl_if_eq:NnTF \l__problems_pts_tl {0}
    {\bool_set_false:N \l__problems_has_pts_bool}
    {\bool_set_true:N \l__problems_has_pts_bool}
  \tl_if_eq:NnTF \l__problems_min_tl {0}
    {\bool_set_false:N \l__problems_has_min_bool}
    {\bool_set_true:N \l__problems_has_min_bool}
  \int_gzero:N \g__problems_subproblem_int

  \stex_style_apply:
  \_stex_do_id:
  \__problems_activate_macros:
}{
  \addtocounter{pts}{\l__problems_pts_tl}
  \addtocounter{min}{\l__problems_min_tl}
  \__problems_record_problem:
  \stex_style_apply:
  \stex_if_do_html:T{ \end{stex_annotate_env} }
}{
  \par\noindent\problemheader
  \bool_if:NT \c__problems_pts_bool {
    \tl_if_eq:NnF \l__problems_pts_tl {0}{
      \marginpar{\l__problems_pts_tl{}~\problem@kw@pts\smallskip}
    }
```

```
  }
  \bool_if:NT \c__problems_min_bool {
    \tl_if_eq:NnF \l__problems_min_tl {0} {
      \marginpar{\l__problems_min_tl{}~\problem@kw@minutes\smallskip}
    }
  }
  \par
  \stex_ignore_spaces_and_pars:
}{
  \par\bigskip
%  \bool_if:NT \c__problems_test_bool \pagebreak
}{s}

\tl_set:Nn \problemheader {
  \textbf{\sproblemautorefname{~}\thesproblem
    \tl_if_empty:NF \thistitle {
      {~}(\thistitle)
    }
  }
}

\cs_new_protected:Nn \__problems_record_problem: {
  \exp_args:NNe \iow_now:Nn \@auxout {
    \problem@restore {\thesproblem}{\l__problems_pts_tl}{\l__problems_min_tl}
  }
}

\cs_new_protected:Npn \problem@restore #1 #2 #3 {}
```

subproblem (*env.*)

```
\int_new:N \g__problems_subproblem_int

\stex_new_stylable_env:nnnnnnn {subproblem} {O{}}{
  \stex_keys_set:nn{problem}{#1}
  \bool_if:NF \l__problems_in_problem_bool{
    \ifstexhtml\else
      \par{\bfseries WARNING~subproblem~to~be~used~in~some~problem}\par
    \fi
    \__problems_activate_macros:
    \bool_set_true:N \l__problems_in_problem_bool
    \tl_set:Nn \l__problems_pts_tl{0}
    \tl_set:Nn \l__problems_min_tl{0}
  }
  \str_if_empty:NT \l_stex_key_name_str {
    \stex_file_split_off_lang:NN \l__problems_path_seq \g_stex_current_file
    \seq_get_right:NN \l__problems_path_seq \l_stex_key_name_str
  }
  \stex_if_do_html:T {
    \str_if_empty:NT \l_stex_key_name_str {
      \stex_file_split_off_lang:NN \l__problems_path_seq \g_stex_current_file
      \seq_get_right:NN \l__problems_path_seq \l_stex_key_name_str
    }
    \exp_args:Nne \begin{stex_annotate_env} {
      shtml:subproblem={\l_stex_key_name_str},
```

```
      shtml:language={ \l_stex_current_language_str},
      shtml:autogradable={\bool_if:NTF \l_stex_key_autogradable_bool {true}{false}}
    }
    \stex_annotate_invisible:n{}
    \tl_if_empty:NF \l_stex_key_title_tl {
      \exp_args:No \stexdoctitle \l_stex_key_title_tl
    }
  }
  \int_gincr:N \g__problems_subproblem_int
  \bool_if:NF \l__problems_has_pts_bool {
    \tl_gset:Nx \l__problems_pts_tl {\int_eval:n {\l__problems_pts_tl + \l_stex_key_pts_tl}}
  }
  \bool_if:NF \l__problems_has_min_bool {
    \tl_gset:Nx \l__problems_min_tl {\int_eval:n {\l__problems_min_tl + \l_stex_key_min_tl}}
  }
  \stex_if_smsmode:F \stex_style_apply:
}{
  \stex_if_smsmode:F \stex_style_apply:
  \stex_if_do_html:T{ \end{stex_annotate_env} }
}{
  \begin{list}{}{
    \setlength\topsep{0pt}
    \setlength\parsep{0pt}
    \setlength\rightmargin{0pt}
  }\item[\int_use:N \g__problems_subproblem_int .]
  \bool_if:NT \c__problems_pts_bool {
    \bool_if:NF \l__problems_has_pts_bool {
      \marginpar{\smallskip\l_stex_key_pts_tl{}~\problem@kw@pts}
    }
  }
  \bool_if:NT \c__problems_min_bool {
    \bool_if:NF \l__problems_has_min_bool{
      \marginpar{\smallskip\l_stex_key_min_tl{}~\problem@kw@minutes}
    }
  }
}{
  \end{list}
}{}
```

\includeproblem

