

The sTeX3 Package Collection *

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

2022-09-12

Abstract

sTeX is a collection of L^AT_EX packages that allow to markup documents semantically without leaving the document format.

Running ‘pdflatex’ over sTeX-annotated documents formats them into normal-looking PDF. But sTeX also comes with a conversion pipeline into semantically annotated HTML5, which can host semantic added-value services that make the documents active (i.e. interactive and user-adaptive) and essentially turning L^AT_EX into a document format for (mathematical) knowledge management (MKM).

sTeX augments L^AT_EX with

- *semantic macros* that denote and distinguish between mathematical concepts, operators, etc. independent of their notational presentation,
- a powerful *module system* that allows for authoring and importing individual fragments containing document text and/or semantic macros, independent of – and without hard coding – directory paths relative to the current document, and
- a mechanism for exporting sTeX documents to (modular) XHTML, preserving all the semantic information for semantically informed knowledge management services.

This is the full documentation of sTeX. It consists of four parts:

- **Part I** is a general manual for the sTeX package and associated software. It is primarily directed at end-users who want to use sTeX to author semantically enriched documents.
- **Part II** documents the macros provided by the sTeX package. It is primarily directed at package authors who want to build on sTeX, but can also serve as a reference manual for end-users.
- **Part III** documents additional packages that build on sTeX, primarily its module system. These are not part of the sTeX package itself, but useful additions enabled by sTeX package functionality.
- **Part IV** is the detailed documentation of the sTeX package implementation.

*Version 3.2 (last revised 2022-09-12)

Contents

I	Manual	1
1	What is sTeX?	2
2	Setup	3
2.1	Setting up the sTeX Package	3
2.1.1	Minimal Setup for the sTeX Package	3
2.1.2	GIT-based Setup for the sTeX Development Version	3
2.1.3	Setting your MathHub Directory	4
2.2	Setting up the sTeX IDE	4
2.2.1	The sTeX VSCode Extension	4
2.2.2	Setting up MMT	4
2.3	Manual Setup	6
2.3.1	sTeX Archives (Manual Setup)	6
2.3.2	Manual Setup for Active Documents and Knowledge Management Services	6
3	The sTeX IDE	7
4	A First sTeX Document	8
4.1	OMDOC/xhtml Conversion	11
4.2	MMT/OMDOC Conversion	12
5	Creating sTeX Content	14
5.1	How Knowledge is Organized in sTeX	14
5.2	sTeX Archives	15
5.2.1	The Local MathHub-Directory	15
5.2.2	The Structure of sTeX Archives	16
5.2.3	MANIFEST.MF-Files	16
5.2.4	Using Files in sTeX Archives Directly	17
5.3	Module, Symbol and Notation Declarations	18
5.3.1	The smodule-Environment	18
5.3.2	Declaring New Symbols and Notations	20
	Operator Notations	24
5.3.3	Argument Modes	24
	Mode-b Arguments	24
	Mode-a Arguments	25
	Mode-B Arguments	26
5.3.4	Type and Definiens Components	27
5.3.5	Precedences and Automated Bracketing	28
5.3.6	Variables	30
5.3.7	Variable Sequences	31
5.4	Module Inheritance and Structures	33
5.4.1	Multilinguality and Translations	33
5.4.2	Simple Inheritance and Namespaces	34
5.4.3	The mathstructure Environment	36
5.4.4	The copymodule Environment	39

5.4.5	The <code>interpretmodule</code> Environment	40
5.5	Primitive Symbols (The <code>sTeX</code> Metatheory)	41
6	Using <code>sTeX</code> Symbols	42
6.1	<code>\symref</code> and its variants	42
6.2	Marking Up Text and On-the-Fly Notations	43
7	<code>sTeX</code> Statements	47
7.1	Definitions, Theorems, Examples, Paragraphs	47
7.2	Proofs	50
7.3	Highlighting and Presentation Customizations	55
8	Cross References	57
9	Additional Packages	59
9.1	<code>Tikzinput</code> : Treating TIKZ code as images	59
9.2	Modular Document Structuring	60
9.2.1	Introduction	60
9.2.2	Package Options	60
9.2.3	Document Fragments	60
9.2.4	Ending Documents Prematurely	62
9.2.5	Global Document Variables	62
9.3	Slides and Course Notes	62
9.3.1	Introduction	62
9.3.2	Package Options	63
9.3.3	Notes and Slides	63
9.3.4	Customizing Header and Footer Lines	64
9.3.5	Frame Images	65
9.3.6	Excursions	66
9.4	Representing Problems and Solutions	67
9.4.1	Introduction	67
9.4.2	Problems and Solutions	67
9.4.3	Markup for Added-Value Services	69
	Multiple Choice Blocks	69
	Filling-In Concrete Solutions	70
9.4.4	Including Problems	71
9.4.5	Testing and Spacing	72
9.5	Homeworks, Quizzes and Exams	72
9.5.1	Introduction	72
9.5.2	Package Options	72
9.5.3	Assignments	73
9.5.4	Including Assignments	73
9.5.5	Typesetting Exams	73
II	Documentation	75

10	<code>sTeX</code>-Basics	76
10.1	Macros and Environments	76
10.1.1	HTML Annotations	76
10.1.2	Babel Languages	77
10.1.3	Auxiliary Methods	77
11	<code>sTeX</code>-MathHub	78
11.1	Macros and Environments	78
11.1.1	Files, Paths, URIs	78
11.1.2	MathHub Archives	79
11.1.3	Using Content in Archives	80
12	<code>sTeX</code>-References	81
12.1	Macros and Environments	81
12.1.1	Setting Reference Targets	81
12.1.2	Using References	82
13	<code>sTeX</code>-Modules	83
13.1	Macros and Environments	83
13.1.1	The <code>smodule</code> environment	85
14	<code>sTeX</code>-Module Inheritance	87
14.1	Macros and Environments	87
14.1.1	SMS Mode	87
14.1.2	Imports and Inheritance	88
15	<code>sTeX</code>-Symbols	90
15.1	Macros and Environments	90
16	<code>sTeX</code>-Terms	92
16.1	Macros and Environments	92
17	<code>sTeX</code>-Structural Features	94
17.1	Macros and Environments	94
17.1.1	Structures	94
18	<code>sTeX</code>-Statements	95
18.1	Macros and Environments	95
19	<code>sTeX</code>-Proofs: Structural Markup for Proofs	96
20	<code>sTeX</code>-Metatheory	97
20.1	Symbols	97
III	Extensions	98
21	Tikzinput: Treating TIKZ code as images	99
21.1	Macros and Environments	99
22	document-structure: Semantic Markup for Open Mathematical Documents in \LaTeX	100

23	NotesSlides – Slides and Course Notes	101
24	problem.sty: An Infrastructure for formatting Problems	102
25	hwexam.sty/cls: An Infrastructure for formatting Assignments and Exams	103
IV	Implementation	104
26	STEX-Basics Implementation	105
26.1	The STEXDocument Class	105
26.2	Preliminaries	106
26.3	Messages and logging	106
26.4	HTML Annotations	107
26.5	Babel Languages	109
26.6	Persistence	111
26.7	Auxiliary Methods	111
27	STEX-MathHub Implementation	115
27.1	Generic Path Handling	115
27.2	PWD and kpsewhich	117
27.3	File Hooks and Tracking	118
27.4	MathHub Repositories	119
27.5	Using Content in Archives	124
28	STEX-References Implementation	129
28.1	Document URIs and URLs	129
28.2	Setting Reference Targets	131
28.3	Using References	134
29	STEX-Modules Implementation	141
29.1	The smodule environment	145
29.2	Invoking modules	151
30	STEX-Module Inheritance Implementation	153
30.1	SMS Mode	153
30.2	Inheritance	157
31	STEX-Symbols Implementation	163
31.1	Symbol Declarations	163
31.2	Notations	171
31.3	Variables	181
32	STEX-Terms Implementation	190
32.1	Symbol Invocations	190
32.2	Terms	198
32.3	Notation Components	203
32.4	Variables	205
32.5	Sequences	208

33	STEX-Structural Features Implementation	209
33.1	Imports with modification	210
33.2	The feature environment	218
33.3	Structure	218
34	STEX-Statements Implementation	230
34.1	Definitions	230
34.2	Assertions	236
34.3	Examples	239
34.4	Logical Paragraphs	242
35	The Implementation	247
35.1	Proofs	247
36	STEX-Others Implementation	256
37	STEX-Metatheory Implementation	258
38	Tikzinput Implementation	261
39	document-structure.sty Implementation	264
39.1	Package Options	264
39.2	Document Structure	265
39.3	Front and Backmatter	269
39.4	Global Variables	271
40	NotesSlides – Implementation	272
40.1	Class and Package Options	272
40.2	Notes and Slides	274
40.3	Header and Footer Lines	278
40.4	Frame Images	280
40.5	Sectioning	281
40.6	Excursions	284
41	The Implementation	286
41.1	Package Options	286
41.2	Problems and Solutions	287
41.3	Markup for Added Value Services	294
41.4	Multiple Choice Blocks	294
41.5	Filling in Concrete Solutions	295
41.6	Including Problems	296
41.7	Reporting Metadata	297
41.8	Testing and Spacing	298
42	Implementation: The hwexam Package	300
42.1	Package Options	300
42.2	Assignments	301
42.3	Including Assignments	304
42.4	Typesetting Exams	305
42.5	Leftovers	307

Part I

Manual



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



Boxes like this one explain how some \LaTeX concept relates to the MMT/OMDoc system, philosophy or language; see [MMT; Koh06] for introductions.

Chapter 1

What is sTeX?

Formal systems for mathematics (such as interactive theorem provers) have the potential to significantly increase both the accessibility of published knowledge, as well as the confidence in its veracity, by rendering the precise semantics of statements machine actionable. This allows for a plurality of added-value services, from semantic search up to verification and automated theorem proving. Unfortunately, their usefulness is hidden behind severe barriers to accessibility; primarily related to their surface languages reminiscent of programming languages and very unlike informal standards of presentation.

sTeX minimizes this gap between informal and formal mathematics by integrating formal methods into established and widespread authoring workflows, primarily L^AT_EX, via non-intrusive semantic annotations of arbitrary informal document fragments. That way formal knowledge management services become available for informal documents, accessible via an IDE for authors and via generated *active* documents for readers, while remaining fully compatible with existing authoring workflows and publishing systems.

Additionally, an extensible library of reusable document fragments is being developed, that serve as reference targets for global disambiguation, intermediaries for content exchange between systems and other services.

Every component of the system is designed modularly and extensibly, and thus lay the groundwork for a potential full integration of interactive theorem proving systems into established informal document authoring workflows.

The general sTeX workflow combines functionalities provided by several pieces of software:

- The sTeX package collection to use semantic annotations in L^AT_EX documents,
- RuS_{TeX} [RT] to convert `tex` sources to (semantically enriched) `xhtml`,
- The MMT system [MMT], that extracts semantic information from the thus generated `xhtml` and provides semantically informed added value services. Notably, MMT integrates the RuS_{TeX} system already.

Chapter 2

Setup

There are two ways of using sTeX : as a

1. way of writing \LaTeX more modularly (object-oriented Math) for creating PDF documents or
2. foundation for authoring active documents in HTML5 instrumented with knowledge management services.

Both are legitimate and useful. The first requires a significantly smaller tool-chain, so we describe it first. The second requires a much more substantial toolchain of knowledge management systems.

Luckily, the sTeX -IDE will take care of much of the setup required for the full toolchain, if you are willing to use it.

2.1 Setting up the sTeX Package

2.1.1 Minimal Setup for the sTeX Package

In the best of all worlds, there is no setup, as you already have a new version of \TeX Live on your system as a \LaTeX enthusiast. If not now is the time to install it; see [TL]. You can usually update \TeX Live via a package manager or the \TeX Live manager **tlmgr**. sTeX requires a \TeX kernel newer than February 2022.

Alternatively, you can install sTeX from CTAN, the Comprehensive \TeX Archive Network; see [ST] for details. We assume you have the sTeX package in at least version 3.2 (September 2022).

2.1.2 GIT-based Setup for the sTeX Development Version

If you want use the latest and greatest sTeX packages that have not even been released to CTAN, then you can directly clone them from the sTeX development repository [sTeX] by the following command-line instructions:

```
cd <stexdir>
git clone https://github.com/slatex/sTeX.git
```

and keep it updated by pulling updates via `git pull` in the cloned sTeX directory. Make sure to either clone the sTeX repository into a local `texmf-tree` or to update your `TEXINPUTS` environment variable, e.g. by placing the following line in your `.bashrc`:

```
export TEXINPUTS="$(TEXINPUTS):<sTeXDIR>//:"
```

2.1.3 Setting your MathHub Directory

One of sTeX’s features is a proper *module system* of interconnected document snippets for mathematical content. Analogously to *object-oriented programming*, it allows for “object-oriented mathematics” via individual combinable and, importantly, *reusable* modules, developed collaboratively.

To make use of such modules, the sTeX system needs to be told where to find them. There are several ways to do so (see [subsection 5.2.1](#)), but the most convenient way to do so is via a system variable.

To do so, create a directory **MathHub** somewhere on your local file system and set the environment variable **MATHHUB** to the file path to that directory.

In linux, you can do so by writing

```
export MATHHUB="/path/to/your/MathHub"
```

in your `~/.profile` (for all shells) or `~/.bashrc` (for the bash terminal only) file.

2.2 Setting up the sTeX IDE

The sTeX IDE consists of two components using the *Language Server Protocol (LSP)*: A *client* in the form of a VSCode extension, and a *server* included in the MMT system. Installing the extension will open up a setup routine that will guide you through the rest.

2.2.1 The sTeX VSCode Extension

If you have not already, you should first install the VSCode editor available at <https://code.visualstudio.com/>.

Next, open VSCode and install the sTeX extension by clicking on the *extensions* menu on the very left of the VSCode window and searching for “sTeX” in the “*Search Extensions in Marketplace*” field, as in [Figure 1](#), and clicking the *Install*-button of the sTeX extension by KWARC.

2.2.2 Setting up Mmt

Next, open any directory (File → Open Folder...) that contains a `.tex`-file, and a setup window as in [Figure 2](#) will pop up. Click on the highlighted link ‘*here*’ and download the latest version of the `MMT.jar` file (at least version 23.0.0) anywhere you like. Then click the “*Browse...*”-button and select your freshly downloaded `MMT.jar`.

If you have already set a system variable for your MathHub-directory, you are now done and can click “*Finish*”. If you have not, you can now also enter a directory path in the lower text field, and the VSCode extension will attempt to globally set one up for you, depending on your operating system.

Once you click “*Finish*”, the client will connect to <https://stexmmt.mathhub.info/:sTeX>, query for available archives, download the core libraries required for all (or most) semantic services (MMT/urtheories and sTeX/meta-inf) and set up RuSTeX for you automatically.

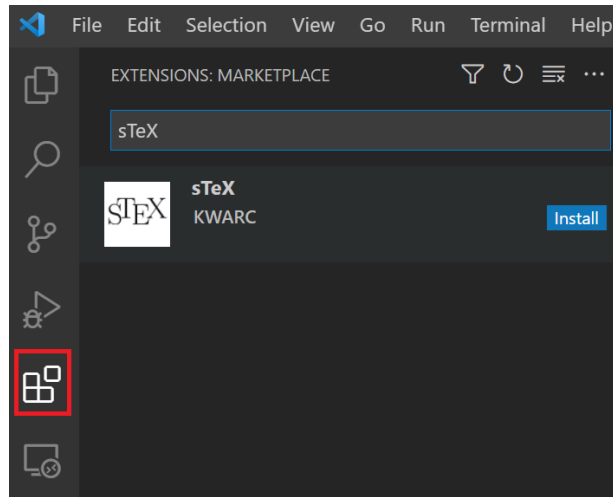


Figure 1: Installing the sTeX extension for VSCode

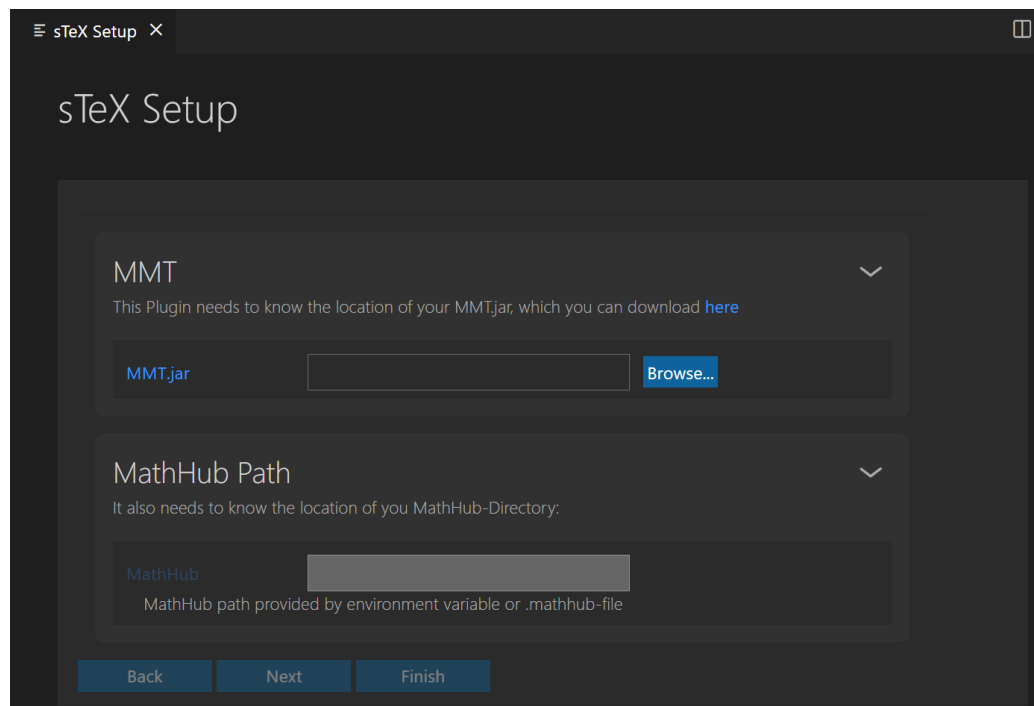


Figure 2: sTeX Setup Routine

2.3 Manual Setup

In lieu of using the $\text{\S}\text{\TeX}$ IDE, we can do the following:

2.3.1 $\text{\S}\text{\TeX}$ Archives (Manual Setup)

Writing semantically annotated $\text{\S}\text{\TeX}$ becomes much easier, if we can use well-designed libraries of already annotated content. $\text{\S}\text{\TeX}$ provides such libraries as $\text{\S}\text{\TeX}$ archives – i.e. GIT repositories at <https://gl.mathhub.info> – most prominently the SMGLoM libraries at <https://gl.mathhub.info/smgglom>.

To do so, we set up a **local MathHub** by creating a MathHub directory `<mhdir>`. Every $\text{\S}\text{\TeX}$ archive as an **archive path** `<apath>` and a name `<archive>`. We can clone the $\text{\S}\text{\TeX}$ archive by the following command-line instructions:

```
cd <mhdir>/<apath>
git clone https://gl.mathhub.info/smgglom/<archive>.git
```

Note that $\text{\S}\text{\TeX}$ archives often depend on other archives, thus you should be prepared to clone these as well – e.g. if `pdflatex` reports missing files. To make sure that $\text{\S}\text{\TeX}$ too knows where to find its archives, we need to set a global system variable `MATHHUB`, that points to your local MathHub-directory (see [section 5.2](#)).

```
export MATHHUB="<mhdir>"
```

2.3.2 Manual Setup for Active Documents and Knowledge Management Services

Foregoing on the $\text{\S}\text{\TeX}$ IDE, we will need several additional (on top of the minimal setup above) pieces of software; namely:

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

Following the setup routine (Step 3) will entail designating a **MathHub**-directory on your local file system, where the MMT system will look for $\text{\S}\text{\TeX}$ /MMT content archives.

- **$\text{\S}\text{\TeX}$ Archives** If we only care about \LaTeX and generating `pdfs`, we do not technically need MMT at all; however, we still need the `MATHHUB` system variable to be set. Furthermore, MMT can make downloading content archives we might want to use significantly easier, since it makes sure that all dependencies of (often highly interrelated) $\text{\S}\text{\TeX}$ archives are cloned as well.

Once set up, we can run `mmt` in a shell and download an archive along with all of its dependencies like this: `lmh install <name-of-repository>`, or a whole *group* of archives; for example, `lmh install smgglom` will download all `smgglom` archives.

- **Ru $\text{\S}\text{\TeX}$** The MMT system will also set up Ru $\text{\S}\text{\TeX}$ for you, which is used to generate (semantically annotated) `xhtml` from tex sources. In lieu of using MMT, you can also download and use Ru $\text{\S}\text{\TeX}$ directly [here](#).

Chapter 3

The \TeX IDE

Chapter 4

A First sTeX Document

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

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

Compiling this document with `pdflatex` should yield the output

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

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

Theorem 0.2. The [geometric series converges](#) towards 1.

Move your cursor over the various highlighted parts of the document – depending on your pdf viewer, this should yield some interesting (but possibly for now cryptic) information.

Remark 4.0.1:

Note that all of the highlighting, tooltips, coloring and the environment headers come from `stexthm` – by default, the amount of additional packages loaded is kept to a minimum and all the presentations can be customized, see [section 7.3](#).

Let’s investigate this document in detail to understand the respective parts of the \TeX markup infrastructure:

```
smodule (env.) \begin{smodule}{GeometricSeries}
...
\end{smodule}
```

First, we open a new *module* called `GeometricSeries`. The main purpose of the `smodule` environment is to group the contents and associate it with a *globally unique* identifier (URI), which is computed from the name `GeometricSeries` and the document context.

(Depending on your pdf viewer), the URI should pop up in a tooltip if you hover over the word [geometric series](#).

```
\importmodule \importmodule[smglom/calculus]{series}
\importmodule \importmodule[smglom/arithmetics]{realarith}
```

Next, we *import* two modules – `series` from the \TeX archive `smglom/calculus`, and `realarith` from the \TeX archive `smglom/arithmetics`. If we investigate these archives, we find the files `series.en.tex` and `realarith.en.tex` (respectively) in their respective source-folders, which contain the statements `\begin{smodule}{series}` and `\begin{smodule}{realarith}` (respectively).

The `\importmodule`-statements make all \TeX symbols and associated semantic macros (e.g. `\infinitesum`, `\realdive`, `\realpower`) in the imported module available to the current module `GeometricSeries`. The module `GeometricSeries` “exports” all of these symbols to all modules imports it via an `\importmodule{GeometricSeries}` instruction. Additionally it exports the local symbol `\geometricSeries`.

```
\usemodule
```

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

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

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

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

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

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

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

\symname ... is the `\symname{?series}`

The `\symname`-command prints the name of a symbol, highlights it (based on customizable settings) and associates the text printed with the corresponding symbol.

Note that the argument of `\symref` can be an imported symbol (here the `series` symbol is imported from the `series` module). \LaTeX tries to determine the full symbol URI from the argument. If there are name clashes in or with the imported symbols, the name of the exporting module can be prepended to the symbol name before the `?` character.

If you hover over the word `series` in the pdf output, you should see a tooltip showing the full URI of the symbol used.

\symref The `\symname`-command is a special case of the more general `\symref`-command, which allows customizing the precise text associated with a symbol. `\symref` takes two arguments: the first is the symbol name (or macro name), and the second a variant verbalization of the symbol, e.g. an inflection variant, a different language or a synonym. In our example `\symname{?series}` abbreviates `\symref{?series}{series}`.

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

```

\[\defeq{\geometricSeries}{\definiens{
  \infinitesum{\svar{n}}{1}{
    \realdivide[frac]{1}{
      \realpower{2}{\svar{n}}
    }
  }}
}\].\]

```

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

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

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

4.1 OMDoc/xhtml Conversion

So, if we run `pdflatex` on our document, then \TeX yields pretty colors and tooltips¹. But \TeX becomes a lot more powerful if we additionally convert our document to `xhtml` while preserving all the \TeX markup in the result.

TODO VSCode Plugin

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

```

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

```

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

```

<mi resource="1" property="stex:arg">2</mi>
<mrow resource="2" property="stex:arg">
  <mi resource="var://n" property="stex:OMV">n</mi>
</mrow>
</msup>
</mrow>
</mfrac>
</mrow>
</mrow>
</mrow>

```

...containing all the semantic information. The MMT system can extract from this the following OPENMATH snippet:

```

<OMBIND>
  <OMID name="...?series?infinitiesum"/>
  <OMV name="n"/>
  <OMLIT name="1"/>
  <OMA>
    <OMS name="...?realarith?division"/>
    <OMLIT name="1"/>
    <OMA>
      <OMS name="...realarith?exponentiation"/>
      <OMLIT name="2"/>
      <OMV name="n"/>
    </OMA>
  </OMA>
</OMBIND>

```

...giving us the full semantics of the snippet, allowing for a plurality of knowledge management services – in particular when serving the `xhtml`.

Remark 4.1.1:

Note that the `html` when opened in a browser will look slightly different than the `pdf` when it comes to highlighting semantic content – that is because naturally `html` allows for much more powerful features than `pdf` does. Consequently, the `html` is intended to be served by a system like MMT, which can pick up on the semantic information and offer much more powerful highlighting, linking and similar features, and being customizable by *readers* rather than being prescribed by an author.

Additionally, not all browsers (most notably Chrome) support MATHML natively, and might require additional external JavaScript libraries such as MathJax to render mathematical formulas properly.

4.2 Mmt/OMDoc Conversion

Another way to convert our document to *actual* MMT/OMDOC is to put it in an `TEX` archive (see [section 5.2](#)) and have MMT take care of everything.

Assuming the above file is `source/demo.tex` in an `TEX` archive `MyTest`, you can run MMT and do `build MyTest stex-omdoc demo.tex` to convert the document to both `xhtml` (which you will find in `xhtml/demo.xhtml` in the archive) and formal MMT/OMDOC, which you can subsequently view in the MMT browser (see <https://>

uniformal.github.io/doc/applications/server.html#the-mmt-web-site for details).

Chapter 5

Creating sTeX Content

We can use sTeX by simply including the package with `\usepackage{stex}`, or – primarily for individual fragments to be included in other documents – by using the sTeX document class with `\documentclass{stex}` which combines the standalone document class with the stex package.

Both the stex package and document class offer the following options:

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

mathhub (*(⟨directory⟩)*) MathHub folder to search for repositories – this is not necessary if the MATHHUB system variable is set.

writesms (*(⟨boolean⟩)*) with this package option, sTeX will write the contents of all external modules imported via `\importmodule` or `\usemodule` into a file `\jobname.sms` (analogously to the table of contents `.toc`-file).

usesms (*(⟨boolean⟩)*) subsequently tells sTeX to read the generated sms-file at the beginning of the document. This allows for e.g. collaborating on documents without all authors having to have all used archives and modules available – one author can load the modules with **writesms**, and the rest can use the modules with **usesms**. Furthermore, the sms file can be submitted alongside a `tex`-file, effectively making it “standalone”.

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

debug (*(⟨log-prefix⟩*)*) Logs debugging information with the given prefixes to the terminal, or all if **all** is given. Largely irrelevant for the majority of users.

5.1 How Knowledge is Organized in sTeX

sTeX content is organized on multiple levels:

1. sTeX **archives** (see [section 5.2](#)) contain individual `.tex`-files.
2. These may contain sTeX **modules**, introduced via `\begin{smodule}{ModuleName}`.

3. Modules contain $\text{\S}\text{\TeX}$ **symbol declarations**, introduced via `\symdecl{symbolname}`, `\symdef{symbolname}` and some other constructions. Most symbols have a *notation* that can be used via a *semantic macro* `\symbolname` generated by symbol declarations.
4. $\text{\S}\text{\TeX}$ **expressions** finally are built up from usages of semantic macros.

- $\text{\S}\text{\TeX}$ archives are simultaneously MMT archives, and the same directory structure is consequently used.
 - $\text{\S}\text{\TeX}$ modules correspond to OMDOC/MMT *theories*. `\importmodules` (and similar constructions) induce MMT `\includes` and other *theory morphisms*, thus giving rise to a *theory graph* in the OMDOC sense [RK13].
 - Symbol declarations induce OMDOC/MMT *constants*, with optional (formal) *type* and *definiens* components.
 - Finally, $\text{\S}\text{\TeX}$ expressions are converted to OMDOC/MMT terms, which use the abstract syntax (and XML encoding) of OPENMATH [Bus+04].

$\hookrightarrow M \rightarrow$
 $\hookrightarrow M \rightarrow$
 $\hookrightarrow T \rightarrow$

5.2 $\text{\S}\text{\TeX}$ Archives

5.2.1 The Local MathHub-Directory

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

All $\text{\S}\text{\TeX}$ archives need to exist in the local MathHub-directory. $\text{\S}\text{\TeX}$ knows where this folder is via one of four means:

1. If the $\text{\S}\text{\TeX}$ package is loaded with the option `mathhub=/path/to/mathhub`, then $\text{\S}\text{\TeX}$ 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 $\text{\S}\text{\TeX}$ -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, $\text{\S}\text{\TeX}$ will attempt to retrieve the system variable `MATHHUB`, assuming it will point to the local MathHub-directory. Since this variant needs setting up only *once* and is machine-specific (rather than defined in tex code), it is compatible with collaborating and sharing tex content, and hence recommended.
4. Finally, if all else fails, $\text{\S}\text{\TeX}$ will look for a file `~/.stex/mathhub.path`. If this file exists, $\text{\S}\text{\TeX}$ 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.

5.2.2 The Structure of \TeX Archives

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

Each such archive needs two subdirectories:

- `/source` – this is where all your tex files go.
- `/META-INF` – a directory containing a single file `MANIFEST.MF`, the content of which we will consider shortly

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

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

We recommend the following additional directory structure in the `source`-folder of an \TeX archive:

- `/source/mod/` – individual \TeX modules, containing symbol declarations, notations, and `\begin{spargraph}[type=symdoc,for=...]` environments for “encyclopaedic” symbol documentations
- `/source/def/` – definitions
- `/source/ex/` – examples
- `/source/thm/` – theorems, lemmata and proofs; preferably proofs in separate files to allow for multiple proofs for the same statement
- `/source/snip/` – individual text snippets such as remarks, explanations etc.
- `/source/frag/` – individual document fragments, ideally only `\inputrefing` snippets, definitions, examples etc. in some desirable order
- `/source/tikz/` – tikz images, as individual `.tex`-files
- `/source/PIC/` – image files.

5.2.3 MANIFEST.MF-Files

The `MANIFEST.MF` in the `META-INF`-directory consists of key-value-pairs, informing \TeX (and associated software) of various properties of an archive. For example, the `MANIFEST.MF` of the `smglom/calculus`-archive looks like this:

```
id: smglom/calculus
source-base: http://mathhub.info/smgglom/calculus
narration-base: http://mathhub.info/smgglom/calculus
dependencies: smglom/arithmetic,smglom/sets,smglom/topology,
              smglom/mv,smglom/linear-algebra,smglom/algebra
responsible: Michael.Kohlhase@FAU.de
title: Elementary Calculus
```

<code>teaser: Terminology for the mathematical study of change.</code> <code>description: desc.html</code>

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

`id`: The name of the archive, including its group (e.g. `smglom/calculus`),

`source-base` or

`ns`: The namespace from which all symbol and module URIs in this repository are formed, see (TODO),

`narration-base`: The namespace from which all document URIs in this repository are formed, see (TODO),

`url-base`: The URL that is formed as a basis for *external references*, see (TODO),

`dependencies`: All archives that this archive depends on. \TeX ignores this field, but MMT can pick up on them to resolve dependencies, e.g. for `lmh install`.

5.2.4 Using Files in \TeX Archives Directly

Several macros provided by \TeX allow for directly including files in repositories. These are:

`\mhinput` `\mhinput`[Some/Archive]{some/file} directly inputs the file `some/file` in the `source-` folder of `Some/Archive`.

`\inputref` `\inputref`[Some/Archive]{some/file} behaves like `\mhinput`, but wraps the input in a `\begingroup ... \endgroup`. When converting to `xhtml`, the file is not input at all, and instead an `html`-annotation is inserted that references the file, e.g. for lazy loading.

In the majority of practical cases `\inputref` is likely to be preferred over `\mhinput` because it leads to less duplication in the generated `xhtml`.

`\ifinput` Both `\mhinput` and `\inputref` set `\ifinput` to “true” during input. This allows for selectively including e.g. bibliographies only if the current file is not being currently included in a larger document.

`\addmhbibresource` `\addmhbibresource`[Some/Archive]{some/file} searches for a file like `\mhinput` does, but calls `\addbibresource` to the result and looks for the file in the archive root directory directly, rather than the `source` directory. Typical invocations are

- `\addmhbibresource{lib/refs.bib}`, which specifies a bibliography in the `lib` folder in the local archive or
- `\addmhbibresource[HW/meta-inf]{lib/refs.bib}` in another.

`\libinput` `\libinput{some/file}` searches for a file `some/file` in

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

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

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

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

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

Remark 5.2.1:

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

```
1 \documentclass{stex}
2 \libinput{preamble}
3 \begin{document}
4   ...
5   \ifinputref \else \libinput{postamble} \fi
6 \end{document}
```

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

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

5.3 Module, Symbol and Notation Declarations

5.3.1 The `smodule`-Environment

`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.

`type` ($\langle string \rangle^*$) for use in customizations.
`deprecate` ($\langle module \rangle$) if set, will throw a warning when loaded, urging to use $\langle module \rangle$ instead.
`id` ($\langle string \rangle$) for cross-referencing.
`ns` ($\langle URI \rangle$) the namespace to use. *Should not be used, unless you know precisely what you're doing.* If not explicitly set, is computed using `\stex_modules_current_namespace:`.
`lang` ($\langle language \rangle$) if not set, computed from the current file name (e.g. `foo.en.tex`).
`sig` ($\langle language \rangle$) if the current file is a translation of a file with the same base name but a different language suffix, setting `sig=<lang>` will preload the module from that language file. This helps ensuring that the (formal) content of both modules is (almost) identical across languages and avoids duplication.
`creators` ($\langle string \rangle^*$) names of the creators.
`contributors` ($\langle string \rangle^*$) names of contributors.
`srccite` ($\langle string \rangle$) a source citation for the content of this module.

\hookrightarrow An sTeX module corresponds to an MMT/OMDOC *theory*. As such it
 \hookrightarrow gets assigned a module URI (*universal resource identifier*) of the form
 \hookrightarrow `<namespace>?<module-name>`.

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

Example 1

Input:

```

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

```

Output:

Hello World

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

For example:

Example 2

Input:

```

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

```

Output:

```

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

```

5.3.2 Declaring New Symbols and Notations

Inside an `smodule` environment, we can declare new \TeX symbols.

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

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

\hookrightarrow `\symdecl` introduces a new OMDoc/MMT constant in the current module (=OMDoc/MMT theory). Correspondingly, they get assigned the URI `<module-URI>?<constant-name>`.

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

Example 3

Input:

```

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

```

Output:

```

Given a foo, we can...

```

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

Example 4

Input:

```

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

```

Output:

```

this is a symbol taking two arguments.

```

So far we have gained exactly ... nothing by adding the arity information: we cannot do anything with the arguments in the text.

We will now see what we can gain with more machinery.

\notation We probably want to supply a notation as well, in which case we can finally actually use the semantic macro in math mode. We can do so using the **\notation** command, like this:

Example 5

Input:

```

1 \notation{binarysymbol}{\text{First: }#1\text{; Second: }#2}
2 $\binarysymbol{a}{b}$

```

Output:

```

First: a; Second: b

```

\hookrightarrow Applications of semantic macros, such as $\binarysymbol{a}{b}$ are translated to
 \rightarrow MMT/OMDOC as OMA-terms with head `<OMS name="...?binarysymbol"/>`.
 \rightsquigarrow Semantic macros with no arguments correspond to OMS directly.

\comp For many semantic services e.g. semantic highlighting or **wikification** (linking user-visible notation components to the definition of the respective symbol they come from), we need to specify the notation components. Unfortunately, there is currently no way the \TeX engine can infer this by itself, so we have to specify it manually in the notation specification. We can do so with the **\comp** command.

We can introduce a new notation **highlight** for \binarysymbol that fixes this flaw, which we can subsequently use with $\binarysymbol[\text{highlight}]$:

Example 6

Input:

```
1 \notation{binarysymbol}[highlight]
2   {\comp{\text{First: }}#1\comp{\text{; Second: }}#2}
3 $\binarysymbol[highlight]{a}{b}$
```

Output:

First: a ; Second: b



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



Note that it is required that

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

Both criteria are not just required for technical reasons, but conceptionally meaningful:

The underlying principle is that the arguments to a semantic macro represent *arguments to the mathematical operation* represented by a symbol. For example, a semantic macro `\addition{a}{b}` taking two arguments would represent *the actual addition of (mathematical objects) a and b*. It should therefore be impossible for a or b to be part of a notation component of `\addition`.

Similarly, a semantic macro can not conceptually be part of the notation of `\addition`, since a semantic macro represents a *distinct mathematical concept* with *its own semantics*, whereas notations are syntactic representations of the very symbol to which the notation belongs.

If you want an argument to a semantic macro to be a purely syntactic parameter, then you are likely somewhat confused with respect to the distinction between the precise *syntax* and *semantics* of the symbol you are trying to declare (which happens quite often even to experienced \TeX users), and might want to give those another thought - quite likely, the macro 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.

`\symdef` In the vast majority of cases where a symbol declaration should come with a semantic macro, we will want to supply a notation immediately. For that reason, the `\symdef` command combines the functionality of both `\symdecl` and `\notation` with the optional arguments of both:

Example 7

Input:

```
1 \symdef{newbinarysymbol}[hl,args=2]
2   {\comp{\text{1.: }}#1\comp{\text{; 2.: }}#2}
3 $\newbinarysymbol{a}{b}$
```

Output:

```
1.: a; 2.: b
```

We just declared a new symbol `newbinarysymbol` with `args=2` and immediately provided it with a notation with identifier `hl`. Since `hl` is the *first* (and so far, only) notation supplied for `newbinarysymbol`, using `\newbinarysymbol` without optional argument defaults to this notation.

But one man’s meat is another man’s poison: it is very subjective what the “default notation” of an operator should be. Different communities have different practices. For instance, the complex unit is written as i in Mathematics and as j in electrical engineering. So to allow modular specification and facilitate re-use of document fragments $\text{\texttt{STEX}}$ allows to re-set notation defaults.

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

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

`\textsymdecl` In the less mathematical settings where we want a symbol and semantic macro for some concept with a notation *beyond* its mere name, but which should also be available in $\text{\texttt{TEX}}$ ’s text mode, the command `\textsymdecl` is useful. For example, we can declare a symbol `openmath` with the notation `\textsc{OpenMath}` using `\textsymdecl{openmath}[name=OpenMath]{\textsc{OpenMath}}`. The `\openmath` yields `OPENMATH` both in text and math mode.

Operator Notations

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

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

Example 8

Input:

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

Output:

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

\hookrightarrow `\symbolname!` is translated to OMDoc/MMT as `<OMS name="...?symbolname"/>` directly.

5.3.3 Argument Modes

The notations so far used *simple* arguments which we call *mode-i* arguments. Declaring a new symbol with `\symdecl{foo}[args=3]` is equivalent to writing `\symdecl{foo}[args=iii]`, indicating that the semantic macro takes three *mode-i* arguments. However, there are three more argument modes which we will investigate now, namely *mode-b*, *mode-a* and *mode-B* arguments.

Mode-b Arguments

A *mode-b* argument represents a *variable* that is *bound* by the symbol in its application, making the symbol a *binding operator*. Typical examples of binding operators are e.g. sums \sum , products \prod , integrals \int , quantifiers like \forall and \exists , that λ -operator, etc.

\hookrightarrow *Mode-b* arguments behave exactly like *mode-i* arguments within $\text{T}_{\text{E}}\text{X}$, but applications of binding operators, i.e. symbols with *mode-b* arguments, are translated to *OMBIND*-terms in OMDoc/MMT, rather than *OMA*.

For example, we can implement a summation operator binding an index variable and taking lower and upper index bounds and the expression to sum over like this:

Example 9

Input:

```
1 \symdef{summation}[args=biii]
2 {\mathop{\comp{\sum}}_{\#1}\comp{=}\#2}^{\#3}\#4}
3 $\summation{\svar{x}}{\#1}\{\svar{n}\}\{\svar{x}}^{\#2}$
```

Output:

$$\sum_{x=1}^n x^2$$

where the variable x is now *bound* by the `\summation`-symbol in the expression.

Mode-a Arguments

Mode-a arguments represent a *flexary argument sequence*, i.e. a sequence of arguments of arbitrary length. Formally, operators that take arbitrarily many arguments don't "exist", but in informal mathematics, they are ubiquitous. Mode-a arguments allow us to write e.g. `\addition{a,b,c,d,e}` rather than having to write something like `\addition{a}\addition{b}\addition{c}\addition{d}\addition{e}`!

`\notation` (and consequently `\symdef`, too) take one additional argument for each mode-a argument that indicates how to "accumulate" a comma-separated sequence of arguments. This is best demonstrated on an example.

Let's say we want an operator representing quantification over an ascending chain of elements in some set, i.e. `\ascendingchain{S}{a,b,c,d,e}{t}` should yield $\forall a <_S b <_S c <_S d <_S e. t$. The "base"-notation for this operator is simply `\comp{\forall} \#2 \comp{. ,} \#3`, where `\#2` represents the full notation fragment *accumulated* from `{a,b,c,d,e}`.

The *additional* argument to `\notation` (or `\symdef`) takes the same arguments as the base notation and two *additional* arguments `\#1` and `\#2` representing successive pairs in the mode-a argument, and accumulates them into `\#2`, i.e. to produce $a <_S b <_S c <_S d <_S e$, we do `\#1 \comp{<}_{\#1} \#2`:

Example 10

Input:

```
1 \symdef{ascendingchain}[args=iaai]
2 {\comp{\forall} \#2 \comp{. ,} \#3}
3 {\#1 \comp{<}_{\#1} \#2}
4
5 Tadaa: $\ascendingchain{S}{a,b,c,d,e}{t}$
```

Output:

Tadaa: $\forall a <_S b <_S c <_S d <_S e. t$

If this seems overkill, keep in mind that you will rarely need the single-hash arguments `#1,#2` etc. in the `a`-notation-argument. For a much more representative and simpler example, we can introduce flexary addition via:

Example 11

Input:

```
1 \symdef{addition}[args=a]{#1}{##1 \comp{+} ##2}
2
3 Tadaa: $\addition{a,b,c,d,e}$
```

Output:

Tadaa: $a+b+c+d+e$

The `assoc`-key We mentioned earlier that “formally”, flexary arguments don’t really “exist”. Indeed, formally, addition is usually defined as a binary operation, quantifiers bind a single variable etc.

Consequently, we can tell $\text{\texttt{STeX}}$ (or, rather, $\text{\texttt{MMT/OMDoc}}$) how to “resolve” flexary arguments by providing `\symdecl` or `\symdef` with an optional `assoc`-argument, as in `\symdecl{addition}[args=a,assoc=bin]`. The possible values for the `assoc`-key are:

`bin`: A binary, associative argument, e.g. as in `\addition`

`binl`: A binary, left-associative argument, e.g. $a^{b^{c^d}}$, which stands for $((a^b)^c)^d$

`binr`: A binary, right-associative argument, e.g. as in $A \rightarrow B \rightarrow C \rightarrow D$, which stands for $A \rightarrow (B \rightarrow (C \rightarrow D))$

`pre`: Successively prefixed, e.g. as in $\forall x. y. z. P$, which stands for $\forall x. \forall y. \forall z. P$

`conj`: Conjunctive, e.g. as in $a = b = c = d$ or $a, b, c, d \in A$, which stand for $a = d \wedge b = d \wedge c = d$ and $a \in A \wedge b \in A \wedge c \in A \wedge d \in A$, respectively

`pwconj`: Pairwise conjunctive, e.g. as in $a \neq b \neq c \neq d$, which stands for $a \neq b \wedge a \neq c \wedge a \neq d \wedge b \neq c \wedge b \neq d \wedge c \neq d$

As before, at the PDF level, this annotation is invisible (and without effect), but at the level of the generated $\text{\texttt{OMDoc/MMT}}$ this leads to more semantical expressions.

Mode-B Arguments

Finally, mode-B arguments simply combine the functionality of both `a` and `b` - i.e. they represent an arbitrarily long sequence of variables to be bound, e.g. for implementing quantifiers:

Example 12

Input:

```

1 \symdef{quantforall}[args=Bi]
2   {\comp{\forall}#1\comp{.}#2}
3   {##1\comp{,}##2}
4
5 \$\quantforall{\svar{x},\svar{y},\svar{z}}{P}$

```

Output:

$\forall x,y,z.P$

5.3.4 Type and Definiens Components

`\symdecl` and `\symdef` take two more optional arguments. \TeX largely ignores them (except for special situations we will talk about later), but MMT can pick up on them for additional services. These are the `type` and `def` keys, which expect expressions in math-mode (ideally using semantic macros, of course!)

The `type` and `def` keys correspond to the `type` and `definiens` components of

- \hookrightarrow OMDoc/MMT constants.
- \hookrightarrow Correspondingly, the name “type” should be taken with a grain of salt, since
- \hookrightarrow OMDoc/MMT – being foundation-independent – does not a priori implement a fixed typing system.

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

Example 13

Input:

```

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

```

Output:

`addition` is an operation $\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$

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

Example 14

Input:

```

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

```

Output:

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

5.3.5 Precedences and Automated Bracketing

Having done `\addition`, the obvious next thing to implement is `\multiplication`. This is straight-forward in theory:

Example 15

Input:

```

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

```

Output:

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

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

Example 16

Input:

```

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

```

Output:

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

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

Example 17

Input:

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

Output:

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

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

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

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

Example 18

Input:

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

Output:

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

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

`\infprec`
`\neginfprec`

It is occasionally useful to have “infinitely” high or low precedences to enforce or forbid automated bracketing entirely, e.g. for bracket-like notations such as intervals – for those purposes, `\infprec` and `\neginfprec` exist (which are implemented as the maximal and minimal integer values accordingly).

More precisely, each notation takes

1. One *operator precedence* and
2. one *argument precedence* for each argument.

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

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

When $\text{\S}\text{\TeX}$ steps into an argument of a semantic macro, it sets p_d to the respective argument precedence of the notation used.

In the example above:



1. $\text{\S}\text{\TeX}$ starts out with $p_d = \text{\code{\infprec}}$.
2. $\text{\S}\text{\TeX}$ encounters `\addition` with $p_{op} = 100$. Since $100 \not> \text{\code{\infprec}}$, it inserts no parentheses.
3. Next, $\text{\S}\text{\TeX}$ encounters the two arguments for `\addition`. Both have no specifically provided argument precedence, so $\text{\S}\text{\TeX}$ uses $p_d = p_{op} = 100$ for both and recurses.
4. Next, $\text{\S}\text{\TeX}$ encounters `\multiplication{b,...}`, whose notation has $p_{op} = 50$.
5. We compare to the current downward precedence p_d set by `\addition`, arriving at $p_{op} = 50 \not> 100 = p_d$, so $\text{\S}\text{\TeX}$ again inserts no parentheses.
6. Since the notation of `\multiplication` has no explicitly set argument precedences, $\text{\S}\text{\TeX}$ uses the operator precedence for all arguments of `\multiplication`, hence sets $p_d = p_{op} = 50$ and recurses.
7. Next, $\text{\S}\text{\TeX}$ encounters the inner `\addition{c,...}` whose notation has $p_{op} = 100$.
8. We compare to the current downward precedence p_d set by `\multiplication`, arriving at $p_{op} = 100 > 50 = p_d$ – which finally prompts $\text{\S}\text{\TeX}$ to insert parentheses, and we proceed as before.

5.3.6 Variables

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

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

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

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

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

Example 19

Input:

```

1 \vardef{varf}[
2   name=f,
3   type=\funtype{\Nat}{\Nat},
4   op=f,
5   args=1,
6   prec=0;\neginfprec
7 ]{\comp{f}#1}
8 \vardef{varn}[name=n,type=\Nat]{\comp{n}}
9 \vardef{varx}[name=x,type=\Nat]{\comp{x}}
10
11 Given a function $\varf!:\funtype{\Nat}{\Nat}$,
12 by $\addition{\varf!,\varn}$ we mean the function\rustexBREAK
13 $\fun{\varx}{\varf{\addition{\varx,\varn}}}$

```

Output:

Given a function $f : \mathbb{N} \rightarrow \mathbb{N}$, by $f+n$ we mean the function $x \mapsto f(x+n)$

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

TODO: bind=forall/exists

5.3.7 Variable Sequences

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

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

This is best shown by example:

Example 20

Input:

```

1 \vardef{varn}[name=n,type=\Nat]{\comp{n}}
2 \varseq{seqa}[name=a,type=\Nat]{1}{\varn}{\comp{a}_{#1}}
3
4 The $i$th index of $\seqa!$ is $\seqa{i}$$.

```

Output:

The i th index of a_1, \dots, a_n is a_i .

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

TODO: more notations for invoking sequences.

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

Example 21

Input:

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

Output:

$a_1 + \dots + a_n$

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

Example 22

Input:

```

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

```

Output:

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

We can also explicitly provide a “middle” segment to be used, like such:

Example 23

Input:

```

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

```

Output:

$$a_1^1, \dots, a_n^1, a_1^2, \dots, a_1^m, \dots, a_n^m \text{ and } a_1^1 + \dots + a_n^1 + a_1^2 + \dots + a_1^m + \dots + a_n^m$$

5.4 Module Inheritance and Structures

The \TeX features for modular document management are inherited from the OM-Doc/MMT model that organizes knowledge into a graph, where the nodes are theories (called modules in \TeX) and the edges are truth-preserving mappings (called theory morphisms in MMT). We have already seen modules/theories above.

Before we get into theory morphisms in \TeX we will see a very simple application of modules: managing multilinguality modularly.

5.4.1 Multilinguality and Translations

If we load the \TeX document class or package with the option `lang=<lang>`, \TeX will load the appropriate `babel` language for you – e.g. `lang=de` will load the `babel` language `ngerman`. Additionally, it makes \TeX aware of the current document being set in (in this example) *german*. This matters for reasons other than mere `babel`-purposes, though:

Every *module* is assigned a language. If no \TeX package option is set that allows for inferring a language, \TeX will check whether the current file name ends in e.g. `.en.tex` (or `.de.tex` or `.fr.tex`, or...) and set the language accordingly. Alternatively, a language can be explicitly assigned via `\begin{smodule}[lang=<language>]{Foo}`.

Technically, each `smodule`-environment induces *two* OMDoc/MMT theories:
 \hookrightarrow `\begin{smodule}[lang=<lang>]{Foo}` generates a theory `some/namespace?Foo` that only contains the “formal” part of the module – i.e. exactly the content that is exported when using `\importmodule`.
 \rightsquigarrow Additionally, MMT generates a *language theory* `some/namespace/Foo?<lang>` that includes `some/namespace?Foo` and contains all the other document content – variable declarations, includes for each `\usemodule`, etc.

Notably, the language suffix in a filename is ignored for `\usemodule`, `\importmodule` and in generating/computing URIs for modules. This however allows for providing *translations* for modules between languages without needing to duplicate content:

If a module `Foo` exists in e.g. `english` in a file `Foo.en.tex`, we can provide a file `Foo.de.tex` right next to it, and write `\begin{smodule}[sig=en]{Foo}`. The `sig`-key then signifies, that the “signature” of the module is contained in the *english* version of the module, which is immediately imported from there, just like `\importmodule` would.

Additionally to translating the informal content of a module file to different languages, it also allows for customizing notations between languages. For example, the *least common multiple* of two numbers is often denoted as `lcm(a,b)` in english, but is called *kleinstes gemeinsames Vielfaches* in german and consequently denoted as `kgV(a,b)` there.

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

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

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

5.4.2 Simple Inheritance and Namespaces

`\importmodule` `\importmodule[Some/Archive]{path?ModuleName}` is only allowed within an `smodule`-environment and makes the symbols declared in `ModuleName` available therein. Additionally the symbols of `ModuleName` will be exported if the current module is imported somewhere else via `\importmodule`.

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

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



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

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

- If `\begin{smodule}{Foo}` occurs in a file `/path/to/file/Foo[.<lang>].tex` which does not belong to an archive, the namespace is `file://path/to/file`.
- If the same statement occurs in a file `/path/to/file/bar[.<lang>].tex`, the namespace is `file://path/to/file/bar`.

In other words: outside of archives, the namespace corresponds to the file URI



with the filename dropped iff it is equal to the module name, and ignoring the (optional) language suffix.

If the current file is in an archive, the procedure is the same except that the initial segment of the file path up to the archive's `source`-folder is replaced by the archive's namespace URI.



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

- `\importmodule{Foo}` outside of an archive refers to module `Foo` in the current namespace. Consequently, `Foo` must have been declared earlier in the same document or, if not, in a file `Foo[.<lang>].tex` in the same directory.
- The same statement *within* an archive refers to either the module `Foo` declared earlier in the same document, or otherwise to the module `Foo` in the archive's top-level namespace. In the latter case, it has to be declared in a file `Foo[.<lang>].tex` directly in the archive's `source`-folder.
- Similarly, in `\importmodule{some/path?Foo}` the path `some/path` refers to either the sub-directory and relative namespace path of the current directory and namespace outside of an archive, or relative to the current archive's top-level namespace and `source`-folder, respectively.

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

- Similarly, `\importmodule[Some/Archive]{some/path?Foo}` is resolved like the previous cases, but relative to the archive `Some/Archive` in the mathhub-directory.
- Finally, `\importmodule{full://uri?Foo}` naturally refers to the module `Foo` in the namespace `full://uri`. Since the file this module is declared in can not be determined directly from the URI, the module must be in memory already, e.g. by being referenced earlier in the same document. Since this is less compatible with a modular development, using full URIs directly is strongly discouraged, unless the module is declared in the current file directly.

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



For persistency reasons, everything in an `\STEXexport` is digested by T_EX in the L^AT_EX3-category code scheme. This means that the characters `_` and `:` are considered *letters* and valid parts of control sequence names, and space characters are



ignored entirely. For spaces, use the character `~` instead, and keep in mind, that if you want to use subscripts, you should use `\c_math_subscript_token` instead of `_`!

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

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

5.4.3 The `mathstructure` Environment

A common occurrence in mathematics is bundling several interrelated “declarations” together into *structures*. For example:

- A *monoid* is a structure $\langle M, \circ, e \rangle$ with $\circ : M \times M \rightarrow M$ and $e \in M$ such that...
- A *topological space* is a structure $\langle X, \mathcal{T} \rangle$ where X is a set and \mathcal{T} is a topology on X
- A *partial order* is a structure $\langle S, \leq \rangle$ where \leq is a binary relation on S such that...

This phenomenon is important and common enough to warrant special support, in particular because it requires being able to *instantiate* such structures (or, rather, structure *signatures*) in order to talk about (concrete or variable) *particular* monoids, topological spaces, partial orders etc.

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

How this works is again best demonstrated by example:

Example 24

Input:

```

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

Output:

A *monoid* is...

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

Example 25

Input:

```

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

Output:

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

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

Example 26

Input:

```

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

Output:

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

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

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

`\instantiate` and `mathstructure` make use of the *Theories-as-Types* paradigm (see [MRK18]):

- `mathstructure{<name>}` simply creates a nested theory with name $\hookrightarrow M \rightarrow$ `<name>-structure`. The *constant* `<name>` is defined as `Mod(<name>-structure)`
- $\hookrightarrow M \rightarrow$ – a *dependent record type with manifest fields*, the fields of which are generated
- $\rightsquigarrow T \rightsquigarrow$ from (and correspond to) the constants in `<name>-structure`.

`\instantiate` generates a constant whose definiens is a record term of type `Mod(<name>-structure)`, with the fields assigned based on the respective key-value-list.

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

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

`\varinstantiate` The answer is that we can also instantiate a `mathstructure` with a *variable*. The syntax of `\varinstantiate` is equivalent to that of `\instantiate`, but all of the key-value-pairs are optional, and if not explicitly assigned (to a symbol *or* a variable declared with `\vardef`) inherit their notation from the one in the `mathstructure` environment.

This allows us to do things like:

Example 27

Input:

```
1 \varinstantiate{varM}{monoid}{M}
2
3 A \symname{monoid} is a structure
4 $\varM! := \mathstrut{\varM{universe}, \varM{op}!, \varM{unit}}$
5 such that
6 $\varM{op}!: \funtype{\varM{universe}, \varM{universe}}{\varM{universe}}$ ...
```

Output:

A `monoid` is a structure $M := \langle U, \circ, e \rangle$ such that $\circ : U \times U \rightarrow U$...

.

and

Example 28

Input:

```

1 \varinstantiate{varMb}{monoid}{M_2}[universe = Int]
2
3 Let $\varMb! := \mathstrut{\varMb{universe}, \varMb{op}!, \varMb{unit}}$
4 be a \symname{monoid} on $\Int$ ...

```

Output:

Let $M_2 := \langle \mathbb{Z}, \circ, e \rangle$ be a monoid on \mathbb{Z} ...

.

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

usestructure (*env.*) The `usestructure{<struct>}` environment is used in multilingual settings as a parallel to the `mathstructure`. It opens a group and then issues a `\usemodule{.../<struct>-structure}` that gives the body access to all the semantic macros in the referenced structure.

5.4.4 The copymodule Environment

TODO: explain

Given modules:

Example 29

Input:

```

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

```

Output:

.

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

Example 30

Input:

```

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

```

Output:

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

TODO: explain donotclone

5.4.5 The interpretmodule Environment

TODO: explain

Example 31

Input:

```

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

```

Output:

5.5 Primitive Symbols (The $\text{\texttt{sTeX}}$ Metatheory)

The `stex-metatheory` package contains $\text{\texttt{sTeX}}$ symbols so ubiquitous, that it is virtually impossible to describe any flexiformal content without them, or that are required to annotate even the most primitive symbols with meaningful (foundation-independent) “type”-annotations, or required for basic structuring principles (theorems, definitions). As such, it serves as the default meta theory for any $\text{\texttt{sTeX}}$ module.

We can also see the `stex-metatheory` as a foundation of mathematics in the sense of [Rab15], albeit an informal one (the ones discussed there are all formal foundations). The state of the `stex-metatheory` is necessarily incomplete, and will stay so for a long while: It arises as a collection of empirically useful symbols that are collected as more and more mathematics are encoded in $\text{\texttt{sTeX}}$ and are classified as foundational.

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

We make this theory part of the $\text{\texttt{sTeX}}$ collection due to the obiquity of the symbols involved. Note however, that the metatheory is for all practical purposes a “normal” $\text{\texttt{sTeX}}$ module, and the symbols contained “normal” $\text{\texttt{sTeX}}$ symbols.

Chapter 6

Using \TeX Symbols

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

6.1 `\symref` and its variants

<code>\symref</code> <code>\symname</code>	We have already seen <code>\symname</code> and <code>\symref</code> , the latter being the more general. <code>\symref{<symbolname>}{<code>}</code> marks-up <code><code></code> as referencing <code><symbolname></code> . Since quite often, the <code><code></code> should be (a variant of) the name of the symbol anyway, we also have <code>\symname{<symbolname>}</code> .
---	--

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

Example 32

Input:

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

Output:

A natural number is...

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

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

Example 33

Input:

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

Output:

Natural numbers are...



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

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

However, especially in `\syname` (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 `\syname{natural-number}` is... rather than A `\syname{Nat}` is.... \TeX attempts to handle this case thusly:

If `string` does *not* correspond to a semantic macro `\string` and does *not* contain a `?`, then \TeX checks all symbols currently in scope until it finds one, whose name is `string`. If `string` is of the form `pre?name`, \TeX first looks through all modules currently in scope, whose full 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 `\syname{Integers?addition}` or `\syname{RealNumbers?addition}` in the case where several `additions` are in scope.

6.2 Marking Up Text and On-the-Fly Notations

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

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

Example 34

Input:

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

Output:

The sum of n and m is...

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

\hookrightarrow As expected, the above example is translated to OMDoc/MMT as an
 \hookrightarrow OMA with `<OMS name="...?addition"/>` as head and `<OMV name="n"/>` and
 \hookrightarrow `<OMV name="m"/>` as arguments.



Note the difference in treating “arguments” between math mode and text mode. In math mode the (in this case two) tokens/groups following the `\addition` macro are treated as arguments to the addition function, whereas in text mode the group following `\addition` is taken to be the ad-hoc presentation. We drill in on this now.

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

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

Example 35

Input:

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

Output:

```
Addition is...
```

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

`\arg` also allows us to switch the order of arguments around and “hide” arguments: For example, `\arg[3]{<code>}` signifies that `<code>` represents the *third* argument to the current operator, and `\arg*[i]{<code>}` signifies that `<code>` represents the *i*th argument, but it should not produce any output (it is exported in the `xhtml` however, so that MMT and other systems can pick up on it).¹

Example 36

Input:

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

¹EDNOTE: MK: I do not understand why we have to/want to give the second `arg*`; I think this must be elaborated on.

Output:

adding k yields...

Note that since the second `\arg` has no explicit argument number, it automatically represents the first not-yet-given argument – i.e. in this case the first one.²

The same syntax can be used in math mod as well. This allows us to spontaneously introduce new notations on the fly. We can activate it using the starred variants of semantic macros:

Example 37

Input:

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

Output:

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

If we take features like `\inputref` and `\mhinput` (and the `sfragment`-environment, see [subsection 9.2.1](#)) 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 inputted 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.

The `\sref` command attempts to do precisely that. Unlike plain `\ref`, `\autoref` etc., `\sref` refers to not just a *label*, but instead a pair consisting of a *label* and the *document* in whose context we want to refer to it. Conversely, every *document* (i.e. standalone compilable `.tex`-file) keeps track of the “names” (*Definition 3.1* etc.) for every label as determined in the context of the document, and stores them in a dedicated file `\jobname.sref`. Additionally, every document has a “reference name” (e.g. “*the section on Foo*”). This allows us to refer to “label x in document D ” to yield “*Definition 1 in the section on Foo*”. And of course, \TeX can decide based on the current document

²EdNOTE: MK: I do not understand this at all.

to either refer to the label by its “full name” or directly as e.g. *Definition 3.1* depending on whether the label occurs in the current document anyway (and link to it accordingly).

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, but allows for additional disambiguation between possibly duplicate labels across files, and
- (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[archive=<archive1>,file=<file>]
{\<label>}[archive=<archive2>,in=<document-context>,title=<title>]
```

This command references *<label>* (declared in *<file>* in *<archive1>*). If the object (section, figure, etc.) with that label occurs ultimately 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 *<document-context>* in *<archive2>*.

For example, the reference to the `sfragment`-environment above will appear as “subsection 7.2.1 (Introduction) in the \TeX 3 manual” if you are reading this in the package documentation for `stex-references` directly, but as a linked “subsection 7.2.1” in the full documentation or manual. This is achieved using

```
\sref[file=stex-document-structure]{sec:ds:intro}[in=../stex-manual,title={the \sTeX}3 document]
```

For a further example, the following:

Part III

will say “Part III” (and link accordingly) in the full documentation, and “Part III (Extensions) in the full \TeX 3 documentation” everywhere else. This is achieved using

```
\sref[file=../stex-doc]{part:extends}[in=../stex-doc,title={the full \sTeX}3 document]
```

```
\extref\sref[archive=<archive1>,file=<file>]
{\<label>}[archive=<archive2>,in=<document-context>,title=<title>]}
```

The `\extref`-command behaves exactly like `\sref`, but takes *required* the document context argument and will always use it for generating the document text, regardless of whether the label occurs in the current document.

Chapter 7

STEX Statements

7.1 Definitions, Theorems, Examples, Paragraphs

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

The corresponding environments for that are:

- `sdefinition` for definitions,
- `sassertion` for assertions, i.e. propositions that are declared to be *true*, such as theorems, lemmata, axioms,
- `sexample` for examples and counterexamples, and
- `sparagraph` for “other” semantic paragraphs, such as comments, remarks, conjectures, etc.

The *presentation* of these environments can be customized to use e.g. predefined `theorem-environments`, see [section 7.3](#) for details.

All of these environments take optional arguments in the form of `key=value`-pairs. Common to all of them are the keys `id=` (for cross-referencing, see [chapter 8](#)), `type=` for customization (see [section 7.3](#)) and additional information (e.g. definition principles, “difficulty” etc), as well as `title=` (for giving the paragraph a title), and finally `for=`.

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

Example 38

Input:

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

Output:

Example 7.1.1 (An Example). $2+3$ is 5, $2\cdot 3$ is 6.

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

\hookrightarrow M \rightarrow The special `type=symdoc` for **sparagraph** is intended to be used for “informal definitions”, or encyclopedia-style descriptions for symbols.
 \hookrightarrow M \rightarrow The MMT system can use those (in lieu of an actual **sdefinition** in scope) to
 \hookrightarrow T \rightarrow present to users, e.g. when hovering over symbols.

`\definiens` Additionally, **sdefinition** (and **sparagraph** with `type=symdoc`) introduces `\definiens`[<optional sym which marks up <code> as being the explicit *definiens* of <optional symbolname> (in case `for=` has multiple symbols)].

All four statement environments – i.e. **sdefinition**, **sassertion**, **sexample**, and **sparagraph** – also take an optional parameter `name=` – if this one is given a value, the environment will generate a *symbol* by that name (but with no semantic macro). Not only does this allow for `\symref` et al, it allows us to resume our earlier example for monoids much more nicely:³

Example 39

Input:

³EdNOTE: MK: we should reference the example explicitly here.

```

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

```

Output:

A **monoid** is a structure $\langle U, \circ, e \rangle$ where $\circ : U \rightarrow U$ and $e \in U$ such that

Axiom 7.1.2 (Associativity). \circ is associative

Axiom 7.1.3 (Unit). $x \circ e = x$ for all $x \in U$

An example for a **monoid** is...

EdN:4

The main difference to before⁴ is that the two **sassertions** now have **name=** attributes. Thus the **mathstructure monoid** now contains two additional symbols, namely the axioms for associativity and that e is a unit. Note that both symbols do not represent the mere *propositions* that e.g. \circ is associative, but *the assertion that it is actually true* that \circ is associative.

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

```

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

```

⁴EdNOTE: MK: reference

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

7.2 Proofs

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

Its central component is the `sproof`-environment, whose body consists of:

- *subproofs* via the `subproof`-environment,
- *proof steps* via the `\spfstep`, `\eqstep` `\assumption`, and `\conclude` macros, and
- *comments*, via normal text without special markup.

`sproof`, `subproof` and the various proof step macros take the following optional arguments:

`id` ($\langle string \rangle$) for referencing,

`method` ($\langle string \rangle$) the proof method (e.g. contradiction, induction,...)

`term` ($\langle token list \rangle$) the (ideally semantically-marked up) proposition that is derived/proven by this proof/subproof/proof step.

Additionally, they take one mandatory argument for the document text to be annotated, or (in the case of the environments) as an introductory description of the proof itself. Since the latter often contains the `term` to be derived as text, alternatively to providing it as an optional argument, the mandatory argument can use the `\yield`-macro to mark it up in the text.

The `sproof` and `subproof` environments additionally take two optional arguments:

`for` the symbol identifier/name corresponding to the `sassertion` to be proven. This too subsumes `\yield` and the `term`-argument.

`hide` In the pdf, this only shows the mandatory argument text and hides the body of the environment. In the HTML (as served by MMT), the bodies of all `proof` and `subproof` environments are *collapsible*, and `hide` collapses the body by default.

```

1 \begin{sassertion}[type=theorem,name=sqrt2irr]
2   \conclusion{\irrational{\arg{\realroot{2}}$ is \comp{irrational}}}.
3 \end{sassertion}
4
5 \begin{sproof}[for=sqrt2irr,method=contradiction]{By contradiction}
6   \assumption{Assume \yield{\rational{\arg{\realroot{2}}$ is
7     \comp{rational}}}}
8   \begin{subproof}[method=straightforward]{Then
9     \yield{\$eq{\ratfrac{\intpow{\vara}{2}}{\intpow{\varb}{2}}{2}$
10       for some $\inset{\vara,\varb}\PosInt$ with
11       \coprime{\arg{\vara},\arg{\varb}$ \comp{coprime}}}}
```

²Of course, \LaTeX can not check that the assertions are the “correct” ones – but if the assertions (both in monoid as well as those for addition and zero) are properly marked up, MMT can. **TODO: should**

```

12 \assumption{By assumption, \yield{there are
13 $\inset{\vara,\varb}\PosInt$ with
14 $\realroot{2}=\ratfrac{\vara}{\varb}$}}
15 \spfstep{wlog, we can assume \coprime{$\arg{\vara},\arg{\varb}$}
16 to be \comp{coprime}}
17 % a comment:
18 If not, reduce the fraction until numerator and denominator
19 are coprime, and let the resulting components be
20 $\vara$ and $\varb$
21 \spfstep{Then \yield{$\eq{\intpow{\ratfrac{\vara}{\varb}}{2}{2}$}}
22 \eqstep{\ratfrac{\intpow{\vara}{2}}{\intpow{\varb}{2}}}}
23 \end{subproof}
24 \begin{subproof}[term=\divides{2}{\vara},method=straightforward]{
25 Then $\vara$ is even}
26 \spfstep{Multiplying the equation by $\intpow{\varb}{2}$ yields
27 $\yield{\eq{\intpow{\vara}{2}{\inttimes{2}{\intpow{\varb}{2}}}}}$}
28 \spfstep[term=\divides{2}{\intpow{\vara}{2}}]{Hence
29 $\intpow{\vara}{2}$ is even}
30 \conclude[term=\divides{2}{\vara}]{Hence $\vara$ is even as well}
31 % another comment:
32 Hint: Think about the prime factorizations of $\vara$ and
33 $\intpow{\vara}{2}$
34 \end{subproof}
35 \begin{subproof}[term=\divides{2}{\varb},method=straightforward,]{
36 Then $\varb$ is also even}
37 \spfstep{Since $\vara$ is even, we have \yield{some $\varc$ $
38 such that $\eq{\inttimes{2}{\varc}}{\vara}$}}
39 \spfstep{Plugging into the above, we get
40 \yield{$\eq{\intpow{\inttimes{2}{\vara}}{2}
41 {\inttimes{2}{\intpow{\varb}{2}}}$}}
42 \eqstep{\inttimes{4}{\intpow{\vara}{2}}}
43 \spfstep{Dividing both sides by $2$ yields
44 \yield{$\eq{\intpow{\varb}{2}{\inttimes{2}{\intpow{\vara}{2}}}$}}
45 \spfstep[term=\divides{2}{\intpow{\varb}{2}}]{Hence
46 $\intpow{\varb}{2}$ is even}
47 \conclude[term=\divides{2}{\varb}]{Hence $\varb$ is even}
48 % one more comment:
49 By the same argument as above
50 \end{subproof}
51 \conclude[term=\contradiction]{Contradiction to $\vara,\varb$ being
52 \symname{coprime}.}
53 \end{spproof}

```

which will produce:

Theorem 7.2.1. $\sqrt{2}$ is *irrational*.

Proof: By contradiction

1. Assume $\sqrt{2}$ is *rational*
2. Then $(\frac{a}{b})^2=2$ for some $a,b \in \mathbb{Z}^+$ with a,b *coprime*
 - 2.1. By assumption, there are $a,b \in \mathbb{Z}^+$ with $\sqrt{2} = \frac{a}{b}$
 - 2.2. wlog, we can assume a,b to be *coprime*

If not, reduce the fraction until numerator and denominator are coprime, and let the re-

ulting components be a and b

2.3. Then $(\frac{a}{b})^2=2$

$$= \frac{a^2}{b^2}$$

3. Then a is even

3.1. Multiplying the equation by b^2 yields $a^2=2b^2$

3.2. Hence a^2 is even

\Rightarrow Hence a is even as well

Hint: Think about the prime factorizations of a and a^2

4. Then b is also even

4.1. Since a is even, we have some c such that $2c=a$

4.2. Plugging into the above, we get $(2a)^2=2b^2$

$$= 4a^2$$

4.3. Dividing both sides by 2 yields $b^2=2a^2$

4.4. Hence b^2 is even

\Rightarrow Hence b is even

By the same argument as above

\Rightarrow Contradiction to a, b being coprime.

□

If we mark all subproofs with `hide`, we will obtain the following instead:

Theorem 7.2.2. $\sqrt{2}$ is irrational.

Proof: By contradiction

1. Assume $\sqrt{2}$ is rational

2. Then $(\frac{a}{b})^2=2$ for some $a, b \in \mathbb{Z}^+$ with a, b coprime

3. Then a is even

4. Then b is also even

\Rightarrow Contradiction to a, b being coprime.

□

However, the hidden subproofs will still be shown in the HTML, only in an expandable section which is collapsed by default.

The above style of writing proofs is usually called *structured proofs*. They have a huge advantage over the traditional purely prosaic style, in that (as the name suggests) the actual *structure* of the proof is made explicit, which almost always makes it considerably more comprehensible. We, among many others, encourage the general use of structured proofs.

Alas, most proofs are not written in this style, and we would do users a disservice by insisting on this style. For that reason, the `spfblock` environment turns all subproofs and proof step macros into presentationally neutral *inline* annotations, as in the induction step of the following example:

```
1 \begin{sproof}[id=simple-proof,method=induction]
2   {We prove that  $\sum_{i=1}^n 2i-1=n^2$  by induction over  $n$ }
```

```

3 For the induction we have to consider three cases: % <- a comment
4 \begin{subproof}{ $n=1$ }
5   \spfstep*{then we compute  $1=1^2$ }
6 \end{subproof}
7 \begin{subproof}{ $n=2$ }
8   This case is not really necessary, but we do it for the
9   fun of it (and to get more intuition).
10  \spfstep*{We compute  $1+3=2^2=4$ .}
11 \end{subproof}
12 \begin{subproof}{ $n>1$ }\begin{spfblock}
13   \assumption[id=ind-hyp]{
14     Now, we assume that the assertion is true for a certain  $k \geq 1$ ,
15     i.e.  $\mathsf{yield}\{\sum_{i=1}^k (2i-1) = k^2\}$ .
16   }
17
18   We have to show that we can derive the assertion for  $n=k+1$  from
19   this assumption, i.e.  $\sum_{i=1}^{k+1} (2i-1) = (k+1)^2$ .
20
21   \spfstep{
22     We obtain  $\mathsf{yield}\{\sum_{i=1}^{k+1} (2i-1) =$ 
23      $\sum_{i=1}^k (2i-1) + 2(k+1) - 1\}$ 
24     \spfjust{by \splitsum{\comp{splitting the sum}}
25     \arg*{\mathsf{yield}\{\sum_{i=1}^{k+1} (2i-1) = (k+1)^2\}}}.
26   }
27   \spfstep{
28     Thus we have  $\mathsf{yield}\{\sum_{i=1}^{k+1} (2i-1) = k^2 + 2k + 1\}$ 
29     \spfjust{by \symname{induction-hypothesis}}.
30   }
31   \conclude{
32     We can \spfjust{\simplification{\comp{simplify} the right-hand side
33     \arg*{ $k^2 + 2k + 1$ }} to
34      $(k+1)^2$ , which proves the assertion.
35   }
36 \end{subproof}\end{spfblock}
37 \conclude{
38   We have considered all the cases, so we have proven the assertion.
39 }
40 \end{spproof}

```

This yields the following result:

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

For the induction we have to consider three cases:

1. $n = 1$

then we compute $1 = 1^2$

2. $n = 2$

This case is not really necessary, but we do it for the fun of it (and to get more intuition).

We compute $1 + 3 = 2^2 = 4$.

3. $n > 1$

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

We have to show that we can derive the assertion for $n = k + 1$ from this assumption,

i.e. $\sum_{i=1}^{k+1} (2i - 1) = (k + 1)^2$.
 We obtain $\sum_{i=1}^{k+1} 2i - 1 = \sum_{i=1}^k 2i - 1 + 2(k + 1) - 1$ by [splitting the sum](#). Thus
 we have $\sum_{i=1}^{k+1} (2i - 1) = k^2 + 2k + 1$ by [induction hypothesis](#). We can [simplify](#) the
 right-hand side to $k + 1^2$, which proves the assertion.
 \Rightarrow We have considered all the cases, so we have proven the assertion. □

sproof (*env.*) The **sproof** environment is the main container for proofs. It takes an optional **KeyVal** argument that allows to specify the **id** (identifier) and **for** (for which assertion is this a proof) keys. The regular argument of the **proof** environment contains an introductory comment, that may be used to announce the proof style. The **proof** environment contains a sequence of **spfstep**, **spfcomment**, and **spfcases** environments that are used to markup the proof steps.

\spfilea The **\spfilea** macro allows to give a one-paragraph description of the proof idea.

\spfsketch For one-line proof sketches, we use the **\spfsketch** macro, which takes the same optional argument as **sproof** and another one: a natural language text that sketches the proof.

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

\yield See above

\spfjust This evidence is marked up with the **\spfjust** macro in the **stex-proofs** package. This environment totally invisible to the formatted result; it wraps the text in the proof step that corresponds to the evidence (ideally, a semantically marked-up term).

\assumption The **\assumption** macro allows to mark up a (justified) assumption.

\justarg

subproof (*env.*) The **subproof** environment is used to mark up a subproof. This environment takes an optional **KeyVal** argument for semantic annotations and a second argument that allows to specify an introductory comment (just like in the **proof** environment). The **method** key can be used to give the name of the proof method executed to make this subproof.

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

The `stex-proofs` package provides the `\sproofend` macro for this.

`\sProofEndSymbol` If a different symbol for the proof end is to be used (e.g. *q.e.d*), then this can be obtained by specifying it using the `\sProofEndSymbol` configuration macro (e.g. by specifying `\sProofEndSymbol{q.e.d}`).

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

7.3 Highlighting and Presentation Customizations

The environments starting with `s` (i.e. `smodule`, `sassertion`, `sexample`, `sdefinition`, `sparagraph` and `sproof`) by default produce no additional output whatsoever (except for the environment content of course). Instead, the document that uses them (whether directly or e.g. via `\inputref`) can decide how these environments are supposed to look like.

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

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

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

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

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

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

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

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

```

1 \stexpatchassertion[theorem]
2   {\ifx\sassertiontitle\@empty
3     \begin{theorem}
4   \else
5     \begin{theorem}[\sassertiontitle]
6   \fi}
7 {\end{theorem}}

```

Or, if we want *all kinds of sdefinitions* to use a predefined definition-environment irrespective of their type=, then we can issue the following customization patch:

```

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

```

<hr/>	
<code>\compemph</code>	Apart from the environments, we can control how \TeX highlights variables, notation
<code>\varemp</code>	components, <code>\symrefs</code> and <code>\definiendums</code> , respectively.
<code>\symrefemph</code>	To do so, we simply redefine these four macros. For example, to highlight nota-
<code>\defemph</code>	tion components (i.e. everything in a <code>\comp</code>) in blue, as in this document, we can do
	<code>\def\compemph#1{\textcolor{blue}{#1}}</code> . By default, <code>\compemph</code> et al do nothing.

<hr/>	
<code>\compemph@uri</code>	For each of the four macros, there exists an additional macro that takes the full URI of
<code>\varemp@uri</code>	the relevant symbol currently being highlighted as a second argument. That allows us to
<code>\symrefemph@uri</code>	e.g. use pdf tooltips and links. For example, this document uses ⁵
<code>\defemph@uri</code>	
	<pre> 1 \protected\def\symrefemph@uri#1#2{ 2 \pdftooltip{ 3 \symrefemph{#1} 4 }{ 5 URI:~\detokenize{#2} 6 } 7 } </pre>

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

Chapter 8

Cross References

If we take features like `\inputref` and `\mhinput` (and the `sfragment`-environment, see [subsection 9.2.1](#)) 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 inputted 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.

The `\sref` command attempts to do precisely that. Unlike plain `\ref`, `\autoref` etc., `\sref` refers to not just a *label*, but instead a pair consisting of a *label* and the *document* in whose context we want to refer to it. Conversely, every *document* (i.e. standalone compilable `.tex`-file) keeps track of the “names” (*Definition 3.1* etc.) for every label as determined in the context of the document, and stores them in a dedicated file `\jobname.sref`. Additionally, every document has a “reference name” (e.g. “*the section on Foo*”). This allows us to refer to “label *x* in document *D*” to yield “*Definition 1 in the section on Foo*”. And of course, `TEX` can decide based on the current document to either refer to the label by its “full name” or directly as e.g. *Definition 3.1* depending on whether the label occurs in the current document anyway (and link to it accordingly).

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, but allows for additional disambiguation between possibly duplicate labels across files, and
- (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=<file>]
{<label>}[archive=<archive2>,in=<document-context>,title=<title>]`

This command references *<label>* (declared in *<file>* in *<archive1>*). If the object (section, figure, etc.) with that label occurs ultimately 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 *<document-context>* in *<archive2>*.

For example, the reference to the `sfragment`-environment above will appear as “subsection 7.2.1 (Introduction) in the `gTeX3` manual” if you are reading this in the package documentation for `stex-references` directly, but as a linked “subsection 7.2.1” in the full documentation or manual. This is achieved using

```
\sref[file=stex-document-structure]{sec:ds:intro}[in=./stex-manual,title={the \sTeX}]
```

For a further example, the following:

Part III

will say “Part III” (and link accordingly) in the full documentation, and “Part III (Extensions) in the full `gTeX3` documentation” everywhere else. This is achieved using

```
\sref[file=./stex-doc]{part:extends}[in=./stex-doc,title={the full \sTeX}3 document]
```

\extref `\sref[archive=<archive1>,file=<file>]
{<label>}[archive=<archive2>,in=<document-context>,title=<title>]}`

The `\extref`-command behaves exactly like `\sref`, but takes *required* the document context argument and will always use it for generating the document text, regardless of whether the label occurs in the current document.

Chapter 9

Additional Packages

9.1 Tikzinput: Treating TIKZ code as images

image The behavior of the `tikzinput` 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}
3 \usetikzpackage{...}
4 \begin{document}
5   \begin{tikzpicture}
6     ...
7   \end{tikzpicture}
8 \end{document}
```

The `standalone` class is a minimal L^AT_EX class that when loaded in a document that uses the `standalone` package: the preamble and the `document` 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 L^AT_EX over it separately, e.g. for generating an image file from it.

\tikzinput 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>}`.

`\ctikzinput` is a version of `\tikzinput` that is centered.

<code>\mhtikzinput</code>	<code>\mhtizkinput</code> is a variant of <code>\tikzinput</code> that treats its file path argument as a relative path in a math archive in analogy to <code>\inputref</code> . To give the archive path, we use the <code>mhrepos=</code> key. Again, <code>\cmhtizkinput</code> is a version of <code>\mhtikzinput</code> that is centered.
<code>\cmhtikzinput</code>	

<code>\libusetikzlibrary</code>	Sometimes, we want to supply archive-specific TIKZ libraries in the <code>lib</code> folder of the archive or the <code>meta-inf/lib</code> of the archive group. Then we need an analogon to <code>\libinput</code> for <code>\usetikzlibrary</code> . The <code>stex-tikzinput</code> package provides the <code>libusetikzlibrary</code> for this purpose.
---------------------------------	---

9.2 Modular Document Structuring

9.2.1 Introduction

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

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

DAG models of documents allow to replace the “Copy and Paste” in the source document with a label-and-reference model where document are shared in the document source and the formatter does the copying during document formatting/presentation.

9.2.2 Package Options

The `document-structure` package accepts the following options:

<code>class=<name></code>	load <code><name>.cls</code> instead of <code>article.cls</code>
<code>topsect=<sect></code>	The top-level sectioning level; the default for <code><sect></code> is <code>section</code>

9.2.3 Document Fragments

`sfragment` (*env.*) The structure of the document is given by nested `sfragment` environments. In the \LaTeX route, the `sfragment` environment is flexibly mapped to sectioning commands, inducing the proper sectioning level from the nesting of `sfragment` environments. Correspondingly, the `sfragment` environment takes an optional key/value argument for metadata followed by a regular argument for the (section) title of the sfragment. The optional metadata argument has the keys `id` for an identifier, `creators` and `contributors` for the Dublin Core metadata [DCM03]. The option `short` allows to give a short title for the generated section. If the title contains semantic macros, we need to give the `loadmodules` key (it needs no value). For instance we would have

```

1 \begin{smodule}{foo}
2   \symdef{bar}{Ba_r}
3   ...
4   \begin{sfragment}[id=sec.bardriv,loadmodules]
5     {Introducing $\protect\bar$ Derivations}

```

TeX automatically computes the sectioning level, from the nesting of `sfragment` environments.

But sometimes, we want to skip levels (e.g. to use a `\subsection*` as an introduction for a chapter).

`blindfragment` (*env.*) Therefore the `document-structure` package provides a variant `blindfragment` that does not produce markup, but increments the sectioning level and logically groups document parts that belong together, but where traditional document markup relies on convention rather than explicit markup. The `blindfragment` environment is useful e.g. for creating frontmatter at the correct level. The example below shows a typical setup for the outer document structure of a book with parts and chapters.