```
\stex_keys_define:nnnn{ includeproblem }{
  \str_clear:N \l_stex_key_mhrepos_str
}{
  archive .str_set:N    = \l_stex_key_mhrepos_str,
  unknown .code:n = {}
}{}

\NewDocumentCommand\includeproblem{O{} m}{
  \group_begin:
  \tl_set:Nn \l_problem_inputproblem_keys_tl {#1}
  \stex_keys_set:nn{includeproblem}{#1}
  \exp_args:Nno \use:nn{\inputref[}\l_stex_key_mhrepos_str]{#2}
  \group_end:
}
```

```
8543
```

(*End of definition for* `\includeproblem`. *This function is documented on page 114.*)

`solution` (*env.*)

```
8544 \int_new:N \g_problem_id_counter
8545 \dim_new:N \l_stex_key_testspace_dim
8546 \stex_keys_define:nnnn{ solution }{
8547   \str_clear:N \l_stex_key_answerclass_str
8548   \dim_zero:N \l_stex_key_testspace_dim
8549 }{
8550   testspace   .dim_set:N = \l_stex_key_testspace_dim,
8551   answerclass .str_set:N = \l_stex_key_answerclass_str
8552 }{id,title,style}
8553
8554 \cs_new_protected:Nn \__problems_solution_start:n {
8555   \stex_keys_set:nn{ solution }{#1}
8556   \str_if_empty:NT \l_stex_key_id_str {
8557     \int_gincr:N \g_problem_id_counter
8558     \str_set:Nx \l_stex_key_id_str {
8559       SOLUTION_\int_use:N \g_problem_id_counter
8560     }
8561   }
8562   \stex_if_do_html:T{
8563     \begin{stex_annotate_env}{
8564       shtml:solution=\l_stex_key_id_str,
8565       shtml:answerclass={\l_stex_key_answerclass_str}
8566     }
8567   }
8568   \stex_style_apply:
8569 }
8570
8571 \stex_new_stylable_env:nnnnnnn { solution }{ O{} }{
8572   \stex_if_do_html:TF{
8573     \__problems_solution_start:n{#1}
8574   }{
8575     \ifsolutions
8576       \__problems_solution_start:n{#1}
8577     \else
8578       \stex_keys_set:nn{ solution }{#1}
8579       \testspace{\l_stex_key_testspace_dim}
8580       \setbox\l_tmpa_box\vbox\bgroup
8581     \fi
8582   }
8583 }{
8584   \stex_if_do_html:TF{
8585     \stex_style_apply:
8586     \end{stex_annotate_env}
8587   }{
8588     \ifsolutions
8589       \stex_style_apply:
8590       \stex_if_do_html:T{
8591         \end{stex_annotate_env}
8592       }
```

```
    \else
      \egroup
    \fi
  }
}{
  \par\smallskip\rule[.3em]{\linewidth}{0.4pt}\newline\smallskip
  \noindent\emph{\problem@kw@solution\tl_if_empty:NF \l_stex_key_title_tl{
    {~}\l_stex_key_title_tl \str_if_empty:NF \l_stex_key_answerclass_str {
      (Answer Class: \l_stex_key_answerclass_str)
    }
  } :~}
}{
  \par\rule[.3em]{\linewidth}{0.4pt}\newline
}{}

\stex_deactivate_macro:Nn \solution {sproblem~environments}
```

\startsolutions
\stopsolutions

```
\cs_new_protected:Npn \startsolutions{
  \global\solutionstrue
}
\cs_new_protected:Npn \stopsolutions{
  \global\solutionsfalse
}
```

(*End of definition for* `\startsolutions` *and* `\stopsolutions`. *These functions are documented on page* 111.)

hint (*env.*)