```

1 \begin{document}
2 \begin{blindfragment}
3 \begin{blindfragment}
4 \begin{frontmatter}
5 \maketitle\newpage
6 \begin{sfragment}{Preface}
7 ... <<preface>> ...
8 \end{sfragment}
9 \clearpage\setcounter{tocdepth}{4}\tableofcontents\clearpage
10 \end{frontmatter}
11 \end{blindfragment}
12 ... <<introductory remarks>> ...
13 \end{blindfragment}
14 \begin{sfragment}{Introduction}
15 ... <<intro>> ...
16 \end{sfragment}
17 ... <<more chapters>> ...
18 \bibliographystyle{alpha}\bibliography{kwarc}
19 \end{document}

```

Here we use two levels of `blindfragment`:

- The outer one groups the introductory parts of the book (which we assume to have a sectioning hierarchy topping at the part level). This `blindfragment` makes sure that the introductory remarks become a “chapter” instead of a “part”.
- The inner one groups the frontmatter³ and makes the preface of the book a section-level construct. The `frontmatter` environment also suppresses numbering as is traditional for prefaces.

`\skipfragment` The `\skipfragment` “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`.

³We shied away from redefining the `frontmatter` to induce a `blindfragment`, but this may be the “right” way to go in the future.

<hr/> <code>\currentsectionlevel</code> <hr/>	The <code>\currentsectionlevel</code> macro supplies the name of the current sectioning level, e.g. “chapter”, or “subsection”. <code>\CurrentSectionLevel</code> is the capitalized variant. They are useful to write something like “In this <code>\currentsectionlevel</code> , we will...” in an <code>sfragment</code> environment, where we do not know which sectioning level we will end up.
<code>\CurrentSectionLevel</code> <hr/>	

9.2.4 Ending Documents Prematurely

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

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

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 \TeX preamble of the course notes file.

9.2.5 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.

<hr/> <code>\setSGvar</code> <hr/>	<code>\setSGvar{<vname>}{<text>}</code> to set the global variable <code><vname></code> to <code><text></code> and <code>\useSGvar{<vname>}</code> to reference it.
<code>\useSGvar</code> <hr/>	

<hr/> <code>\ifSGvar</code> <hr/>	With <code>\ifSGvar</code> we can test for the contents of a global variable: the macro call <code>\ifSGvar{<vname>}{<val>}{<ctext>}</code> tests the content of the global variable <code><vname></code> , only if (after expansion) it is equal to <code><val></code> , the conditional text <code><ctext></code> is formatted.
-----------------------------------	---

9.3 Slides and Course Notes

9.3.1 Introduction

The `notesslides` document class is derived from `beamer.cls` [Tana], 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 \TeX 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.

9.3.2 Package Options

The `notesslides` class takes a variety of class options:

<code>slides</code>	The options <code>slides</code> and <code>notes</code> switch between slides mode and notes mode (see subsection 9.3.3).
<code>notes</code>	
<code>sectocframes</code>	If the option <code>sectocframes</code> is given, then for the <code>sfragments</code> , special frames with the <code>sfragment</code> title (and number) are generated.
<code>frameimages</code>	If the option <code>frameimages</code> is set, then slide mode also shows the <code>\frameimage</code> -generated frames (see ??). If also the <code>fiboxed</code> option is given, the slides are surrounded by a box.
<code>fiboxed</code>	

9.3.3 Notes and Slides

`frame` (*env.*) Slides are represented with the `frame` environment just like in the `beamer` class, see [\[Tanb\]](#) 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 L^AT_EX 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 `\notesttrue` or `\notestfalse` 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

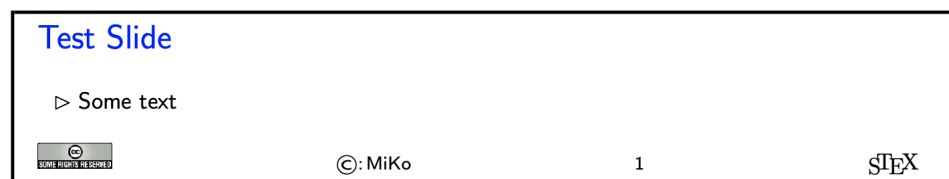
`nexample` (*env.*) level of source indenting). Similarly, we have the `nfragment`, `ndefinition`, `nexample`,

`nsproof` (*env.*) `nsproof`, and `nassertion` environments.

`nassertion` (*env.*)

9.3.4 Customizing Header and Footer Lines

The `notesslides` package and class comes with a simple default theme named `sTeX` that provided by the `beamerthemesTeX`. 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 height is variable) is



The footer line can be customized. In particular the logos.

\setslidelogo The default logo provided by the `notesslides` package is the \TeX 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 Attribution-ShareAlike license by default to strengthen the public domain. If package `hyperref` is loaded, then we can attach a hyperlink to the license logo. `\setlicensing[<url>]{<logo name>}` is used for customization, where `<url>` is optional.

9.3.5 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 \TeX notes.

\frameimage In this case we can use `\frameimage[<opt>]{<path>}`, where `<opt>` are the options of `\includegraphics` from the `graphicx` package [CR99] and `<path>` is the file path (extension can be left off like in `\includegraphics`). We have added the `label` key that allows to give a frame label that can be referenced like a regular `beamer` frame.

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: 

9.3.6 Excursions

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

```
1 \excursion{founif}{../fragments/founif.en}
2 {We will cover first-order unification in}
3 ...
4 \begin{appendix}\printexcursions\end{appendix}
```

It generates a paragraph that references the excursion whose source is in the file `../fragments/founif.en.tex` and automatically books the file for the `\printexcursions` command that is used here to put it into the appendix. We will look at the mechanics now.

`\excursion` The `\excursion{<ref>}{<path>}{<text>}` is syntactic sugar for

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

`\activateexcursion` Here `\activateexcursion{<path>}` augments the `\printexcursions` macro by a call
`\printexcursion` `\inputref{<path>}`. In this way, the `\printexcursions` macro (usually in the appendix)
`\excursionref` 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

```
1 \begin{note}
2 \begin{sfragment}[id=<id>]{Excursions}
3   \inputref{<path>}
4   \printexcursions
5 \end{sfragment}
6 \end{note}
```



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

9.4 Representing Problems and Solutions

9.4.1 Introduction

The `problem` package supplies an infrastructure that allows specify problem. Problems are text fragments that come with auxiliary functions: `hints`, `notes`, and `solutions`⁴. 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.

9.4.2 Problems and Solutions

solutions	The <code>problem</code> package takes the options <code>solutions</code> (should solutions be output?), <code>notes</code>
notes	(should the problem notes be presented?), <code>hints</code> (do we give the hints?), <code>gnotes</code> (do we
hints	show grading notes?), <code>pts</code> (do we display the points awarded for solving the problem?),
gnotes	<code>min</code> (do we display the estimated minutes for problem soling). If theses are specified, then
pts	the corresponding auxiliary parts of the problems are output, otherwise, they remain
min	invisible.
boxed	The <code>boxed</code> option specifies that problems should be formatted in framed boxes so
test	that they are more visible in the text. Finally, the <code>test</code> 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 40

Input:

⁴for the moment multiple choice problems are not supported, but may well be in a future version

```

1 \documentclass{article}
2 \usepackage[solutions,hints,pts,min]{problem}
3 \begin{document}
4   \begin{sproblem}[id=elephants,pts=10,min=2,title=Fitting Elephants]
5     How many Elephants can you fit into a Volkswagen beetle?
6     \begin{hint}
7       Think positively, this is simple!
8     \end{hint}
9     \begin{exnote}
10      Justify your answer
11    \end{exnote}
12  \begin{solution}[for=elephants]
13    Four, two in the front seats, and two in the back.
14    \begin{gnote}
15      if they do not give the justification deduct 5 pts
16    \end{gnote}
17  \end{solution}
18 \end{sproblem}
19 \end{document}

```

Output:

Problem 9.4.1 (Fitting Elephants)
 How many Elephants can you fit into a Volkswagen beetle?

Hint: Think positively, this is simple!

Note: Justify your answer

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

Grading: if they do not give the justification deduct 5 pts

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

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

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

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

9.4.3 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 meta-data 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 41

Input:

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

Output:

Problem 9.4.2 (Functions)

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

☐ def

Correct!

☐ function

Wrong! *that is for C and C++*

☐ fun

Wrong! *that is for Standard ML*

☐ public static void

Wrong! *that is for Java*

In “exam mode” where disable solutions (here via \stopsolutions)

Example 42

Input:

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

Output:

Problem 9.4.3 (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 takes⁶ an a single argument, which contains a concrete solution (i.e. a number, a string, ...), which generates a fill-in-box in test mode:

Example 43

Input:

```
1 \stopsolutions
2 \begin{problem}[id=elephants.fillin,title=Fitting Elephants]
3   How many Elephants can you fit into a Volkswagen beetle? \fillinsol{4}
4 \end{problem}
```

Output:

Problem 9.4.4 (Fitting Elephants)

How many Elephants can you fit into a Volkswagen beetle?
and the actual solution in solutions mode:

Example 44

Input:

```
1 \begin{problem}[id=elephants.fillin,title=Fitting Elephants]
2   How many Elephants can you fit into a Volkswagen beetle? \fillinsol{4}
3 \end{problem}
```

Output:

Problem 9.4.5 (Fitting Elephants)

How many Elephants 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.⁷

9.4.4 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

⁷EDNOTE: 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.

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.

9.4.5 Testing and Spacing

The `problem` package is often used by the `hwexam` package, which is used to create homework assignments and exams. Both of these have a “test mode” (invoked by the package option `test`), where certain information –master solutions or feedback – is not shown in the presentation.

`\testspace` `\testspace` takes an argument that expands to a dimension, and leaves vertical space accordingly. Specific instances exist: `\testsmallspace`, `\testsmallspace`, `\testsmallspace` `\testsmallspace` give small (1cm), medium (2cm), and big (3cm) vertical space.
`\testsmallspace` `\testnewpage` makes a new page in `test` mode, and `\testemptypage` generates an empty page with the cautionary message that this page was intentionally left empty.
`\testnewpage`
`\testemptypage`

9.5 Homeworks, Quizzes and Exams

9.5.1 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.

9.5.2 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.

9.5.3 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 **title** — the ordinal of the **assignment** environment), **title** (for the assignment title; this is **type** referenced in the title of the assignment sheet), **type** (for the assignment type; e.g. “quiz”, **given** or “homework”), **given** (for the date the assignment was given), and **due** (for the date the assignment is due).

9.5.4 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.

9.5.5 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.

```
reqpts 1 \title{320101 General Computer Science (Fall 2010)}
        2 \begin{testheading}[duration=one hour,min=60,reqpts=27]
        3   Good luck to all students!
        4 \end{testheading}
```

Will result in

Name:

Matriculation Number:

320101 General Computer Science (Fall 2010)

2022-09-12

You have one hour (sharp) for the test;

Write the solutions to the sheet.

The estimated time for solving this exam is 60 minutes, leaving you 0 minutes for revising your exam.

You can reach 40 points if you solve all problems. You will only need 27 points for a perfect score, i.e. 13 points are bonus points.

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

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

	To be used for grading, do not write here													
prob.	9.4.1	9.4.2	9.4.3	9.4.4	9.4.5	1.1	2.1	2.2	2.3	3.1	3.2	3.3	Sum	grade
total	10					4	4	6	6	4	4	2	40	
reached														

good luck

EdN:8

8

⁸EDNOTE: MK: The first three “problems” come from the stex examples above, how do we get rid of this?

Part II
Documentation

Chapter 10

STEX-Basics

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

10.1 Macros and Environments

<code>\sTeX</code>	Both print this ST _E X logo.
<code>\stex</code>	

<code>\stex_debug:nn</code>	<code>\stex_debug:nn {<log-prefix>} {<message>}</code>
-----------------------------	--

Logs *<message>*, if the package option `debug` contains *<log-prefix>*.

10.1.1 HTML Annotations

<code>\if@latexml</code>	L ^A T _E X2e conditional for L ^A T _E XML
--------------------------	---

<code>\latexml_if_p: *</code>	L ^A T _E X3 conditionals for L ^A T _E XML.
<code>\latexml_if:TF *</code>	

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

<code>\stex_suppress_html:n</code>	Temporarily disables HTML annotations in its argument code
------------------------------------	--

We have four macros for annotating generated HTML (via L^AT_EXML or R_US_TE_X) with attributes:

<code>\stex_annotate:nnn</code>	<code>\stex_annotate:nnn {⟨property⟩} {⟨resource⟩} {⟨content⟩}</code>
<code>\stex_annotate_invisible:nnn</code>	
<code>\stex_annotate_invisible:n</code>	

Annotates the HTML generated by `⟨content⟩` with

`property="stex:⟨property⟩", resource="⟨resource⟩".`

`\stex_annotate_invisible:n` adds the attributes

`stex:visible="false", style="display:none".`

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

<code>stex_annotate_env (env.)</code>	<code>\begin{stex_annotate_env}{⟨property⟩}{⟨resource⟩}</code> <code>⟨content⟩</code> <code>\end{stex_annotate_env}</code> behaves like <code>\stex_annotate:nnn {⟨property⟩} {⟨resource⟩} {⟨content⟩}</code> .
---------------------------------------	--

10.1.2 Babel Languages

<code>\c_stex_languages_prop</code>
<code>\c_stex_language_abbrevs_prop</code>

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

10.1.3 Auxiliary Methods

<code>\stex_deactivate_macro:Nn</code>	<code>\stex_deactivate_macro:Nn⟨cs⟩{⟨environments⟩}</code>
<code>\stex_reactivate_macro:N</code>	

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

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

<code>\ignorespacesandpars</code>	ignores white space characters and <code>\par</code> control sequences. Expands tokens in the process.
-----------------------------------	--

Chapter 11

STEX-MathHub

This sub package provides code for handling ST_EX archives, files, file paths and related methods.

11.1 Macros and Environments

<code>\stex_kpsewhich:n</code>	<code>\stex_kpsewhich:n</code> executes <code>kpsewhich</code> and stores the return in <code>\l_stex_kpsewhich_return_str</code> . This does not require shell escaping.
--------------------------------	---

11.1.1 Files, Paths, URIs

<code>\stex_path_from_string:Nn</code>	<code>\stex_path_from_string:Nn</code> $\langle path-variable \rangle$ $\{\langle string \rangle\}$ turns the $\langle string \rangle$ into a path by splitting it at <code>/</code> -characters and stores the result in $\langle path-variable \rangle$. Also applies <code>\stex_path_canonicalize:N</code> .
--	--

<code>\stex_path_to_string:NN</code>	The inverse; turns a path into a string and stores it in the second argument variable, or leaves it in the input stream.
<code>\stex_path_to_string:N</code>	

<code>\stex_path_canonicalize:N</code>	Canonicalizes the path provided; in particular, resolves <code>.</code> and <code>..</code> path segments.
--	--

<code>\stex_path_if_absolute_p:N</code> *	Checks whether the path provided is <i>absolute</i> , i.e. starts with an empty segment
<code>\stex_path_if_absolute:N\underline{T}</code> *	

<code>\c_stex_pwd_seq</code>	Store the current working directory as path-sequence and string, respectively, and the (heuristically guessed) full path to the main file, based on the PWD and <code>\jobname</code> .
<code>\c_stex_pwd_str</code>	
<code>\c_stex_mainfile_seq</code>	
<code>\c_stex_mainfile_str</code>	

<code>\g_stex_currentfile_seq</code>	The file being currently processed (respecting <code>\input</code> etc.)
--------------------------------------	--

<code>\stex_filestack_push:n</code>	Push and pop (repectively) a file path to the file stack, to keep track of the current file.
<code>\stex_filestack_pop:</code>	Are called in hooks <code>file/before</code> and <code>file/after</code> , respectively.

11.1.2 MathHub Archives

<code>\mathhub</code>	We determine the path to the local MathHub folder via one of four means, in order of
<code>\c_stex_mathhub_seq</code>	precedence:
<code>\c_stex_mathhub_str</code>	

1. The `mathhub` package option, or
2. the `\mathhub-macro`, if it has been defined before the `\usepackage{stex}`-statement, or
3. the `MATHHUB` system variable, or
4. a path specified in `~/.stex/mathhub.path`.

In all four cases, `\c_stex_mathhub_seq` and `\c_stex_mathhub_str` are set accordingly.

<code>\l_stex_current_repository_prop</code>
--

Always points to the *current* MathHub repository (if we currently are in one). Has the following fields corresponding to the entries in the `MANIFEST.MF`-file:

- `id`: The name of the archive, including its group (e.g. `smglom/calculus`),
- `ns`: The content namespace (for modules and symbols),
- `narr`: the narration namespace (for document references),
- `docurl`: The URL that is used as a basis for *external references*,
- `deps`: All archives that this archive depends on (currently not in use).

<code>\stex_set_current_repository:n</code>

Sets the current repository to the one with the provided ID. calls `__stex_mathhub_do_manifest:n`, so works whether this repository's `MANIFEST.MF`-file has already been read or not.

<code>\stex_require_repository:n</code>	Calls <code>__stex_mathhub_do_manifest:n</code> iff the corresponding archive property list does not already exist, and adds a corresponding definition to the <code>.sms</code> -file.
---	--

<code>\stex_in_repository:nn</code>	<code>\stex_in_repository:nn{<i>repository-name</i>}{<i>code</i>}</code>
-------------------------------------	--

Change the current repository to `{repository-name}` (or not, if `{repository-name}` is empty), and passes its ID on to `{code}` as `#1`. Switches back to the previous repository after executing `{code}`.

11.1.3 Using Content in Archives

<hr/> <code>\mhpath</code> *	<code>\mhpath{<archive-ID>}{<filename>}</code>
	Expands to the full path of file <code><filename></code> in repository <code><archive-ID></code> . Does not check whether the file or the repository exist.
<hr/> <code>\inputref</code> <hr/> <code>\mhinput</code>	<code>\inputref[<archive-ID>]{<filename>}</code> Both <code>\input</code> the file <code><filename></code> in archive <code><archive-ID></code> (relative to the <code>source-</code> subdirectory). <code>\mhinput</code> does so directly. <code>\inputref</code> does so within an <code>\begingroup... \endgroup-</code> block, and skips it in <code>html-mode</code> , inserting a <i>reference</i> to the file instead. Both also set <code>\ifinputref</code> to true.
<hr/> <code>\addmhbibresource</code>	<code>\inputref[<archive-ID>]{<filename>}</code> Adds a <code>.bib</code> -file <code><filename></code> in archive <code><archive-ID></code> (relative to the top-directory of the archive!).
<hr/> <code>\libinput</code>	<code>\libinput{<filename>}</code> Inputs <code><filename>.tex</code> from the <code>lib</code> folders in the current archive and the <code>meta-inf-</code> archive of the current archive group(s) (if existent) in descending order. Throws an error if no file by that name exists in any of the relevant <code>lib</code> -folders.
<hr/> <code>\libusepackage</code>	<code>\libusepackage[<args>]{<filename>}</code> Like <code>\libinput</code> , but looks for <code>.sty</code> -files and calls <code>\usepackage[<meta{args}>]{<Arg{filename}>}</code> instead of <code>\input</code> . Throws an error, if none or more than one suitable package file is found.
<hr/> <code>\mhgraphics</code> <hr/> <code>\cmhgraphics</code>	<i>If</i> the <code>graphicx</code> package is loaded, these macros are defined at <code>\begin{document}</code> . <code>\mhgraphics</code> takes the same arguments as <code>\includegraphics</code> , with the additional optional key <code>mhrepos</code> . It then resolves the file path in <code>\mhgraphics[mhrepos=Foo/Bar]{foo/bar.png}</code> relative to the <code>source</code> -folder of the <code>Foo/Bar</code> -archive. <code>\cmhgraphics</code> additional wraps the image in a <code>center</code> -environment.
<hr/> <code>\lstinputmhlisting</code> <hr/> <code>\clstinputmhlisting</code>	Like <code>\mhgraphics</code> , but only defined if the <code>listings</code> -package is loaded, and with <code>\lstinputlisting</code> instead of <code>\includegraphics</code> .

Chapter 12

STEX-References

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

12.1 Macros and Environments

<code>\stex_get_document_uri:</code>	Computes the current document uri from the current archive's narr -field and its location relative to the archive's source -directory. Reference targets are computed from this URI and the reference-id.
--------------------------------------	---

<code>\l_stex_current_docns_str</code>	Stores its result in <code>\l_stex_current_docns_str</code>
--	---

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

<code>\l_stex_current_docurl_str</code>	Stores its result in <code>\l_stex_current_docurl_str</code>
---	--

12.1.1 Setting Reference Targets

<code>\stex_ref_new_doc_target:n</code>	<code>\stex_ref_new_doc_target:n{<id>}</code> Sets a new reference target with id <code><id></code> .
---	--

<code>\stex_ref_new_sym_target:n</code>	<code>\stex_ref_new_sym_target:n{<uri>}</code> Sets a new reference target for the symbol <code><uri></code> .
---	---

12.1.2 Using References

`\sref` `\sref[<opt-args>]{<id>}`

References the label with if *<id>*. Optional arguments: **TODO**

`\srefsym` `\srefsym[<opt-args>]{<symbol>}`

Like `\sref`, but references the *canonical label* for the provided symbol. The canonical target is the last of the following occurring in the document:

- A `\definiendum` or `\definame` for *<symbol>*,
- The `sassertion`, `sexample` or `sparagraph` with `for=<symbol>` that generated *<symbol>* in the first place, or
- A `\sparagraph` with `type=symdoc` and `for=<symbol>`.

`\srefsymuri` `\srefsymuri{<URI>}{<text>}`

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

Chapter 13

sTeX-Modules

This sub package contains code related to Modules

13.1 Macros and Environments

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

<hr/> <hr/> <code>\c_stex_module_<URI>_prop</code>	A property list with the following fields: <ul style="list-style-type: none"><code>name</code> The <i>name</i> of the module,<code>ns</code> the <i>namespace</i> in field <code>ns</code>,<code>file</code> the <i>file</i> containing the module, as a sequence of path fragments<code>lang</code> the module's <i>language</i>,<code>sig</code> the language of the signature module, if the current file is a translation from some other language,<code>deprecate</code> if this module is deprecated, the module that replaces it,<code>meta</code> the metatheory of the module.
--	---

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

<hr/> <hr/> <code>\c_stex_module_<URI>_constants</code>	The names of all constants declared in the module
---	---

<hr/> <hr/> <code>\c_stex_module_<URI>_constants</code>	The full URIs of all modules imported in this module
---	--

<hr/> <hr/>	<hr/>
<code>\l_stex_current_module_str</code>	<code>\l_stex_current_module_str</code> always contains the URI of the current module (if existent).
<hr/> <hr/>	
<code>\l_stex_all_modules_seq</code>	Stores full URIs for all modules currently in scope.
<hr/> <hr/>	
<code>\stex_if_in_module_p: *</code> <code>\stex_if_in_module:TF *</code>	Conditional for whether we are currently in a module
<hr/> <hr/>	
<code>\stex_if_module_exists_p:n *</code> <code>\stex_if_module_exists:nTF *</code>	
	Conditional for whether a module with the provided URI is already known.
<hr/> <hr/>	
<code>\stex_add_to_current_module:n</code> <code>\STEXexport</code>	
	Adds the provided tokens to the <code>_code</code> control sequence of the current module. <code>\stex_add_to_current_module:n</code> is used internally, <code>\STEXexport</code> is intended for users and additionally executes the provided code immediately.
<hr/> <hr/>	
<code>\stex_add_constant_to_current_module:n</code>	
	Adds the declaration with the provided name to the <code>_constants</code> control sequence of the current module.
<hr/> <hr/>	
<code>\stex_add_import_to_current_module:n</code>	
	Adds the module with the provided full URI to the <code>_imports</code> control sequence of the current module.
<hr/> <hr/>	
<code>\stex_collect_imports:n</code>	Iterates over all imports of the provided (full URI of a) module and stores them as a topologically sorted list – including the provided module as the last element – in <code>\l_stex_collect_imports_seq</code>
<hr/> <hr/>	
<code>\stex_do_up_to_module:n</code>	Code that is <i>exported</i> from module (such as symbol declarations) should be local <i>to the current module</i> . For that reason, ideally all symbol declarations and similar commands should be called directly in the module environment, however, that is not always feasible, e.g. in structural features or <code>sparapraphs</code> . <code>\stex_do_up_to_module</code> therefore executes the provided code repeatedly in an <code>\aftergroup</code> up until the group level is equal to that of the innermost smodule environment.

\stex_modules_current_namespace:

Computes the current namespace as follows:

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

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

13.1.1 The smodule environment

module (*env.*) `\begin{module}[\langle options \rangle]{\langle name \rangle}`

Opens a new module with name `\langle name \rangle`. Options are:

title (`\langle token list \rangle`) to display in customizations.

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

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

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

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

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

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

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

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

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

\stex_module_setup:nn `\stex_module_setup:nn{\langle params \rangle}{\langle name \rangle}`

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

\stexpatchmodule `\stexpatchmodule [\langle type \rangle] {\langle begincode \rangle} {\langle endcode \rangle}`

Customizes the presentation for those `smodule`-environments with `type=\langle type \rangle`, or all others if no `\langle type \rangle` is given.

\STEXModule `\STEXModule {\langle fragment \rangle}`

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

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

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

Chapter 14

STEX-Module Inheritance

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

14.1 Macros and Environments

14.1.1 SMS Mode

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

`\g_stex_smsmode_allowedmacros_tl`

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

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

`\g_stex_smsmode_allowedmacros_escape_tl`

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

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

`\g_stex_smsmode_allowedenvs_seq`

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

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

`\stex_if_smsmode_p:` ★ Tests whether SMS mode is currently active.
`\stex_if_smsmode:` *TF* ★

<code>\stex_file_in_smsmode:nn</code>	<code>\stex_in_smsmode:nn {<filename>} {<code>}</code>
---------------------------------------	--

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

<code>\stex_smsmode_do:</code>	Starts gobbling tokens until one is encountered that is allowed in SMS mode.
--------------------------------	--

14.1.2 Imports and Inheritance

<code>\importmodule</code>	<code>\importmodule[<archive-ID>]{<module-path>}</code>
----------------------------	---

Imports a module by reading it from a file and “activating” it. `STEX` determines the module and its containing file by passing its arguments on to `\stex_import_module_path:nn`.

<code>\usemodule</code>	<code>\importmodule[<archive-ID>]{<module-path>}</code>
-------------------------	---

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

$\backslash\text{stex_import_module_uri:nn}$	$\backslash\text{stex_import_module_uri:nn}$ $\{\langle\text{archive-ID}\rangle\}$ $\{\langle\text{module-path}\rangle\}$
---	--

Determines the URI of a module by splitting $\langle\text{module-path}\rangle$ into $\langle\text{path}\rangle?\langle\text{name}\rangle$. If $\langle\text{module-path}\rangle$ does *not* contain a ?-character, we consider it to be the $\langle\text{name}\rangle$, and $\langle\text{path}\rangle$ to be empty.

If $\langle\text{archive-ID}\rangle$ is empty, it is automatically set to the ID of the current archive (if one exists).

1. If $\langle\text{archive-ID}\rangle$ is empty:

(a) If $\langle\text{path}\rangle$ is empty, then $\langle\text{name}\rangle$ must have been declared earlier in the same file and retrievable from $\backslash\text{g_stex_modules_in_file_seq}$, or a file with name $\langle\text{name}\rangle.\langle\text{lang}\rangle.\text{tex}$ must exist in the same folder, containing a module $\langle\text{name}\rangle$.

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

(b) If $\langle\text{path}\rangle$ is not empty, it must point to the relative path of the containing file as well as the namespace.

2. Otherwise:

(a) If $\langle\text{path}\rangle$ is empty, then $\langle\text{name}\rangle$ must have been declared earlier in the same file and retrievable from $\backslash\text{g_stex_modules_in_file_seq}$, or a file with name $\langle\text{name}\rangle.\langle\text{lang}\rangle.\text{tex}$ must exist in the top **source** folder of the archive, containing a module $\langle\text{name}\rangle$.

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

(b) If $\langle\text{path}\rangle$ is not empty, it must point to the path of the containing file as well as the namespace, relative to the namespace of the archive.

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

$\backslash\text{l_stex_import_name_str}$ $\backslash\text{l_stex_import_archive_str}$ $\backslash\text{l_stex_import_path_str}$ $\backslash\text{l_stex_import_ns_str}$	stores the result in these four variables.
---	--

$\backslash\text{stex_import_require_module:nnnn}$	$\{\langle\text{ns}\rangle\}$ $\{\langle\text{archive-ID}\rangle\}$ $\{\langle\text{path}\rangle\}$ $\{\langle\text{name}\rangle\}$
---	---

Checks whether a module with URI $\langle\text{ns}\rangle?\langle\text{name}\rangle$ already exists. If not, it looks for a plausible file that declares a module with that URI.

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

Chapter 15

$\text{\texttt{S}\text{\textsf{T}\text{\textsf{E}\text{\textsf{X}}}}$ -Symbols

Code related to symbol declarations and notations

15.1 Macros and Environments

$\text{\texttt{\textbackslash symdecl}}$	$\text{\texttt{\textbackslash symdecl}}\{\langle\textit{macroname}\rangle\}[\langle\textit{args}\rangle]$
--	---

Declares a new symbol with semantic macro $\text{\texttt{\textbackslash macroname}}$. Optional arguments are:

- **name**: An (OMDOC) name. By default equal to $\langle\textit{macroname}\rangle$.
- **type**: An (ideally semantic) term, representing a *type*. Not used by $\text{\texttt{S}\text{\textsf{T}\text{\textsf{E}\text{\textsf{X}}}}$, but passed on to MMT for semantic services.
- **def**: An (ideally semantic) term, representing a *definiens*. Not used by $\text{\texttt{S}\text{\textsf{T}\text{\textsf{E}\text{\textsf{X}}}}$, but passed on to MMT for semantic services.
- **args**: Specifies the “signature” of the semantic macro. Can be either an integer $0 \leq n \leq 9$, or a (more precise) sequence of the following characters:
 - i** a “normal” argument, e.g. $\text{\texttt{\textbackslash symdecl}}\{\texttt{plus}}\}[\texttt{args=ii}]$ allows for $\text{\texttt{\textbackslash plus}}\{2\}\{2\}$.
 - a** an *associative* argument; i.e. a sequence of arbitrarily many arguments provided as a comma-separated list, e.g. $\text{\texttt{\textbackslash symdecl}}\{\texttt{plus}}\}[\texttt{args=a}]$ allows for $\text{\texttt{\textbackslash plus}}\{2,2,2\}$.
 - b** a *variable* argument. Is treated by $\text{\texttt{S}\text{\textsf{T}\text{\textsf{E}\text{\textsf{X}}}}$ like an **i**-argument, but an application is turned into an $\text{\texttt{OMBind}}$ in OMDOC, binding the provided variable in the subsequent arguments of the operator; e.g. $\text{\texttt{\textbackslash symdecl}}\{\texttt{forall}}\}[\texttt{args=bi}]$ allows for $\text{\texttt{\textbackslash forall}}\{x\}\{\textit{in}\}\text{\texttt{\textbackslash Nat}}\}\{x\geq 0\}$.

<hr/> <hr/> <code>\stex_symdecl_do:n</code>	<p>Implements the core functionality of <code>\symdecl</code>, and is called by <code>\symdecl</code> and <code>\symdef</code>.</p> <p>Ultimately stores the symbol $\langle URI \rangle$ in the property list <code>\l_stex_symdecl_<URI>_prop</code> with fields:</p> <ul style="list-style-type: none"> • <code>name</code> (string), • <code>module</code> (string), • <code>notations</code> (sequence of strings; initially empty), • <code>type</code> (token list), • <code>args</code> (string of is, as and bs), • <code>arity</code> (integer string), • <code>assocs</code> (integer string; number of associative arguments),
<hr/> <hr/> <code>\stex_all_symbols:n</code>	<p>Iterates over all currently available symbols. Requires two <code>\seq_map_break:</code> to break fully.</p>
<hr/> <hr/> <code>\stex_get_symbol:n</code>	<p>Computes the full URI of a symbol from a macro argument, e.g. the macro name, the macro itself, the full URI...</p>
<hr/> <hr/> <code>\notation</code>	<p><code>\notation[<args>]{<symbol>}{<notations⁺>}</code></p> <p>Introduces a new notation for $\langle symbol \rangle$, see <code>\stex_notation_do:nn</code></p>
<hr/> <hr/> <code>\stex_notation_do:nn</code>	<p><code>\stex_notation_do:nn{<URI>}{<notations⁺>}</code></p> <p>Implements the core functionality of <code>\notation</code>, and is called by <code>\notation</code> and <code>\symdef</code>.</p> <p>Ultimately stores the notation in the property list <code>\g_stex_notation_<URI>#<variant>#<lang>_prop</code> with fields:</p> <ul style="list-style-type: none"> • <code>symbol</code> (URI string), • <code>language</code> (string), • <code>variant</code> (string), • <code>opprec</code> (integer string), • <code>argprec</code> (sequence of integer strings)
<hr/> <hr/> <code>\symdef</code>	<p><code>\symdef[<args>]{<symbol>}{<notations⁺>}</code></p> <p>Combines <code>\symdecl</code> and <code>\notation</code> by introducing a new symbol and assigning a new notation for it.</p>

Chapter 16

STEX-Terms

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

16.1 Macros and Environments

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

`\symref` `\symref{<symbol>}{<text>}`
 shortcut for `\STEXsymbol{<symbol>}! [<text>]`

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

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

`\STEXInternalTermMathOMSiiii` `<URI><fragment><precedence><body>`
`\STEXInternalTermMathOMAiiai`
`\STEXInternalTermMathOMBiiii`

Annotates `<body>` as an OMDOC-term (OMID, OMA or OMBIND, respectively) with head symbol `<URI>`, generated by the specific notation `<fragment>` with (upwards) operator precedence `<precedence>`. Inserts parentheses according to the current downwards precedence and operator precedence.

`\STEXInternalTermMathArgiii` `\stex_term_arg:nnn<int><prec><body>`

Annotates `<body>` as the `<int>`th argument of the current OMA or OMBIND, with (downwards) argument precedence `<prec>`.

<hr/> <code>\STEXInternalTermMathAssocArgiiii</code> <hr/>	<code>\stex_term_arg:nnn⟨int⟩⟨prec⟩⟨notation⟩⟨type⟩⟨body⟩</code>
	Annotates $\langle body \rangle$ as the $\langle int \rangle$ th (associative) <i>sequence</i> argument (as comma-separated list of terms) of the current OMA or OMBIND, with (downwards) argument precedence $\langle prec \rangle$ and associative notation $\langle notation \rangle$.
<hr/> <code>\infpref</code> <code>\neginfpref</code> <hr/>	Maximal and minimal notation precedences.
<hr/> <code>\dobrackets</code> <hr/>	<code>\dobrackets {⟨body⟩}</code> Puts $\langle body \rangle$ in parentheses; scaled if in display mode unscaled otherwise. Uses the current \TeX brackets (by default (and)), which can be changed temporarily using <code>\withbrackets</code> .
<hr/> <code>\withbrackets</code> <hr/>	<code>\withbrackets ⟨left⟩ ⟨right⟩ {⟨body⟩}</code> Temporarily (i.e. within $\langle body \rangle$) sets the brackets used by \TeX for automated bracketing (by default (and)) to $\langle left \rangle$ and $\langle right \rangle$. Note that $\langle left \rangle$ and $\langle right \rangle$ need to be allowed after <code>\left</code> and <code>\right</code> in display-mode.
<hr/> <code>\stex_term_custom:nn</code> <hr/>	<code>\stex_term_custom:nn{⟨URI⟩}{⟨args⟩}</code> Implements custom one-time notation. Invoked by <code>\stex_invoke_symbol:n</code> in text mode, or if followed by <code>*</code> in math mode, or whenever followed by <code>!</code> .
<hr/> <code>\comp</code> <code>\compemph</code> <code>\compemph@uri</code> <code>\defemph</code> <code>\defemph@uri</code> <code>\symrefemph</code> <code>\symrefemph@uri</code> <code>\varemp</code> <code>\varemp@uri</code> <hr/>	<code>\comp{⟨args⟩}</code> Marks $\langle args \rangle$ as a notation component of the current symbol for highlighting, linking, etc. The precise behavior is governed by <code>\@comp</code> , which takes as additional argument the URI of the current symbol. By default, <code>\@comp</code> adds the URI as a PDF tooltip and colors the highlighted part in blue. <code>\@defemph</code> behaves like <code>\@comp</code> , and can be similarly redefined, but marks an expression as <i>definiendum</i> (used by <code>\definiendum</code>)
<hr/> <code>\STEXinvisible</code> <hr/>	Exports its argument as OMDOC (invisible), but does not produce PDF output. Useful e.g. for semantic macros that take arguments that are not part of the symbolic notation.
<hr/> <code>\ellipses</code> <hr/>	TODO

Chapter 17

TeX-Structural Features

Code related to structural features

17.1 Macros and Environments

17.1.1 Structures

`mathstructure` (*env.*) TODO

Chapter 18

TeX-Statements

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

18.1 Macros and Environments

`symboldoc (env.) \begin{<symboldoc>}{<symbols>} <text> \end{<symboldoc>}`

Declares *<text>* to be a (natural language, encyclopaedic) description of $\{<symbols>\}$ (a comma separated list of symbol identifiers).

Chapter 19

sTeX-Proofs: Structural Markup for Proofs

Chapter 20

sT_EX-Metatheory

20.1 Symbols

Part III
Extensions

Chapter 21

Tikzinput: Treating TIKZ code as images

21.1 Macros and Environments

Chapter 22

document-structure: Semantic Markup for Open Mathematical Documents in L^AT_EX

Chapter 23

NotesSlides – Slides and Course Notes

Chapter 24

`problem.sty`: An Infrastructure for formatting Problems

Chapter 25

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

Part IV
Implementation

Chapter 26

\TeX -Basics Implementation

26.1 The \TeX Document Class

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

```
1 \<cls>
2
3 %%%%%%%%% basics.dtx %%%%%%%%%
4
5 \RequirePackage{expl3,l3keys2e}
6 \ProvidesExplClass{stex}{2022/08/08}{3.2.0}{sTeX document class}
7
8 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
9 \ProcessOptions
10
11 \bool_set_true:N \c_stex_document_class_bool
12
13 \RequirePackage{stex}
14
15 \stex_html_backend:TF {
16   \LoadClass{article}
17 }{
18   \LoadClass[border=1px,varwidth,crop=false]{standalone}
19   \setlength\textwidth{15cm}
20 }
21 \RequirePackage{standalone}
22
23
24 \clist_if_empty:NT \c_stex_languages_clist {
25   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
26   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
27   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
28   \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
29     \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
30       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
```



```

31     }
32   }
33   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
34   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
35     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
36     \prop_if_in:NoT \c_stex_languages_prop \l_tmpa_str {
37       \stex_debug:nn{language} {Language~\l_tmpa_str~
38         inferred~from~file~name}
39     }
40   }
41 }
42 }
43 </cls>

```

26.2 Preliminaries

```

44 <*package>
45
46 %%%%%%%%% basics.dtx %%%%%%%%%
47
48 \RequirePackage{expl3,l3keys2e,ltxcmds}
49 \ProvidesExplPackage{stex}{2022/08/08}{3.2.0}{sTeX package}
50
51 \bool_if_exist:NF \c_stex_document_class_bool {
52   \bool_set_false:N \c_stex_document_class_bool
53   \RequirePackage{standalone}
54 }
55
56 \message{^^J*~This~is~sTeX~version~3.2.0~*^^J}
57
58 %\RequirePackage{morewrites}
59 %\RequirePackage{amsmath}
60
61 Package options:
62 \keys_define:nn { stex } {
63   debug      .clist_set:N = \c_stex_debug_clist ,
64   lang       .clist_set:N = \c_stex_languages_clist ,
65   mathhub    .tl_set_x:N  = \mathhub ,
66   usesms     .bool_set:N  = \c_stex_persist_mode_bool ,
67   writesms   .bool_set:N  = \c_stex_persist_write_mode_bool ,
68   image      .bool_set:N  = \c_tikzinput_image_bool ,
69   unknown    .code:n      = {}
70 }
71 \ProcessKeysOptions { stex }

```

\stex The sTeX logo:

\sTeX `\RequirePackage{stex-logo}` % externalized for backwards-compatibility reasons

(End definition for `\stex` and `\sTeX`. These functions are documented on page 76.)

26.3 Messages and logging

```

72 <@@=stex_log>
    Warnings and error messages
73 \msg_new:nnn{stex}{error/unknownlanguage}{
74   Unknown~language:~#1
75 }
76 \msg_new:nnn{stex}{warning/nomathhub}{
77   MATHHUB~system~variable~not~found~and~no~
78   \detokenize{\mathhub}~value~set!
79 }
80 \msg_new:nnn{stex}{error/deactivated-macro}{
81   The~\detokenize{#1}~command~is~only~allowed~in~#2!
82 }

```

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

```

83 \cs_new_protected:Nn \stex_debug:nn {
84   \clist_if_in:NnTF \c_stex_debug_clist { all } {
85     \msg_set:nnn{stex}{debug / #1}{
86       \\Debug~#1:~#2\\
87     }
88     \msg_none:nn{stex}{debug / #1}
89   }{
90     \clist_if_in:NnT \c_stex_debug_clist { #1 } {
91       \msg_set:nnn{stex}{debug / #1}{
92         \\Debug~#1:~#2\\
93       }
94       \msg_none:nn{stex}{debug / #1}
95     }
96   }
97 }

```

(End definition for \stex_debug:nn. This function is documented on page 76.)

Redirecting messages:

```

98 \clist_if_in:NnTF \c_stex_debug_clist {all} {
99   \msg_redirect_module:nnn{ stex }{ none }{ term }
100 }{
101   \clist_map_inline:Nn \c_stex_debug_clist {
102     \msg_redirect_name:nnn{ stex }{ debug / #1 }{ term }
103   }
104 }
105
106 \stex_debug:nn{log}{debug~mode~on}

```

26.4 HTML Annotations

```

107 <@@=stex_annotate>

```

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

```

108 \tl_new:N \l_stex_html_arg_tl

```

(End definition for \l_stex_html_arg_tl and \c_stex_html_emptyarg_tl. These variables are documented on page ??.)

`_stex_html_checkempty:n`

```

109 \cs_new_protected:Nn \_stex_html_checkempty:n {
110   \tl_set:Nn \l_stex_html_arg_tl { #1 }
111   \tl_if_empty:NT \l_stex_html_arg_tl {
112     \tl_set_eq:NN \l_stex_html_arg_tl \c_stex_html_emptyarg_tl
113   }
114 }

```

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

`\stex_if_do_html_p:` Whether to (locally) produce HTML output

`\stex_if_do_html:TF`

```

115 \bool_new:N \_stex_html_do_output_bool
116 \bool_set_true:N \_stex_html_do_output_bool
117
118 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
119   \bool_if:nTF \_stex_html_do_output_bool
120     \prg_return_true: \prg_return_false:
121 }

```

(End definition for `\stex_if_do_html:TF`. This function is documented on page 76.)

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

```

122 \cs_new_protected:Nn \stex_suppress_html:n {
123   \exp_args:Nne \use:nn {
124     \bool_set_false:N \_stex_html_do_output_bool
125     #1
126   }{
127     \stex_if_do_html:T {
128       \bool_set_true:N \_stex_html_do_output_bool
129     }
130   }
131 }

```

(End definition for `\stex_suppress_html:n`. This function is documented on page 76.)

`\stex_annotate_html:nn`

`\stex_annotate_invisible:n`

`\stex_annotate_invisible:nnn`

We define four macros for introducing attributes in the HTML output. The definitions depend on the “backend” used (L^AT_EX_ML, R_US_TE_X, p_DF_LA_TE_X).

The p_DF_LA_TE_X-macros largely do nothing; the R_US_TE_X-implementations are pretty clear in what they do, the L^AT_EX_ML-implementations resort to perl bindings.