```

\stex_keys_define:nnnn{ problemenv }{}{}{id,title,style}

\cs_new_protected:Nn \__problems_hint_start:n {
  \stex_keys_set:nn{ problemenv }{#1}
  \str_if_empty:NT \l_stex_key_id_str {
    \int_gincr:N \g_problem_id_counter
    \str_set:Nx \l_stex_key_id_str {
      HINT_\int_use:N \g_problem_id_counter
    }
  }
  \stex_if_do_html:T{
    \begin{stex_annotate_env}{
      shtml:problemhint=\l_stex_key_id_str
    }
  }
  \stex_style_apply:
}

\stex_new_stylable_env:nnnnnnn { hint }{ O{} }{
  \stex_if_do_html:TF{
    \__problems_hint_start:n{#1}
  }{
    \bool_if:NTF \c__problems_hints_bool {
      \__problems_hint_start:n{#1}
```

```
    }{
      \setbox\l_tmpa_box\vbox\bgroup
    }
  }
}{
  \stex_if_do_html:TF{
    \stex_style_apply:
    \end{stex_annotate_env}
  }{
    \bool_if:NTF \c__problems_hints_bool {
      \stex_style_apply:
      \stex_if_do_html:T{
        \end{stex_annotate_env}
      }
    }{
      \egroup
    }
  }
}{
  \par\smallskip\rule[.3em]{\linewidth}{0.4pt}\newline\smallskip
  \noindent\emph{\problem@kw@hint\tl_if_empty:NF \l_stex_key_title_tl{
    {~}\l_stex_key_title_tl
  } :~}
}{
  \par\rule[.3em]{\linewidth}{0.4pt}\newline
}{}
\stex_deactivate_macro:Nn \hint {sproblem~environments}
```

exnote (*env.*)

```
\cs_new_protected:Nn \__problems_exnote_start:n {
  \stex_keys_set:nn{ problemenv }{#1}
  \str_if_empty:NT \l_stex_key_id_str {
    \int_gincr:N \g_problem_id_counter
    \str_set:Nx \l_stex_key_id_str {
      EXNOTE_\int_use:N \g_problem_id_counter
    }
  }
  \stex_if_do_html:T{
    \begin{stex_annotate_env}{
      shtml:problemnote=\l_stex_key_id_str
    }
  }
  \stex_style_apply:
}

\stex_new_stylable_env:nnnnnnn { exnote }{ O{} }{
  \stex_if_do_html:TF{
    \__problems_exnote_start:n{#1}
  }{
    \bool_if:NTF \c__problems_notes_bool {
      \__problems_exnote_start:n{#1}
    }{
      \setbox\l_tmpa_box\vbox\bgroup
    }
```

```
  }
}{
  \stex_if_do_html:TF{
    \stex_style_apply:
    \end{stex_annotate_env}
  }{
    \bool_if:NTF \c__problems_notes_bool {
      \stex_style_apply:
      \stex_if_do_html:T{
        \end{stex_annotate_env}
      }
    }{
      \egroup
    }
  }
}{
  \par\smallskip\rule[.3em]{\linewidth}{0.4pt}\newline\smallskip
  \noindent\emph{\problem@kw@note\tl_if_empty:NF \l_stex_key_title_tl{
    {~}\l_stex_key_title_tl
  } :~}
}{
  \par\rule[.3em]{\linewidth}{0.4pt}\newline
}{}
\stex_deactivate_macro:Nn \exnote {sproblem~environments}
```

gnote (*env.*)

```
\int_new:N \l__problems_anscls_int

\cs_new_protected:Nn \__problems_gnote_start:n {
  \stex_keys_set:nn{ problemenv }{#1}
  \str_if_empty:NT \l_stex_key_id_str {
    \int_gincr:N \g_problem_id_counter
    \str_set:Nx \l_stex_key_id_str {
      GNOTE_\int_use:N \g_problem_id_counter
    }
  }

  \stex_if_do_html:T{
    \begin{stex_annotate_env}{
      shtml:problemgnote=\l_stex_key_id_str
    }
  }
  \stex_style_apply:
}

\stex_new_stylable_env:nnnnnnn { gnote }{ O{} }{
  \stex_if_do_html:TF{
    \__problems_gnote_start:n{#1}
  }{
    \bool_if:NTF \c__problems_gnotes_bool {
      \__problems_gnote_start:n{#1}
    }{
      \setbox\l_tmpa_box\vbox\bgroup
    }
```

```
  }
  \stex_reactivate_macro:N \anscls
}{
  \stex_if_do_html:TF{
    \stex_style_apply:
    \end{stex_annotate_env}
  }{
    \bool_if:NTF \c__problems_gnotes_bool {
      \stex_style_apply:
      \stex_if_do_html:T{
        \end{stex_annotate_env}
      }
    }{
      \egroup
    }
  }
}{
  \par\smallskip\rule[.3em]{\linewidth}{0.4pt}\newline\smallskip
  \noindent\emph{\problem@kw@grading\str_if_empty:NF \l_stex_key_title_tl{
    {~}\l_stex_key_title_tl
  } :~}
}{
  \par\rule[.3em]{\linewidth}{0.4pt}\newline
}{}
\stex_deactivate_macro:Nn \gnote {sproblem~environments}


\stex_keys_define:nnnn{ anscls }{
  \str_clear:N \l_stex_key_pts_str
  \tl_clear:N \l_stex_key_feedback_tl
}{
  pts     .str_set:N = \l_stex_key_pts_str,
  feedback .tl_set:N = \l_stex_key_feedback_str,
  update  .code:n    = {}
}{id}
\newcommand \anscls [2][] {
  \stex_keys_set:nn{ anscls }{#1}
  \str_if_empty:NT \l_stex_key_id_str {
    \int_incr:N \l__problems_anscls_int
    \str_set:Nx \l_stex_key_id_str {
      AC\int_use:N \l__problems_anscls_int
    }
  }
  \begin{list}{}{
    \setlength\topsep{0pt}
    \setlength\parsep{0pt}
    \setlength\rightmargin{0pt}
  }\item[\l_stex_key_id_str]
    \stex_if_do_html:TF{
      \stex_annotate:nn{
        shtml:answerclass={\l_stex_key_id_str}
        \str_if_empty:NF \l_stex_key_pts_str{
          ,shtml:answerclass-pts={\l_stex_key_pts_str}
        }
```

```
        \str_if_empty:NF \l_stex_key_feedback_str{
          ,shtml:answerclass-feedback={\l_stex_key_feedback_str}
        }
      }{
        #2
      }
    }{#2}
    \str_if_empty:NF \l_stex_key_pts_str {\par
      ~ \problem@kw@points:~\l_stex_key_pts_str
    }
    \str_if_empty:NF \l_stex_key_feedback_str {\par
      ~ \problem@kw@feedback :~\l_stex_key_feedback_str
    }
  \end{list}
}
\stex_deactivate_macro:Nn \anscls {gnote~environments}
```

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

```
\def\pts#1{
  \bool_if:NT \c__problems_pts_bool {
    \stex_annotate:nn{shtml:problempoints={#1}}{\marginpar{#1~\problem@kw@pts}}
  }\hbox_unpack:N\c_empty_box
}
\def\min#1{
  \bool_if:NT \c__problems_min_bool {
    \stex_annotate:nn{shtml:problemminutes={}}{\marginpar{#1~\problem@kw@minutes}}
  }\hbox_unpack:N\c_empty_box
}
```

mcb (*env.*)

```
\stex_new_stylable_env:nnnnnnn{mcb}{O{}}{
  \stex_keys_set:nn{style}{#1}
  \stex_if_do_html:T{
    \tl_set:Nn\problem_mcc_box_tl{}
    \exp_args:Nne \begin{stex_annotate_env}{
      shtml:multiple-choice-block={}
      \clist_if_empty:NF \l_stex_key_style_clist {,
        shtml:styles={\l_stex_key_style_clist}
      }
    }
  }
  \stex_deactivate_macro:Nn \mcb {sproblem~environments}
  \stex_deactivate_macro:Nn \scb {sproblem~environments}
  \stex_deactivate_macro:Nn \solution {sproblem~environments}
  \stex_deactivate_macro:Nn \hint {sproblem~environments}
  \stex_deactivate_macro:Nn \exnote {sproblem~environments}
  \stex_deactivate_macro:Nn \gnote {sproblem~environments}
  \stex_reactivate_macro:N \mcc
  \cs_set:Nn \__problems_mccline:n{
    \begin{list}{}{
      \setlength\topsep{0pt}
      \setlength\parsep{0pt}
      \setlength\rightmargin{0pt}
    }\item[\problem_mcc_box_tl] ##1 \end{list}
```

```
  }
  \stex_style_apply:
}{
  \stex_style_apply:
  \stex_if_do_html:T{
    \end{stex_annotate_env}
  }
}{\par}{}{}
\stexstylemcb[inline]{
  \cs_set:Nn \__problems_mccline:n{\problem_mcc_box_tl{~} #1}
}{}

\stex_deactivate_macro:Nn \mcb {sproblem~environments}
```

we define the keys for the mcc macro