```

132 \ifcsname if@rustex\endcsname\else
133   \expandafter\newif\csname if@rustex\endcsname
134   \@rustexfalse
135 \fi
136 \ifcsname if@latexml\endcsname\else
137   \expandafter\newif\csname if@latexml\endcsname
138   \@latexmlfalse
139 \fi
140 \tl_if_exist:NF\stex@backend{
141   \if@rustex
142     \def\stex@backend{rustex}
143   \else
144     \if@latexml
145       \def\stex@backend{latexml}
146     \else

```

```

147     \cs_if_exist:NTF\HCode{
148         \def\stex@backend{tex4ht}
149     }{
150         \def\stex@backend{pdflatex}
151     }
152     \fi
153     \fi
154 }
155 \input{stex-backend-\stex@backend.cfg}
156
157 \newif\ifstexhtml
158 \stex_html_backend:TF\stexhtmltrue\stexhtmlfalse
159

```

(End definition for `\stex_annotate:nnn`, `\stex_annotate_invisible:n`, and `\stex_annotate_invisible:nnn`. These functions are documented on page 77.)

26.5 Babel Languages

```

160 <@@=stex_language>

```

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

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

```

161 \exp_args:NNx \prop_const_from_keyval:Nn \c_stex_languages_prop { \tl_to_str:n {
162     en = english ,
163     de = ngerman ,
164     ar = arabic ,
165     bg = bulgarian ,
166     ru = russian ,
167     fi = finnish ,
168     ro = romanian ,
169     tr = turkish ,
170     fr = french
171 }}
172
173 \exp_args:NNx \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop { \tl_to_str:n {
174     english   = en ,
175     ngerman   = de ,
176     arabic    = ar ,
177     bulgarian = bg ,
178     russian   = ru ,
179     finnish   = fi ,
180     romanian  = ro ,
181     turkish   = tr ,
182     french    = fr
183 }}
184 % todo: chinese simplified (zhs)
185 %         chinese traditional (zht)

```

(End definition for `\c_stex_languages_prop` and `\c_stex_language_abbrevs_prop`. These variables are documented on page 77.)

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

```

186 \cs_new_protected:Nn \stex_set_language:Nn {
187     \str_set:Nx \l_tmpa_str {#2}
188     \prop_get:NoNT \c_stex_languages_prop \l_tmpa_str #1 {

```

```

189 \ifx\@onlypreamble\@notprerr
190 \ltx@ifpackageloaded{babel}{
191 \exp_args:No \selectlanguage #1
192 }{}
193 \else
194 \exp_args:No \str_if_eq:nnTF #1 {turkish} {
195 \RequirePackage[#1,shorthands=:!]{babel}
196 }{
197 \RequirePackage[#1]{babel}
198 }
199 \fi
200 }
201 }
202
203 \clist_if_empty:NF \c_stex_languages_clist {
204 \bool_set_false:N \l_tmpa_bool
205 \clist_clear:N \l_tmpa_clist
206 \clist_map_inline:Nn \c_stex_languages_clist {
207 \str_set:Nx \l_tmpa_str {#1}
208 \str_if_eq:nnT {#1}{tr}{
209 \bool_set_true:N \l_tmpa_bool
210 }
211 \prop_get:NoNTF \c_stex_languages_prop \l_tmpa_str \l_tmpa_str {
212 \clist_put_right:No \l_tmpa_clist \l_tmpa_str
213 } {
214 \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
215 }
216 }
217 \stex_debug:nn{lang} {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
218 \bool_if:NTF \l_tmpa_bool {
219 \RequirePackage[\clist_use:Nn \l_tmpa_clist,,shorthands=:!]{babel}
220 }{
221 \RequirePackage[\clist_use:Nn \l_tmpa_clist,]{babel}
222 }
223 }
224
225 \AtBeginDocument{
226 \stex_html_backend:T {
227 \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
228 \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
229 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
230 \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
231 \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
232 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
233 \stex_debug:nn{basics} {Language~\l_tmpa_str~
234 inferred~from~file~name}
235 \stex_annotate_invisible:nnn{language}{ \l_tmpa_str }{}
236 }
237 }
238 }
239

```

26.6 Persistence

```

240 <@@=stex_persist>
241 \bool_if:NTF \c_stex_persist_mode_bool {
242   \def \stex_persist:n #1 {}
243   \def \stex_persist:x #1 {}
244 }{
245   \bool_if:NTF \c_stex_persist_write_mode_bool {
246     \iow_new:N \c__stex_persist_iow
247     \iow_open:Nn \c__stex_persist_iow{\jobname.sms}
248     \AtEndDocument{
249       \iow_close:N \c__stex_persist_iow
250     }
251     \cs_new_protected:Nn \stex_persist:n {
252       \tl_set:Nn \l_tmpa_tl { #1 }
253       \regex_replace_all:nnN { \cP\# } { \cO\# } \l_tmpa_tl
254       \regex_replace_all:nnN { \ } { \~ } \l_tmpa_tl
255       \exp_args:NNo \iow_now:Nn \c__stex_persist_iow \l_tmpa_tl
256     }
257     \cs_generate_variant:Nn \stex_persist:n {x}
258   }{
259     \def \stex_persist:n #1 {}
260     \def \stex_persist:x #1 {}
261   }
262 }

```

26.7 Auxiliary Methods

\stex_deactivate_macro:Nn

```

263 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
264   \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
265   \def#1{
266     \msg_error:nnnn{stex}{error/deactivated-macro}{\detokenize{#1}}{#2}
267   }
268 }

```

(End definition for \stex_deactivate_macro:Nn. This function is documented on page 77.)

\stex_reactivate_macro:N

```

269 \cs_new_protected:Nn \stex_reactivate_macro:N {
270   \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname
271 }

```

(End definition for \stex_reactivate_macro:N. This function is documented on page 77.)

\ignorespacesandpars

```

272 \protected\def\ignorespacesandpars{
273   \begingroup\catcode13=10\relax
274   \@ifnextchar\par{
275     \endgroup\expandafter\ignorespacesandpars\@gobble
276   }{
277     \endgroup
278   }
279 }

```

```

280
281 \cs_new_protected:Nn \stex_copy_control_sequence:NNN {
282   \tl_set:Nx \_tmp_args_tl {\cs_argument_spec:N #2}
283   \exp_args:NNo \tl_remove_all:Nn \_tmp_args_tl \c_hash_str
284   \int_set:Nn \l_tmpa_int {\tl_count:N \_tmp_args_tl}
285
286   \tl_clear:N \_tmp_args_tl
287   \int_step_inline:nn \l_tmpa_int {
288     \tl_put_right:Nx \_tmp_args_tl {{\exp_not:n{####}\exp_not:n{##1}}}
289   }
290
291   \tl_set:Nn #3 {\cs_generate_from_arg_count:NNnn #1 \cs_set:Npn}
292   \tl_put_right:Nx #3 { {\int_use:N \l_tmpa_int}{
293     \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
294     \exp_after:wN\exp_after:wN\exp_after:wN\exp_after:wN {
295       \exp_after:wN #2 \_tmp_args_tl
296     }
297   }}
298 }
299 \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {cNN}
300 \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {NcN}
301 \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {ccN}
302
303 \cs_new_protected:Nn \stex_copy_control_sequence_ii:NNN {
304   \tl_set:Nx \_tmp_args_tl {\cs_argument_spec:N #2}
305   \exp_args:NNo \tl_remove_all:Nn \_tmp_args_tl \c_hash_str
306   \int_set:Nn \l_tmpa_int {\tl_count:N \_tmp_args_tl}
307
308   \tl_clear:N \_tmp_args_tl
309   \int_step_inline:nn \l_tmpa_int {
310     \tl_put_right:Nx \_tmp_args_tl {{\exp_not:n{#####}\exp_not:n{##1}}}
311   }
312
313   \edef \_tmp_args_tl {
314     \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
315     \exp_after:wN\exp_after:wN\exp_after:wN {
316       \exp_after:wN #2 \_tmp_args_tl
317     }
318   }
319
320   \exp_after:wN \def \exp_after:wN \_tmp_args_tl
321   \exp_after:wN ##\exp_after:wN 1 \exp_after:wN ##\exp_after:wN 2
322   \exp_after:wN { \_tmp_args_tl }
323
324   \edef \_tmp_args_tl {
325     \exp_after:wN \exp_not:n \exp_after:wN {
326       \_tmp_args_tl {####1}{####2}
327     }
328   }
329
330   \tl_set:Nn #3 {\cs_generate_from_arg_count:NNnn #1 \cs_set:Npn}
331   \tl_put_right:Nx #3 { {\int_use:N \l_tmpa_int}{
332     \exp_after:wN\exp_not:n\exp_after:wN{\_tmp_args_tl}
333   }}

```

```

334 }
335
336 \cs_generate_variant:Nn \stex_copy_control_sequence_ii:NNN {cNN}
337 \cs_generate_variant:Nn \stex_copy_control_sequence_ii:NNN {NcN}
338 \cs_generate_variant:Nn \stex_copy_control_sequence_ii:NNN {ccN}

```

(End definition for \ignorespacesandpars. This function is documented on page 77.)

\MMTrule

```

339 \NewDocumentCommand \MMTrule {m m}{
340   \seq_set_split:Nnn \l_tmpa_seq , {#2}
341   \int_zero:N \l_tmpa_int
342   \stex_annotate_invisible:nnn{mmtrule}{scala://#1}{
343     \seq_if_empty:NF \l_tmpa_seq {
344       $\seq_map_inline:Nn \l_tmpa_seq {
345         \int_incr:N \l_tmpa_int
346         \stex_annotate:nnn{arg}{i\int_use:N \l_tmpa_int}{##1}
347       }$
348     }
349   }
350 }
351
352 \NewDocumentCommand \MMTinclude {m}{
353   \stex_annotate_invisible:nnn{import}{#1}{-}
354 }
355
356 \tl_new:N \g_stex_document_title
357 \cs_new_protected:Npn \STEXTtitle #1 {
358   \tl_if_empty:NT \g_stex_document_title {
359     \tl_gset:Nn \g_stex_document_title { #1 }
360   }
361 }
362 \cs_new_protected:Nn \stex_document_title:n {
363   \tl_if_empty:NT \g_stex_document_title {
364     \tl_gset:Nn \g_stex_document_title { #1 }
365     \stex_annotate_invisible:n{\noindent
366       \stex_annotate:nnn{doctitle}{-}{ #1 }
367     \par}
368   }
369 }
370 \AtBeginDocument {
371   \let \STEXTtitle \stex_document_title:n
372   \tl_if_empty:NF \g_stex_document_title {
373     \stex_annotate_invisible:n{\noindent
374       \stex_annotate:nnn{doctitle}{-}{ \g_stex_document_title }
375     \par}
376   }
377   \let \stex_maketitle:\maketitle
378   \def \maketitle{
379     \tl_if_empty:NF \@title {
380       \exp_args:No \stex_document_title:n \@title
381     }
382     \stex_maketitle:
383   }

```



```

384 }
385
386 \cs_new_protected:Nn \stex_par: {
387   \mode_if_vertical:F{
388     \if@minipage\else\if@nobreak\else\par\fi\fi
389   }
390 }
391
392 \end{package}

```

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

Chapter 27

STEX -MathHub Implementation

```
393 <*package>
394
395 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
396
397 <@@=stex_path>
398
399 Warnings and error messages
400 \msg_new:nnn{stex}{error/norepository}{
401   No~archive~#1~found~in~#2
402 }
403 \msg_new:nnn{stex}{error/notinarchive}{
404   Not~currently~in~an~archive,~but~\detokenize{#1}~
405   needs~one!
406 }
407 \msg_new:nnn{stex}{error/nofile}{
408   \detokenize{#1}~could~not~find~file~#2
409 }
410 \msg_new:nnn{stex}{error/twofiles}{
411   \detokenize{#1}~found~two~candidates~for~#2
412 }
```

27.1 Generic Path Handling

We treat paths as L^AT_EX3-sequences (of the individual path segments, i.e. separated by a /-character) unix-style; i.e. a path is absolute if the sequence starts with an empty entry.

`\stex_path_from_string:Nn`

```
411 \cs_new_protected:Nn \stex_path_from_string:Nn {
412   \stex_debug:nn{files}{#2}
413   \str_set:Nx \l_tmpa_str { #2 }
414   \str_if_empty:NTF \l_tmpa_str {
415     \seq_clear:N #1
416   }{
417     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
418     \sys_if_platform_windows:T{
```

```

419 \seq_clear:N \l_tmpa_tl
420 \seq_map_inline:Nn #1 {
421   \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
422   \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
423 }
424 \seq_set_eq:NN #1 \l_tmpa_tl
425 }
426 \stex_path_canonicalize:N #1
427 }
428 \stex_debug:nn{files}{Yields: \stex_path_to_string:N#1}
429 }
430

```

(End definition for `\stex_path_from_string:Nn`. This function is documented on page 78.)

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

```

431 \cs_new_protected:Nn \stex_path_to_string:NN {
432   \exp_args:NNe \str_set:Nn #2 { \seq_use:Nn #1 / }
433 }
434
435 \cs_new:Nn \stex_path_to_string:N {
436   \seq_use:Nn #1 /
437 }

```

(End definition for `\stex_path_to_string:NN` and `\stex_path_to_string:N`. These functions are documented on page 78.)

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

```

438 \str_const:Nn \c__stex_path_dot_str {.}
439 \str_const:Nn \c__stex_path_up_str {..}

```

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

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

```

440 \cs_new_protected:Nn \stex_path_canonicalize:N {
441   \seq_if_empty:NF #1 {
442     \seq_clear:N \l_tmpa_seq
443     \seq_get_left:NN #1 \l_tmpa_tl
444     \str_if_empty:NT \l_tmpa_tl {
445       \seq_put_right:Nn \l_tmpa_seq {}
446     }
447     \seq_map_inline:Nn #1 {
448       \str_set:Nn \l_tmpa_tl { ##1 }
449       \str_if_eq:NNF \l_tmpa_tl \c__stex_path_dot_str {
450         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
451           \seq_if_empty:NNTF \l_tmpa_seq {
452             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
453               \c__stex_path_up_str
454             }
455           }{
456             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
457             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
458               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
459                 \c__stex_path_up_str

```

```

460         }
461       }{
462         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
463       }
464     }
465   }{
466     \str_if_empty:NF \l_tmpa_tl {
467       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
468     }
469   }
470 }
471 }
472 \seq_gset_eq:NN #1 \l_tmpa_seq
473 }
474 }

```

(End definition for `\stex_path_canonicalize:N`. This function is documented on page 78.)

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

```

475 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
476   \seq_if_empty:NTF #1 {
477     \prg_return_false:
478   }{
479     \seq_get_left:NN #1 \l_tmpa_tl
480     \sys_if_platform_windows:TF{
481       \str_if_in:NnTF \l_tmpa_tl {:}{
482         \prg_return_true:
483       }{
484         \prg_return_false:
485       }
486     }{
487       \str_if_empty:NTF \l_tmpa_tl {
488         \prg_return_true:
489       }{
490         \prg_return_false:
491       }
492     }
493   }
494 }

```

(End definition for `\stex_path_if_absolute:N`. This function is documented on page 78.)

27.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

495 \str_new:N\l_stex_kpsewhich_return_str
496 \cs_new_protected:Nn \stex_kpsewhich:n {\begingroup
497   \catcode'\ =12
498   \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
499   \tl_gset_eq:NN \l_tmpa_tl \l_tmpa_tl
500   \endgroup
501   \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
502   \tl_trim_spaces:N \l_stex_kpsewhich_return_str
503 }

```

(End definition for `\stex_kpsewhich:n`. This function is documented on page 78.)

We determine the PWD

`\c_stex_pwd_seq`
`\c_stex_pwd_str`

```
504 \sys_if_platform_windows:TF{
505   \begingroup\escapechar=-1\catcode'\=12
506   \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
507   \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
508   \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_
509   }}{
510   \stex_kpsewhich:n{-var-value~PWD}
511   }
512
513 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
514 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
515 \stex_debug:nn {mathhub} {PWD:~\str_use:N\c_stex_pwd_str}
```

(End definition for `\c_stex_pwd_seq` and `\c_stex_pwd_str`. These variables are documented on page 78.)

27.3 File Hooks and Tracking

516 `<@@=stex_files>`

We introduce hooks for file inputs that keep track of the absolute paths of files used. This will be useful to keep track of modules, their archives, namespaces etc.

Note that the absolute paths are only accurate in `\input`-statements for paths relative to the PWD, so they shouldn't be relied upon in any other setting than for \TeX -purposes.

`\g_stex_files_stack` keeps track of file changes

```
517 \seq_gclear_new:N\g_stex_files_stack
```

(End definition for `\g_stex_files_stack`.)

`\c_stex_mainfile_seq`
`\c_stex_mainfile_str`

```
518 \str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
519 \stex_path_from_string:Nn \c_stex_mainfile_seq
520   \c_stex_mainfile_str
```

(End definition for `\c_stex_mainfile_seq` and `\c_stex_mainfile_str`. These variables are documented on page 78.)

`\g_stex_currentfile_seq`

```
521 \seq_gclear_new:N\g_stex_currentfile_seq
```

(End definition for `\g_stex_currentfile_seq`. This variable is documented on page 79.)

`\stex_filestack_push:n`

```
522 \cs_new_protected:Nn \stex_filestack_push:n {
523   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
524   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
525     \stex_path_from_string:Nn\g_stex_currentfile_seq{
526       \c_stex_pwd_str/#1
527     }
528   }
```

```

528 }
529 \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
530 \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
531 \stex_get_document_uri:
532 }

```

(End definition for `\stex_filestack_push:n`. This function is documented on page 79.)

`\stex_filestack_pop:`

```

533 \cs_new_protected:Nn \stex_filestack_pop: {
534   \seq_if_empty:NF\g__stex_files_stack{
535     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
536   }
537   \seq_if_empty:NTF\g__stex_files_stack{
538     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
539   }{
540     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
541     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
542   }
543   \stex_get_document_uri:
544 }

```

(End definition for `\stex_filestack_pop:`. This function is documented on page 79.)

Hooks for the current file:

```

545 \AddToHook{file/before}{
546   \tl_if_empty:NTF\CurrentFilePath{
547     \stex_filestack_push:n{\CurrentFile}
548   }{
549     \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
550   }
551 }
552 \AddToHook{file/after}{
553   \stex_filestack_pop:
554 }

```

27.4 MathHub Repositories

```

555 <@=stex_mathhub>

```

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

The path to the mathhub directory. If the `\mathhub`-macro is not set, we query `kpsewhich` for the MATHHUB system variable.

```

556 \str_if_empty:NTF\mathhub{
557   \sys_if_platform_windows:TF{
558     \begingroup\escapechar=-1\catcode'\=12
559     \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str MATHHUB\c_percent_str}
560     \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
561     \exp_args:NNx\str_if_eq:ont\l_stex_kpsewhich_return_str{\c_percent_str MATHHUB\c_percent_str}
562     \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_kpsewhich_return_str}}
563   }{
564     \stex_kpsewhich:n{-var-value-MATHHUB}
565   }
566   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
567 }

```

```

568 \str_if_empty:NT \c_stex_mathhub_str {
569   \sys_if_platform_windows:TF{
570     \begingroup\escapechar=-1\catcode'\=12
571     \exp_args:Nx\stex_kpsewhich:n{-var-value-HOME}
572     \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
573     \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_s
574   }{
575     \stex_kpsewhich:n{-var-value-HOME}
576   }
577   \ior_open:NnT \g_tmpa_ior{\l_stex_kpsewhich_return_str / .stex / mathhub.path}{
578     \begingroup\escapechar=-1\catcode'\=12
579     \ior_str_get:NN \g_tmpa_ior \l_tmpa_str
580     \sys_if_platform_windows:T{
581       \exp_args:NNx\str_replace_all:Nnn\l_tmpa_str{\c_backslash_str}/
582     }
583     \str_gset_eq:NN \c_stex_mathhub_str\l_tmpa_str
584     \endgroup
585     \ior_close:N \g_tmpa_ior
586   }
587 }
588 \str_if_empty:NTF\c_stex_mathhub_str{
589   \msg_warning:nn{stex}{warning/nomathhub}
590 }{
591   \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
592   \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
593 }
594 }{
595   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
596   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
597     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
598       \c_stex_pwd_str/\mathhub
599     }
600   }
601   \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
602   \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
603 }

```

(End definition for `\mathhub`, `\c_stex_mathhub_seq`, and `\c_stex_mathhub_str`. These variables are documented on page 79.)

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

```

604 \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
605   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
606     \str_set:Nx \l_tmpa_str { #1 }
607     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
608     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
609     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
610     \__stex_mathhub_find_manifest:N \l_tmpa_seq
611     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
612       \msg_error:nnxx{stex}{error/norepository}{#1}{
613         \stex_path_to_string:N \c_stex_mathhub_str
614       }
615       \input{Fatal-Error!}

```

```

616     } {
617     \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
618     }
619   }
620 }

```

(End definition for __stex_mathhub_do_manifest:n.)

\l_stex_mathhub_manifest_file_seq

```

621 \seq_new:N\l__stex_mathhub_manifest_file_seq

```

(End definition for \l_stex_mathhub_manifest_file_seq.)

__stex_mathhub_find_manifest:N

Attempts to find the MANIFEST.MF in some file path and stores its path in \l__stex_mathhub_manifest_file_seq:

```

622 \cs_new_protected:Nn \__stex_mathhub_find_manifest:N {
623   \seq_set_eq:NN\l_tmpa_seq #1
624   \bool_set_true:N\l_tmpa_bool
625   \bool_while_do:Nn \l_tmpa_bool {
626     \seq_if_empty:NTF \l_tmpa_seq {
627       \bool_set_false:N\l_tmpa_bool
628     }{
629       \file_if_exist:nTF{
630         \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
631       }{
632         \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
633         \bool_set_false:N\l_tmpa_bool
634       }{
635         \file_if_exist:nTF{
636           \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
637         }{
638           \seq_put_right:Nn\l_tmpa_seq{META-INF}
639           \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
640           \bool_set_false:N\l_tmpa_bool
641         }{
642           \file_if_exist:nTF{
643             \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
644           }{
645             \seq_put_right:Nn\l_tmpa_seq{meta-inf}
646             \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
647             \bool_set_false:N\l_tmpa_bool
648           }{
649             \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
650           }
651         }
652       }
653     }
654   }
655   \seq_set_eq:NN\l__stex_mathhub_manifest_file_seq\l_tmpa_seq
656 }

```

(End definition for __stex_mathhub_find_manifest:N.)

\c_stex_mathhub_manifest_ior

File variable used for MANIFEST-files

```

657 \ior_new:N \c__stex_mathhub_manifest_ior

```


(End definition for `\c__stex_mathhub_manifest_ior.`)

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

```

658 \cs_new_protected:Nn \_stex_mathhub_parse_manifest:n {
659   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
660   \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
661   \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
662     \str_set:Nn \l_tmpa_str {##1}
663     \exp_args:NNo \seq_set_split:Nnn
664       \l_tmpb_seq \c_colon_str \l_tmpa_str
665     \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
666       \exp_args:NNe \str_set:Nn \l_tmpb_tl {
667         \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
668       }
669       \exp_args:No \str_case:nnTF \l_tmpa_tl {
670         {id} {
671           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
672             { id } \l_tmpb_tl
673         }
674         {narration-base} {
675           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
676             { narr } \l_tmpb_tl
677         }
678         {url-base} {
679           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
680             { docurl } \l_tmpb_tl
681         }
682         {source-base} {
683           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
684             { ns } \l_tmpb_tl
685         }
686         {ns} {
687           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
688             { ns } \l_tmpb_tl
689         }
690         {dependencies} {
691           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
692             { deps } \l_tmpb_tl
693         }
694       }{}{}
695     }{}
696   }
697   \ior_close:N \c__stex_mathhub_manifest_ior
698   \stex_persist:x {
699     \prop_set_from_keyval:cn{ c_stex_mathhub_#1_manifest_prop }{
700       \exp_after:wN \prop_to_keyval:N \csname c_stex_mathhub_#1_manifest_prop\endcsname
701     }
702   }
703 }
```

(End definition for `_stex_mathhub_parse_manifest:n.`)

`\stex_set_current_repository:n`

```

704 \cs_new_protected:Nn \stex_set_current_repository:n {
```

```

705 \stex_require_repository:n { #1 }
706 \prop_set_eq:Nc \l_stex_current_repository_prop {
707   c_stex_mathhub_#1_manifest_prop
708 }
709 }

```

(End definition for `\stex_set_current_repository:n`. This function is documented on page 79.)

`\stex_require_repository:n`

```

710 \cs_new_protected:Nn \stex_require_repository:n {
711   \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
712     \stex_debug:nn{mathhub}{Opening~archive:~#1}
713     \__stex_mathhub_do_manifest:n { #1 }
714   }
715 }

```

(End definition for `\stex_require_repository:n`. This function is documented on page 79.)

`\l_stex_current_repository_prop` Current MathHub repository

```

716 %\prop_new:N \l_stex_current_repository_prop
717 \bool_if:NF \c_stex_persist_mode_bool {
718   \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
719   \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
720     \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
721   } {
722     \__stex_mathhub_parse_manifest:n { main }
723     \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
724     \l_tmpa_str
725     \prop_set_eq:cN { c_stex_mathhub_#1_manifest_prop }
726     \c_stex_mathhub_main_manifest_prop
727     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
728     \stex_debug:nn{mathhub}{Current~repository:~
729     \prop_item:Nn \l_stex_current_repository_prop {id}
730   }
731 }
732 }

```

(End definition for `\l_stex_current_repository_prop`. This variable is documented on page 79.)

`\stex_in_repository:nn` Executes the code in the second argument in the context of the repository whose ID is provided as the first argument.

```

733 \cs_new_protected:Nn \stex_in_repository:nn {
734   \str_set:Nx \l_tmpa_str { #1 }
735   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
736   \str_if_empty:NTF \l_tmpa_str {
737     \prop_if_exist:NTF \l_stex_current_repository_prop {
738       \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
739       \exp_args:Ne \l_tmpa_cs{
740         \prop_item:Nn \l_stex_current_repository_prop { id }
741       }
742     }{
743       \l_tmpa_cs{}
744     }
745   }{

```

```

746 \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}
747 \stex_require_repository:n \l_tmpa_str
748 \str_set:Nx \l_tmpa_str { #1 }
749 \exp_args:Nne \use:nn {
750   \stex_set_current_repository:n \l_tmpa_str
751   \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
752 }{
753   \stex_debug:nn{mathhub}{switching~back~to:~
754     \prop_if_exist:NTF \l_stex_current_repository_prop {
755       \prop_item:Nn \l_stex_current_repository_prop { id }::~
756       \meaning\l_stex_current_repository_prop
757     }{
758       no~repository
759     }
760   }
761   \prop_if_exist:NTF \l_stex_current_repository_prop {
762     \stex_set_current_repository:n {
763       \prop_item:Nn \l_stex_current_repository_prop { id }
764     }
765   }{
766     \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
767   }
768 }
769 }
770 }

```

(End definition for `\stex_in_repository:nn`. This function is documented on page 79.)

27.5 Using Content in Archives

`\mhpath`

```

771 \def \mhpath #1 #2 {
772   \exp_args:Ne \tl_if_empty:nTF{#1}{
773     \c_stex_mathhub_str /
774     \prop_item:Nn \l_stex_current_repository_prop { id }
775     / source / #2
776   }{
777     \c_stex_mathhub_str / #1 / source / #2
778   }
779 }

```

(End definition for `\mhpath`. This function is documented on page 80.)

`\inputref`

`\mhinput`

```

780 \newif \ifinputref \inputreffalse
781
782 \cs_new_protected:Nn \__stex_mathhub_mhinput:nn {
783   \stex_in_repository:nn {#1} {
784     \ifinputref
785       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
786     \else
787       \inputreftrue
788       \input{ \c_stex_mathhub_str / ##1 / source / #2 }

```

```

789     \inputreffalse
790   \fi
791 }
792 }
793 \NewDocumentCommand \mhinput { 0{} m}{
794   \__stex_mathhub_mhinput:nn{ #1 }{ #2 }
795 }
796
797 \cs_new_protected:Nn \__stex_mathhub_inputref:nn {
798   \stex_in_repository:nn {#1} {
799     \stex_html_backend:TF {
800       \str_clear:N \l_tmpa_str
801       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
802         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
803       }
804
805       \tl_if_empty:nTF{ ##1 }{
806         \IfFileExists{#2}{
807           \stex_annotate_invisible:nnn{inputref}{
808             \l_tmpa_str / #2
809           }{}
810         }{
811           \input{#2}
812         }
813       }{
814         \IfFileExists{ \c_stex_mathhub_str / ##1 / source / #2 }{
815           \stex_annotate_invisible:nnn{inputref}{
816             \l_tmpa_str / #2
817           }{}
818         }{
819           \input{ \c_stex_mathhub_str / ##1 / source / #2 }
820         }
821       }
822
823     }{
824       \begingroup
825       \inputreftrue
826       \tl_if_empty:nTF{ ##1 }{
827         \input{#2}
828       }{
829         \input{ \c_stex_mathhub_str / ##1 / source / #2 }
830       }
831     } \endgroup
832   }
833 }
834 }
835 \NewDocumentCommand \inputref { 0{} m}{
836   \__stex_mathhub_inputref:nn{ #1 }{ #2 }
837 }

```

(End definition for `\inputref` and `\mhinput`. These functions are documented on page 80.)

`\addmhbibresource`

```

838 \cs_new_protected:Nn \__stex_mathhub_mhbibresource:nn {

```

```

839 \stex_in_repository:nn {#1} {
840   \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
841 }
842 }
843 \newcommand\addmhbibresource[2][]{
844   \__stex_mathhub_mhbibresource:nn{ #1 }{ #2 }
845 }

```

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

\libinput

```

846 \cs_new_protected:Npn \libinput #1 {
847   \prop_if_exist:NF \l_stex_current_repository_prop {
848     \msg_error:nnn{stex}{error/notinarchive}\libinput
849   }
850   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
851     \msg_error:nnn{stex}{error/notinarchive}\libinput
852   }
853   \seq_clear:N \l__stex_mathhub_libinput_files_seq
854   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
855   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
856
857   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
858     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #1.tex}
859     \IfFileExists{ \l_tmpa_str }{
860       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
861     }{}
862     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
863     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
864   }
865
866   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}
867   \IfFileExists{ \l_tmpa_str }{
868     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
869   }{}
870
871   \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
872     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
873   }{
874     \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
875       \input{ ##1 }
876     }
877   }
878 }

```

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

\libusepackage

```

879 \NewDocumentCommand \libusepackage {0{ } m} {
880   \prop_if_exist:NF \l_stex_current_repository_prop {
881     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
882   }
883   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
884     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
885   }

```

```

886 \seq_clear:N \l__stex_mathhub_libinput_files_seq
887 \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
888 \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
889
890 \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
891   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #2}
892   \IfFileExists{ \l_tmpa_str.sty }{
893     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
894   }{
895     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
896     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
897   }
898
899 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2}
900 \IfFileExists{ \l_tmpa_str.sty }{
901   \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
902 }{
903
904 \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
905   \msg_error:nxxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
906 }{
907   \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
908     \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
909       \usepackage[#1]{ #1 }
910     }
911   }{
912     \msg_error:nxxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
913   }
914 }
915 }

```

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

`\mhgraphics`
`\cmhgraphics`

```

916 \AddToHook{begindocument}{
917 \ltx@ifpackageloaded{graphicx}{
918   \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
919   \providecommand\mhgraphics[2] [] {%
920     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
921     \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}
922   }
923   \providecommand\cmhgraphics[2] [] {\begin{center}\mhgraphics[#1]{#2}\end{center}}
924 }{

```

(End definition for `\mhgraphics` and `\cmhgraphics`. These functions are documented on page 80.)

`\lstinputmhlisting`
`\clstinputmhlisting`

```

925 \ltx@ifpackageloaded{listings}{
926   \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
927   \newcommand\lstinputmhlisting[2] [] {%
928     \def\lst@mhrepos{}\setkeys{lst}{#1}%
929     \lstinputlisting[#1]{\mhp\lst@mhrepos{#2}}
930   }
931   \newcommand\clstinputmhlisting[2] [] {\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
932 }{

```

933

934 `\end{package}`

(End definition for `\lstinputmhlisting` and `\clstinputmhlisting`. These functions are documented on page 80.)

Chapter 28

STEX -References Implementation

```
935 <*package>
936
937 %%%%%%%%% stex-references.dtx %%%%%%%%%
938
939 <@@=stex_refs>
    Warnings and error messages
940 \msg_new:nnn{stex}{error/extrefmissing}{
941   Missing~in~or~cite~value~for~\detokenize{\extref}!
942 }
943 \msg_new:nnn{stex}{warning/smsmissing}{
944   .sref~file~#1~doesn't~exist!
945 }
946 \msg_new:nnn{stex}{warning/smslabelmissing}{
947   No~label~#2~in~.sref~file~#1!
948 }
```

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

```
949 \iow_new:N \c__stex_refs_refs_iow
950 \AtBeginDocument{
951   \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
952 }
953 \AtEndDocument{
954   \iow_close:N \c__stex_refs_refs_iow
955 }
```

28.1 Document URIs and URLs

`\l_stex_current_docns_str`

```
956 \str_new:N \l_stex_current_docns_str
```

(End definition for `\l_stex_current_docns_str`. This variable is documented on page 81.)

`\stex_get_document_uri:`

```
957 \cs_new_protected:Nn \stex_get_document_uri: {
```



```

958 \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
959 \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
960 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
961 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
962 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
963
964 \str_clear:N \l_tmpa_str
965 \prop_if_exist:NT \l_stex_current_repository_prop {
966   \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
967     \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
968   }
969 }
970
971 \str_if_empty:NTF \l_tmpa_str {
972   \str_set:Nx \l_stex_current_docns_str {
973     file:/\stex_path_to_string:N \l_tmpa_seq
974   }
975 }{
976   \bool_set_true:N \l_tmpa_bool
977   \bool_while_do:Nn \l_tmpa_bool {
978     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
979     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
980       {source} { \bool_set_false:N \l_tmpa_bool }
981     }{}{
982       \seq_if_empty:NT \l_tmpa_seq {
983         \bool_set_false:N \l_tmpa_bool
984       }
985     }
986   }
987
988   \seq_if_empty:NTF \l_tmpa_seq {
989     \str_gset_eq:NN \l_stex_current_docns_str \l_tmpa_str
990   }{
991     \str_gset:Nx \l_stex_current_docns_str {
992       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
993     }
994   }
995 }
996 %\stex_get_document_url:
997 }

```

(End definition for `\stex_get_document_uri`:. This function is documented on page 81.)

`\l_stex_current_docurl_str`

```

998 \str_new:N \l_stex_current_docurl_str

```

(End definition for `\l_stex_current_docurl_str`. This variable is documented on page 81.)

`\stex_get_document_url:`

```

999 \cs_new_protected:Nn \stex_get_document_url: {
1000   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1001   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1002   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1003   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1004   \seq_put_right:No \l_tmpa_seq \l_tmpb_str

```

```

1005
1006 \str_clear:N \l_tmpa_str
1007 \prop_if_exist:NT \l_stex_current_repository_prop {
1008   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
1009     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
1010       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
1011     }
1012   }
1013 }
1014
1015 \str_if_empty:NTF \l_tmpa_str {
1016   \str_set:Nx \l_stex_current_docurl_str {
1017     file:/\stex_path_to_string:N \l_tmpa_seq
1018   }
1019 }{
1020   \bool_set_true:N \l_tmpa_bool
1021   \bool_while_do:Nn \l_tmpa_bool {
1022     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1023     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1024       {source} { \bool_set_false:N \l_tmpa_bool }
1025     }{}{
1026       \seq_if_empty:NT \l_tmpa_seq {
1027         \bool_set_false:N \l_tmpa_bool
1028       }
1029     }
1030   }
1031
1032   \seq_if_empty:NTF \l_tmpa_seq {
1033     \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
1034   }{
1035     \str_set:Nx \l_stex_current_docurl_str {
1036       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
1037     }
1038   }
1039 }
1040 }

```

(End definition for `\stex_get_document_url`.. This function is documented on page [81](#).)

28.2 Setting Reference Targets

```

1041 \str_const:Nn \c__stex_refs_url_str{URL}
1042 \str_const:Nn \c__stex_refs_ref_str{REF}
1043 \str_new:N \l__stex_refs_curr_label_str
1044 % @currentlabel -> number
1045 % @currentlabelname -> title
1046 % @currentHref -> name.number <- id of some kind
1047 % @currentcounter <- name/id
1048 % \#autorefname <- "Section"
1049 % \theH# -> \arabic{section}
1050 % \the# -> number
1051 % \hyper@makecurrent{#}
1052 \int_new:N \l__stex_refs_unnamed_counter_int

```

Restoring references from .sref-files

\STEXInternalSrefRestoreTarget

1053 \cs_new_protected:Npn \STEXInternalSrefRestoreTarget #1#2#3#4#5 {}

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

\stex_ref_new_doc_target:n

```

1054 \seq_new:N \g_stex_ref_files_seq
1055
1056 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
1057   %\stex_get_document_uri:
1058   \str_clear:N \l__stex_refs_curr_label_str
1059   \str_set:Nx \l_tmpa_str { #1 }
1060   \str_if_empty:NT \l_tmpa_str {
1061     \int_gincr:N \l__stex_refs_unnamed_counter_int
1062     \str_set:Nx \l_tmpa_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
1063   }
1064   \str_set:Nx \l__stex_refs_curr_label_str {
1065     \l_stex_current_docns_str?\l_tmpa_str
1066   }
1067
1068   \exp_args:Noo \STEXInternalAuxAddDocRef\l_stex_current_docns_str\l_tmpa_str
1069
1070   %\seq_if_exist:cF{g__stex_refs_labels_\l_tmpa_str_seq}{
1071   %   \seq_new:c {g__stex_refs_labels_\l_tmpa_str_seq}
1072   %}
1073   %\seq_if_in:coF{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str {
1074   %   \seq_gput_right:co{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str
1075   %}
1076
1077
1078   \stex_if_smsmode:TF {
1079     %\stex_get_document_url:
1080     %\str_gset_eq:cN {sref_url_\l__stex_refs_curr_label_str_str}\l_stex_current_docurl_str
1081     %\str_gset_eq:cN {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_url_str
1082   }{
1083     \iow_now:Nx \c__stex_refs_refs_iow {
1084       \STEXInternalSrefRestoreTarget
1085       {\l_stex_current_docns_str}
1086       {\l_tmpa_str}
1087       {\@currentcounter}
1088       {\@currentlabel}
1089       {\tl_if_exist:NT\@currentlabelname{\exp_args:No\unexpanded\@currentlabelname}}
1090     }
1091     %\iow_now:Nx \c__stex_refs_refs_iow {
1092     %   {\l_stex_current_docns_str?\l_tmpa_str}~=={\use:c{\@currentcounter autorefname}~\@currentcounter}
1093     %   \stex_debug:nn{sref}{New~label~\l__stex_refs_curr_label_str~at~\use:c{\use:c{\@currentcounter}~\@currentcounter}}
1094     %   \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
1095     %   \immediate\write\auxout{\STEXInternalAuxAddDocRef{\l_stex_current_docns_str}{\l_tmpa_str}{\l__stex_refs_curr_label_str}}
1096     %   \str_gset:cx {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_ref_str
1097     %}
1098   }
1099   \NewDocumentCommand \slabel {m} {\stex_ref_new_doc_target:n {#1}}

```

(End definition for `\stex_ref_new_doc_target:n`. This function is documented on page 81.)

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

```

1100 \cs_new_protected:Npn \STEXInternalAuxAddDocRef #1 #2 {
1101   \exp_args:NNx \seq_if_in:NnTF \g_stex_ref_files_seq {\detokenize{#1}} {
1102     \exp_args:Nnx \seq_if_in:cnF{g_stex_ref_ #1 _seq}{\detokenize{#2}}{
1103       \exp_args:Nnx \seq_gput_left:cn{g_stex_ref_ #1 _seq}{\detokenize{#2}}
1104     }
1105   }{
1106     \exp_args:NNx \seq_gput_right:Nn \g_stex_ref_files_seq {\detokenize{#1}}
1107     %\seq_if_exist:cF{g_stex_ref_ #1 _seq}{
1108       \seq_new:c{g_stex_ref_ #1 _seq} % <- seq_new throws errors??
1109     %}
1110     \exp_args:Nnx \seq_gput_left:cn{g_stex_ref_ #1 _seq}{\detokenize{#2}}
1111   }
1112
1113   %\str_set:Nn \l_tmpa_str {#1?#2}
1114   %\str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
1115   %\seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
1116     % \seq_new:c {g__stex_refs_labels_#2_seq}
1117     %}
1118   %\seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
1119     % \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
1120     %}
1121 }

```

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

```

1122 \AtEndDocument{
1123   \def\STEXInternalAuxAddDocRef#1 #2 {}{}
1124 }

```

`\stex_ref_new_sym_target:n`

```

1125 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
1126
1127   % \stex_if_smsmode:TF {
1128   %   \str_if_exist:cF{sref_sym_#1_type}{
1129   %     \stex_get_document_url:
1130   %     \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str
1131   %     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
1132   %   }
1133   % }{
1134   %   \str_if_empty:NF \l__stex_refs_curr_label_str {
1135   %     \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
1136   %     \immediate\write\@auxout{
1137   %       \exp_not:N\expandafter\def\exp_not:N\csname \exp_not:N\detokenize{sref_sym_#1_label
1138   %       \l__stex_refs_curr_label_str
1139   %     }
1140   %   }
1141   % }
1142 % }
1143 }

```

(End definition for `\stex_ref_new_sym_target:n`. This function is documented on page 81.)