```
\cs_new_protected:Nn \__problems_do_yes_param:Nn {
  \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
    \bool_set_true:N #1
  }{
    \bool_set_false:N #1
  }
}
\stex_keys_define:nnnn{mcc}{
  \tl_clear:N \l_stex_key_feedback_tl
  \bool_set_false:N \l_stex_key_T_bool
  \tl_clear:N \l_stex_key_Ttext_tl
  \tl_clear:N \l_stex_key_Ftext_tl
}{
  feedback  .tl_set:N     = \l_stex_key_feedback_tl ,
  T         .code:n     = {\bool_set_true:N \l_stex_key_T_bool} ,
  F         .code:n       = {\bool_set_false:N \l_stex_key_T_bool} ,
  Ttext     .tl_set:N     = \l_stex_key_Ttext_tl ,
  Ftext     .tl_set:N     = \l_stex_key_Ftext_tl ,
}{id}

```

\mcc

```
\tl_set:Nn \problem_mcc_box_tl {
  \ltx@ifpackageloaded{amssymb}{$\square$}{
    \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
  }
}
\newcommand\mcc[2][]{
  \stex_keys_set:nn{mcc}{#1}
  \tl_set:Nn \l_tmpa_tl{
    \bool_if:NTF \l_stex_key_T_bool {
      \tl_if_empty:NTF \l_stex_key_Ttext_tl \problem@kw@correct \l_stex_key_Ttext_tl
    }{
      \tl_if_empty:NTF \l_stex_key_Ftext_tl \problem@kw@wrong \l_stex_key_Ftext_tl
    }
    \tl_if_empty:NF \l_stex_key_feedback_tl {
      \\\emph{\l_stex_key_feedback_tl}
    }
  }
```

```
  \__problems_mccline:n{
    \stex_if_do_html:TF{
      \stex_annotate:nn{shtml:mcc={
        \bool_if:NTF \l_stex_key_T_bool {true}{false}
      }}{
        #2\stex_annotate:nn{shtml:mcc-solution={}}{\l_tmpa_tl}
      }
    }{
        #2\ifsolutions\footnote{\l_tmpa_tl}\fi
    }
  }
}
\stex_deactivate_macro:Nn \mcc {mcb~environments}
```

(*End of definition for* \mcc. *This function is documented on page 112.*)

scb (*env.*)

```
\tl_set:Nn\problem_scc_box_tl{$\bigcirc$}

\stex_new_stylable_env:nnnnnnn{scb}{O{}}{
  \stex_keys_set:nn{style}{#1}
  \stex_if_do_html:T{
    \exp_args:Nne\begin{stex_annotate_env}{
      shtml:single-choice-block={}
      \clist_if_empty:NF \l_stex_key_style_clist {,
        shtml:styles={\l_stex_key_style_clist}
      }
    }
    \tl_set:Nn\problem_scc_box_tl{}
  }
  \stex_deactivate_macro:Nn \mcb {sproblem~environments}
  \stex_deactivate_macro:Nn \scb {sproblem~environments}
  \stex_deactivate_macro:Nn \solution {sproblem~environments}
  \stex_deactivate_macro:Nn \hint {sproblem~environments}
  \stex_deactivate_macro:Nn \exnote {sproblem~environments}
  \stex_deactivate_macro:Nn \gnote {sproblem~environments}
  \stex_reactivate_macro:N \scc
  \cs_set:Nn \__problems_sccline:n{
    \begin{list}{}{
      \setlength\topsep{0pt}
      \setlength\parsep{0pt}
      \setlength\rightmargin{0pt}
    }\item[\problem_scc_box_tl] ##1 \end{list}
  }
  \stex_style_apply:
}{
  \stex_style_apply:
  \stex_if_do_html:T{
    \end{stex_annotate_env}
  }
}{\par}{}{}
\stexstylescb[inline]{
```

```
  \cs_set:Nn \__problems_sccline:n{\problem_scc_box_tl{~} #1}
}{}

\stex_deactivate_macro:Nn \scb {sproblem~environments}
```

\scc

```

\newcommand\scc[2][]{
  \stex_keys_set:nn{mcc}{#1}
  \tl_set:Nn \l_tmpa_tl{
    \bool_if:NTF \l_stex_key_T_bool {
      \tl_if_empty:NTF \l_stex_key_Ttext_tl \problem@kw@correct \l_stex_key_Ttext_tl
    }{
      \tl_if_empty:NTF \l_stex_key_Ftext_tl \problem@kw@wrong \l_stex_key_Ftext_tl
    }
    \tl_if_empty:NF \l_stex_key_feedback_tl {
      \\\emph{\l_stex_key_feedback_tl}
    }
  }

  \__problems_sccline:n{
    \stex_if_do_html:TF{
      \stex_annotate:nn{shtml:scc={
        \bool_if:NTF \l_stex_key_T_bool {true}{false}
      }}{
        #2\stex_annotate:nn{shtml:scc-solution={}}{\l_tmpa_tl}
      }
    }{
        #2\ifsolutions\footnote{\l_tmpa_tl}\fi
    }
  }
}
\stex_deactivate_macro:Nn \scc {scb~environments}


\newcommand\yesTnoF{
  \begin{scb}[style=inline]
    \scc[T]{yes}~\scc[F]{no}
  \end{scb}
}
\newcommand\yesFnoT{
  \begin{scb}[style=inline]
    \scc[F]{yes}~\scc[T]{no}
  \end{scb}
}
\newcommand\trueTfalseF{
  \begin{scb}[style=inline]
    \scc[T]{true}~\scc[F]{false}
  \end{scb}
}
\newcommand\trueFfalseT{
  \begin{scb}[style=inline]
    \scc[F]{true}~\scc[T]{false}
  \end{scb}
```

```
}
```

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

\fillinsol

```
\stex_keys_define:nnnn{fillinsol}{
  \tl_clear:N \l__problems_fillin_solution_tl
  \dim_zero:N \l_stex_key_testspace_dim
}{
  testspace   .dim_set:N = \l_stex_key_testspace_dim,
  exact    .code:n = {\__problems_parse_fillin_arg:nnnn{exact}#1 },
  numrange .code:n = {\__problems_parse_fillin_arg:nnnn{numrange}#1 },
  regex    .code:n = {\__problems_parse_fillin_arg:nnnn{regex}#1 }
}{}

\cs_new:Nn \__problems_parse_fillin_arg:nnnn {
  \stex_if_do_html:TF{
    \tl_set:Nn \l_tmpa_tl{#3}
    \tl_set:Nn \l_tmpb_tl{T}
    \stex_annotate_invisible:nn{
      shtml:fillin-case={#1},
      shtml:fillin-case-value={#2},
      shtml:fillin-case-verdict={
        \tl_if_eq:NNTF\l_tmpa_tl\l_tmpb_tl{true}{false}
      },
    }{#4}
  }{
    \tl_put_right:Nn \l__problems_fillin_solution_tl {
      #1{~} & #2{~} & #3{~} & #4{~} \\
    }
  }
}

\newcommand\fillinsol[2][]{
  \quad
  \mode_if_math:TF{
    \hbox{\__problems_fillinsol:nn{#1}{$#2$}}
  }{
    \__problems_fillinsol:nn{#1}{#2}
  }
  \quad
}
\cs_new_protected:Nn \__problems_fillinsol:nn {
  \stex_keys_set:nn{fillinsol}{#1}
  \stex_if_do_html:TF{
    \stex_annotate:nn{shtml:fillinsol={}}{ \_stex_annotate_force_break:n{
      #2
      \l__problems_fillin_solution_tl
    } }
  }{
    \ifsolutions
      \textcolor{red}{\fbox{#2}}
      \tl_if_empty:NF \l__problems_fillin_solution_tl {
        \footnote{
```

```
            \halign{ ##\hfil & ##\hfil &##\hfil&##\hfil \cr
              \textbf{type }&\textbf{case }&\textbf{verdict }&\textbf{feedback }\cr
              \l__problems_fillin_solution_tl
            }
          }
        }
      \else
        \fbox{\dim_compare:nNnTF\l_stex_key_testspace_dim={0pt}{
          \phantom{\huge{#2}}
        }{
          \hspace{\l_stex_key_testspace_dim}
        }}
      \fi
    }
}
\stex_deactivate_macro:Nn \fillinsol {sproblem~environments}
```

(*End of definition for* `\fillinsol`. *This function is documented on page 114.*)

`\testemptypage`

```
\newcommand\testemptypage[1][]{%
\bool_if:NT \c__problems_test_bool {\ \vfill\begin{center}\hwexam@kw@testemptypage\end{cente
}
```

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

`\testspace`

```
\newcommand\testspace[1]{\bool_if:NT \c__problems_test_bool {\vspace*{#1}}}
\newcommand\testsmallspace{\testspace{1cm}}
\newcommand\testmedspace{\testspace{2cm}}
\newcommand\testbigspace{\testspace{3cm}}
```

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

`\testnewpage`