28.3 Using References

`\sref` Optional arguments:

```

1144
1145 \keys_define:nn { stex / sref / 1 } {
1146   archive .str_set_x:N = \l__stex_refs_repo_str,
1147   file     .str_set_x:N = \l__stex_refs_file_str,
1148   % TODO get rid of this
1149   fallback .code:n = {},
1150   pre      .code:n = {},
1151   post     .code:n = {}
1152 }
1153 \cs_new_protected:Nn \__stex_refs_args_i:n {
1154   \str_clear:N \l__stex_refs_repo_str
1155   \str_clear:N \l__stex_refs_file_str
1156   \keys_set:nn { stex / sref / 1 } { #1 }
1157 }
1158 \keys_define:nn { stex / sref / 2 } {
1159   in       .str_set_x:N = \l__stex_refs_in_str,
1160   archive  .str_set_x:N = \l__stex_refs_repob_str,
1161   title    .tl_set:N = \l__stex_refs_title_tl
1162 }
1163 \cs_new_protected:Nn \__stex_refs_args_ii:n {
1164   \str_clear:N \l__stex_refs_in_str
1165   \tl_clear:N \l__stex_refs_title_tl
1166   \str_clear:N \l__stex_refs_repob_str
1167   \keys_set:nn { stex / sref / 2 } { #1 }
1168 }

```

The actual macro:

```

1169 \NewDocumentCommand \sref { 0{} m 0{} }{
1170   \__stex_refs_args_i:n{#1}
1171   \__stex_refs_args_ii:n{#3}
1172   \str_clear:N \l__stex_refs_uri_str
1173   \__stex_refs_find_uri:n{#2}
1174   \__stex_refs_do_sref:n{#2}
1175 }
1176 \NewDocumentCommand \extref { 0{} m m }{
1177   \__stex_refs_args_i:n{#1}
1178   \__stex_refs_args_ii:n{#3}
1179   \str_if_empty:NT \l__stex_refs_in_str {
1180     \msg_error:nn{stex}{error/extrefmissing}
1181   }
1182   \str_clear:N \l__stex_refs_uri_str
1183   \__stex_refs_find_uri:n{#2}
1184   \__stex_refs_do_sref_in:n{#2}
1185 }
1186
1187 \cs_new_protected:Nn \__stex_refs_find_uri:n {
1188   \stex_debug:nn{sref}{File:~\l__stex_refs_file_str^^JRepo:\l__stex_refs_repo_str}
1189   \str_if_empty:NTF \l__stex_refs_file_str {
1190     \stex_debug:nn{sref}{Empty.~Checking~current~file~for~#1}
1191     \seq_if_exist:cT{g_stex_ref_\l_stex_current_docns_str_seq}{
1192       \seq_map_inline:cn{g_stex_ref_\l_stex_current_docns_str_seq}{

```

```

1193     \str_if_eq:nnT{#1}{##1}{
1194         \str_set_eq:NN \l__stex_refs_uri_str \l_stex_current_docns_str
1195         \stex_debug:nn{sref}{Found.}
1196         \seq_map_break:
1197     }
1198 }
1199 }
1200 \str_if_empty:NT \l__stex_refs_uri_str {
1201     \stex_debug:nn{sref}{Checking~other~files}
1202     \seq_map_inline:Nn \g_stex_ref_files_seq {
1203         \stex_debug:nn{sref}{##1...}
1204         \seq_map_inline:cn{g_stex_ref_##1_seq}{
1205             \str_if_eq:nnT{#1}{####1}{
1206                 \stex_debug:nn{sref}{Found~##1}
1207                 \str_set:Nn \l__stex_refs_uri_str {##1}
1208                 \seq_map_break:n{\seq_map_break:}
1209             }
1210         }
1211     }
1212 }
1213 }{
1214     \str_if_empty:NTF \l__stex_refs_repo_str {
1215         \prop_if_exist:NTF \l_stex_current_repository_prop {
1216             \prop_get:NnN \l_stex_current_repository_prop { ns } \l__stex_refs_uri_str
1217             \str_set:Nx \l__stex_refs_uri_str {\l__stex_refs_uri_str / \l__stex_refs_file_str}
1218             \stex_path_from_string:Nn \l_tmpb_seq \l__stex_refs_uri_str
1219             \str_set:Nx \l__stex_refs_uri_str {\stex_path_to_string:N \l_tmpb_seq}
1220         }{
1221             \stex_path_from_string:Nn \l_tmpb_seq {
1222                 \stex_path_to_string:N \g_stex_currentfile_seq/ .. / \l__stex_refs_file_str
1223             }
1224             \str_set:Nx \l__stex_refs_uri_str {file:/\stex_path_to_string:N \l_tmpb_seq}
1225         }
1226     }{
1227         \stex_require_repository:n \l__stex_refs_repo_str
1228         \prop_get:cnN { c_stex_mathhub_\l__stex_refs_repo_str_manifest_prop } { ns } \l__stex_refs_uri_str
1229         \str_set:Nx \l__stex_refs_uri_str {\l__stex_refs_uri_str / \l__stex_refs_file_str}
1230         \stex_path_from_string:Nn \l_tmpb_seq \l__stex_refs_uri_str
1231         \str_set:Nx \l__stex_refs_uri_str {\stex_path_to_string:N \l_tmpb_seq}
1232     }
1233 }
1234 }
1235
1236 \cs_new_protected:Nn \__stex_refs_do_autoref:n{
1237     \cs_if_exist:cTF{autoref}{
1238         \exp_args:Nx\autoref{sref_#1}
1239     }{
1240         \exp_args:Nx\ref{sref_#1}
1241     }
1242 }
1243
1244 \cs_new_protected:Nn \__stex_refs_do_sref:n {
1245     \str_if_empty:NTF \l__stex_refs_uri_str {
1246         \str_if_empty:NTF \l__stex_refs_in_str {

```

```

1247     \stex_debug:nn{sref}{autoref-on~#1}
1248     \__stex_refs_do_autoref:n{#1}
1249   }{
1250     \stex_debug:nn{sref}{srefin-on~#1}
1251     \__stex_refs_do_sref_in:n{#1}
1252   }
1253 }{
1254   \exp_args:NNo \seq_if_in:NnTF \g_stex_ref_files_seq \l__stex_refs_uri_str {
1255     \exp_args:Nnx \seq_if_in:cnTF{g_stex_ref\_l__stex_refs_uri_str_seq}{\detokenize{#1}}{
1256       \stex_debug:nn{sref}{Reference~found~in~ref~files;~autoref-on~#1}
1257       \__stex_refs_do_autoref:n{\l__stex_refs_uri_str?#1}
1258     }{
1259       \str_if_empty:NTF \l__stex_refs_in_str {
1260         \stex_debug:nn{sref}{in~empty;~autoref-on~#1}
1261         \__stex_refs_do_autoref:n{#1}
1262       }{
1263         \stex_debug:nn{sref}{in~non-empty;~srefin-on~#1}
1264         \__stex_refs_do_sref_in:n{#1}
1265       }
1266     }
1267   }{
1268     \str_if_empty:NTF \l__stex_refs_in_str {
1269       \stex_debug:nn{sref}{in~empty;~autoref-on~#1}
1270       \__stex_refs_do_autoref:n{#1}
1271     }{
1272       \stex_debug:nn{sref}{in~non-empty;~srefin-on~#1}
1273       \__stex_refs_do_sref_in:n{#1}
1274     }
1275   }
1276 }
1277 }
1278
1279 \cs_new_protected:Nn \__stex_refs_restore_target:nnnnn {
1280   \str_if_empty:NTF \l__stex_refs_uri_str {
1281     \exp_args:No \str_if_eq:nnT \l__stex_refs_id_str {#2}{
1282       \tl_set:Nn \l__stex_refs_return_tl {
1283         \use:c{#3autorefname}~#4\tl_if_empty:nF{#5}{~(5)}~in~
1284         \tl_if_empty:nTF\l__stex_refs_title_tl{
1285           ???
1286         }\l__stex_refs_title_tl
1287       }
1288     }
1289   }{
1290     \stex_debug:nn{sref}{\l__stex_refs_uri_str{~ == ~ #1 ~ ?}
1291     \exp_args:No \str_if_eq:nnT \l__stex_refs_uri_str {#1}{
1292       \stex_debug:nn{sref}{\l__stex_refs_id_str~ == ~ #2 ~ ?}
1293       \exp_args:No \str_if_eq:nnT \l__stex_refs_id_str {#2}{
1294         \stex_debug:nn{sref}{success!}
1295         \tl_set:Nn \l__stex_refs_return_tl {
1296           \use:c{#3autorefname}~#4\tl_if_empty:nF{#5}{~(5)}~in~
1297           \tl_if_empty:nTF\l__stex_refs_title_tl{
1298             ???
1299           }\l__stex_refs_title_tl
1300         }

```

```

1301         \endinput
1302     }
1303 }
1304 }
1305 }
1306
1307 \cs_new_protected:Nn \__stex_refs_do_sref_in:n {
1308   \stex_debug:nn{sref}{In: \l__stex_refs_in_str^^JRepo:\l__stex_refs_repo_str}
1309   \stex_debug:nn{sref}{URI: \l__stex_refs_uri_str?#1}
1310   %\msg_warning:nnn{stex}{warning/smsmissing}{<filename>}
1311   \begingroup\catcode13=9\relax\catcode10=9\relax
1312   \str_if_empty:NTF \l__stex_refs_repob_str {
1313     \prop_if_exist:NTF \l_stex_current_repository_prop {
1314       \str_set:Nx \l_tmpa_str {
1315         \c_stex_mathhub_str /
1316         \prop_item:Nn \l_stex_current_repository_prop { id }
1317         / source / \l__stex_refs_in_str .sref
1318       }
1319     }{
1320       \str_set:Nx \l_tmpa_str {
1321         \stex_path_to_string:N \g_stex_currentfile_seq/ .. / \l__stex_refs_in_str . sref
1322       }
1323     }
1324   }{
1325     \str_set:Nx \l_tmpa_str {
1326       \c_stex_mathhub_str / \l__stex_refs_repob_str
1327       / source / \l__stex_refs_in_str . sref
1328     }
1329   }
1330   \stex_path_from_string:Nn \l_tmpb_seq \l_tmpa_str
1331   \stex_path_to_string:NN \l_tmpb_seq \l_tmpa_str
1332   \stex_debug:nn{sref}{File: \l_tmpa_str}
1333   \exp_args:No \IfFileExists \l_tmpa_str {
1334     \tl_clear:N \l__stex_refs_return_tl
1335     \str_set:Nn \l__stex_refs_id_str {#1}
1336     \let\STEXInternalSrefRestoreTarget\__stex_refs_restore_target:nnnnn
1337     \use:c{@ @ input}{\l_tmpa_str}
1338     \exp_args:No \tl_if_empty:nTF \l__stex_refs_return_tl {
1339       \exp_args:Nnno \msg_warning:nnnn{stex}{warning/smslabelmissing}\l_tmpa_str{#1}
1340       \__stex_refs_do_autoref:n{
1341         \str_if_empty:NF\l__stex_refs_uri_str{\l__stex_refs_uri_str?}#1
1342       }
1343     }{
1344       \l__stex_refs_return_tl
1345     }
1346   }{
1347     \exp_args:Nnno \msg_warning:nnn{stex}{warning/smsmissing}\l_tmpa_str
1348     \__stex_refs_do_autoref:n{
1349       \str_if_empty:NF\l__stex_refs_uri_str{\l__stex_refs_uri_str?}#1
1350     }
1351   }
1352   \endgroup
1353 }
1354

```



```

1355 % \__stex_refs_args:n { #1 }
1356 % \str_if_empty:NTF \l__stex_refs_indocument_str {
1357 %   \str_set:Nx \l_tmpa_str { #2 }
1358 %   \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
1359 %   \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
1360 %     \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str_seq}{
1361 %       \seq_get_left:cNF {g__stex_refs_labels_\l_tmpa_str_seq} \l_tmpa_str {
1362 %         \str_clear:N \l_tmpa_str
1363 %       }
1364 %     }{
1365 %       \str_clear:N \l_tmpa_str
1366 %     }
1367 %   }{
1368 %     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1369 %     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1370 %     \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
1371 %     \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str_seq}{
1372 %       \str_set_eq:NN \l_tmpc_str \l_tmpa_str
1373 %       \str_clear:N \l_tmpa_str
1374 %       \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str_seq} {
1375 %         \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
1376 %           \str_range:nnn { ##1 }{-\l_tmpa_int}{ -1 }
1377 %         }{
1378 %           \seq_map_break:n {
1379 %             \str_set:Nn \l_tmpa_str { ##1 }
1380 %           }
1381 %         }
1382 %       }
1383 %     }{
1384 %       \str_clear:N \l_tmpa_str
1385 %     }
1386 %   }
1387 %   \str_if_empty:NTF \l_tmpa_str {
1388 %     \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_li
1389 %   }{
1390 %     \str_if_eq:cNTF {sref_\l_tmpa_str_type} \c__stex_refs_ref_str {
1391 %       \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1392 %         \cs_if_exist:cTF{autoref}{
1393 %           \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1394 %         }{
1395 %           \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1396 %         }
1397 %       }{
1398 %         \ltx@ifpackageloaded{hyperref}{
1399 %           \hyperref[sref_\l_tmpa_str]\l__stex_refs_linktext_tl
1400 %         }{
1401 %           \l__stex_refs_linktext_tl
1402 %         }
1403 %       }
1404 %     }{
1405 %       \ltx@ifpackageloaded{hyperref}{
1406 %         \href{\use:c{sref_url_\l_tmpa_str_str}}{\tl_if_empty:NTF \l__stex_refs_linktext_
1407 %       }{
1408 %         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_ref

```

```

1409 %      }
1410 %      }
1411 %    }
1412 %  }{
1413 %    % TODO
1414 %  }
1415 %}

```

(End definition for `\sref`. This function is documented on page 82.)

`\srefsym`

```

1416 \NewDocumentCommand \srefsym { 0{} m}{
1417   \stex_get_symbol:n { #2 }
1418   \__stex_refs_sym_aux:nn{#1}{\l_stex_get_symbol_uri_str}
1419 }
1420
1421 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1422
1423 % \str_if_exist:cTF {sref_sym_#2 _label_str }{
1424 %   \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1425 % }{
1426 %   \__stex_refs_args:n { #1 }
1427 %   \str_if_empty:NTF \l__stex_refs_indocument_str {
1428 %     \tl_if_exist:cTF{sref_sym_#2 _type}{
1429 %       % doc uri in \l_tmpb_str
1430 %       \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1431 %       \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1432 %         % reference
1433 %         \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1434 %           \cs_if_exist:cTF{autoref}{
1435 %             \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1436 %           }{
1437 %             \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1438 %           }
1439 %         }{
1440 %           \ltx@ifpackageloaded{hyperref}{
1441 %             \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1442 %           }{
1443 %             \l__stex_refs_linktext_tl
1444 %           }
1445 %         }
1446 %       }{
1447 %         % URL
1448 %         \ltx@ifpackageloaded{hyperref}{
1449 %           \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl
1450 %             }{
1451 %               \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_r
1452 %             }
1453 %         }
1454 %       }{
1455 %         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_
1456 %       }
1457 %     }{
1458 %       % TODO

```

```

1459 %   }
1460 %   }
1461 }

```

(End definition for \srefsym. This function is documented on page 82.)

\srefsymuri

```

1462 \cs_new_protected:Npn \srefsymuri #1 #2 { % TODO
1463   #2%\__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1464 }

```

(End definition for \srefsymuri. This function is documented on page 82.)

```

1465 \</package>

```

Chapter 29

STEX -Modules Implementation

```
1466 <*package>
1467
1468 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1469
1470 <@@=stex_modules>
1471
1472     Warnings and error messages
1473 \msg_new:nnn{stex}{error/unknownmodule}{
1474     No~module~#1~found
1475 }
1476 \msg_new:nnn{stex}{error/syntax}{
1477     Syntax~error:~#1
1478 }
1479 \msg_new:nnn{stex}{error/siglanguage}{
1480     Module~#1~declares~signature~#2,~but~does~not~
1481     declare~its~language
1482 }
1483 \msg_new:nnn{stex}{warning/deprecated}{
1484     #1~is~deprecated;~please~use~#2~instead!
1485 }
1486 \msg_new:nnn{stex}{error/conflictingmodules}{
1487     Conflicting~imports~for~module~#1
1488 }
```

`\l_stex_current_module_str` The current module:

```
1488 \str_new:N \l_stex_current_module_str
```

(End definition for \l_stex_current_module_str. This variable is documented on page 84.)

`\l_stex_all_modules_seq` Stores all available modules

```
1489 \seq_new:N \l_stex_all_modules_seq
```

(End definition for \l_stex_all_modules_seq. This variable is documented on page 84.)

```

\stex_if_in_module_p:
\stex_if_in_module:TF
1490 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1491   \str_if_empty:NTF \l_stex_current_module_str
1492   \prg_return_false: \prg_return_true:
1493 }

(End definition for \stex_if_in_module:TF. This function is documented on page 84.)

```

```

\stex_if_module_exists_p:n
\stex_if_module_exists:nTF
1494 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1495   \prop_if_exist:cTF { c_stex_module_#1_prop }
1496   \prg_return_true: \prg_return_false:
1497 }

(End definition for \stex_if_module_exists:nTF. This function is documented on page 84.)

```

```

\stex_add_to_current_module:n
\STEXexport
1498 \cs_new_protected:Nn \stex_execute_in_module:n { \stex_if_in_module:T {
1499   \stex_add_to_current_module:n { #1 }
1500   \stex_do_up_to_module:n { #1 }
1501 }}
1502 \cs_generate_variant:Nn \stex_execute_in_module:n {x}
1503
1504 \cs_new_protected:Nn \stex_add_to_current_module:n {
1505   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str_code} { #1 }
1506 }
1507 \cs_generate_variant:Nn \stex_add_to_current_module:n {x}
1508 \cs_new_protected:Npn \STEXexport {
1509   \ExplSyntaxOn
1510   \__stex_modules_export:n
1511 }
1512 \cs_new_protected:Nn \__stex_modules_export:n {
1513   \ignorespacesandpars#1\ExplSyntaxOff
1514   \stex_add_to_current_module:n { \ignorespacesandpars#1}
1515   \stex_smsmode_do:
1516 }
1517 \let \stex_module_export_helper:n \use:n
1518 \stex_deactivate_macro:Nn \STEXexport {module~environments}

(End definition for \stex_add_to_current_module:n and \STEXexport. These functions are documented
on page 84.)

```

```

\stex_add_constant_to_current_module:n
1519 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1520   \str_set:Nx \l_tmpa_str { #1 }
1521   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str_constants} { \l_tmpa_str }
1522 }

(End definition for \stex_add_constant_to_current_module:n. This function is documented on page
84.)

```

```

\stex_add_import_to_current_module:n
1523 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1524   \str_set:Nx \l_tmpa_str { #1 }
1525   \exp_args:Nno

```

```

1526 \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1527 \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1528 }
1529 }

```

(End definition for `\stex_add_import_to_current_module:n`. This function is documented on page 84.)

`\stex_collect_imports:n`

```

1530 \cs_new_protected:Nn \stex_collect_imports:n {
1531 \seq_clear:N \l_stex_collect_imports_seq
1532 \__stex_modules_collect_imports:n {#1}
1533 }
1534 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1535 \seq_map_inline:cn {c_stex_module_#1_imports} {
1536 \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1537 \__stex_modules_collect_imports:n { ##1 }
1538 }
1539 }
1540 \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1541 \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1542 }
1543 }

```

(End definition for `\stex_collect_imports:n`. This function is documented on page 84.)

`\stex_do_up_to_module:n`

```

1544 \int_new:N \l__stex_modules_group_depth_int
1545 \cs_new_protected:Nn \stex_do_up_to_module:n {
1546 \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1547 #1
1548 }{
1549 #1
1550 \expandafter \tl_gset:Nn
1551 \csname l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1552 \expandafter\expandafter\expandafter\endcsname
1553 \expandafter\expandafter\expandafter { \csname
1554 l__stex_modules_aftergroup_\l_stex_current_module_str _tl\endcsname #1 }
1555 \aftergroup\__stex_modules_aftergroup_do:
1556 }
1557 }
1558 \cs_generate_variant:Nn \stex_do_up_to_module:n {x}
1559 \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1560 \stex_debug:nn{aftergroup}{\cs_meaning:c{
1561 l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1562 }}}
1563 \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1564 \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1565 \tl_gclear:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1566 }{
1567 \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1568 \aftergroup\__stex_modules_aftergroup_do:
1569 }
1570 }
1571 \cs_new_protected:Nn \stex_reset_up_to_module:n {
1572 \expandafter\let\csname l__stex_modules_aftergroup_#1_tl\endcsname\undefined

```

1573 }

(End definition for `\stex_do_up_to_module:n`. This function is documented on page 84.)

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

1574

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

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

```
1575 \str_new:N \l_stex_module_ns_str
1576 \str_new:N \l_stex_module_subpath_str
1577 \cs_new_protected:Nn __stex_modules_compute_namespace:nN {
1578   \seq_set_eq:NN \l_tmpa_seq #2
1579   % split off file extension
1580   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str % <- filename
1581   \exp_args:Nnno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1582   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str % <- filename without suffixes
1583   \seq_put_right:No \l_tmpa_seq \l_tmpb_str % <- file path including name without suffixes
1584
1585   \bool_set_true:N \l_tmpa_bool
1586   \bool_while_do:Nn \l_tmpa_bool {
1587     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1588     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1589       {source} { \bool_set_false:N \l_tmpa_bool }
1590     }{}{
1591       \seq_if_empty:NT \l_tmpa_seq {
1592         \bool_set_false:N \l_tmpa_bool
1593       }
1594     }
1595   }
1596
1597   \stex_path_to_string:NN \l_tmpa_seq \l_stex_module_subpath_str
1598   % \l_tmpa_seq <- sub-path relative to archive
1599   \str_if_empty:NTF \l_stex_module_subpath_str {
1600     \str_set:Nx \l_stex_module_ns_str {#1}
1601   }{
1602     \str_set:Nx \l_stex_module_ns_str {
1603       #1/\l_stex_module_subpath_str
1604     }
1605   }
1606 }
1607
1608 \cs_new_protected:Nn \stex_modules_current_namespace: {
1609   \str_clear:N \l_stex_module_subpath_str
1610   \prop_if_exist:NTF \l_stex_current_repository_prop {
1611     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1612     __stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1613   }{
1614     % split off file extension
1615     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1616     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
```

```

1617 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1618 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1619 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1620 \str_set:Nx \l_stex_module_ns_str {
1621   file:/\stex_path_to_string:N \l_tmpa_seq
1622 }
1623 }
1624 }

```

(End definition for `\stex_modules_current_namespace:.` This function is documented on page 85.)

29.1 The smodule environment

smodule arguments:

```

1625 \keys_define:nn { stex / module } {
1626   title      .tl_set:N      = \smodulename ,
1627   type       .str_set_x:N   = \smodulename ,
1628   id         .str_set_x:N   = \smoduleid ,
1629   deprecate  .str_set_x:N   = \l_stex_module_deprecate_str ,
1630   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1631   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1632   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1633   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1634   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1635   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1636   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1637 }
1638
1639 \cs_new_protected:Nn \__stex_modules_args:n {
1640   \str_clear:N \smodulename
1641   \str_clear:N \smodulename
1642   \str_clear:N \smoduleid
1643   \str_clear:N \l_stex_module_ns_str
1644   \str_clear:N \l_stex_module_deprecate_str
1645   \str_clear:N \l_stex_module_lang_str
1646   \str_clear:N \l_stex_module_sig_str
1647   \str_clear:N \l_stex_module_creators_str
1648   \str_clear:N \l_stex_module_contributors_str
1649   \str_clear:N \l_stex_module_meta_str
1650   \str_clear:N \l_stex_module_srccite_str
1651   \keys_set:nn { stex / module } { #1 }
1652 }
1653
1654 % module parameters here? In the body?
1655

```

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

```

1656 \cs_new_protected:Nn \stex_module_setup:nn {
1657   \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1658   \str_set:Nx \l_stex_module_name_str { #2 }
1659   \__stex_modules_args:n { #1 }

```


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

Are we in a nested module?

```

1660 \stex_if_in_module:TF {
1661   % Nested module
1662   \prop_get:cnN {c_stex_module\_l_stex_current_module_str _prop}
1663   { ns } \l_stex_module_ns_str
1664   \str_set:Nx \l_stex_module_name_str {
1665     \prop_item:cn {c_stex_module\_l_stex_current_module_str _prop}
1666     { name } / \l_stex_module_name_str
1667   }
1668   \str_if_empty:NT \l_stex_module_lang_str {
1669     \str_set:Nx \l_stex_module_lang_str {
1670       \prop_item:cn {c_stex_module\_l_stex_current_module_str _prop}
1671       { lang }
1672     }
1673   }
1674 }{
1675   % not nested:
1676   \str_if_empty:NT \l_stex_module_ns_str {
1677     \stex_modules_current_namespace:
1678     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1679     / {\l_stex_module_ns_str}
1680     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1681     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1682       \str_set:Nx \l_stex_module_ns_str {
1683         \stex_path_to_string:N \l_tmpa_seq
1684       }
1685     }
1686   }
1687 }

```

Next, we determine the language of the module:

```

1688 \str_if_empty:NT \l_stex_module_lang_str {
1689   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1690   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1691   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1692   \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
1693     \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
1694       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
1695     }
1696   }
1697   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1698   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
1699     \seq_pop_right:NN \l_tmpa_seq \l_stex_module_lang_str
1700     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1701       inferred~from~file~name}
1702   }
1703 }
1704
1705 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
1706   \exp_args:NNo \stex_set_language:Nn \l_tmpa_str \l_stex_module_lang_str
1707 }}

```

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

```

1708 \str_if_empty:NTF \l_stex_module_sig_str {
1709   \exp_args:Nnx \prop_gset_from_keyval:cn {
1710     c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1711   } {
1712     name      = \l_stex_module_name_str ,
1713     ns        = \l_stex_module_ns_str ,
1714     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1715     lang      = \l_stex_module_lang_str ,
1716     sig       = \l_stex_module_sig_str ,
1717     deprecate = \l_stex_module_deprecate_str ,
1718     meta      = \l_stex_module_meta_str
1719   }
1720   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1721   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1722   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _copymodules}
1723   \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1724   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1725 \str_if_empty:NT \l_stex_module_meta_str {
1726   \str_set:Nx \l_stex_module_meta_str {
1727     \c_stex_metatheory_ns_str ? Metatheory
1728   }
1729 }
1730 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1731   \bool_set_true:N \l_stex_in_meta_bool
1732   \exp_args:Nx \stex_add_to_current_module:n {
1733     \bool_set_true:N \l_stex_in_meta_bool
1734     \stex_activate_module:n {\l_stex_module_meta_str}
1735     \bool_set_false:N \l_stex_in_meta_bool
1736   }
1737   \stex_activate_module:n {\l_stex_module_meta_str}
1738   \bool_set_false:N \l_stex_in_meta_bool
1739 }
1740 }{
1741   \str_if_empty:NT \l_stex_module_lang_str {
1742     \msg_error:nnxx{stex}{error/siglanguage}{
1743       \l_stex_module_ns_str?\l_stex_module_name_str
1744     }{\l_stex_module_sig_str}
1745   }
1746   \stex_debug:nn{modules}{Signature~\l_stex_module_sig_str~for~\l_stex_module_ns_str?\l_st
1747   \stex_if_module_exists:nTF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1748     \stex_debug:nn{modules}{(already exists)}
1749   }{
1750     \stex_debug:nn{modules}{(needs loading)}
1751     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1752     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1753     \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1754     \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1755     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1756     \str_set:Nx \l_tmpa_str {
1757       \stex_path_to_string:N \l_tmpa_seq /

```

```

1758     \l_tmpa_str . \l_stex_module_sig_str .tex
1759   }
1760   \IfFileExists \l_tmpa_str {
1761     \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1762       \str_clear:N \l_stex_current_module_str
1763       \seq_clear:N \l_stex_all_modules_seq
1764       \stex_debug:nn{modules}{Loading~signature}
1765     }
1766   }{
1767     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1768   }
1769 }
1770 \stex_if_smsmode:F {
1771   \stex_activate_module:n {
1772     \l_stex_module_ns_str ? \l_stex_module_name_str
1773   }
1774 }
1775 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1776 }
1777 \str_if_empty:NF \l_stex_module_deprecate_str {
1778   \msg_warning:nnxx{stex}{warning/deprecated}{
1779     Module~\l_stex_current_module_str
1780   }{
1781     \l_stex_module_deprecate_str
1782   }
1783 }
1784 \seq_put_right:Nx \l_stex_all_modules_seq {
1785   \l_stex_module_ns_str ? \l_stex_module_name_str
1786 }
1787 \tl_clear:c{l__stex_modules_aftergroup_\l_stex_module_ns_str ? \l_stex_module_name_str _tl}
1788 }

```

(End definition for `\stex_module_setup:nn`. This function is documented on page 85.)

smodule (*env.*) The module environment.

```

\__stex_modules_begin_module: implements \begin{smodule}
1789 \cs_new_protected:Nn \__stex_modules_begin_module: {
1790   \stex_reactivate_macro:N \STEXexport
1791   \stex_reactivate_macro:N \importmodule
1792   \stex_reactivate_macro:N \symdecl
1793   \stex_reactivate_macro:N \notation
1794   \stex_reactivate_macro:N \symdef
1795
1796   \stex_debug:nn{modules}{
1797     New~module:\\
1798     Namespace:~\l_stex_module_ns_str\\
1799     Name:~\l_stex_module_name_str\\
1800     Language:~\l_stex_module_lang_str\\
1801     Signature:~\l_stex_module_sig_str\\
1802     Metatheory:~\l_stex_module_meta_str\\
1803     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1804   }
1805

```

```

1806 \stex_if_do_html:T{
1807   \begin{stex_annotate_env} {theory} {
1808     \l_stex_module_ns_str ? \l_stex_module_name_str
1809   }
1810
1811   \stex_annotate_invisible:nnn{header}{} {
1812     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1813     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1814     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1815       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1816     }
1817     \str_if_empty:NF \smoduletype {
1818       \stex_annotate:nnn{type}{\smoduletype}{}
1819     }
1820   }
1821 }
1822 % TODO: Inherit metatheory for nested modules?
1823 }
1824 \iffalse \end{stex_annotate_env} \fi %^^A make syntax highlighting work again

```

(End definition for `_stex_modules_begin_module:.`)

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

```

1825 \cs_new_protected:Nn \_stex_modules_end_module: {
1826   \stex_debug:nn{modules}{Closing module~\prop_item:cn {c_stex_module\_l_stex_current_module}
1827   \stex_reset_up_to_module:n \l_stex_current_module_str
1828   \stex_if_smsmode:T {
1829     \stex_persist:x {
1830       \prop_set_from_keyval:cn{c_stex_module\_l_stex_current_module_str _prop}{
1831         \exp_after:wN \prop_to_keyval:N \csname c_stex_module\_l_stex_current_module_str _pr
1832       }
1833       \seq_set_from_clist:cn{c_stex_module\_l_stex_current_module_str _constants}{
1834         \seq_use:cn{c_stex_module\_l_stex_current_module_str _constants},
1835       }
1836       \seq_set_from_clist:cn{c_stex_module\_l_stex_current_module_str _imports}{
1837         \seq_use:cn{c_stex_module\_l_stex_current_module_str _imports},
1838       }
1839       \tl_set:cn {c_stex_module\_l_stex_current_module_str _code}
1840     }
1841     \exp_after:wN \let \exp_after:wN \l_tmpa_tl \csname c_stex_module\_l_stex_current_module
1842     \exp_after:wN \stex_persist:n \exp_after:wN { \exp_after:wN { \l_tmpa_tl } }
1843   }
1844 }

```

(End definition for `_stex_modules_end_module:.`)

The core environment

```

1845 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1846 \NewDocumentEnvironment { smodule } { 0 } { m } {
1847   \stex_module_setup:nn{#1}{#2}
1848   %\par
1849   \stex_if_smsmode:F{
1850     \tl_if_empty:NF \smoduletitle {
1851       \exp_args:No \stex_document_title:n \smoduletitle
1852     }

```

```

1853 \tl_clear:N \l_tmpa_tl
1854 \clist_map_inline:Nn \smodulotype {
1855   \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1856     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1857   }
1858 }
1859 \tl_if_empty:NTF \l_tmpa_tl {
1860   \__stex_modules_smodule_start:
1861 }{
1862   \l_tmpa_tl
1863 }
1864 }
1865 \__stex_modules_begin_module:
1866 \str_if_empty:NF \smoduleid {
1867   \stex_ref_new_doc_target:n \smoduleid
1868 }
1869 \stex_smsmode_do:
1870 } {
1871   \__stex_modules_end_module:
1872   \stex_if_smsmode:F {
1873     \end{stex_annotate_env}
1874     \clist_set:Nn \l_tmpa_clist \smodulotype
1875     \tl_clear:N \l_tmpa_tl
1876     \clist_map_inline:Nn \l_tmpa_clist {
1877       \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1878         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1879       }
1880     }
1881     \tl_if_empty:NTF \l_tmpa_tl {
1882       \__stex_modules_smodule_end:
1883     }{
1884       \l_tmpa_tl
1885     }
1886   }
1887 }

```

\stexpatchmodule

```

1888 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1889 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1890
1891 \newcommand\stexpatchmodule[3] [] {
1892   \str_set:Nx \l_tmpa_str{ #1 }
1893   \str_if_empty:NTF \l_tmpa_str {
1894     \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1895     \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1896   }{
1897     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1898     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1899   }
1900 }

```

(End definition for \stexpatchmodule. This function is documented on page 85.)

29.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n
1901 \NewDocumentCommand \STEXModule { m } {
1902   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1903   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1904   \tl_set:Nn \l_tmpa_tl {
1905     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1906   }
1907   \seq_map_inline:Nn \l_stex_all_modules_seq {
1908     \str_set:Nn \l_tmpb_str { ##1 }
1909     \str_if_eq:eeT { \l_tmpa_str } {
1910       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1911     } {
1912       \seq_map_break:n {
1913         \tl_set:Nn \l_tmpa_tl {
1914           \stex_invoke_module:n { ##1 }
1915         }
1916       }
1917     }
1918   }
1919   \l_tmpa_tl
1920 }
1921
1922 \cs_new_protected:Nn \stex_invoke_module:n {
1923   \stex_debug:nn{modules}{Invoking~module~#1}
1924   \peek_charcode_remove:NTF ! {
1925     \__stex_modules_invoke_uri:nN { #1 }
1926   } {
1927     \peek_charcode_remove:NTF ? {
1928       \__stex_modules_invoke_symbol:nn { #1 }
1929     } {
1930       \msg_error:nnx{stex}{error/syntax}{
1931         ?~or~!~expected~after~
1932         \c_backslash_str STEXModule{#1}
1933       }
1934     }
1935   }
1936 }
1937
1938 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1939   \str_set:Nn #2 { #1 }
1940 }
1941
1942 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1943   \stex_invoke_symbol:n{#1?#2}
1944 }

```

(End definition for `\STEXModule` and `\stex_invoke_module:n`. These functions are documented on page 85.)

```

\stex_activate_module:n
1945 \bool_new:N \l_stex_in_meta_bool
1946 \bool_set_false:N \l_stex_in_meta_bool

```

```

1947 \cs_new_protected:Nn \stex_activate_module:n {
1948   \stex_debug:nn{modules}{Activating~module~#1}
1949   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1950     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1951     \use:c{ c_stex_module_#1_code }
1952   }
1953 }

```

(End definition for `\stex_activate_module:n`. This function is documented on page 86.)

`mmtinterface` (*env.*)

```

1954 \NewDocumentEnvironment { mmtinterface } { 0{} m m } {
1955   \begin{smodule}[#1]{#3}
1956   \str_set:Nx \l_stex_module_mmtfor_str {#2}
1957   \MMTinclude{#2}
1958   \stex_reactivate_macro:N \mmtdecl
1959   \stex_reactivate_macro:N \mmtdef
1960 }{
1961   \end{smodule}
1962 }

1963 \</package>

```

Chapter 30

STEX -Module Inheritance Implementation

```
1964 <*package>
1965
1966 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1967
```

30.1 SMS Mode

```
1968 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1969 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1970 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1971 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1972
1973 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1974   \makeatletter
1975   \makeatother
1976   \ExplSyntaxOn
1977   \ExplSyntaxOff
1978   \rustexBREAK
1979 }
1980
1981 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1982   \symdef
1983   \importmodule
1984   \notation
1985   \symdecl
1986   \STEXexport
1987   \inlineass
1988   \inlinedef
1989   \inlineex
1990   \endinput
1991   \setnotation
```



```

1992 \copynotation
1993 \assign
1994 \renamedekl
1995 \donotcopy
1996 \instantiate
1997 \textsymdecl
1998 }
1999
2000 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
2001   \tl_to_str:n {
2002     smodule,
2003     copymodule,
2004     interpretmodule,
2005     realization,
2006     sdefinition,
2007     sexample,
2008     sassertion,
2009     sparagraph,
2010     mathstructure,
2011     extstructure,
2012     extstructure*
2013   }
2014 }

```

(End definition for `\g_stex_smsmode_allowedmacros_tl`, `\g_stex_smsmode_allowedmacros_escape_tl`, and `\g_stex_smsmode_allowedenvs_seq`. These variables are documented on page 87.)

`\stex_if_smsmode_p:`

`\stex_if_smsmode:TF`

```

2015 \bool_new:N \g__stex_smsmode_bool
2016 \bool_set_false:N \g__stex_smsmode_bool
2017 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
2018   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
2019 }

```

(End definition for `\stex_if_smsmode:TF`. This function is documented on page 87.)

`__stex_smsmode_in_smsmode:nn`

```

2020 \cs_new_protected:Nn \__stex_smsmode_in_smsmode:nn { \stex_suppress_html:n {
2021   \vbox_set:Nn \l_tmpa_box {
2022     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
2023     \bool_gset_true:N \g__stex_smsmode_bool
2024     #2
2025     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
2026   }
2027   \box_clear:N \l_tmpa_box
2028 } }

```

(End definition for `__stex_smsmode_in_smsmode:nn`.)

`\stex_file_in_smsmode:nn`

```

2029 \quark_new:N \q__stex_smsmode_break
2030
2031 \NewDocumentCommand \__stex_smsmode_importmodule: { 0{} m } {
2032   \seq_gput_right:Nn \l__stex_smsmode_importmodules_seq {{#1}{#2}}
2033   \stex_smsmode_do:

```

```

2034 }
2035
2036 \cs_new_protected:Nn \__stex_smsmode_module:nn {
2037   \__stex_modules_args:n{#1}
2038   \stex_if_in_module:F {
2039     \str_if_empty:NF \l_stex_module_sig_str {
2040       \stex_modules_current_namespace:
2041       \str_set:Nx \l_stex_module_name_str { #2 }
2042       \stex_if_module_exists:nF{\l_stex_module_ns_str?\l_stex_module_name_str}{
2043         \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
2044         \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
2045         \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
2046         \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
2047         \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
2048         \str_set:Nx \l_tmpa_str {
2049           \stex_path_to_string:N \l_tmpa_seq /
2050           \l_tmpa_str . \l_stex_module_sig_str .tex
2051         }
2052         \IfFileExists \l_tmpa_str {
2053           \exp_args:NNx \seq_gput_right:Nn \l__stex_smsmode_sigmodules_seq \l_tmpa_str
2054         }{
2055           \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
2056         }
2057       }
2058     }
2059   }
2060 }
2061
2062 \prg_new_conditional:Nnn \__stex_smsmode_check_import_pair:nn {T,F,TF} {
2063   %\stex_debug:nn{import-pair}{\detokenize{#1}~{#2}}
2064   \tl_if_empty:nTF{#1}{
2065     \prop_if_exist:NTF \l_stex_current_repository_prop
2066     {
2067       %\stex_debug:nn{import-pair}{in repository \prop_item:Nn \l_stex_current_repository_
2068       \prg_return_true:
2069     } {
2070       \seq_set_split:Nnn \l_tmpa_seq ? {#2}
2071       \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
2072       \tl_if_empty:NT \l_tmpa_tl {
2073         \seq_pop_left:NN \l_tmpa_seq \l_tmpa_tl
2074       }
2075       %\stex_debug:nn{import-pair}{\seq_use:Nn \l_tmpa_seq,~of~length~\seq_count:N \l_tmpa
2076       \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1
2077       \prg_return_true: \prg_return_false:
2078     }
2079   }\prg_return_true:
2080 }
2081
2082 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
2083   \stex_filestack_push:n{#1}
2084   \seq_gclear:N \l__stex_smsmode_importmodules_seq
2085   \seq_gclear:N \l__stex_smsmode_sigmodules_seq
2086   % ----- new -----
2087   \__stex_smsmode_in_smsmode:nn{#1}{

```

```

2088 \let\importmodule\__stex_smsmode_importmodule:
2089 \let\stex_module_setup:nn\__stex_smsmode_module:nn
2090 \let\__stex_modules_begin_module:\relax
2091 \let\__stex_modules_end_module:\relax
2092 \seq_clear:N \g_stex_smsmode_allowedenvs_seq
2093 \exp_args:NNx \seq_put_right:Nn \g_stex_smsmode_allowedenvs_seq {\tl_to_str:n{smodule}}
2094 \tl_clear:N \g_stex_smsmode_allowedmacros_tl
2095 \tl_clear:N \g_stex_smsmode_allowedmacros_escape_tl
2096 \tl_put_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\importmodule}
2097 \everyeof{\q__stex_smsmode_break\noexpand}
2098 \expandafter\expandafter\expandafter
2099 \stex_smsmode_do:
2100 \csname @ @ input\endcsname "#1"\relax
2101
2102 \seq_map_inline:Nn \l__stex_smsmode_sigmodules_seq {
2103   \stex_filestack_push:n{##1}
2104   \expandafter\expandafter\expandafter
2105   \stex_smsmode_do:
2106   \csname @ @ input\endcsname "##1"\relax
2107   \stex_filestack_pop:
2108 }
2109 }
2110 % ----- new -----
2111 \__stex_smsmode_in_smsmode:nn{#1} {
2112   #2
2113   % ----- new -----
2114   \begin{group}
2115   %\stex_debug:nn{smsmode}{Here:~\seq_use:Nn\l__stex_smsmode_importmodules_seq, }
2116   \seq_map_inline:Nn \l__stex_smsmode_importmodules_seq {
2117     \__stex_smsmode_check_import_pair:nnT ##1 { \begin{group}
2118       \stex_import_module_uri:nn ##1
2119       \stex_import_require_module:nnnn
2120       \l_stex_import_ns_str
2121       \l_stex_import_archive_str
2122       \l_stex_import_path_str
2123       \l_stex_import_name_str \end{group}
2124     }
2125   }
2126   \end{group}
2127   \stex_debug:nn{smsmode}{Actually~loading~file~#1}
2128   % ----- new -----
2129   \everyeof{\q__stex_smsmode_break\noexpand}
2130   \expandafter\expandafter\expandafter
2131   \stex_smsmode_do:
2132   \csname @ @ input\endcsname "#1"\relax
2133 }
2134 \stex_filestack_pop:
2135 }

```

(End definition for `\stex_file_in_smsmode:nn`. This function is documented on page 88.)

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

```

2136 \cs_new_protected:Npn \stex_smsmode_do: {

```

```

2137 \stex_if_smsmode:T {
2138   \__stex_smsmode_do:w
2139 }
2140 }
2141 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
2142   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
2143     \expandafter\if\expandafter\relax\noexpand#1
2144     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
2145   \else\expandafter\__stex_smsmode_do:w\fi
2146 }{
2147   \__stex_smsmode_do:w % #1
2148 }
2149 }
2150 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
2151   \cs_if_eq:NNTF #1 \q__stex_smsmode_break {
2152     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
2153       #1\__stex_smsmode_do:w
2154     }{
2155       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
2156         #1
2157       }{
2158         \cs_if_eq:NNTF \begin #1 {
2159           \__stex_smsmode_check_begin:n
2160         }{
2161           \cs_if_eq:NNTF \end #1 {
2162             \__stex_smsmode_check_end:n
2163           }{
2164             \__stex_smsmode_do:w
2165           }
2166         }
2167       }
2168     }
2169   }
2170 }
2171
2172 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
2173   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
2174     \begin{#1}
2175   }{
2176     \__stex_smsmode_do:w
2177   }
2178 }
2179 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
2180   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
2181     \end{#1}\__stex_smsmode_do:w
2182   }{
2183     \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
2184   }
2185 }

```

(End definition for `\stex_smsmode_do:.` This function is documented on page 88.)

30.2 Inheritance

```
2186 <@@=stex_importmodule>
```

```
\stex_import_module_uri:nn
```

```

2187 \cs_new_protected:Nn \stex_import_module_uri:nn {
2188   \str_set:Nx \l_stex_import_archive_str { #1 }
2189   \str_set:Nn \l_stex_import_path_str { #2 }
2190
2191   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
2192   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
2193   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
2194
2195   \stex_modules_current_namespace:
2196   \bool_lazy_all:nTF {
2197     {\str_if_empty_p:N \l_stex_import_archive_str}
2198     {\str_if_empty_p:N \l_stex_import_path_str}
2199     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
2200   }{
2201     \str_set_eq:NN \l_stex_import_path_str \l_stex_module_subpath_str
2202     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
2203   }{
2204     \str_if_empty:NT \l_stex_import_archive_str {
2205       \prop_if_exist:NT \l_stex_current_repository_prop {
2206         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
2207       }
2208     }
2209     \str_if_empty:NTF \l_stex_import_archive_str {
2210       \str_if_empty:NF \l_stex_import_path_str {
2211         \stex_path_from_string:Nn \l_tmpb_seq {
2212           \l_stex_module_ns_str / .. / \l_stex_import_path_str
2213         }
2214         \str_set:Nx \l_stex_import_ns_str {\stex_path_to_string:N \l_tmpb_seq}
2215         \str_replace_once:Nnn \l_stex_import_ns_str {file:/} {file://}
2216       }
2217     }{
2218       \stex_require_repository:n \l_stex_import_archive_str
2219       \prop_get:cnN { c_stex_mathhub\_l_stex_import_archive_str _manifest_prop } { ns }
2220       \l_stex_import_ns_str
2221       \str_if_empty:NF \l_stex_import_path_str {
2222         \str_set:Nx \l_stex_import_ns_str {
2223           \l_stex_import_ns_str / \l_stex_import_path_str
2224         }
2225       }
2226     }
2227   }
2228 }

```

(End definition for `\stex_import_module_uri:nn`. This function is documented on page 89.)

[illegible]

(End definition for \l stex import name str and others. These variables are documented on page 89.)

```

\stex_import_require_module:nnnnn {{\ns}} {{\archive-ID}} {{\path}} {{\name}}
2233 \cs_new_protected:Nn \stex_import_require_module:nnnnn {
2234 \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
2235
2236 \stex_debug:nn{requiremodule}{Here:\\~1:~#1\\~2:~#2\\~3:~#3\\~4:~#4}
2237
2238 \exp_args:NNxx \seq_set_split:Nnn \l_tmpa_seq {\tl_to_str:n{/}} {#4}
2239 \seq_get_left:NN \l_tmpa_seq \l_tmpc_str
2240
2241 %\stex_debug:nn{requiremodule}{Top~module:\l_tmpc_str}
2242
2243 % archive
2244 \str_set:Nx \l_tmpa_str { #2 }
2245 \str_if_empty:NTF \l_tmpa_str {
2246 \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
2247 \seq_put_right:Nn \l_tmpa_seq {...}
2248 } {
2249 \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
2250 \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
2251 \seq_put_right:Nn \l_tmpa_seq { source }
2252 }
2253
2254 % path
2255 \str_set:Nx \l_tmpb_str { #3 }
2256 \str_if_empty:NTF \l_tmpb_str {
2257 \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / \l_tmpc_str }
2258
2259 \ltx@ifpackageloaded{babel} {
2260 \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
2261 { \language } \l_tmpb_str {
2262 \msg_error:nnx{stex}{error/unknownlanguage}{\language}
2263 }
2264 } {
2265 \str_clear:N \l_tmpb_str
2266 }
2267
2268 \stex_debug:nn{modules}{Checking~a1~\l_tmpa_str.\l_tmpb_str.tex}
2269 \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2270 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2271 }{
2272 \stex_debug:nn{modules}{Checking~a2~\l_tmpa_str.tex}
2273 \IfFileExists{ \l_tmpa_str.tex }{
2274 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2275 }{
2276 % try english as default
2277 \stex_debug:nn{modules}{Checking~a3~\l_tmpa_str.en.tex}
2278 \IfFileExists{ \l_tmpa_str.en.tex }{
2279 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2280 }{
2281 \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2282 }
2283 }
2284 }
2285

```

```

2286 } {
2287   \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
2288   \seq_concat:NNN \l_tmpb_seq \l_tmpa_seq \l_tmpb_seq
2289
2290   \ltx@ifpackageloaded{babel} {
2291     \exp_args:NnX \prop_get:NnNF \c_stex_language_abbrevs_prop
2292       { \language } \l_tmpb_str {
2293       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
2294     }
2295   } {
2296     \str_clear:N \l_tmpb_str
2297   }
2298
2299   \stex_path_canonicalize:N \l_tmpb_seq
2300   \stex_path_to_string:NN \l_tmpb_seq \l_tmpa_str
2301
2302   \stex_debug:nn{modules}{Checking~b1~\l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex}
2303   \IfFileExists{ \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }{
2304     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }
2305   }{
2306     \stex_debug:nn{modules}{Checking~b2~\l_tmpa_str/\l_tmpc_str.tex}
2307     \IfFileExists{ \l_tmpa_str/\l_tmpc_str.tex }{
2308       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.tex }
2309     }{
2310       % try english as default
2311       \stex_debug:nn{modules}{Checking~b3~\l_tmpa_str/\l_tmpc_str.en.tex}
2312       \IfFileExists{ \l_tmpa_str/\l_tmpc_str.en.tex }{
2313         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.en.tex }
2314       }{
2315         \stex_debug:nn{modules}{Checking~b4~\l_tmpa_str.\l_tmpb_str.tex}
2316         \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2317           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2318         }{
2319           \stex_debug:nn{modules}{Checking~b4~\l_tmpa_str.tex}
2320           \IfFileExists{ \l_tmpa_str.tex }{
2321             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2322           }{
2323             % try english as default
2324             \stex_debug:nn{modules}{Checking~b5~\l_tmpa_str.en.tex}
2325             \IfFileExists{ \l_tmpa_str.en.tex }{
2326               \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2327             }{
2328               \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2329             }
2330           }
2331         }
2332       }
2333     }
2334   }
2335 }
2336
2337 \str_if_eq:eeF{\g__stex_importmodule_file_str}{\seq_use:Nn \g_stex_currentfile_seq /}{
2338   \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
2339     \seq_clear:N \l_stex_all_modules_seq

```

```

2340     \str_clear:N \l_stex_current_module_str
2341     \str_set:Nx \l_tmpb_str { #2 }
2342     \str_if_empty:NF \l_tmpb_str {
2343       \stex_set_current_repository:n { #2 }
2344     }
2345     \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
2346   }
2347
2348   \stex_if_module_exists:nF { #1 ? #4 } {
2349     \msg_error:nnx{stex}{error/unknownmodule}{
2350       #1?#4~(in~file~\g__stex_importmodule_file_str)
2351     }
2352   }
2353 }
2354
2355 }
2356 \stex_activate_module:n { #1 ? #4 }
2357 }

```

(End definition for `\stex_import_require_module:nnnn`. This function is documented on page 89.)

`\importmodule`

```

2358 \NewDocumentCommand \importmodule { 0{} m } {
2359   \stex_import_module_uri:nn { #1 } { #2 }
2360   \stex_debug:nn{modules}{Importing~module:~
2361     \l_stex_import_ns_str ? \l_stex_import_name_str
2362   }
2363   \stex_import_require_module:nnnn
2364   { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2365   { \l_stex_import_path_str } { \l_stex_import_name_str }
2366   \stex_if_smsmode:F {
2367     \stex_annotate_invisible:nnn
2368     {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
2369   }
2370   \exp_args:Nx \stex_add_to_current_module:n {
2371     \stex_import_require_module:nnnn
2372     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2373     { \l_stex_import_path_str } { \l_stex_import_name_str }
2374   }
2375   \exp_args:Nx \stex_add_import_to_current_module:n {
2376     \l_stex_import_ns_str ? \l_stex_import_name_str
2377   }
2378   \stex_smsmode_do:
2379   \ignorespacesandpars
2380 }
2381 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

(End definition for `\importmodule`. This function is documented on page 88.)

`\usemodule`

```

2382 \NewDocumentCommand \usemodule { 0{} m } {
2383   \stex_if_smsmode:F {
2384     \stex_import_module_uri:nn { #1 } { #2 }
2385     \stex_import_require_module:nnnn
2386     { \l_stex_import_ns_str } { \l_stex_import_archive_str }

```



```

2387     { \l_stex_import_path_str } { \l_stex_import_name_str }
2388     \stex_annotate_invisible:nnn
2389     {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
2390   }
2391   \stex_smsmode_do:
2392   \ignorespacesandpars
2393 }

```

(End definition for \usemodule. This function is documented on page 88.)

```

2394 \cs_new_protected:Nn \stex_csl_to_imports:Nn {
2395   \tl_if_empty:nF{#2}{
2396     \clist_set:Nn \l_tmpa_clist {#2}
2397     \clist_map_inline:Nn \l_tmpa_clist {
2398       \tl_if_head_eq_charcode:nNTF {##1} [{
2399         #1 ##1
2400       }{
2401         #1{##1}
2402       }
2403     }
2404   }
2405 }
2406 \cs_generate_variant:Nn \stex_csl_to_imports:Nn {No}
2407
2408
2409 </package>

```

Chapter 31

STEX -Symbols Implementation

```
2410 <*package>
2411
2412 %%%%%%%%%% symbols.dtx %%%%%%%%%%
2413
2414 Warnings and error messages
2415 \msg_new:nnn{stex}{error/wrongargs}{
2416   args~value~in~symbol~declaration~for~#1~
2417   needs~to~be~i,~a,~b~or~B,~but~#2~given
2418 }
2419 \msg_new:nnn{stex}{error/unknownsymbol}{
2420   No~symbol~#1~found!
2421 }
2422 \msg_new:nnn{stex}{error/seqlength}{
2423   Expected~#1~arguments;~got~#2!
2424 }
2425 \msg_new:nnn{stex}{error/unknownnotation}{
2426   Unknown~notation~#1~for~#2!
2427 }
```

31.1 Symbol Declarations

```
2427 <@@=stex_symdecl>
\stex_all_symbols:n Map over all available symbols
2428 \cs_new_protected:Nn \stex_all_symbols:n {
2429   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
2430   \seq_map_inline:Nn \l_stex_all_modules_seq {
2431     \seq_map_inline:cn{c_stex_module_##1_constants}{
2432       \__stex_symdecl_all_symbols_cs{##1?####1}
2433     }
2434   }
2435 }
```

(End definition for `\stex_all_symbols:n`. This function is documented on page 91.)

`\STEXsymbol`

```
2436 \NewDocumentCommand \STEXsymbol { m } {
2437   \stex_get_symbol:n { #1 }
2438   \exp_args:No
2439   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
2440 }
```

(End definition for `\STEXsymbol`. This function is documented on page 92.)

`symdecl` arguments:

```
2441 \keys_define:nn { stex / symdecl } {
2442   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2443   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2444   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
2445   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,
2446   align     .str_set:N    = \l_stex_symdecl_align_str , % TODO(?)
2447   gfc       .str_set:N    = \l_stex_symdecl_gfc_str , % TODO(?)
2448   def       .tl_set:N    = \l_stex_symdecl_definiens_tl ,
2449   reorder   .str_set_x:N = \l_stex_symdecl_reorder_str ,
2450   argnames  .clist_set:N = \l_stex_symdecl_argnames_clist ,
2451   assoc     .choices:nn  =
2452             {bin,binl,binr,pre,conj,pwconj}
2453             {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}
2454 }
2455
2456 \bool_new:N \l_stex_symdecl_make_macro_bool
2457
2458 \cs_new_protected:Nn \__stex_symdecl_args:n {
2459   \str_clear:N \l_stex_symdecl_name_str
2460   \str_clear:N \l_stex_symdecl_args_str
2461   \str_clear:N \l_stex_symdecl_deprecate_str
2462   \str_clear:N \l_stex_symdecl_reorder_str
2463   \str_clear:N \l_stex_symdecl_assoctype_str
2464   \bool_set_false:N \l_stex_symdecl_local_bool
2465   \tl_clear:N \l_stex_symdecl_type_tl
2466   \tl_clear:N \l_stex_symdecl_definiens_tl
2467   \clist_clear:N \l_stex_symdecl_argnames_clist
2468
2469   \keys_set:nn { stex / symdecl } { #1 }
2470 }
```

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

```
2471
2472 \NewDocumentCommand \symdecl { s m O{} } {
2473   \__stex_symdecl_args:n { #3 }
2474   \IfBooleanTF #1 {
2475     \bool_set_false:N \l_stex_symdecl_make_macro_bool
2476   } {
2477     \bool_set_true:N \l_stex_symdecl_make_macro_bool
2478   }
2479   \stex_symdecl_do:n { #2 }
2480   \stex_smsmode_do:
2481 }
```

```

2482
2483 \cs_new_protected:Nn \stex_symdecl_do:nn {
2484   \__stex_symdecl_args:n{#1}
2485   \bool_set_false:N \l_stex_symdecl_make_macro_bool
2486   \stex_symdecl_do:n{#2}
2487 }
2488
2489 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

(End definition for \symdecl. This function is documented on page 90.)

\stex_symdecl_do:n

```

2490 \cs_new_protected:Nn \stex_symdecl_do:n {
2491   \stex_if_in_module:F {
2492     % TODO throw error? some default namespace?
2493   }
2494
2495   \str_if_empty:NT \l_stex_symdecl_name_str {
2496     \str_set:Nx \l_stex_symdecl_name_str { #1 }
2497   }
2498
2499   \prop_if_exist:cT { l_stex_symdecl_
2500     \l_stex_current_module_str ?
2501     \l_stex_symdecl_name_str
2502     _prop
2503   }{
2504     % TODO throw error (beware of circular dependencies)
2505   }
2506
2507   \prop_clear:N \l_tmpa_prop
2508   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
2509   \seq_clear:N \l_tmpa_seq
2510   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
2511   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
2512
2513   \str_if_empty:NT \l_stex_symdecl_deprecate_str {
2514     \str_if_empty:NF \l_stex_module_deprecate_str {
2515       \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
2516     }
2517   }
2518   \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
2519
2520   \exp_args:No \stex_add_constant_to_current_module:n {
2521     \l_stex_symdecl_name_str
2522   }
2523
2524   % arity/args
2525   \int_zero:N \l_tmpb_int
2526
2527   \bool_set_true:N \l_tmpa_bool
2528   \str_map_inline:Nn \l_stex_symdecl_args_str {
2529     \token_case_meaning:NnF ##1 {
2530       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2531       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }

```

```

2532     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2533     {\tl_to_str:n a} {
2534         \bool_set_false:N \l_tmpa_bool
2535         \int_incr:N \l_tmpb_int
2536     }
2537     {\tl_to_str:n B} {
2538         \bool_set_false:N \l_tmpa_bool
2539         \int_incr:N \l_tmpb_int
2540     }
2541 }{
2542     \msg_error:nnxx{stex}{error/wrongargs}{
2543         \l_stex_current_module_str ?
2544         \l_stex_symdecl_name_str
2545     }{##1}
2546 }
2547 }
2548
2549 \bool_if:NTF \l_tmpa_bool {
2550     % possibly numeric
2551     \str_if_empty:NTF \l_stex_symdecl_args_str {
2552         \prop_put:Nnn \l_tmpa_prop { args } {}
2553         \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2554     }{
2555         \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
2556         \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2557         \str_clear:N \l_tmpa_str
2558         \int_step_inline:nn \l_tmpa_int {
2559             \str_put_right:Nn \l_tmpa_str i
2560         }
2561         \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
2562     }
2563 } {
2564     \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
2565     \prop_put:Nnx \l_tmpa_prop { arity }
2566     { \str_count:N \l_stex_symdecl_args_str }
2567 }
2568 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2569
2570 \tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
2571     \prop_put:Nnx \l_tmpa_prop { defined }{ false }
2572 }{
2573     \prop_put:Nnx \l_tmpa_prop { defined }{ true }
2574 }
2575
2576 % argnames
2577
2578 \clist_clear:N \l_tmpa_clist
2579 \int_step_inline:nn {\prop_item:Nn \l_tmpa_prop {arity}} {
2580     \clist_if_empty:NTF \l_stex_symdecl_argnames_clist {
2581         \clist_put_right:Nn \l_tmpa_clist {##1}
2582     }{
2583         \clist_pop:NN \l_stex_symdecl_argnames_clist \l_tmpa_tl
2584         \exp_args:NNx \clist_put_right:Nn \l_tmpa_clist {\c_dollar_str\l_tmpa_tl}
2585     }

```

```

2586 }
2587 \prop_put:Nn \l_tmpa_prop {argnames} {\clist_use:Nn \l_tmpa_clist ,}
2588
2589 % semantic macro
2590
2591 \bool_if:NT \l_stex_symdecl_make_macro_bool {
2592   \exp_args:Nx \stex_do_up_to_module:n {
2593     \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2594       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2595     }}
2596   }
2597 }
2598
2599 \stex_debug:nn{symbols}{New~symbol:~
2600   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2601   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2602   Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
2603   Definiens:~\exp_not:o {\l_stex_symdecl_definiens_tl}
2604 }
2605
2606 % circular dependencies require this:
2607 \stex_if_do_html:T {
2608   \stex_annotate_invisible:nnn {symdecl} {
2609     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2610   } {
2611     \tl_if_empty:NF \l_stex_symdecl_type_tl {
2612       \stex_annotate_invisible:nnn{type}{\l_stex_symdecl_type_tl$}
2613     }
2614     \stex_annotate_invisible:nnn{args}{\prop_item:Nn \l_tmpa_prop { args }}{}
2615     \stex_annotate_invisible:nnn{macroname}{#1}{}
2616     \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2617       \stex_annotate_invisible:nnn{definiens}{}
2618       {\l_stex_symdecl_definiens_tl$}
2619     }
2620     \str_if_empty:NF \l_stex_symdecl_assoc_type_str {
2621       \stex_annotate_invisible:nnn{assoc_type}{\l_stex_symdecl_assoc_type_str}{}
2622     }
2623     \str_if_empty:NF \l_stex_symdecl_reorder_str {
2624       \stex_annotate_invisible:nnn{reorderargs}{\l_stex_symdecl_reorder_str}{}
2625     }
2626   }
2627 }
2628 \prop_if_exist:cF {
2629   \l_stex_symdecl_
2630   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2631   _prop
2632 } {
2633   \bool_if:NTF \l_stex_symdecl_local_bool \stex_do_up_to_module:x \stex_execute_in_module:
2634     \__stex_symdecl_restore_symbol:nnnnnnnn
2635       {\l_stex_symdecl_name_str}
2636       { \prop_item:Nn \l_tmpa_prop {args} }
2637       { \prop_item:Nn \l_tmpa_prop {arity} }
2638       { \prop_item:Nn \l_tmpa_prop {assoc} }
2639       { \prop_item:Nn \l_tmpa_prop {defined} }

```

```

2640         {\bool_if:NT \l_stex_symdecl_make_macro_bool {#1} }
2641         {\l_stex_current_module_str}
2642         { \prop_item:Nn \l_tmpa_prop {argnames} }
2643     }
2644 }
2645 }
2646 \cs_new_protected:Nn \__stex_symdecl_restore_symbol:nnnnnnnn {
2647     \prop_clear:N \l_tmpa_prop
2648     \prop_put:Nnn \l_tmpa_prop { module } { #7 }
2649     \prop_put:Nnn \l_tmpa_prop { name } { #1 }
2650     \prop_put:Nnn \l_tmpa_prop { args } { #2 }
2651     \prop_put:Nnn \l_tmpa_prop { arity } { #3 }
2652     \prop_put:Nnn \l_tmpa_prop { assocs } { #4 }
2653     \prop_put:Nnn \l_tmpa_prop { defined } { #5 }
2654     \prop_put:Nnn \l_tmpa_prop { argnames } { #8 }
2655     \tl_if_empty:nF{#6}{
2656         \tl_set:cx{#6}{\stex_invoke_symbol:n{\detokenize{#7 ? #1}}}
2657     }
2658     \prop_set_eq:cN{\l_stex_symdecl_ \detokenize{#7 ? #1} _prop}\l_tmpa_prop
2659     \seq_clear:c{\l_stex_symdecl_ \detokenize{#7 ? #1} _notations}
2660 }

```

(End definition for `\stex_symdecl_do:n`. This function is documented on page 91.)

`\textsymdecl`

```

2661
2662 \keys_define:nn { stex / textsymdecl } {
2663     name      .str_set_x:N = \l__stex_symdecl_name_str ,
2664     type      .tl_set:N    = \l__stex_symdecl_type_tl
2665 }
2666
2667 \cs_new_protected:Nn \stex_textsymdecl_args:n {
2668     \str_clear:N \l__stex_symdecl_name_str
2669     \tl_clear:N \l__stex_symdecl_type_tl
2670     \clist_clear:N \l_stex_symdecl_argnames_clist
2671     \keys_set:nn { stex / textsymdecl } { #1 }
2672 }
2673
2674 \NewDocumentCommand \textsymdecl {m O{} m} {
2675     \stex_textsymdecl_args:n { #2 }
2676     \str_if_empty:NTF \l__stex_symdecl_name_str {
2677         \__stex_symdecl_args:n{name=#1,#2}
2678     }{
2679         \__stex_symdecl_args:n{#2}
2680     }
2681     \bool_set_true:N \l_stex_symdecl_make_macro_bool
2682     \stex_symdecl_do:n{#1-sym}
2683     \stex_execute_in_module:n{
2684         \cs_set_nopar:cpn{#1name}{
2685             \ifvmode\hbox_unpack:N\c_empty_box\fi
2686             \ifmmode\hbox{#3}\else#3\fi\hspace
2687         }
2688         \cs_set_nopar:cpn{#1}{
2689             \ifmmode\csname#1-sym\expandafter\endcsname\else

```

```

2690     \ifvmode\hbox_unpack:N\c_empty_box\fi
2691     \symref{#1-sym}{#3}\expandafter\xspace
2692     \fi
2693   }
2694 }
2695 \stex_execute_in_module:x{
2696   \__stex_notation_restore_notation:nnnnn
2697   {\l_stex_current_module_str?\tl_if_empty:N\l_stex_symdecl_name_str{#1}\l_stex_symdecl_name_str{#1}}{0}
2698   {\exp_not:n{\STEXInternalTermMathOMSiiii{\STEXInternalCurrentSymbolStr}{}}{\neginfprec}{\neginfprec}}{0}
2699   \comp{\hbox{#3}}\STEXInternalSymbolAfterInvokationTL
2700   }}}
2701   {}
2702   {}
2703 }
2704 \stex_smsmode_do:
2705 }

```

(End definition for `\textsymdecl`. This function is documented on page 23.)

`\stex_get_symbol:n`

```

2706 \str_new:N \l_stex_get_symbol_uri_str
2707
2708 \cs_new_protected:Nn \stex_get_symbol:n {
2709   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2710     \tl_set:Nn \l_tmpa_tl { #1 }
2711     \__stex_symdecl_get_symbol_from_cs:
2712   }{
2713     % argument is a string
2714     % is it a command name?
2715     \cs_if_exist:cTF { #1 }{
2716       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2717       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2718       \str_if_empty:N\l_tmpa_str {
2719         \exp_args:Nx \cs_if_eq:NNTF {
2720           \tl_head:N \l_tmpa_tl
2721         } \stex_invoke_symbol:n {
2722           \__stex_symdecl_get_symbol_from_cs:
2723         }{
2724           \__stex_symdecl_get_symbol_from_string:n { #1 }
2725         }
2726       } {
2727         \__stex_symdecl_get_symbol_from_string:n { #1 }
2728       }
2729     }{
2730       % argument is not a command name
2731       \__stex_symdecl_get_symbol_from_string:n { #1 }
2732       % \l_stex_all_symbols_seq
2733     }
2734   }
2735   \str_if_eq:eeF {
2736     \prop_item:cn {
2737       \l_stex_symdecl\l_stex_get_symbol_uri_str _prop
2738     }{ deprecate }
2739   }{}{

```



```

2740     \msg_warning:nnxx{stex}{warning/deprecated}{
2741       Symbol~\l_stex_get_symbol_uri_str
2742     }{
2743       \prop_item:cn {l_stex_symdecl_l\l_stex_get_symbol_uri_str _prop}{ deprecate }
2744     }
2745   }
2746 }
2747
2748 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2749   \tl_set:Nn \l_tmpa_tl {
2750     \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2751   }
2752   \str_set:Nn \l_tmpa_str { #1 }
2753
2754   %\int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2755
2756   \str_if_in:NnTF \l_tmpa_str ? {
2757     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
2758     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
2759     \str_set:Nx \l_tmpb_str {\seq_use:Nn \l_tmpa_seq ?}
2760   }{
2761     \str_clear:N \l_tmpb_str
2762   }
2763   \str_if_empty:NnTF \l_tmpb_str {
2764     \seq_map_inline:Nn \l_stex_all_modules_seq {
2765       \seq_map_inline:cn{c_stex_module_##1_constants}{
2766         \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2767           \seq_map_break:n{\seq_map_break:n{
2768             \tl_set:Nn \l_tmpa_tl {
2769               \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2770             }
2771           }}
2772         }
2773       }
2774     }
2775   }{
2776     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpb_str }
2777     \seq_map_inline:Nn \l_stex_all_modules_seq {
2778       \str_if_eq:eeT{ \l_tmpb_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2779         \seq_map_inline:cn{c_stex_module_##1_constants}{
2780           \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2781             \seq_map_break:n{\seq_map_break:n{
2782               \tl_set:Nn \l_tmpa_tl {
2783                 \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2784               }
2785             }}
2786           }
2787         }
2788       }
2789     }
2790   }
2791
2792   \l_tmpa_tl
2793 }

```

```

2794
2795 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
2796   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2797     { \tl_tail:N \l_tmpa_tl }
2798   \tl_if_single:NTF \l_tmpa_tl {
2799     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2800       \exp_after:wN \str_set:Nn \exp_after:wN
2801         \l_stex_get_symbol_uri_str \l_tmpa_tl
2802     }{
2803       % TODO
2804       % tail is not a single group
2805     }
2806   }{
2807     % TODO
2808     % tail is not a single group
2809   }
2810 }

```

(End definition for `\stex_get_symbol:n`. This function is documented on page 91.)

31.2 Notations

```

2811 <@@=stex_notation>
      notation arguments:
2812 \keys_define:nn { stex / notation } {
2813   % lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2814   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2815   prec     .str_set_x:N = \l__stex_notation_prec_str ,
2816   op       .tl_set:N = \l__stex_notation_op_tl ,
2817   primary .bool_set:N = \l__stex_notation_primary_bool ,
2818   primary .default:n = {true} ,
2819   hints    .str_set_x:N = \l__stex_notation_hints_str,
2820   unknown .code:n = \str_set:Nx
2821     \l__stex_notation_variant_str \l_keys_key_str
2822 }
2823
2824 \cs_new_protected:Nn \_stex_notation_args:n {
2825   % \str_clear:N \l__stex_notation_lang_str
2826   \str_clear:N \l__stex_notation_variant_str
2827   \str_clear:N \l__stex_notation_prec_str
2828   \str_clear:N \l__stex_notation_hints_str
2829   \tl_clear:N \l__stex_notation_op_tl
2830   \bool_set_false:N \l__stex_notation_primary_bool
2831
2832   \keys_set:nn { stex / notation } { #1 }
2833 }

```

`\notation`

```

2834 \NewDocumentCommand \notation { s m O{}} {
2835   \_stex_notation_args:n { #3 }
2836   \tl_clear:N \l_stex_symdecl_definiens_tl
2837   \stex_get_symbol:n { #2 }
2838   \tl_set:Nn \l_stex_notation_after_do_tl {

```

```

2839 \__stex_notation_final:
2840 \IfBooleanTF#1{
2841   \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2842 }{}
2843 \stex_smsmode_do:\ignorespacesandpars
2844 }
2845 \stex_notation_do:nnnnn
2846 { \prop_item:cn {\l_stex_symdecl\l_stex_get_symbol_uri_str _prop } { args } }
2847 { \prop_item:cn { \l_stex_symdecl\l_stex_get_symbol_uri_str _prop } { arity } }
2848 { \l__stex_notation_variant_str }
2849 { \l__stex_notation_prec_str }
2850 }
2851 \stex_deactivate_macro:Nn \notation {module-environments}

```

(End definition for \notation. This function is documented on page 91.)

\stex_notation_do:nnnnn

```

2852 \seq_new:N \l__stex_notation_precedences_seq
2853 \tl_new:N \l__stex_notation_opprec_tl
2854 \int_new:N \l__stex_notation_currarg_int
2855 \tl_new:N \STEXInternalSymbolAfterInvokationTL
2856
2857 \cs_new_protected:Nn \stex_notation_do:nnnnn {
2858   \let\STEXInternalCurrentSymbolStr\relax
2859   \seq_clear:N \l__stex_notation_precedences_seq
2860   \tl_clear:N \l__stex_notation_opprec_tl
2861   \str_set:Nx \l__stex_notation_args_str { #1 }
2862   \str_set:Nx \l__stex_notation_arity_str { #2 }
2863   \str_set:Nx \l__stex_notation_suffix_str { #3 }
2864   \str_set:Nx \l__stex_notation_prec_str { #4 }
2865
2866   % precedences
2867   \str_if_empty:NTF \l__stex_notation_prec_str {
2868     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2869       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2870     }{
2871       \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2872     }
2873   } {
2874     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2875       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2876       \int_step_inline:nn { \l__stex_notation_arity_str } {
2877         \exp_args:NNo
2878         \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2879       }
2880     }{
2881       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2882       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2883         \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2884         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2885           \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2886             \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2887           \seq_map_inline:Nn \l_tmpa_seq {
2888             \seq_put_right:Nn \l__stex_notation_precedences_seq { ##1 }

```

```

2889     }
2890   }
2891   }{
2892     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2893       \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2894     }{
2895       \tl_set:No \l__stex_notation_opprec_tl { 0 }
2896     }
2897   }
2898 }
2899 }
2900
2901 \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2902 \int_step_inline:nn { \l__stex_notation_arity_str } {
2903   \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
2904     \exp_args:NNo
2905     \seq_put_right:No \l__stex_notation_precedences_seq {
2906       \l__stex_notation_opprec_tl
2907     }
2908   }
2909 }
2910 \tl_clear:N \l_stex_notation_dummyargs_tl
2911
2912 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2913   \exp_args:NNe
2914   \cs_set:Npn \l_stex_notation_macrocode_cs {
2915     \STEXInternalTermMathOMSiiii { \STEXInternalCurrentSymbolStr }
2916     { \l__stex_notation_suffix_str }
2917     { \l__stex_notation_opprec_tl }
2918     { \exp_not:n { #5 } }
2919   }
2920   \l_stex_notation_after_do_tl
2921 }{
2922   \str_if_in:NnTF \l__stex_notation_args_str b {
2923     \exp_args:Nne \use:nn
2924     {
2925       \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2926       \cs_set:Npn \l__stex_notation_arity_str } { {
2927         \STEXInternalTermMathOMBiiii { \STEXInternalCurrentSymbolStr }
2928         { \l__stex_notation_suffix_str }
2929         { \l__stex_notation_opprec_tl }
2930         { \exp_not:n { #5 } }
2931       } }
2932   }{
2933     \str_if_in:NnTF \l__stex_notation_args_str B {
2934       \exp_args:Nne \use:nn
2935       {
2936         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2937         \cs_set:Npn \l__stex_notation_arity_str } { {
2938           \STEXInternalTermMathOMBiiii { \STEXInternalCurrentSymbolStr }
2939           { \l__stex_notation_suffix_str }
2940           { \l__stex_notation_opprec_tl }
2941           { \exp_not:n { #5 } }
2942         } }

```

```

2943   }{
2944     \exp_args:Nne \use:nn
2945     {
2946       \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2947       \cs_set:Npn \l__stex_notation_arity_str } { {
2948         \STEXInternalTermMathOMAiiai { \STEXInternalCurrentSymbolStr }
2949         { \l__stex_notation_suffix_str }
2950         { \l__stex_notation_opprec_tl }
2951         { \exp_not:n { #5 } }
2952       } }
2953     }
2954   }
2955
2956   \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2957   \int_zero:N \l__stex_notation_currarg_int
2958   \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2959   \__stex_notation_arguments:
2960 }
2961 }

```

(End definition for \stex_notation_do:nnnnn. This function is documented on page ??.)

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

```

2962 \cs_new_protected:Nn \__stex_notation_arguments: {
2963   \int_incr:N \l__stex_notation_currarg_int
2964   \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2965     \l_stex_notation_after_do_tl
2966   }{
2967     \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2968     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2969     \str_if_eq:VnTF \l_tmpa_str a {
2970       \__stex_notation_argument_assoc:nn{a}
2971     }{
2972       \str_if_eq:VnTF \l_tmpa_str B {
2973         \__stex_notation_argument_assoc:nn{B}
2974       }{
2975         \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpb_str
2976         \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2977           { \STEXInternalTermMathArgiii
2978             { \l_tmpa_str\int_use:N \l__stex_notation_currarg_int }
2979             { \l_tmpb_str }
2980             { ###\int_use:N \l__stex_notation_currarg_int }
2981           }
2982         }
2983         \__stex_notation_arguments:
2984       }
2985     }
2986   }
2987 }

```

(End definition for __stex_notation_arguments:.)

_stex_notation_argument_assoc:nn

```

2988 \cs_new_protected:Nn \__stex_notation_argument_assoc:nn {

```

```

2989
2990 \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2991   {\l__stex_notation_arity_str}{
2992     #2
2993   }
2994 \int_zero:N \l_tmpa_int
2995 \tl_clear:N \l_tmpa_tl
2996 \str_map_inline:Nn \l__stex_notation_args_str {
2997   \int_incr:N \l_tmpa_int
2998   \tl_put_right:Nx \l_tmpa_tl {
2999     \str_if_eq:nnTF {##1}{a}{ } {} }{
3000     \str_if_eq:nnTF {##1}{B}{ } {} }{
3001     {\_stex_term_arg:nn{##1}\int_use:N \l_tmpa_int}{##### \int_use:N \l_tmpa_int}
3002   }
3003 }
3004 }
3005 }
3006 \exp_after:wN\exp_after:wN\exp_after:wN \def
3007 \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
3008 \exp_after:wN\exp_after:wN\exp_after:wN ##
3009 \exp_after:wN\exp_after:wN\exp_after:wN 1
3010 \exp_after:wN\exp_after:wN\exp_after:wN ##
3011 \exp_after:wN\exp_after:wN\exp_after:wN 2
3012 \exp_after:wN\exp_after:wN\exp_after:wN {
3013   \exp_after:wN \exp_after:wN \exp_after:wN
3014   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
3015     \exp_after:wN \l_tmpa_cs \l_tmpa_tl
3016   }
3017 }
3018
3019 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
3020 \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
3021   \STEXInternalTermMathAssocArgiiii
3022   { \int_use:N \l__stex_notation_currarg_int }
3023   { \l_tmpa_str }
3024   { ####\int_use:N \l__stex_notation_currarg_int }
3025   { \l_tmpa_cs {####1} {####2} }
3026   {#1}
3027 } }
3028 \__stex_notation_arguments:
3029 }

```

(End definition for _stex_notation_argument_assoc:nn.)

_stex_notation_final: Called after processing all notation arguments

```

3030 \cs_new_protected:Nn \__stex_notation_restore_notation:nnnnn {
3031   \cs_generate_from_arg_count:cNnn{stex_notation_\detokenize{#1} \c_hash_str \detokenize{#2}}
3032   \cs_set_nopar:Npn {#3}{#4}
3033   \tl_if_empty:nF {#5}{
3034     \tl_set:cn{stex_op_notation_\detokenize{#1} \c_hash_str \detokenize{#2}_cs}{\comp{ #5 }
3035   }
3036   \seq_if_exist:cT { l_stex_symdecl_\detokenize{#1} _notations }{
3037     \seq_put_right:cx { l_stex_symdecl_\detokenize{#1} _notations } { \detokenize{#2} }
3038   }

```

```

3039 }
3040
3041 \cs_new_protected:Nn \__stex_notation_final: {
3042
3043   \stex_execute_in_module:x {
3044     \__stex_notation_restore_notation:nnnnn
3045     {\l_stex_get_symbol_uri_str}
3046     {\l__stex_notation_suffix_str}
3047     {\l__stex_notation_arity_str}
3048     {
3049       \exp_after:wN \exp_after:wN \exp_after:wN
3050       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
3051       { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \STEXInt
3052     }
3053     {\exp_args:No \exp_not:n \l__stex_notation_op_tl }
3054   }
3055
3056   \stex_debug:nn{symbols}{
3057     Notation~\l__stex_notation_suffix_str
3058     ~for~\l_stex_get_symbol_uri_str^^J
3059     Operator~precedence:~\l__stex_notation_opprec_tl^^J
3060     Argument~precedences:~
3061     \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
3062     Notation: \cs_meaning:c {
3063       stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
3064       \l__stex_notation_suffix_str
3065       _cs
3066     }
3067   }
3068   % HTML annotations
3069   \stex_if_do_html:T {
3070     \stex_annotate_invisible:nnn { notation }
3071     { \l_stex_get_symbol_uri_str } {
3072       \stex_annotate_invisible:nnn { notationfragment }
3073       { \l__stex_notation_suffix_str }{}
3074       \stex_annotate_invisible:nnn { precedence }
3075       { \l__stex_notation_prec_str }{}
3076
3077       \int_zero:N \l_tmpa_int
3078       \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
3079       \tl_clear:N \l_tmpa_tl
3080       \int_step_inline:nn { \l__stex_notation_arity_str }{
3081         \int_incr:N \l_tmpa_int
3082         \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
3083         \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_rema
3084         \str_if_eq:VnTF \l_tmpb_str a {
3085           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3086             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3087             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3088           } }
3089         }{
3090           \str_if_eq:VnTF \l_tmpb_str B {
3091             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3092               \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,

```

```

3093         \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{\}
3094     } }
3095     }{
3096         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3097             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{\}
3098         } }
3099     }
3100 }
3101 }
3102 \stex_annotate_invisible:nnn { notationcomp }{\}{
3103     \str_set:Nx \STEXInternalCurrentSymbolStr {\l_stex_get_symbol_uri_str }
3104     $ \exp_args:Nno \use:nn { \use:c {
3105         stex_notation_ \STEXInternalCurrentSymbolStr
3106         \c_hash_str \l__stex_notation_suffix_str_cs
3107     } } { \l_tmpa_tl } $
3108 }
3109 \tl_if_empty:NF \l__stex_notation_op_tl {
3110     \stex_annotate_invisible:nnn { notationopcomp }{\}{
3111         $\l__stex_notation_op_tl$
3112     }
3113 }
3114 }
3115 }
3116 }

```

(End definition for `_stex_notation_final:`.)

`\setnotation`

```

3117 \keys_define:nn { stex / setnotation } {
3118 % lang .tl_set_x:N = \l__stex_notation_lang_str ,
3119 variant .tl_set_x:N = \l__stex_notation_variant_str ,
3120 unknown .code:n = \str_set:Nx
3121     \l__stex_notation_variant_str \l_keys_key_str
3122 }
3123
3124 \cs_new_protected:Nn \_stex_setnotation_args:n {
3125 % \str_clear:N \l__stex_notation_lang_str
3126 \str_clear:N \l__stex_notation_variant_str
3127 \keys_set:nn { stex / setnotation } { #1 }
3128 }
3129
3130 \cs_new_protected:Nn \_stex_notation_setnotation:nn {
3131 \seq_if_exist:cn{\l_stex_symdecl_#1_notations}{
3132     \seq_remove_all:cn { l_stex_symdecl_#1_notations }{ #2 }
3133     \seq_put_left:cn { l_stex_symdecl_#1_notations }{ #2 }
3134 }
3135 }
3136
3137 \cs_new_protected:Nn \stex_setnotation:n {
3138 \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1_notations }
3139 { \l__stex_notation_variant_str }{
3140     \stex_execute_in_module:x{ \_stex_notation_setnotation:nn {#1}{\l__stex_notation_vari
3141     \stex_debug:nn {notations}{
3142         Setting~default~notation~

```



```

3143         {\l__stex_notation_variant_str }~for~
3144         #1 \\\
3145         \expandafter\meaning\csname
3146         l_stex_symdecl_#1 _notations\endcsname
3147     }
3148 }{
3149     \msg_error:nnxx{stex}{unknownnotation}{\l__stex_notation_variant_str}{#1}
3150 }
3151 }
3152
3153 \NewDocumentCommand \setnotation {m m} {
3154     \stex_get_symbol:n { #1 }
3155     \_stex_setnotation_args:n { #2 }
3156     \stex_setnotation:n{\l_stex_get_symbol_uri_str}
3157     \stex_smsmode_do:\ignorespacesandpars
3158 }
3159
3160 \cs_new_protected:Nn \stex_copy_notations:nn {
3161     \stex_debug:nn {notations}{
3162         Copying~notations~from~#2~to~#1\\
3163         \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
3164     }
3165     \tl_clear:N \l_tmpa_tl
3166     \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
3167         \tl_put_right:Nn \l_tmpa_tl { {##### #1} }
3168     }
3169     \seq_map_inline:cn {l_stex_symdecl_#2_notations}{\begingroup
3170         \stex_debug:nn{Here}{Here:~##1}
3171         \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
3172         \edef \l_tmpa_tl {
3173             \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
3174             \exp_after:wN\exp_after:wN\exp_after:wN {
3175                 \exp_after:wN \l_tmpa_cs \l_tmpa_tl
3176             }
3177         }
3178
3179         \exp_after:wN \def \exp_after:wN \l_tmpa_tl
3180         \exp_after:wN #####\exp_after:wN 1 \exp_after:wN #####\exp_after:wN 2
3181         \exp_after:wN { \l_tmpa_tl }
3182
3183         \edef \l_tmpa_tl {
3184             \exp_after:wN \exp_not:n \exp_after:wN {
3185                 \l_tmpa_tl {##### 1}{##### 2}
3186             }
3187         }
3188
3189         \stex_debug:nn{Here}{Here:~\expandafter\detokenize\expandafter{\l_tmpa_tl}}
3190
3191     \stex_execute_in_module:x {
3192         \_stex_notation_restore_notation:nnnnn
3193         {#1}{##1}
3194         { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }
3195         { \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl} }
3196         {

```

```

3197         \cs_if_exist:cT{stex_op_notation_ #2\c_hash_str ##1 _cs}{
3198             \exp_args:NNo\exp_args:No\exp_not:n{\csname stex_op_notation_ #2\c_hash_str ##1
3199             }
3200         }
3201     }\endgroup
3202 }
3203 }
3204
3205 \NewDocumentCommand \copynotation {m m} {
3206     \stex_get_symbol:n { #1 }
3207     \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
3208     \stex_get_symbol:n { #2 }
3209     \exp_args:Noo
3210     \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
3211     \stex_smsmode_do:\ignorespacesandpars
3212 }
3213

```

(End definition for \setnotation. This function is documented on page 23.)

\symdef