```
\newcommand\testnewpage{\bool_if:NT \c__problems_test_bool {\newpage}}
```

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

```
⟨/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.

```
⟨*package⟩
\ProvidesExplPackage{hwexam}{2023/10/13}{3.4.0}{homework assignments and exams}
\RequirePackage{l3keys2e}

\keys_define:nn {hwexam / pkg}{
multiple  .default:n  = { false },
```

```
multiple .bool_set:N  = \c_hwexam_multiple_bool,
unknown   .code:n  = {
\PassOptionsToPackage{\CurrentOption}{problem}
}
}
\ProcessKeysOptions{ hwexam /pkg }
\RequirePackage{problem}
```

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

```
\AddToHook{begindocument}{
\ExplSyntaxOn\makeatletter
\input{hwexam-english.ldf}
\ltx@ifpackageloaded{babel}{
\clist_set:Nx \l_tmpa_clist {\exp_args:No \tl_to_str:n \bbl@loaded}
\exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
\input{hwexam-ngerman.ldf}
}
\exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
\input{hwexam-finnish.ldf}
}
\exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
\input{hwexam-french.ldf}
}
\exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
\input{hwexam-russian.ldf}
}
}{}
\makeatother\ExplSyntaxOff
}
```

(*End of 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.*)

```
\stex_keys_define:nnnn{ assignment }{
  \tl_clear:N \l_stex_key_number_tl
\tl_clear:N \l_stex_key_given_tl
\tl_clear:N \l_stex_key_due_tl
}{
  number  .tl_set:N     = \l_stex_key_number_tl,
  given   .tl_set:N     = \l_stex_key_given_tl,
  due     .tl_set:N     = \l_stex_key_due_tl,
unknown .code:n = {}
}{id,title,style}

\newcounter{assignment}
\stex_new_stylable_env:nnnnnnn {assignment}{O{}}{
  \cs_if_exist:NTF \l_hwexam_includeassignment_keys_tl {
```

```
    \tl_put_left:Nn \l_hwexam_includeassignment_keys_tl {#1,}
    \exp_args:Nno \stex_keys_set:nn{assignment}{
      \l_hwexam_includeassignment_keys_tl
    }
  }{
    \stex_keys_set:nn{assignment}{#1}
  }
\tl_if_empty:NF \l_stex_key_number_tl {
\global\setcounter{assignment}{\int_eval:n{\l_stex_key_number_tl-1}}
}
\global\refstepcounter{assignment}
\setcounter{sproblem}{0}
\def\thesproblem{\theassignment.\arabic{sproblem}}
\stex_style_apply:
\_stex_do_id:
}{
\stex_style_apply:
}{
\par\begin{center}
\textbf{\Large\assignmentautorefname~\theassignment
\tl_if_empty:NF \l_stex_key_title_tl {
{~}---~\l_stex_key_title_tl
}
}\par\smallskip
\textbf{
\tl_if_empty:NF \l_stex_key_given_tl {
\hwexam@kw@given :~\l_stex_key_given_tl\quad
}
\tl_if_empty:NF \l_stex_key_due_tl {
\hwexam@kw@due :~\l_stex_key_due_tl\quad
}
}
\end{center}
\par\bigskip
}{
\par\pagebreak
}{}
```

\includeassignment

```
\NewDocumentCommand\includeassignment{O{} m}{
\group_begin:
\tl_set:Nn \l_hwexam_includeassignment_keys_tl {#1}
\stex_keys_set:nn{includeproblem}{#1}
\exp_args:Nno \use:nn{\inputref[}\l_stex_key_mhrepos_str]{#2}
\group_end:
}
```

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

Restoring information about problems:

```
\prop_new:N \c_@@_problems_prop
\tl_set:Nn \c_@@_total_mins_tl {0}
\tl_set:Nn \c_@@_total_pts_tl {0}
\int_new:N \c_@@_total_problems_int
\cs_set_protected:Npn \problem@restore #1 #2 #3 {
```

```
\int_gincr:N \c_@@_total_problems_int
\prop_gput:Nnn \c_@@_problems_prop {#1}{{#2}{#3}}
\tl_gset:Nx \c_@@_total_pts_tl { \int_eval:n { \c_@@_total_pts_tl + #2 }}
\tl_gset:Nx \c_@@_total_mins_tl { \int_eval:n { \c_@@_total_mins_tl + #2 }}
}
```