```

3214 \keys_define:nn { stex / symdef } {
3215     name .str_set_x:N = \l_stex_symdecl_name_str ,
3216     args .str_set_x:N = \l_stex_symdecl_args_str ,
3217     type .tl_set:N = \l_stex_symdecl_type_tl ,
3218     def .tl_set:N = \l_stex_symdecl_definiens_tl ,
3219     reorder .str_set_x:N = \l_stex_symdecl_reorder_str ,
3220     op .tl_set:N = \l__stex_notation_op_tl ,
3221     % lang .str_set_x:N = \l__stex_notation_lang_str ,
3222     variant .str_set_x:N = \l__stex_notation_variant_str ,
3223     prec .str_set_x:N = \l__stex_notation_prec_str ,
3224     argnames .clist_set:N = \l_stex_symdecl_argnames_clist ,
3225     assoc .choices:nn =
3226         {bin,binl,binr,pre,conj,pwconj}
3227         {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
3228     unknown .code:n = \str_set:Nx
3229         \l__stex_notation_variant_str \l_keys_key_str
3230 }
3231
3232 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
3233     \str_clear:N \l_stex_symdecl_name_str
3234     \str_clear:N \l_stex_symdecl_args_str
3235     \str_clear:N \l_stex_symdecl_assoctype_str
3236     \str_clear:N \l_stex_symdecl_reorder_str
3237     \bool_set_false:N \l_stex_symdecl_local_bool
3238     \tl_clear:N \l_stex_symdecl_type_tl
3239     \tl_clear:N \l_stex_symdecl_definiens_tl
3240     \clist_clear:N \l_stex_symdecl_argnames_clist
3241     % \str_clear:N \l__stex_notation_lang_str
3242     \str_clear:N \l__stex_notation_variant_str
3243     \str_clear:N \l__stex_notation_prec_str
3244     \tl_clear:N \l__stex_notation_op_tl
3245
3246     \keys_set:nn { stex / symdef } { #1 }

```

```

3247 }
3248
3249 \NewDocumentCommand \symdef { m O{} } {
3250   \__stex_notation_symdef_args:n { #2 }
3251   \bool_set_true:N \l_stex_symdecl_make_macro_bool
3252   \stex_symdecl_do:n { #1 }
3253   \tl_set:Nn \l_stex_notation_after_do_tl {
3254     \__stex_notation_final:
3255     \stex_smsmode_do:\ignorespacesandpars
3256   }
3257   \str_set:Nx \l_stex_get_symbol_uri_str {
3258     \l_stex_current_module_str ? \l_stex_symdecl_name_str
3259   }
3260   \exp_args:Nx \stex_notation_do:nnnnn
3261     { \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str_prop } { args } }
3262     { \prop_item:cn { l_stex_symdecl\l_stex_get_symbol_uri_str_prop } { arity } }
3263     { \l__stex_notation_variant_str }
3264     { \l__stex_notation_prec_str}
3265   }
3266   \stex_deactivate_macro:Nn \symdef {module~environments}
3267
3268   \keys_define:nn { stex / mmtdef } {
3269     name .str_set_x:N = \l_stex_symdecl_name_str ,
3270     args .str_set_x:N = \l_stex_symdecl_args_str ,
3271     reorder .str_set_x:N = \l_stex_symdecl_reorder_str ,
3272     op .tl_set:N = \l__stex_notation_op_tl ,
3273     % lang .str_set_x:N = \l__stex_notation_lang_str ,
3274     variant .str_set_x:N = \l__stex_notation_variant_str ,
3275     prec .str_set_x:N = \l__stex_notation_prec_str ,
3276     argnames .clist_set:N = \l_stex_symdecl_argnames_clist ,
3277     assoc .choices:nn =
3278       {bin,binl,binr,pre,conj,pwconj}
3279       {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
3280     unknown .code:n = \str_set:Nx
3281       \l__stex_notation_variant_str \l_keys_key_str
3282   }
3283   \cs_new_protected:Nn \_stex_mmtdef_args:n {
3284     \str_clear:N \l_stex_symdecl_name_str
3285     \str_clear:N \l_stex_symdecl_args_str
3286     \str_clear:N \l_stex_symdecl_assoctype_str
3287     \str_clear:N \l_stex_symdecl_reorder_str
3288     \bool_set_false:N \l_stex_symdecl_local_bool
3289     \clist_clear:N \l_stex_symdecl_argnames_clist
3290     % \str_clear:N \l__stex_notation_lang_str
3291     \str_clear:N \l__stex_notation_variant_str
3292     \str_clear:N \l__stex_notation_prec_str
3293     \tl_clear:N \l__stex_notation_op_tl
3294
3295     \keys_set:nn { stex / mmtdef } { #1 }
3296   }
3297
3298   \NewDocumentCommand \mmtdef {m O{} }{
3299     \_stex_mmtdef_args:n{ #2 }
3300     \bool_set_true:N \l_stex_symdecl_make_macro_bool

```

```

3301 \str_if_empty:NT \l_stex_symdecl_name_str {
3302   \str_set:Nx \l_stex_symdecl_name_str { #1 }
3303 }
3304 %\tl_set:Nx \l_stex_symdecl_definiens_tl {
3305 %  \stex_annotate:nnn{ OMID }{
3306 %    \l_stex_module_mmtfor_str?\l_stex_symdecl_name_str
3307 %  }{}
3308 %}
3309 \stex_symdecl_do:n { #1 }
3310 \MMTrule{rules.stex.mmt.kwarc.info?SubstitutionRule}{
3311   \stex_annotate:nnn{ OMID }{
3312     \l_stex_current_module_str ? \l_stex_symdecl_name_str
3313   }{},
3314   \stex_annotate:nnn{ OMID }{
3315     \l_stex_module_mmtfor_str?\l_stex_symdecl_name_str
3316   }{}
3317 }
3318 \tl_set:Nn \l_stex_notation_after_do_tl {
3319   \__stex_notation_final:
3320   \stex_smsmode_do:\ignorespacesandpars
3321 }
3322 \str_set:Nx \l_stex_get_symbol_uri_str {
3323   \l_stex_current_module_str ? \l_stex_symdecl_name_str
3324 }
3325 \exp_args:Nx \stex_notation_do:nnnnn
3326 { \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str_prop } { args } }
3327 { \prop_item:cn { l_stex_symdecl\l_stex_get_symbol_uri_str_prop } { arity } }
3328 { \l__stex_notation_variant_str }
3329 { \l__stex_notation_prec_str}
3330 }

```

(End definition for `\symdef`. This function is documented on page 91.)

31.3 Variables

```

3331 <@@=stex_variables>
3332
3333 \keys_define:nn { stex / vardef } {
3334   name .str_set_x:N = \l__stex_variables_name_str ,
3335   args .str_set_x:N = \l__stex_variables_args_str ,
3336   type .tl_set:N = \l__stex_variables_type_tl ,
3337   def .tl_set:N = \l__stex_variables_def_tl ,
3338   op .tl_set:N = \l__stex_variables_op_tl ,
3339   prec .str_set_x:N = \l__stex_variables_prec_str ,
3340   reorder .str_set_x:N = \l__stex_variables_reorder_str ,
3341   argnames .clist_set:N = \l__stex_variables_argnames_clist ,
3342   assoc .choices:nn =
3343     {bin,binl,binr,pre,conj,pwconj}
3344     {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
3345   bind .choices:nn =
3346     {forall,exists}
3347     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
3348 }
3349

```

```

3350 \cs_new_protected:Nn \__stex_variables_args:n {
3351   \str_clear:N \l__stex_variables_name_str
3352   \str_clear:N \l__stex_variables_args_str
3353   \str_clear:N \l__stex_variables_prec_str
3354   \str_clear:N \l__stex_variables_assoctype_str
3355   \str_clear:N \l__stex_variables_reorder_str
3356   \str_clear:N \l__stex_variables_bind_str
3357   \tl_clear:N \l__stex_variables_type_tl
3358   \tl_clear:N \l__stex_variables_def_tl
3359   \tl_clear:N \l__stex_variables_op_tl
3360   \clist_clear:N \l__stex_variables_argnames_clist
3361
3362   \keys_set:nn { stex / vardef } { #1 }
3363 }
3364
3365 \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{}} {
3366   \__stex_variables_args:n {#2}
3367   \str_if_empty:NT \l__stex_variables_name_str {
3368     \str_set:Nx \l__stex_variables_name_str { #1 }
3369   }
3370   \prop_clear:N \l_tmpa_prop
3371   \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
3372
3373   \int_zero:N \l_tmpb_int
3374   \bool_set_true:N \l_tmpa_bool
3375   \str_map_inline:Nn \l__stex_variables_args_str {
3376     \token_case_meaning:NnF ##1 {
3377       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
3378       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
3379       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
3380       {\tl_to_str:n a} {
3381         \bool_set_false:N \l_tmpa_bool
3382         \int_incr:N \l_tmpb_int
3383       }
3384       {\tl_to_str:n B} {
3385         \bool_set_false:N \l_tmpa_bool
3386         \int_incr:N \l_tmpb_int
3387       }
3388     }{
3389       \msg_error:nxxx{stex}{error/wrongargs}{
3390         variable~\l__stex_variables_name_str
3391       }{##1}
3392     }
3393   }
3394   \bool_if:NTF \l_tmpa_bool {
3395     % possibly numeric
3396     \str_if_empty:NTF \l__stex_variables_args_str {
3397       \prop_put:Nnn \l_tmpa_prop { args } {}
3398       \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
3399     }{
3400       \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
3401       \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
3402       \str_clear:N \l_tmpa_str
3403       \int_step_inline:nn \l_tmpa_int {

```

```

3404     \str_put_right:Nn \l_tmpa_str i
3405   }
3406   \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
3407   \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
3408 }
3409 } {
3410   \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
3411   \prop_put:Nnx \l_tmpa_prop { arity }
3412     { \str_count:N \l__stex_variables_args_str }
3413 }
3414 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
3415 \tl_set:cx { #1 } { \stex_invoke_variable:n { \l__stex_variables_name_str } }
3416
3417 % argnames
3418
3419 \clist_clear:N \l_tmpa_clist
3420 \int_step_inline:nn {\prop_item:Nn \l_tmpa_prop {arity}} {
3421   \clist_if_empty:NTF \l__stex_variables_argnames_clist {
3422     \clist_put_right:Nn \l_tmpa_clist {##1}
3423   } {
3424     \clist_pop:NN \l__stex_variables_argnames_clist \l_tmpa_tl
3425     \exp_args:NNx \clist_put_right:Nn \l_tmpa_clist {\c_dollar_str\l_tmpa_tl}
3426   }
3427 }
3428 \prop_put:Nnx \l_tmpa_prop {argnames} {\clist_use:Nn \l_tmpa_clist ,}
3429
3430
3431 \prop_set_eq:cN { l_stex_symdecl_var://\l__stex_variables_name_str _prop } \l_tmpa_prop
3432
3433 \tl_if_empty:NF \l__stex_variables_op_tl {
3434   \cs_set:cpx {
3435     stex_var_op_notation_ \l__stex_variables_name_str _cs
3436   } { \exp_not:N\comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
3437 }
3438
3439 \tl_set:Nn \l_stex_notation_after_do_tl {
3440   \exp_args:Nne \use:nn {
3441     \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str _cs }
3442     \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
3443   } {{
3444     \exp_after:wN \exp_after:wN \exp_after:wN
3445     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
3446     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \STEXInter
3447   }}
3448 \stex_if_do_html:T {
3449   \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
3450     \stex_annotate_invisible:nnn { precedence }
3451     { \l__stex_variables_prec_str }{}
3452   \tl_if_empty:NF \l__stex_variables_type_tl {\stex_annotate_invisible:nnn{type}{\l__stex_variables_type_str}{\l__stex_variables_type_str}}
3453   \stex_annotate_invisible:nnn{args}{\l__stex_variables_args_str }{}
3454   \stex_annotate_invisible:nnn{macroname}{#1}{\l__stex_variables_macroname_str}
3455   \tl_if_empty:NF \l__stex_variables_def_tl {
3456     \stex_annotate_invisible:nnn{definiens}{\l__stex_variables_def_str}
3457     {\l__stex_variables_def_tl}

```

```

3458     }
3459     \str_if_empty:NF \l__stex_variables_assoctype_str {
3460         \stex_annotate_invisible:nnn{assoctype}{\l__stex_variables_assoctype_str}{}
3461     }
3462     \str_if_empty:NF \l__stex_variables_reorder_str {
3463         \stex_annotate_invisible:nnn{reorderargs}{\l__stex_variables_reorder_str}{}
3464     }
3465     \int_zero:N \l_tmpa_int
3466     \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
3467     \tl_clear:N \l_tmpa_tl
3468     \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
3469         \int_incr:N \l_tmpa_int
3470         \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
3471         \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables
3472         \str_if_eq:VnTF \l_tmpb_str a {
3473             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3474                 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3475                 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3476             } }
3477         }{
3478             \str_if_eq:VnTF \l_tmpb_str B {
3479                 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3480                     \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3481                     \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3482                 } }
3483             }{
3484                 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3485                     \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
3486                 } }
3487             }
3488         }
3489     }
3490     \stex_annotate_invisible:nnn { notationcomp }{}{
3491         \str_set:Nx \STEXInternalCurrentSymbolStr {var://\l__stex_variables_name_str }
3492         $ \exp_args:Nno \use:nn { \use:c {
3493             stex_var_notation_\l__stex_variables_name_str_cs
3494         } } { \l_tmpa_tl } $
3495     }
3496     \tl_if_empty:NF \l__stex_variables_op_tl {
3497         \stex_annotate_invisible:nnn { notationopcomp }{}{
3498             $\l__stex_variables_op_tl$
3499         }
3500     }
3501 }
3502 \str_if_empty:NF \l__stex_variables_bind_str {
3503     \stex_annotate_invisible:nnn {bindtype}{\l__stex_variables_bind_str,\l__stex_variab
3504 }
3505 }\ignorespacesandpars
3506 }
3507
3508 \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
3509 }
3510
3511 \cs_new:Nn \stex_reset:N {

```

```

3512 \tl_if_exist:NTF #1 {
3513   \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
3514 }{
3515   \let \exp_not:N #1 \exp_not:N \undefined
3516 }
3517 }
3518
3519 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
3520   \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
3521   \exp_args:Nnx \use:nn {
3522     % TODO
3523     \stex_annotate_invisible:nnn {vardecl}{\clist_use:Nn\l__stex_variables_names,}{
3524       #2
3525     }
3526   }{
3527     \stex_reset:N \varnot
3528     \stex_reset:N \vartype
3529     \stex_reset:N \vardefi
3530   }
3531 }
3532
3533 \NewDocumentCommand \vardef { s } {
3534   \IfBooleanTF#1 {
3535     \__stex_variables_do_complex:nn
3536   }{
3537     \__stex_variables_do_simple:nnn
3538   }
3539 }
3540
3541 \NewDocumentCommand \svar { 0{} m }{
3542   \tl_if_empty:nTF {#1}{
3543     \str_set:Nn \l_tmpa_str { #2 }
3544   }{
3545     \str_set:Nn \l_tmpa_str { #1 }
3546   }
3547   \_stex_term_omv:nn {
3548     var://\l_tmpa_str
3549   }{
3550     \exp_args:Nnx \use:nn {
3551       \def\comp{\_varcomp}
3552       \str_set:Nx \STEXInternalCurrentSymbolStr { var://\l_tmpa_str }
3553       \comp{ #2 }
3554     }{
3555       \_stex_reset:N \comp
3556       \_stex_reset:N \STEXInternalCurrentSymbolStr
3557     }
3558   }
3559 }
3560
3561
3562
3563 \keys_define:nn { stex / varseq } {
3564   name .str_set_x:N = \l__stex_variables_name_str ,
3565   args .int_set:N = \l__stex_variables_args_int ,

```



```

3566 type      .tl_set:N      = \l__stex_variables_type_tl ,
3567 mid        .tl_set:N      = \l__stex_variables_mid_tl ,
3568 bind       .choices:nn    =
3569           {forall,exists}
3570           {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
3571 }
3572
3573 \cs_new_protected:Nn \__stex_variables_seq_args:n {
3574   \str_clear:N \l__stex_variables_name_str
3575   \int_set:Nn \l__stex_variables_args_int 1
3576   \tl_clear:N \l__stex_variables_type_tl
3577   \str_clear:N \l__stex_variables_bind_str
3578
3579   \keys_set:nn { stex / varseq } { #1 }
3580 }
3581
3582 \NewDocumentCommand \varseq {m O{} m m m}{
3583   \__stex_variables_seq_args:n { #2 }
3584   \str_if_empty:NT \l__stex_variables_name_str {
3585     \str_set:Nx \l__stex_variables_name_str { #1 }
3586   }
3587   \prop_clear:N \l_tmpa_prop
3588   \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
3589
3590   \seq_set_from_clist:Nn \l_tmpa_seq {#3}
3591   \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
3592     \msg_error:nnxx{stex}{error/seqlength}
3593     {\int_use:N \l__stex_variables_args_int}
3594     {\seq_count:N \l_tmpa_seq}
3595   }
3596   \seq_set_from_clist:Nn \l_tmpb_seq {#4}
3597   \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
3598     \msg_error:nnxx{stex}{error/seqlength}
3599     {\int_use:N \l__stex_variables_args_int}
3600     {\seq_count:N \l_tmpb_seq}
3601   }
3602   \prop_put:Nnn \l_tmpa_prop {starts} {#3}
3603   \prop_put:Nnn \l_tmpa_prop {ends} {#4}
3604
3605   \cs_generate_from_arg_count:cNnn {stex_varseq\l__stex_variables_name_str _cs}
3606   \cs_set:Npn {\int_use:N \l__stex_variables_args_int} { #5 }
3607
3608   % argnames
3609
3610   \clist_clear:N \l_tmpa_clist
3611   \int_step_inline:nn {\l__stex_variables_args_int} {
3612     \clist_put_right:Nn \l_tmpa_clist {##1}
3613   }
3614   \prop_put:Nnx \l_tmpa_prop {argnames} {\clist_use:Nn \l_tmpa_clist ,}
3615
3616
3617
3618
3619   \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}

```

```

3620 \int_step_inline:nn \l__stex_variables_args_int {
3621   \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
3622 }
3623 \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
3624 \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
3625 \tl_if_empty:NF \l__stex_variables_mid_tl {
3626   \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
3627   \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
3628 }
3629 \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3630 \int_step_inline:nn \l__stex_variables_args_int {
3631   \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
3632 }
3633 \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
3634 \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
3635
3636
3637 \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl
3638
3639 \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}
3640
3641 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3642
3643 \int_step_inline:nn \l__stex_variables_args_int {
3644   \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
3645     \STEXInternalTermMathArgiii{i##1}{0}{\exp_not:n{####}##1}
3646   }}
3647 }
3648
3649 \tl_set:Nx \l_tmpa_tl {
3650   \STEXInternalTermMathOMaiiii { varseq://\l__stex_variables_name_str}{0}{
3651     \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
3652   }
3653 }
3654
3655 \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \STEXInternalSymbolAfterInvokationTL} }
3656
3657 \exp_args:Nno \use:nn {
3658   \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
3659   \cs_set:Npn {\int_use:N \l__stex_variables_args_int}{\l_tmpa_tl}
3660
3661   \stex_debug:nn{sequences}{New~Sequence:~
3662     \expandafter\meaning\csname stex_varseq_\l__stex_variables_name_str _cs\endcsname\\~\\
3663     \prop_to_keyval:N \l_tmpa_prop
3664   }
3665   \prop_set_eq:cN {l_stex_symdecl_varseq://\l__stex_variables_name_str _prop}\l_tmpa_prop
3666
3667   \stex_if_do_html:T{\stex_annotate_invisible:nnn{varseq}{\l__stex_variables_name_str}{
3668     \tl_if_empty:NF \l__stex_variables_type_tl {
3669       \stex_annotate:nnn {type}{\l__stex_variables_type_tl}
3670     }
3671     \stex_annotate:nnn {args}{\int_use:N \l__stex_variables_args_int}{}
3672     \str_if_empty:NF \l__stex_variables_bind_str {
3673       \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{}

```

```

3674 }
3675 \stex_annotate:nnn{startindex}{#3$}
3676 \stex_annotate:nnn{endindex}{#4$}
3677
3678 \tl_clear:N \l_tmpa_tl
3679 \int_step_inline:nn \l__stex_variables_args_int {
3680   \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3681     \stex_annotate:nnn{argmarker}{#1}{}
3682   } }
3683 }
3684 \stex_annotate_invisible:nnn { notationcomp }{}{
3685   \str_set:Nx \STEXInternalCurrentSymbolStr {varseq://\l__stex_variables_name_str }
3686   $ \exp_args:Nno \use:nn { \use:c {
3687     stex_varseq_\l__stex_variables_name_str _cs
3688   } } { \l_tmpa_tl } $
3689 }
3690 \stex_annotate_invisible:nnn { notationopcomp }{}{
3691   $ \prop_item:Nn \l_tmpa_prop { notation } $
3692 }
3693
3694 }}
3695
3696 \ignorespacesandpars
3697 }
3698
3699
3700 \keys_define:nn { stex / mmtdecl } {
3701   name      .str_set_x:N = \l_stex_symdecl_name_str ,
3702   args      .str_set_x:N = \l_stex_symdecl_args_str ,
3703   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,
3704   reorder   .str_set_x:N = \l_stex_symdecl_reorder_str ,
3705   argnames  .clist_set:N = \l_stex_symdecl_argnames_clist ,
3706   assoc     .choices:nn =
3707     {bin,binl,binr,pre,conj,pwconj}
3708     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}
3709 }
3710
3711 \cs_new_protected:Nn \stex_mmtdecl_args:n {
3712   \str_clear:N \l_stex_symdecl_name_str
3713   \str_clear:N \l_stex_symdecl_args_str
3714   \str_clear:N \l_stex_symdecl_deprecate_str
3715   \str_clear:N \l_stex_symdecl_reorder_str
3716   \str_clear:N \l_stex_symdecl_assoctype_str
3717   \bool_set_false:N \l_stex_symdecl_local_bool
3718   \clist_clear:N \l_stex_symdecl_argnames_clist
3719
3720   \keys_set:nn { stex / symdecl } { #1 }
3721 }
3722
3723 \NewDocumentCommand \mmtdecl { s m O{} } {
3724   \stex_mmtdecl_args:n{#3}
3725   \IfBooleanTF #1 {
3726     \bool_set_false:N \l_stex_symdecl_make_macro_bool
3727   } {

```

```

3728   \bool_set_true:N \l_stex_symdecl_make_macro_bool
3729 }
3730 \str_if_empty:NT \l_stex_symdecl_name_str {
3731   \str_set:Nx \l_stex_symdecl_name_str { #1 }
3732 }
3733 %\tl_set:Nx \l_stex_symdecl_definiens_tl {
3734 %   \stex_annotate:nnn{ OMID }{
3735 %     \l_stex_module_mmtfor_str?\l_stex_symdecl_name_str
3736 %   }{}
3737 %}
3738 \stex_symdecl_do:n{#2}
3739 \MMTrule{rules.stex.mmt.kwarc.info?SubstitutionRule}{
3740   \stex_annotate:nnn{ OMID }{
3741     \l_stex_current_module_str ? \l_stex_symdecl_name_str
3742   }{},
3743   \stex_annotate:nnn{ OMID }{
3744     \l_stex_module_mmtfor_str?\l_stex_symdecl_name_str
3745   }{}
3746 }
3747 \stex_smsmode_do:
3748 }
3749
3750 \stex_deactivate_macro:Nn \mmtdecl {mmtinterface~environments}
3751 \stex_deactivate_macro:Nn \mmtdef {mmtinterface~environments}
3752
3753 </package>

```

Chapter 32

STEX -Terms Implementation

```
3754 <*package>
3755
3756 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
3757
3758 <@@=stex_terms>
3759
3760 Warnings and error messages
3761 \msg_new:nnn{stex}{error/nonotation}{
3762   Symbol~#1~invoked,~but~has~no~notation~#2!
3763 }
3764 \msg_new:nnn{stex}{error/notationarg}{
3765   Error~in~parsing~notation~#1
3766 }
3767 \msg_new:nnn{stex}{error/noop}{
3768   Symbol~#1~has~no~operator~notation~for~notation~#2
3769 }
3770 \msg_new:nnn{stex}{error/notallowed}{
3771   Symbol~invocation~#1~not~allowed~in~notation~component~of~#2
3772 }
3773 \msg_new:nnn{stex}{error/doubleargument}{
3774   Argument~#1~of~symbol~#2~already~assigned
3775 }
3776 \msg_new:nnn{stex}{error/overarity}{
3777   Argument~#1~invalid~for~symbol~#2~with~arity~#3
3778 }
3779
```

32.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
3778
3779
3780 \bool_new:N \l_stex_allow_semantic_bool
3781 \bool_set_true:N \l_stex_allow_semantic_bool
3782
```

```

3783 \cs_new_protected:Nn \stex_invoke_symbol:n {
3784   \ifvmode\indent\fi
3785   \bool_if:NTF \l_stex_allow_semantic_bool {
3786     \str_if_eq:eeF {
3787       \prop_item:cn {
3788         l_stex_symdecl_#1_prop
3789       }{ deprecate }
3790     }{}{
3791       \msg_warning:nxxx{stex}{warning/deprecated}{
3792         Symbol~#1
3793       }{
3794         \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
3795       }
3796     }
3797     \if_mode_math:
3798       \exp_after:wN \__stex_terms_invoke_math:n
3799     \else:
3800       \exp_after:wN \__stex_terms_invoke_text:n
3801     \fi: { #1 }
3802   }{
3803     \msg_error:nxxx{stex}{error/notallowed}{#1}{\STEXInternalCurrentSymbolStr}
3804   }
3805 }
3806
3807 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
3808   \peek_charcode_remove:NTF ! {
3809     \__stex_terms_invoke_op_custom:nn {#1}
3810   }{
3811     \__stex_terms_invoke_custom:nn {#1}
3812   }
3813 }
3814
3815 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
3816   \peek_charcode_remove:NTF ! {
3817     % operator
3818     \peek_charcode_remove:NTF * {
3819       % custom op
3820       \__stex_terms_invoke_op_custom:nn {#1}
3821     }{
3822       % op notation
3823       \peek_charcode:NTF [ {
3824         \__stex_terms_invoke_op_notation:nw {#1}
3825       }{
3826         \__stex_terms_invoke_op_notation:nw {#1}[]
3827       }
3828     }
3829   }{
3830     \peek_charcode_remove:NTF * {
3831       \__stex_terms_invoke_custom:nn {#1}
3832       % custom
3833     }{
3834       % normal
3835       \peek_charcode:NTF [ {
3836         \__stex_terms_invoke_notation:nw {#1}

```

```

3837     }{
3838       \_stex_terms_invoke_notation:nw {#1}[]
3839     }
3840   }
3841 }
3842 }
3843
3844
3845 \cs_new_protected:Nn \_stex_terms_invoke_op_custom:nn {
3846   \exp_args:Nnx \use:nn {
3847     \def\comp{\_comp}
3848     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3849     \bool_set_false:N \l_stex_allow_semantic_bool
3850     \stex_mathml_intent:nn{#1}{
3851       \_stex_term_oms:nnn {#1}{#1 \c_hash_str CUSTOM-}{
3852         \comp{ #2 }
3853       }
3854     }
3855   }{
3856     \_stex_reset:N \comp
3857     \_stex_reset:N \STEXInternalCurrentSymbolStr
3858     \bool_set_true:N \l_stex_allow_semantic_bool
3859   }
3860 }
3861
3862 \keys_define:nn { stex / terms } {
3863   % lang      .tl_set_x:N = \l_stex_notation_lang_str ,
3864   variant .tl_set_x:N = \l_stex_notation_variant_str ,
3865   unknown .code:n      = \str_set:Nx
3866     \l_stex_notation_variant_str \l_keys_key_str
3867 }
3868
3869 \cs_new_protected:Nn \_stex_terms_args:n {
3870   % \str_clear:N \l_stex_notation_lang_str
3871   \str_clear:N \l_stex_notation_variant_str
3872
3873   \keys_set:nn { stex / terms } { #1 }
3874 }
3875
3876 \cs_new_protected:Nn \stex_find_notation:nn {
3877   \_stex_terms_args:n { #2 }
3878   \seq_if_empty:cTF {
3879     \l_stex_symdecl_ #1 _notations
3880   } {
3881     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
3882   } {
3883     \str_if_empty:NTF \l_stex_notation_variant_str {
3884       \seq_get_left:cN {\l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
3885     }{
3886       \seq_if_in:cxTF {\l_stex_symdecl_#1_notations}{
3887         \l_stex_notation_variant_str
3888       }{
3889         % \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3890       }{

```

```

3891         \msg_error:nxxx{stex}{error/nonotation}{#1}{
3892         ~\l_stex_notation_variant_str
3893     }
3894 }
3895 }
3896 }
3897 }
3898
3899 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3900     \exp_args:Nnx \use:nn {
3901         \def\comp{\_comp}
3902         \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3903         \stex_find_notation:nn { #1 }{ #2 }
3904         \bool_set_false:N \l_stex_allow_semantic_bool
3905         \cs_if_exist:cTF {
3906             stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3907         }{
3908             \_stex_term_oms:nnn { #1 }{
3909                 #1 \c_hash_str \l_stex_notation_variant_str
3910             }{
3911                 \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3912             }
3913         }{
3914             \int_compare:nNnTF {\prop_item:cn {l_stex_symdecl_#1_prop}{arity}} = 0{
3915                 \cs_if_exist:cTF {
3916                     stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3917                 }{
3918                     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
3919                         \_stex_reset:N \comp
3920                         \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
3921                         \_stex_reset:N \STEXInternalCurrentSymbolStr
3922                         \bool_set_true:N \l_stex_allow_semantic_bool
3923                     }
3924                     \def\comp{\_comp}
3925                     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3926                     \bool_set_false:N \l_stex_allow_semantic_bool
3927                     \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3928                 }{
3929                     \msg_error:nxxx{stex}{error/nonotation}{#1}{
3930                     ~\l_stex_notation_variant_str
3931                     }
3932                 }
3933             }{
3934                 \msg_error:nxxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3935             }
3936         }
3937     }{
3938         \_stex_reset:N \comp
3939         \_stex_reset:N \STEXInternalCurrentSymbolStr
3940         \bool_set_true:N \l_stex_allow_semantic_bool
3941     }
3942 }
3943
3944 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {

```



```

3945 \stex_find_notation:nn { #1 }{ #2 }
3946 \cs_if_exist:cTF {
3947   stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3948 }{
3949   \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
3950     \_stex_reset:N \comp
3951     \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
3952     \_stex_reset:N \STEXInternalCurrentSymbolStr
3953     \bool_set_true:N \l_stex_allow_semantic_bool
3954   }
3955   \def\comp{\_comp}
3956   \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3957   \bool_set_false:N \l_stex_allow_semantic_bool
3958   \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3959 }{
3960   \msg_error:nnxx{stex}{error/nonotation}{#1}{
3961     ~\l_stex_notation_variant_str
3962   }
3963 }
3964 }
3965
3966 \prop_new:N \l__stex_terms_custom_args_prop
3967 \clist_new:N \l_stex_argnames_seq
3968 \seq_new:N \l__stex_terms_tmp_seq
3969
3970 \cs_new_protected:Nn \__stex_terms_custom_comp:n{ \bool_set_false:N \l_stex_allow_semantic_bo
3971
3972 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3973   \exp_args:Nnx \use:nn {
3974     \def\comp{\__stex_terms_custom_comp:n}
3975     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3976     \prop_clear:N \l__stex_terms_custom_args_prop
3977     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3978     \prop_get:cnN {
3979       l_stex_symdecl_#1 _prop
3980     }{ args } \l_tmpa_str
3981     \exp_args:NNx \seq_set_from_clist:Nn \l_stex_argnames_seq {
3982       \prop_item:cn {l_stex_symdecl_#1 _prop}{argnames}
3983     }
3984     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3985     \tl_set:Nn \arg { \__stex_terms_arg: }
3986     \str_if_empty:NTF \l_tmpa_str {
3987       \stex_mathml_intent:nn{#1}{
3988         \_stex_term_oms:nnn {#1}{#1\c_hash_str CUSTOM-}{\ignorespaces#2}
3989       }
3990     }{
3991       \seq_clear:N \l__stex_terms_tmp_seq
3992       \exp_args:Nx\int_step_inline:nn{\prop_item:cn{l_stex_symdecl_#1 _prop}{arity}}{
3993         \tl_set:Nx \l__stex_terms_tmp_tl {\seq_item:Nn \l_stex_argnames_seq {##1}}
3994         \bool_lazy_or:nnT{
3995           \str_if_eq_p:nn{a}{\str_item:Nn \l_tmpa_str{##1}}
3996         }{
3997           \str_if_eq_p:nn{B}{\str_item:Nn \l_tmpa_str{##1}}
3998         }{

```

```

3999         \tl_put_right:Nn \l__stex_terms_tmp_tl +
4000     }
4001     \seq_put_right:No \l__stex_terms_tmp_seq \l__stex_terms_tmp_tl
4002 }
4003 \stex_mathml_intent:nn{
4004     #1[\prop_item:cn {l_stex_symdecl_#1 _prop}]{ args }(
4005     \seq_use:Nn \l__stex_terms_tmp_seq ,
4006 )
4007 ){
4008     \str_if_in:NnTF \l_tmpa_str b {
4009         \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
4010     }{
4011         \str_if_in:NnTF \l_tmpa_str B {
4012             \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
4013         }{
4014             \_stex_term_oma:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
4015         }
4016     }
4017 }
4018 }
4019 % TODO check that all arguments exist
4020 ){
4021     \_stex_reset:N \l_stex_argnames_seq
4022     \_stex_reset:N \STEXInternalCurrentSymbolStr
4023     \_stex_reset:N \arg
4024     \_stex_reset:N \comp
4025     \_stex_reset:N \l__stex_terms_custom_args_prop
4026     %\bool_set_true:N \l_stex_allow_semantic_bool
4027 }
4028 }
4029
4030 \NewDocumentCommand \__stex_terms_arg: { s O{} m}{
4031     \tl_if_empty:nTF {#2}{
4032         \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
4033         \bool_set_true:N \l_tmpa_bool
4034         \bool_do_while:Nn \l_tmpa_bool {
4035             \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int}
4036             \int_incr:N \l_tmpa_int
4037         }{
4038             \bool_set_false:N \l_tmpa_bool
4039         }
4040     }
4041     ){
4042         \int_set:Nn \l_tmpa_int { #2 }
4043     }
4044     \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
4045     \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
4046         \msg_error:nnxxx{stex}{error/overarity}
4047         {\int_use:N \l_tmpa_int}
4048         {\STEXInternalCurrentSymbolStr}
4049         {\str_count:N \l_tmpa_str}
4050     }
4051     \str_set:Nx \l_tmpa_str {\str_item:Nn \l_tmpa_str \l_tmpa_int}
4052     \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {

```

```

4053 \bool_lazy_any:nF {
4054   {\str_if_eq_p:Vn \l_tmpa_str {a}}
4055   {\str_if_eq_p:Vn \l_tmpa_str {B}}
4056 }{
4057   \msg_error:nnxx{stex}{error/doubleargument}
4058   {\int_use:N \l_tmpa_int}
4059   {\STEXInternalCurrentSymbolStr}
4060 }
4061 }
4062 \exp_args:NNx \prop_put:Nnn \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {\ign
4063 \bool_if:NTF \l_stex_allow_semantic_bool \use_i:nn {
4064   \bool_set_true:N \l_stex_allow_semantic_bool
4065   \use:nn
4066 }
4067 {
4068 \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq \l_tmpa_int}{
4069   \IfBooleanTF#1{
4070     \stex_annotate_invisible:n { %TODO
4071       \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{\ignorespaces#3}
4072     }
4073   }{ %TODO
4074     \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{\ignorespaces#3}
4075   }
4076 }}
4077 {\bool_set_false:N \l_stex_allow_semantic_bool}
4078 }
4079
4080
4081 \cs_new_protected:Nn \_stex_term_arg:nn {
4082   \bool_set_true:N \l_stex_allow_semantic_bool
4083   \stex_annotate:nnn{ arg }{ #1 }{ #2 }
4084   \bool_set_false:N \l_stex_allow_semantic_bool
4085 }
4086
4087 \cs_new_protected:Npn \STEXInternalTermMathArgiii #1#2#3 {
4088   \exp_args:Nnx \use:nn
4089   { \int_set:Nn \l__stex_terms_downprec { #2 }
4090     \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq \l_tmpa_int}{
4091       \_stex_term_arg:nn { #1 }{ #3 }
4092     }
4093   }
4094   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
4095 }

```

(End definition for `\stex_invoke_symbol:n`. This function is documented on page 92.)

`\STEXInternalTermMathAssocArgiiii`

```

4096 \cs_new_protected:Npn \STEXInternalTermMathAssocArgiiii #1#2#3#4#5 {
4097   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
4098   \tl_set:Nn \l_tmpb_tl {\STEXInternalTermMathArgiii{#5#1}{#2}}
4099   \tl_if_empty:nTF { #3 }{
4100     \STEXInternalTermMathArgiii{#5#1}{#2}{}
4101   }{
4102     \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 } }{

```

```

4103     \expandafter\if\expandafter\relax\noexpand#3
4104     \tl_set:Nn \l_tmpa_tl {\_stex_terms_math_assoc_arg_maybe_sequence:Nnn#3{#1}{#5}}
4105   \else
4106     \tl_set:Nn \l_tmpa_tl {\_stex_terms_math_assoc_arg_simple:nnn{#1}{#3}{#5}}
4107   \fi
4108   \l_tmpa_tl
4109 }{
4110   \_stex_terms_math_assoc_arg_simple:nnn{#1}{#3}{#5}
4111 }
4112 }
4113 }
4114
4115 \cs_new_protected:Nn \_stex_terms_math_assoc_arg_maybe_sequence:Nnn {
4116   \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
4117   \str_if_empty:NTF \l_tmpa_str {
4118     \exp_args:Nx \cs_if_eq:NNTF {
4119       \tl_head:N #1
4120     } \stex_invoke_sequence:n {
4121       \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
4122       \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
4123       \tl_set:Nx \l_tmpa_tl {\prop_item:cn {l_stex_symdecl_varseq://\l_tmpa_str _prop}{notat
4124       \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
4125       \tl_set:Nx \l_tmpa_tl {\exp_not:N \exp_not:n{
4126         \exp_not:n{\exp_args:Nnx \use:nn} {
4127           \exp_not:n {
4128             \def\comp{\_varcomp}
4129             \str_set:Nn \STEXInternalCurrentSymbolStr
4130             } {varseq://\l_tmpa_str}
4131             \exp_not:n{ ##1 }
4132           }{
4133             \exp_not:n {
4134               \_stex_reset:N \comp
4135               \_stex_reset:N \STEXInternalCurrentSymbolStr
4136             }
4137           }
4138         }}}
4139         \exp_args:Nno \use:n {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
4140         \seq_reverse:N \l_tmpa_seq
4141         \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
4142         \seq_map_inline:Nn \l_tmpa_seq {
4143           \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
4144             \exp_args:Nno
4145             \l_tmpa_cs { ##1 } \l_tmpa_tl
4146           }
4147         }
4148         \tl_set:Nx \l_tmpa_tl {
4149           \_stex_term_omv:nn {varseq://\l_tmpa_str}{
4150             \exp_args:No \exp_not:n \l_tmpa_tl
4151           }
4152         }
4153         \exp_args:No\l_tmpb_tl\l_tmpa_tl
4154       }{
4155         \_stex_terms_math_assoc_arg_simple:nnn{#2} { #1 }{#3}
4156       }

```

```

4157 } {
4158   \_stex_terms_math_assoc_arg_simple:nnn{#2} { #1 }{#3}
4159 }
4160
4161 }
4162
4163 \cs_new_protected:Nn \_stex_terms_math_assoc_arg_simple:nnn {
4164   \clist_set:Nn \l_tmpa_clist{ #2 }
4165   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
4166     \tl_set:Nn \l_tmpa_tl {
4167       \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq #1}{
4168         \_stex_term_arg:nn{A#3#1}{ #2 } }
4169     }
4170   }{
4171     \clist_reverse:N \l_tmpa_clist
4172     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
4173     \tl_set:Nx \l_tmpa_tl {
4174       \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq #1}{
4175         \_stex_term_arg:nn{A#3#1}{
4176           \exp_args:No \exp_not:n \l_tmpa_tl
4177         }
4178       }
4179     }
4180     \clist_map_inline:Nn \l_tmpa_clist {
4181       \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
4182         \l_tmpa_cs {
4183           \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq #1}{
4184             \_stex_term_arg:nn{A#3#1}{##1}
4185           }
4186         } \l_tmpa_tl
4187       }
4188     }
4189   }
4190   \exp_args:No\l_tmpb_tl\l_tmpa_tl
4191 }

```

(End definition for `\STEXInternalTermMathAssocArgiiii`. This function is documented on page 93.)