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

```
\newcommand\correction@table{
\int_compare:nNnT \c_@@_total_problems_int = 0 {
\int_incr:N \c_@@_total_problems_int
\prop_put:Nnn \c_@@_problems_prop {~}{{~}{~}}
}
\tl_clear:N \l_tmpa_tl
\tl_clear:N \l_tmpb_tl
\tl_clear:N \l_tmpc_tl
\prop_map_inline:Nn \c_@@_problems_prop {
\tl_put_right:Nn \l_tmpa_tl { ##1 & }
\tl_put_right:Nx \l_tmpb_tl { \use_i:nn ##2 & }
\tl_put_right:Nn \l_tmpc_tl { & }
}
\resizebox{\textwidth}{!}{%
\exp_args:Nne \begin{tabular}{|l|*{\int_use:N \c_@@_total_problems_int}{c|}c||l|}\hline
&\exp_args:Ne \multicolumn{\int_eval:n{ \c_@@_total_problems_int + 1}}{c||}
{\footnotesize\hwexam@kw@forgrading} &\\\hline
\hwexam@kw@probs & \l_tmpa_tl \hwexam@kw@sum & \hwexam@kw@grade\\\hline
\hwexam@kw@pts & \l_tmpb_tl \c_@@_total_pts_tl & \\\hline
\hwexam@kw@reached & \l_tmpc_tl & \\[.7cm]\hline
\end{tabular}}}
```

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

`\testheading`

```
\def\hwexamheader{\input{hwexam-default.header}}

\def\hwexamminutes{
\tl_if_empty:NTF \hwexam@duration {
{\hwexam@min}~\hwexam@minutes@kw
}{
\hwexam@duration
}
}

\stex_keys_define:nnnn{ hwexam / testheading }{
\tl_clear:N \hwexam@min
\tl_clear:N \hwexam@duration
\tl_clear:N \hwexam@reqpts
\tl_clear:N \hwexam@tools
}{
min  .tl_set:N  = \hwexam@min,
duration .tl_set:N  = \hwexam@duration,
reqpts .tl_set:N  = \hwexam@reqpts,
tools .tl_set:N  = \hwexam@tools
}{}

```

```
\newenvironment{testheading}[1][]{
\stex_keys_set:nn { hwexam / testheading}{#1}

\tl_set_eq:NN \hwexam@totalpts \c_@@_total_pts_tl
\tl_set_eq:NN \hwexam@totalmin \c_@@_total_mins_tl
\tl_set:Nx \hwexam@checktime {\int_eval:n { \hwexam@min - \hwexam@totalmin }}

\newif\if@bonuspoints
\tl_if_empty:NTF \hwexam@reqpts {
\@bonuspointsfalse
}{
\tl_set:Nx \hwexam@bonuspts {
\int_eval:n{\hwexam@totalpts - \hwexam@reqpts}
}
\@bonuspointstrue
}

\makeatletter\hwexamheader\makeatother
}{
\newpage
}
```

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

⟨/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

⟨@@=tikzinput⟩

⟨*package⟩

```
%%%%%%%%%%%%   tikzinput.dtx   %%%%%%%%%%%%
```

```
\ProvidesExplPackage{tikzinput}{2023/10/13}{3.4.0}{tikzinput package}
\RequirePackage{l3keys2e}

\keys_define:nn { tikzinput } {
  image    .bool_set:N   = \c_tikzinput_image_bool,
  image    .default:n    = false ,
  unknown   .code:n       = {}
}

\ProcessKeysOptions { tikzinput }

\bool_if:NTF \c_tikzinput_image_bool {
  \RequirePackage{graphicx}

  \providecommand\usetikzlibrary[]{}
  \newcommand\tikzinput[2][]{\includegraphics[#1]{#2}}
}{
  \RequirePackage{tikz}
  \RequirePackage{standalone}

  \newcommand \tikzinput [2] [] {
    \setkeys{Gin}{#1}
    \ifx \Gin@ewidth \Gin@exclamation
      \ifx \Gin@eheight \Gin@exclamation
        \input { #2 }
      \else
        \resizebox{!}{ \Gin@eheight }{
          \input { #2 }
        }
      \fi
    \else
      \ifx \Gin@eheight \Gin@exclamation
        \resizebox{ \Gin@ewidth }{!}{
          \input { #2 }
        }
      \else
        \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
          \input { #2 }
        }
      \fi
    \fi
  }
}

\newcommand \ctikzinput [2] [] {
  \begin{center}
    \tikzinput [#1] {#2}
  \end{center}
}
```

⟨/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.

C

D

E

F

G

H

I

M

N

O

Q

R

S

stex internal commands:

T