32.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
4192 \tl_const:Nx \infprec {\int_use:N \c_max_int}
4193 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
4194 \int_new:N \l__stex_terms_downprec
4195 \int_set_eq:NN \l__stex_terms_downprec \infprec

```

(End definition for `\infprec`, `\neginfprec`, and `\l__stex_terms_downprec`. These variables are documented on page 93.)

Bracketing:

```

\l__stex_terms_left_bracket_str
\l__stex_terms_right_bracket_str
4196 \tl_set:Nn \l__stex_terms_left_bracket_str (
4197 \tl_set:Nn \l__stex_terms_right_bracket_str )

```

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

`__stex_terms_maybe_brackets:nn` Compares precedences and insert brackets accordingly

```

4198 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
4199   \bool_if:NTF \l__stex_terms_brackets_done_bool {
4200     \bool_set_false:N \l__stex_terms_brackets_done_bool
4201     #2
4202   } {
4203     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
4204       \bool_if:NTF \l__stex_inarray_bool { #2 }{
4205         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
4206         \dobrackets { #2 }
4207       }
4208     }{ #2 }
4209   }
4210 }
```

(End definition for `__stex_terms_maybe_brackets:nn`.)

`\dobrackets`

```

4211 \bool_new:N \l__stex_terms_brackets_done_bool
4212 %\RequirePackage{scalerel}
4213 \cs_new_protected:Npn \dobrackets #1 {
4214   %\ThisStyle{\if D\m@switch
4215   %   \exp_args:Nnx \use:nn
4216   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
4217   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
4218   %   \else
4219   %     \exp_args:Nnx \use:nn
4220   %   {
4221   %     \bool_set_true:N \l__stex_terms_brackets_done_bool
4222   %     \int_set:Nn \l__stex_terms_downprec \infpref
4223   %     \l__stex_terms_left_bracket_str
4224   %     #1
4225   %   }
4226   %   {
4227   %     \bool_set_false:N \l__stex_terms_brackets_done_bool
4228   %     \l__stex_terms_right_bracket_str
4229   %     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
4230   %   }
4231   %\fi}
4232 }
```

(End definition for `\dobrackets`. This function is documented on page 93.)

`\withbrackets`

```

4233 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
4234   \exp_args:Nnx \use:nn
4235   {
4236     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
4237     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
4238     #3
4239   }
4240 }
```

```

4241 \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
4242 {\l__stex_terms_left_bracket_str}
4243 \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
4244 {\l__stex_terms_right_bracket_str}
4245 }
4246 }

```

(End definition for `\withbrackets`. This function is documented on page 93.)

`\STEXinvisible`

```

4247 \cs_new_protected:Npn \STEXinvisible #1 {
4248   \stex_annotate_invisible:n { #1 }
4249 }

```

(End definition for `\STEXinvisible`. This function is documented on page 93.)

OMDoc terms:

`\STEXInternalTermMathOMSiiii`

```

4250 \cs_new_protected:Nn \_stex_term_oms:nnn {
4251   \stex_annotate:nnn{ OMID }{ #2 }{
4252     #3
4253   }
4254 }
4255
4256 \cs_new_protected:Npn \STEXInternalTermMathOMSiiii #1#2#3#4 {
4257   \__stex_terms_maybe_brackets:nn { #3 }{
4258     \stex_mathml_intent:nn{#1} {
4259       \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
4260     }
4261   }
4262 }

```

(End definition for `\STEXInternalTermMathOMSiiii`. This function is documented on page 92.)

`_stex_term_math_omv:nn`

```

4263 \cs_new_protected:Nn \_stex_term_omv:nn {
4264   \stex_annotate:nnn{ OMV }{ #1 }{
4265     #2
4266   }
4267 }

```

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

`\STEXInternalTermMathOMAi`

```

4268 \cs_new_protected:Nn \_stex_term_oma:nnn {
4269   \stex_annotate:nnn{ OMA }{ #2 }{
4270     #3
4271   }
4272 }
4273
4274 \cs_new_protected:Npn \STEXInternalTermMathOMAi #1#2#3#4 {
4275   \exp_args:Nnx \use:nn {
4276     \seq_clear:N \l__stex_terms_tmp_seq
4277     \prop_if_exist:cT{l_stex_symdecl_#1 _prop}{
4278       \exp_args:NNx \seq_set_from_clist:Nn \l_stex_argnames_seq {

```

```

4279     \prop_item:cn {l_stex_symdecl_#1 _prop}{argnames}
4280   }
4281   \exp_args:Nx\int_step_inline:nn{\prop_item:cn{l_stex_symdecl_#1 _prop}{arity}}{
4282     \tl_set:Nx \l__stex_terms_tmp_tl {\seq_item:Nn \l_stex_argnames_seq {##1}}
4283     \bool_lazy_or:nnT{
4284       \str_if_eq_p:nn{a}{\str_item:Nn\l_tmpa_str{##1}}
4285     }{
4286       \str_if_eq_p:nn{B}{\str_item:Nn\l_tmpa_str{##1}}
4287     }{
4288       \tl_put_right:Nn \l__stex_terms_tmp_tl +
4289     }
4290     \seq_put_right:No \l__stex_terms_tmp_seq \l__stex_terms_tmp_tl
4291   }
4292 }
4293 \__stex_terms_maybe_brackets:nn { #3 }{
4294   \stex_mathml_intent:nn{
4295     #1[\prop_item:cn {l_stex_symdecl_#1 _prop}{ args }]{
4296       \seq_use:Nn \l__stex_terms_tmp_seq ,
4297     }
4298   }{
4299     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
4300   }
4301 }
4302 }{
4303   \_stex_reset:N \l_stex_argnames_seq
4304 }
4305 }

```

(End definition for `\STEXInternalTermMathOMAi`. This function is documented on page 92.)

`\STEXInternalTermMathOMBiiii`

```

4306 \cs_new_protected:Nn \_stex_term_ombind:nnn {
4307   \stex_annotate:nnn{ OMBIND }{ #2 }{
4308     #3
4309   }
4310 }
4311
4312 \cs_new_protected:Npn \STEXInternalTermMathOMBiiii #1#2#3#4 {
4313   \exp_args:Nnx \use:nn {
4314     \seq_clear:N \l__stex_terms_tmp_seq
4315     \prop_if_exist:cT{l_stex_symdecl_#1 _prop}{
4316       \exp_args:NNx \seq_set_from_clist:Nn \l_stex_argnames_seq {
4317         \prop_item:cn {l_stex_symdecl_#1 _prop}{argnames}
4318       }
4319       \exp_args:Nx\int_step_inline:nn{\prop_item:cn{l_stex_symdecl_#1 _prop}{arity}}{
4320         \tl_set:Nx \l__stex_terms_tmp_tl {\seq_item:Nn \l_stex_argnames_seq {##1}}
4321         \bool_lazy_or:nnT{
4322           \str_if_eq_p:nn{a}{\str_item:Nn\l_tmpa_str{##1}}
4323         }{
4324           \str_if_eq_p:nn{B}{\str_item:Nn\l_tmpa_str{##1}}
4325         }{
4326           \tl_put_right:Nn \l__stex_terms_tmp_tl +
4327         }
4328         \seq_put_right:No \l__stex_terms_tmp_seq \l__stex_terms_tmp_tl

```



```

4329   }
4330 }
4331 \__stex_terms_maybe_brackets:nn { #3 }{
4332   \stex_mathml_intent:nn{
4333     #1[\prop_item:cn {l_stex_symdecl_#1 _prop}{ args }](
4334       \seq_use:Nn \l__stex_terms_tmp_seq ,
4335     )
4336   }{
4337     \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
4338   }
4339 }
4340 }{
4341   \stex_reset:N \l_stex_argnames_seq
4342 }
4343 }

```

(End definition for `\STEXInternalTermMathOMBiiii`. This function is documented on page 92.)

`\symref`
`\symname`

```

4344 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
4345
4346 \keys_define:nn { stex / symname } {
4347   pre      .tl_set_x:N      = \l__stex_terms_pre_tl ,
4348   post     .tl_set_x:N      = \l__stex_terms_post_tl ,
4349   root     .tl_set_x:N      = \l__stex_terms_root_tl
4350 }
4351
4352 \cs_new_protected:Nn \stex_symname_args:n {
4353   \tl_clear:N \l__stex_terms_post_tl
4354   \tl_clear:N \l__stex_terms_pre_tl
4355   \tl_clear:N \l__stex_terms_root_str
4356   \keys_set:nn { stex / symname } { #1 }
4357 }
4358
4359 \NewDocumentCommand \symref { m m }{
4360   \let\compemph_uri_prev:\compemph@uri
4361   \let\compemph@uri\symrefemph@uri
4362   \STEXsymbol{#1}!\{ #2 }
4363   \let\compemph@uri\compemph_uri_prev:
4364 }
4365
4366 \NewDocumentCommand \synonym { 0{} m m }{
4367   \stex_symname_args:n { #1 }
4368   \let\compemph_uri_prev:\compemph@uri
4369   \let\compemph@uri\symrefemph@uri
4370   % TODO
4371   \STEXsymbol{#2}!\{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
4372   \let\compemph@uri\compemph_uri_prev:
4373 }
4374
4375 \NewDocumentCommand \symname { 0{} m }{
4376   \stex_symname_args:n { #1 }
4377   \stex_get_symbol:n { #2 }
4378   \str_set:Nx \l_tmpa_str {

```

```

4379   \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4380 }
4381 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4382
4383 \let\compemph_uri_prev:\compemph@uri
4384 \let\compemph@uri\symrefemph@uri
4385 \exp_args:NNx \use:nn
4386 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmode*\fi{
4387   \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
4388 } }
4389 \let\compemph@uri\compemph_uri_prev:
4390 }
4391
4392 \NewDocumentCommand \Symname { 0{} m }{
4393   \stex_symname_args:n { #1 }
4394   \stex_get_symbol:n { #2 }
4395   \str_set:Nx \l_tmpa_str {
4396     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4397   }
4398   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4399   \let\compemph_uri_prev:\compemph@uri
4400   \let\compemph@uri\symrefemph@uri
4401   \exp_args:NNx \use:nn
4402   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmode*\fi{
4403     \exp_after:wN \stex_capitalize:n \l_tmpa_str
4404     \l__stex_terms_post_tl
4405   } }
4406   \let\compemph@uri\compemph_uri_prev:
4407 }

```

(End definition for `\symref` and `\symname`. These functions are documented on page 92.)

32.3 Notation Components

```

4408 <@@=stex_notationcomps>

\comp
\compemph@uri
\compemph
\defemph
\defemph@uri
\symrefemph
\symrefemph@uri
\varemp
\varemp@uri

4409 \cs_new_protected:Npn \_comp #1 {
4410   \str_if_empty:NF \STEXInternalCurrentSymbolStr {
4411     \stex_html_backend:TF {
4412       \stex_annotate:nnn { comp }{ \STEXInternalCurrentSymbolStr }{ #1 }
4413     }{
4414       \exp_args:Nnx \compemph@uri { #1 } { \STEXInternalCurrentSymbolStr }
4415     }
4416   }
4417 }
4418
4419 \cs_new_protected:Npn \_varcomp #1 {
4420   \str_if_empty:NF \STEXInternalCurrentSymbolStr {
4421     \stex_html_backend:TF {
4422       \stex_annotate:nnn { varcomp }{ \STEXInternalCurrentSymbolStr }{ #1 }
4423     }{
4424       \exp_args:Nnx \varemp@uri { #1 } { \STEXInternalCurrentSymbolStr }
4425     }

```

```

4426 }
4427 }
4428
4429 \def\comp{\_comp}
4430
4431 \cs_new_protected:Npn \compemph@uri #1 #2 {
4432   \compemph{ #1 }
4433 }
4434
4435
4436 \cs_new_protected:Npn \compemph #1 {
4437   #1
4438 }
4439
4440 \cs_new_protected:Npn \defemph@uri #1 #2 {
4441   \defemph{#1}
4442 }
4443
4444 \cs_new_protected:Npn \defemph #1 {
4445   \textbf{#1}
4446 }
4447
4448 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
4449   \symrefemph{#1}
4450 }
4451
4452 \cs_new_protected:Npn \symrefemph #1 {
4453   \emph{#1}
4454 }
4455
4456 \cs_new_protected:Npn \varemp@uri #1 #2 {
4457   \varemp{#1}
4458 }
4459
4460 \cs_new_protected:Npn \varemp #1 {
4461   #1
4462 }

```

(End definition for `\comp` and others. These functions are documented on page 93.)

`\ellipses`

```

4463 \NewDocumentCommand \ellipses {} { \ldots }

```

(End definition for `\ellipses`. This function is documented on page 93.)

```

\parray
\prmatrix
\parrayline
\parraylineh
\parraycell
4464 \bool_new:N \l_stex_inparray_bool
4465 \bool_set_false:N \l_stex_inparray_bool
4466 \NewDocumentCommand \parray { m m } {
4467   \begingroup
4468   \bool_set_true:N \l_stex_inparray_bool
4469   \begin{array}{#1}
4470     #2
4471   \end{array}
4472 \endgroup

```

```

4473 }
4474
4475 \NewDocumentCommand \prmatrix { m } {
4476   \begingroup
4477   \bool_set_true:N \l_stex_inarray_bool
4478   \begin{matrix}
4479     #1
4480   \end{matrix}
4481   \endgroup
4482 }
4483
4484 \def \maybepline {
4485   \bool_if:NT \l_stex_inarray_bool {\hline}
4486 }
4487
4488 \def \parrayline #1 #2 {
4489   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
4490 }
4491
4492 \def \pmrow #1 { \parrayline{}{ #1 } }
4493
4494 \def \parraylineh #1 #2 {
4495   #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
4496 }
4497
4498 \def \parraycell #1 {
4499   #1 \bool_if:NT \l_stex_inarray_bool {&}
4500 }

```

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

32.4 Variables

```

4501 <@@=stex_variables>

```

\stex_invoke_variable:n Invokes a variable

```

4502 \cs_new_protected:Nn \stex_invoke_variable:n {
4503   \if_mode_math:
4504     \exp_after:wN \__stex_variables_invoke_math:n
4505   \else:
4506     \exp_after:wN \__stex_variables_invoke_text:n
4507   \fi: {#1}
4508 }
4509
4510 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
4511   \peek_charcode_remove:NTF ! {
4512     \__stex_variables_invoke_op_custom:nn {#1}
4513   }{
4514     \__stex_variables_invoke_custom:nn {#1}
4515   }
4516 }
4517
4518
4519 \cs_new_protected:Nn \__stex_variables_invoke_math:n {

```

```

4520 \peek_charcode_remove:NTF ! {
4521   \peek_charcode_remove:NTF ! {
4522     \peek_charcode:NTF [ {
4523       % TODO throw error
4524     }{
4525       \__stex_variables_invoke_op_custom:nn
4526     }
4527   }{
4528     \__stex_variables_invoke_op:n { #1 }
4529   }
4530 }{
4531   \peek_charcode_remove:NTF * {
4532     \__stex_variables_invoke_custom:nn { #1 }
4533   }{
4534     \__stex_variables_invoke_math_ii:n { #1 }
4535   }
4536 }
4537 }
4538
4539 \cs_new_protected:Nn \__stex_variables_invoke_op_custom:nn {
4540   \exp_args:Nnx \use:nn {
4541     \def\comp{\_varcomp}
4542     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4543     \bool_set_false:N \l_stex_allow_semantic_bool
4544     \stex_term_omv:nn {var://#1}{
4545       \comp{ #2 }
4546     }
4547   }{
4548     \stex_reset:N \comp
4549     \stex_reset:N \STEXInternalCurrentSymbolStr
4550     \bool_set_true:N \l_stex_allow_semantic_bool
4551   }
4552 }
4553
4554 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
4555   \cs_if_exist:cTF {
4556     stex_var_op_notation_ #1 _cs
4557   }{
4558     \exp_args:Nnx \use:nn {
4559       \def\comp{\_varcomp}
4560       \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4561       \stex_term_omv:nn { var://#1 }{
4562         \use:c{stex_var_op_notation_ #1 _cs }
4563       }
4564     }{
4565       \stex_reset:N \comp
4566       \stex_reset:N \STEXInternalCurrentSymbolStr
4567     }
4568   }{
4569     \int_compare:nNnTF {\prop_item:cn {l_stex_symdecl_var://#1_prop}{arity}} = 0{
4570       \__stex_variables_invoke_math_ii:n {#1}
4571     }{
4572       \msg_error:nxxx{stex}{error/noop}{variable~#1}{}
4573     }

```

```

4574 }
4575 }
4576
4577 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n #1 {
4578   \cs_if_exist:cTF {
4579     stex_var_notation_#1_cs
4580   }{
4581     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4582       \_stex_reset:N \comp
4583       \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
4584       \_stex_reset:N \STEXInternalCurrentSymbolStr
4585       \bool_set_true:N \l_stex_allow_semantic_bool
4586     }
4587     \def\comp{\_varcomp}
4588     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4589     \bool_set_false:N \l_stex_allow_semantic_bool
4590     \use:c{stex_var_notation_#1_cs}
4591   }{
4592     \msg_error:nnxx{stex}{error/nonotation}{variable-#1}{s}
4593   }
4594 }
4595
4596 \cs_new_protected:Nn \__stex_variables_invoke_custom:nn {
4597   \exp_args:Nnx \use:nn {
4598     \def\comp{\_varcomp}
4599     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4600     \prop_clear:N \l__stex_terms_custom_args_prop
4601     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
4602     \prop_get:cnN {
4603       l_stex_symdecl_var://#1 _prop
4604     }{ args } \l_tmpa_str
4605     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
4606     \tl_set:Nn \arg { \_stex_terms_arg: }
4607     \str_if_empty:NTF \l_tmpa_str {
4608       \_stex_term_omv:nn {var://#1}{\ignorespaces#2}
4609     }{
4610       \str_if_in:NnTF \l_tmpa_str b {
4611         \_stex_term_ombind:nnn {var://#1}{\ignorespaces#2}
4612       }{
4613         \str_if_in:NnTF \l_tmpa_str B {
4614           \_stex_term_ombind:nnn {var://#1}{\ignorespaces#2}
4615         }{
4616           \_stex_term_oma:nnn {var://#1}{\ignorespaces#2}
4617         }
4618       }
4619     }
4620     % TODO check that all arguments exist
4621   }{
4622     \_stex_reset:N \STEXInternalCurrentSymbolStr
4623     \_stex_reset:N \arg
4624     \_stex_reset:N \comp
4625     \_stex_reset:N \l__stex_terms_custom_args_prop
4626     %\bool_set_true:N \l_stex_allow_semantic_bool
4627   }

```

```
4628 }
```

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

32.5 Sequences

```
4629 <@@=stex_sequences>
4630
4631 \cs_new_protected:Nn \stex_invoke_sequence:n {
4632   \peek_charcode_remove:NTF ! {
4633     \stex_term_omv:nn {varseq://#1}{
4634       \exp_args:Nnx \use:nn {
4635         \def\comp{\_varcomp}
4636         \str_set:Nn \STEXInternalCurrentSymbolStr {varseq://#1}
4637         \prop_item:cn{l_stex_symdecl_varseq://#1_prop}{notation}
4638       }{
4639         \stex_reset:N \comp
4640         \stex_reset:N \STEXInternalCurrentSymbolStr
4641       }
4642     }
4643   }{
4644     \bool_set_false:N \l_stex_allow_semantic_bool
4645     \def\comp{\_varcomp}
4646     \str_set:Nn \STEXInternalCurrentSymbolStr {varseq://#1}
4647     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4648       \stex_reset:N \comp
4649       \stex_reset:N \STEXInternalSymbolAfterInvokationTL
4650       \stex_reset:N \STEXInternalCurrentSymbolStr
4651       \bool_set_true:N \l_stex_allow_semantic_bool
4652     }
4653     \use:c { stex_varseq_#1_cs }
4654   }
4655 }
4656 </package>
```

Chapter 33

STEX -Structural Features Implementation

```
4657 ⟨*package⟩
4658
4659 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4660
4661 Warnings and error messages
4662 \msg_new:nnn{stex}{error/copymodule/notallowed}{
4663   Symbol~#1~can~not~be~assigned~in~copymodule~#2
4664 }
4665 \msg_new:nnn{stex}{error/interpretmodule/nodfiniens}{
4666   Symbol~#1~not~assigned~in~interpretmodule~#2
4667 }
4668 \msg_new:nnn{stex}{error/unknownstructure}{
4669   No~structure~#1~found!
4670 }
4671
4672 \msg_new:nnn{stex}{error/unknownfield}{
4673   No~field~#1~in~instance~#2~found!~\#3
4674 }
4675
4676 \msg_new:nnn{stex}{error/keyval}{
4677   Invalid~key=value~pair~#1
4678 }
4679 \msg_new:nnn{stex}{error/instantiate/missing}{
4680   Assignments~missing~in~instantiate:~#1
4681 }
4682 \msg_new:nnn{stex}{error/incompatible}{
4683   Incompatible~signature:~#1~(#2)~and~#3~(#4)
4684 }
4685
```


33.1 Imports with modification

```

4686 <@@=stex_copymodule>
4687 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
4688   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4689     \tl_set:Nn \l_tmpa_tl { #1 }
4690     \__stex_copymodule_get_symbol_from_cs:
4691   }{
4692     % argument is a string
4693     % is it a command name?
4694     \cs_if_exist:cTF { #1 }{
4695       \cs_set_eq:Nc \l_tmpa_tl { #1 }
4696       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4697       \str_if_empty:NTF \l_tmpa_str {
4698         \exp_args:Nx \cs_if_eq:NNTF {
4699           \tl_head:N \l_tmpa_tl
4700         } \stex_invoke_symbol:n {
4701           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
4702         }{
4703           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4704         }
4705       } {
4706         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4707       }
4708     }{
4709       % argument is not a command name
4710       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4711       % \l_stex_all_symbols_seq
4712     }
4713   }
4714 }
4715
4716 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
4717   \str_set:Nn \l_tmpa_str { #1 }
4718   \bool_set_false:N \l_tmpa_bool
4719   \bool_if:NF \l_tmpa_bool {
4720     \tl_set:Nn \l_tmpa_tl {
4721       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
4722     }
4723     \str_set:Nn \l_tmpa_str { #1 }
4724     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4725     \seq_map_inline:Nn #2 {
4726       \str_set:Nn \l_tmpb_str { ##1 }
4727       \str_if_eq:eeT { \l_tmpa_str } {
4728         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4729       } {
4730         \seq_map_break:n {
4731           \tl_set:Nn \l_tmpa_tl {
4732             \str_set:Nn \l_stex_get_symbol_uri_str {
4733               ##1
4734             }
4735           }
4736         }
4737       }

```

```

4738     }
4739     \l_tmpa_tl
4740   }
4741 }
4742
4743 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
4744   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4745     { \tl_tail:N \l_tmpa_tl }
4746   \tl_if_single:NTF \l_tmpa_tl {
4747     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
4748       \exp_after:wN \str_set:Nn \exp_after:wN
4749         \l_stex_get_symbol_uri_str \l_tmpa_tl
4750       \__stex_copymodule_get_symbol_check:n { #1 }
4751     }{
4752       % TODO
4753       % tail is not a single group
4754     }
4755   }{
4756     % TODO
4757     % tail is not a single group
4758   }
4759 }
4760
4761 \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
4762   \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
4763     \msg_error:nxxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
4764       :~\seq_use:Nn #1 {,~}
4765     }
4766   }
4767 }
4768
4769 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
4770   % import module
4771   \stex_import_module_uri:nn { #1 } { #2 }
4772   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
4773   \stex_import_require_module:nnnn
4774     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4775     { \l_stex_import_path_str } { \l_stex_import_name_str }
4776
4777   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
4778   \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
4779
4780   % fields
4781   \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
4782   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4783     \seq_map_inline:cn {c_stex_module_##1_constants}{
4784       \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
4785         ##1 ? ####1
4786       }
4787     }
4788   }
4789
4790   % setup prop
4791   \seq_clear:N \l_tmpa_seq

```

```

4792 \exp_args:Nn \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
4793   name      = \l_stex_current_copymodule_name_str ,
4794   module    = \l_stex_current_module_str ,
4795   from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
4796   includes  = \l_tmpa_seq %,
4797 % fields    = \l_tmpa_seq
4798 }
4799 \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
4800   as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
4801 \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_seq {,
4802 \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,
4803
4804 \stex_if_do_html:T {
4805   \begin{stex_annotate_env} {#4} {
4806     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4807   }
4808   \stex_annotate_invisible:nnn{domain}{\l_stex_import_ns_str ?\l_stex_import_name_str}{}
4809 }
4810 }
4811
4812 \cs_new_protected:Nn \stex_copymodule_end:n {
4813   % apply to every field
4814   \def \l_tmpa_cs ##1 ##2 {#1}
4815
4816   \tl_clear:N \__stex_copymodule_module_tl
4817   \tl_clear:N \__stex_copymodule_exec_tl
4818
4819   %\prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
4820   \seq_clear:N \__stex_copymodule_fields_seq
4821
4822   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4823     \seq_map_inline:cn {c_stex_module_##1_constants}{
4824
4825       \tl_clear:N \__stex_copymodule_curr_symbol_tl % <- wrap in current symbol html
4826       \l_tmpa_cs{##1}{####1}
4827
4828       \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
4829         \str_set_eq:Nc \__stex_copymodule_curr_name_str {l__stex_copymodule_copymodule_##1?####1_name_str}
4830         \stex_if_do_html:T {
4831           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4832             \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?####1_name_str}}
4833           }
4834         }
4835       }{
4836         \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str /
4837       }
4838
4839       \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
4840       \prop_put:Nnx \l_tmpa_prop { name } \__stex_copymodule_curr_name_str
4841       \prop_put:Nnx \l_tmpa_prop { module } \l_stex_current_module_str
4842
4843       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
4844         \stex_if_do_html:T {
4845           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {

```

```

4846         $\stex_annotate_invisible:nnn{definiens}{\exp_after:wN \exp_not:N\csname l__st
4847     }
4848 }
4849 \prop_put:Nnn \l_tmpa_prop { defined } { true }
4850 }
4851
4852 \stex_add_constant_to_current_module:n \__stex_copymodule_curr_name_str
4853 \tl_put_right:Nx \__stex_copymodule_module_tl {
4854     \seq_clear:c {l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_n
4855     \prop_set_from_keyval:cn {
4856         l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_name_str _prop
4857     }{
4858         \prop_to_keyval:N \l_tmpa_prop
4859     }
4860 }
4861
4862 \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
4863     \stex_if_do_html:T {
4864         \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4865             \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1
4866         }
4867     }
4868     \tl_put_right:Nx \__stex_copymodule_module_tl {
4869         \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
4870             \stex_invoke_symbol:n {
4871                 \l_stex_current_module_str ? \__stex_copymodule_curr_name_str
4872             }
4873         }
4874     }
4875 }
4876
4877 \seq_put_right:Nx \__stex_copymodule_fields_seq {\l_stex_current_module_str ? \__stex_
4878
4879 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4880     \stex_copy_notations:nn {\l_stex_current_module_str ? \__stex_copymodule_curr_name_s
4881 }
4882
4883 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4884     \stex_if_do_html:TF{
4885         \stex_annotate_invisible:nnn{assignment} {##1?####1} { \exp_after:wN \exp_not:n \e
4886     }{
4887         \exp_after:wN \exp_not:n \exp_after:wN {\__stex_copymodule_curr_symbol_tl}
4888     }
4889 }
4890 }
4891 }
4892
4893
4894 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \__stex_copymodule_fields_seq
4895 \tl_put_left:Nx \__stex_copymodule_module_tl {
4896     \prop_set_from_keyval:cn {
4897         l_stex_copymodule_ \l_stex_current_module_str? \l_stex_current_copymodule_name_str _pro
4898     }{
4899         \prop_to_keyval:N \l_stex_current_copymodule_prop

```

```

4900     }
4901   }
4902
4903   \seq_gput_right:cx{c_stex_module_\l_stex_current_module_str _copymodules}{
4904     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4905   }
4906
4907   \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4908   \stex_debug:nn{copymodule}{result:\meaning \__stex_copymodule_module_tl}
4909   \stex_debug:nn{copymodule}{output:\meaning \__stex_copymodule_exec_tl}
4910
4911   \__stex_copymodule_exec_tl
4912   \stex_if_do_html:T {
4913     \end{stex_annotate_env}
4914   }
4915 }
4916
4917 \NewDocumentEnvironment {copymodule} { 0{} m m}{
4918   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ copymodule }
4919   \stex_deactivate_macro:Nn \symdecl {module~environments}
4920   \stex_deactivate_macro:Nn \symdef {module~environments}
4921   \stex_deactivate_macro:Nn \notation {module~environments}
4922   \stex_reactivate_macro:N \assign
4923   \stex_reactivate_macro:N \renamedekl
4924   \stex_reactivate_macro:N \donotcopy
4925   \stex_smsmode_do:
4926 }{
4927   \stex_copymodule_end:n {}
4928 }
4929
4930 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
4931   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ interpretmodule }
4932   \stex_deactivate_macro:Nn \symdecl {module~environments}
4933   \stex_deactivate_macro:Nn \symdef {module~environments}
4934   \stex_deactivate_macro:Nn \notation {module~environments}
4935   \stex_reactivate_macro:N \assign
4936   \stex_reactivate_macro:N \renamedekl
4937   \stex_reactivate_macro:N \donotcopy
4938   \stex_smsmode_do:
4939 }{
4940   \stex_copymodule_end:n {
4941     \tl_if_exist:cF {
4942       l__stex_copymodule_copymodule_##1?##2_def_tl
4943     }{
4944       \str_if_eq:eeF {
4945         \prop_item:cn{
4946           l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
4947       }{ true }{
4948         \msg_error:nxxx{stex}{error/interpretmodule/nodéfiniens}{
4949           ##1?##2
4950         }{\l_stex_current_copymodule_name_str}
4951       }
4952     }
4953   }

```

```

4954 }
4955
4956 \iffalse \begin{stex_annotate_env} \fi
4957 \NewDocumentEnvironment {realization} { 0 } { m } {
4958   \stex_copymodule_start:nnnn { #1 } { #2 } { #2 } { realize }
4959   \stex_deactivate_macro:Nn \symdecl {module~environments}
4960   \stex_deactivate_macro:Nn \symdef {module~environments}
4961   \stex_deactivate_macro:Nn \notation {module~environments}
4962   \stex_reactivate_macro:N \donotcopy
4963   \stex_reactivate_macro:N \assign
4964   \stex_smsmode_do:
4965 } {
4966   \stex_import_module_uri:nn { #1 } { #2 }
4967   \tl_clear:N \__stex_copymodule_exec_tl
4968   \tl_set:Nx \__stex_copymodule_module_tl {
4969     \stex_import_require_module:nnnn
4970     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4971     { \l_stex_import_path_str } { \l_stex_import_name_str }
4972   }
4973
4974   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4975     \seq_map_inline:cn {c_stex_module_##1_constants}{
4976       \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str / #1 }
4977       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?###1_def_tl}{
4978         \stex_if_do_html:T {
4979           \tl_put_right:Nx \__stex_copymodule_exec_tl {
4980             \stex_annotate_invisible:nnn{assignment} {##1?###1} {
4981               $\stex_annotate_invisible:nnn{definien}{}{\exp_after:wN \exp_not:N\csname l__
4982             }
4983           }
4984         }
4985         \tl_put_right:Nx \__stex_copymodule_module_tl {
4986           \prop_put:cnn {l_stex_symdecl_##1?###1_prop}{ defined }{ true }
4987         }
4988       }
4989     }
4990
4991     \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4992
4993     \__stex_copymodule_exec_tl
4994     \stex_if_do_html:T {\end{stex_annotate_env}}
4995   }
4996
4997 \NewDocumentCommand \donotcopy { m } {
4998   \str_clear:N \l_stex_import_name_str
4999   \str_set:Nn \l_tmpa_str { #1 }
5000   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
5001   \seq_map_inline:Nn \l_stex_all_modules_seq {
5002     \str_set:Nn \l_tmpb_str { ##1 }
5003     \str_if_eq:eeT { \l_tmpa_str } {
5004       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
5005     } {
5006       \seq_map_break:n {
5007         \stex_if_do_html:T {

```

```

5008         \stex_if_smsmode:F {
5009             \stex_annotate_invisible:nnn{donotcopy}{##1}{
5010                 \stex_annotate:nnn{domain}{##1}{}}
5011         }
5012     }
5013 }
5014 \str_set_eq:NN \l_stex_import_name_str \l_tmpb_str
5015 }
5016 }
5017 \seq_map_inline:cn {c_stex_module_###1_copymodules}{
5018     \str_set:Nn \l_tmpb_str { #####1 }
5019     \str_if_eq:eeT { \l_tmpa_str } {
5020         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
5021     } {
5022         \seq_map_break:n {\seq_map_break:n {
5023             \stex_if_do_html:T {
5024                 \stex_if_smsmode:F {
5025                     \stex_annotate_invisible:nnn{donotcopy}{#####1}{
5026                         \stex_annotate:nnn{domain}{
5027                             \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
5028                         }{}
5029                     }
5030                 }
5031             }
5032             \str_set:Nx \l_stex_import_name_str {
5033                 \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
5034             }
5035         }}
5036     }
5037 }
5038 }
5039 \str_if_empty:NTF \l_stex_import_name_str {
5040     % TODO throw error
5041 }{
5042     \stex_collect_imports:n {\l_stex_import_name_str }
5043     \seq_map_inline:Nn \l_stex_collect_imports_seq {
5044         \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
5045         \seq_map_inline:cn {c_stex_module_###1_constants}{
5046             \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? #####1 }
5047             \bool_lazy_any:nT {
5048                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_name_str}}
5049                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_macroname_str}}
5050                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_def_tl}}
5051             }{
5052                 % TODO throw error
5053             }
5054         }
5055     }
5056     \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
5057     \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_name_str }
5058     \prop_put:Nno \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
5059 }
5060 \stex_smsmode_do:
5061 }

```

```

5062
5063 \NewDocumentCommand \assign { m m }{
5064   \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
5065   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
5066   \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_def_tl}{#2}
5067   \stex_smsmode_do:
5068 }
5069
5070 \keys_define:nn { stex / renamedekl } {
5071   name          .str_set_x:N = \l_stex_renamedekl_name_str
5072 }
5073 \cs_new_protected:Nn \__stex_copymodule_renamedekl_args:n {
5074   \str_clear:N \l_stex_renamedekl_name_str
5075   \keys_set:nn { stex / renamedekl } { #1 }
5076 }
5077
5078 \NewDocumentCommand \renamedekl { O{} m m }{
5079   \__stex_copymodule_renamedekl_args:n { #1 }
5080   \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
5081   \stex_debug:nn{renamedekl}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
5082   \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_macroname_str}{#3}
5083   \str_if_empty:NTF \l_stex_renamedekl_name_str {
5084     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
5085       \l_stex_get_symbol_uri_str
5086     } }
5087   } {
5088     \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_name_str}{\l_stex
5089     \stex_debug:nn{renamedekl}{@~\l_stex_current_module_str ? \l_stex_renamedekl_name_str}
5090     \prop_set_eq:cc {l_stex_symdecl_
5091       \l_stex_current_module_str ? \l_stex_renamedekl_name_str
5092     _prop
5093     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str_prop}
5094     \seq_set_eq:cc {l_stex_symdecl_
5095       \l_stex_current_module_str ? \l_stex_renamedekl_name_str
5096     _notations
5097     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str_notations}
5098     \prop_put:cnx {l_stex_symdecl_
5099       \l_stex_current_module_str ? \l_stex_renamedekl_name_str
5100     _prop
5101     }{ name }{ \l_stex_renamedekl_name_str }
5102     \prop_put:cnx {l_stex_symdecl_
5103       \l_stex_current_module_str ? \l_stex_renamedekl_name_str
5104     _prop
5105     }{ module }{ \l_stex_current_module_str }
5106     \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
5107       \l_stex_current_module_str ? \l_stex_renamedekl_name_str
5108     }
5109     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
5110       \l_stex_current_module_str ? \l_stex_renamedekl_name_str
5111     } }
5112   }
5113   \stex_smsmode_do:
5114 }
5115

```



```

5116 \stex_deactivate_macro:Nn \assign {copymodules}
5117 \stex_deactivate_macro:Nn \renamedekl {copymodules}
5118 \stex_deactivate_macro:Nn \donotcopy {copymodules}
5119
5120

```

33.2 The feature environment

`structural@feature (env.)`

```

5121 <@@=stex_features>
5122
5123 \NewDocumentEnvironment{structural_feature_module}{m m m}{
5124   \stex_if_in_module:F {
5125     \msg_set:nnn{stex}{error/nomodule}{
5126       Structural~Feature~has~to~occur~in~a~module:\\
5127       Feature~#2~of~type~#1\\
5128       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
5129     }
5130     \msg_error:nn{stex}{error/nomodule}
5131   }
5132
5133   \str_set_eq:NN \l_stex_feature_parent_str \l_stex_current_module_str
5134
5135   \stex_module_setup:nn{meta=NONE}{#2 - #1}
5136
5137   \stex_if_do_html:T {
5138     \begin{stex_annotate_env}{feature:#1 }{\l_stex_feature_parent_str ? #2 - #1}
5139     \stex_annotate_invisible:nnn{header}{}{ #3 }
5140   }
5141 }{
5142   \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
5143   \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
5144   \stex_debug:nn{features}{
5145     Feature: \l_stex_last_feature_str
5146   }
5147   \stex_if_do_html:T {
5148     \end{stex_annotate_env}
5149   }
5150 }

```

33.3 Structure

`structure (env.)`

```

5151 <@@=stex_structures>
5152 \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
5153   \prop_if_exist:cF {c_stex_module_ \l_stex_current_module_str _structures}{
5154     \prop_new:c {c_stex_module_ \l_stex_current_module_str _structures}
5155   }
5156   \prop_gput:cxn{c_stex_module_ \l_stex_current_module_str _structures}
5157   {#1}{#2}
5158 }
5159

```

```

5160 \keys_define:nn { stex / features / structure } {
5161   name          .str_set_x:N = \l__stex_structures_name_str ,
5162 }
5163
5164 \cs_new_protected:Nn \__stex_structures_structure_args:n {
5165   \str_clear:N \l__stex_structures_name_str
5166   \keys_set:nn { stex / features / structure } { #1 }
5167 }
5168 \NewDocumentEnvironment{mathstructure}{m O{}}{
5169   \begin{mathstructure_inner}{#1}[#2]
5170     \stex_smsmode_do:
5171     \ignorespacesandpars
5172   }\end{mathstructure_inner}}
5173 \NewDocumentEnvironment{mathstructure_inner}{m O{}}{
5174   \__stex_structures_structure_args:n { #2 }
5175   \str_if_empty:NT \l__stex_structures_name_str {
5176     \str_set:Nx \l__stex_structures_name_str { #1 }
5177   }
5178   \stex_suppress_html:n {
5179     \bool_set_true:N \l_stex_symdecl_make_macro_bool
5180     \exp_args:Nx \stex_symdecl_do:nn {
5181       name = \l__stex_structures_name_str ,
5182       def = {\STEXsymbol{module-type}}{
5183         \STEXInternalTermMathOMSiiii {
5184           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
5185           { ns } ?
5186           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
5187           { name } / \l__stex_structures_name_str - structure
5188         }{}{}{}
5189       }
5190     }{ #1 }
5191   }
5192   \exp_args:Nnnx
5193   \begin{structural_feature_module}{ structure }
5194     { \l__stex_structures_name_str }{}
5195   }{
5196     \end{structural_feature_module}
5197     \stex_reset_up_to_module:n \l_stex_last_feature_str
5198     \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
5199     \seq_clear:N \l_tmpa_seq
5200     \seq_map_inline:Nn \l_stex_collect_imports_seq {
5201       \seq_map_inline:cn{c_stex_module_##1_constants}{
5202         \seq_put_right:Nn \l_tmpa_seq { ##1 ? #####1 }
5203       }
5204     }
5205     \exp_args:Nnno
5206     \prop_gput:cn {c_stex_module_ \l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
5207     \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
5208     \stex_add_structure_to_current_module:nn
5209       \l__stex_structures_name_str
5210       \l_stex_last_feature_str
5211
5212     \stex_execute_in_module:x {
5213       \tl_set:cn { #1 }{

```

```

5214     \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structure
5215   }
5216 }
5217 }
5218
5219 \cs_new:Nn \stex_invoke_structure:nn {
5220   \stex_invoke_symbol:n { #1?#2 }
5221 }
5222
5223 \cs_new_protected:Nn \stex_get_structure:n {
5224   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
5225     \tl_set:Nn \l_tmpa_tl { #1 }
5226     \__stex_structures_get_from_cs:
5227   }{
5228     \cs_if_exist:cTF { #1 }{
5229       \cs_set_eq:Nc \l_tmpa_cs { #1 }
5230       \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
5231       \str_if_empty:NNTF \l_tmpa_str {
5232         \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs} \stex_invoke_structure:nn {
5233           \__stex_structures_get_from_cs:
5234         }{
5235           \__stex_structures_get_from_string:n { #1 }
5236         }
5237       }{
5238         \__stex_structures_get_from_string:n { #1 }
5239       }
5240     }{
5241       \__stex_structures_get_from_string:n { #1 }
5242     }
5243   }
5244 }
5245
5246 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
5247   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
5248   { \tl_tail:N \l_tmpa_tl }
5249   \str_set:Nx \l_tmpa_str {
5250     \exp_after:wN \use_i:nn \l_tmpa_tl
5251   }
5252   \str_set:Nx \l_tmpb_str {
5253     \exp_after:wN \use_ii:nn \l_tmpa_tl
5254   }
5255   \str_set:Nx \l_stex_get_structure_str {
5256     \l_tmpa_str ? \l_tmpb_str
5257   }
5258   \str_set:Nx \l_stex_get_structure_module_str {
5259     \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
5260   }
5261 }
5262
5263 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
5264   \tl_set:Nn \l_tmpa_tl {
5265     \msg_error:nnn{stex}{error/unknownstructure}{#1}
5266   }
5267   \str_set:Nn \l_tmpa_str { #1 }

```

```

5268 \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
5269
5270 \seq_map_inline:Nn \l_stex_all_modules_seq {
5271   \prop_if_exist:cT {c_stex_module_##1_structures} {
5272     \prop_map_inline:cn {c_stex_module_##1_structures} {
5273       \exp_args:No \str_if_eq:nnT \l_tmpa_str {####1}{
5274         %\str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
5275         \prop_map_break:n{\seq_map_break:n{
5276           \tl_set:Nn \l_tmpa_tl {
5277             \str_set:Nn \l_stex_get_structure_str {##1?####1}
5278             \str_set:Nn \l_stex_get_structure_module_str {####2}
5279           }
5280         }}
5281       }
5282     }
5283   }
5284 }
5285 \l_tmpa_tl
5286 }

```

\instantiate

```

5287
5288 \NewDocumentEnvironment{usestructure}{m}{
5289   \stex_get_structure:n {#1}
5290   \exp_args:Nnx \stex_debug:nn{features}{using~structure:~\l_stex_get_structure_module_str}
5291   \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5292 }{}
5293
5294 \keys_define:nn { stex / instantiate } {
5295   name .str_set_x:N = \l__stex_structures_name_str
5296 }
5297 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
5298   \str_clear:N \l__stex_structures_name_str
5299   \keys_set:nn { stex / instantiate } { #1 }
5300 }
5301
5302 \NewDocumentEnvironment{extstructure}{m m O{}}{
5303   \begin{mathstructure_inner}{#1}[#3]
5304     \seq_set_split:Nnn\__stex_structures_extstructure_imports_seq,{#2}
5305     \seq_map_inline:Nn\__stex_structures_extstructure_imports_seq {
5306       \stex_get_structure:n {##1}
5307       \exp_args:Nnx \stex_debug:nn{features}{importing~structure:~\l_stex_get_structure_modu
5308       \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5309       \stex_if_smsmode:F {
5310         \stex_annotate_invisible:nnn
5311         {import} {\l_stex_get_structure_module_str} {}
5312       }
5313       \exp_args:Nx \stex_add_import_to_current_module:n {
5314         \l_stex_get_structure_module_str
5315       }
5316       \exp_args:Nx \stex_add_to_current_module:n {
5317         \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5318       }
5319     }

```

```

5320 \stex_smsmode_do:
5321 \ignorespacesandpars
5322 }{
5323 \end{mathstructure_inner}
5324 }
5325
5326 \NewDocumentEnvironment{extstructure*}{m m O{}}{
5327 % TODO
5328 \begin{extstructure}{#1}{#2}{#3}
5329 }{
5330 \end{extstructure}
5331 }
5332
5333 \NewDocumentCommand \instantiate {m O{ } m m O{}}{
5334 \begin{group
5335 \stex_get_structure:n {#3}
5336 \__stex_structures_instantiate_args:n { #2 }
5337 \str_if_empty:NT \l__stex_structures_name_str {
5338 \str_set:Nn \l__stex_structures_name_str { #1 }
5339 }
5340 \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5341 \seq_clear:N \l__stex_structures_fields_seq
5342 \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
5343 \seq_map_inline:Nn \l_stex_collect_imports_seq {
5344 \seq_map_inline:cn {c_stex_module_##1_constants}{
5345 \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? #####1 }
5346 }
5347 }
5348
5349 \tl_if_empty:nF{#5}{
5350 \seq_set_split:Nnn \l_tmpa_seq , {#5}
5351 \prop_clear:N \l_tmpa_prop
5352 \seq_map_inline:Nn \l_tmpa_seq {
5353 \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
5354 \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
5355 \msg_error:nnn{stex}{error/keyval}{##1}
5356 }
5357 \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
5358 \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
5359 \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
5360 \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}
5361 \exp_args:Nxx \str_if_eq:nnF
5362 {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5363 {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
5364 \msg_error:nnxxx{stex}{error/incompatible}
5365 {\l__stex_structures_dom_str}
5366 {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5367 {\l_stex_get_symbol_uri_str}
5368 {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
5369 }
5370 \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str
5371 }
5372 }
5373

```

```

5374 \seq_map_inline:Nn \l__stex_structures_fields_seq {
5375   \str_set:Nx \l_tmpa_str {field:\l__stex_structures_name_str . \prop_item:cn {l_stex_sy
5376   \stex_debug:nn{instantiate}{Field~\l_tmpa_str :~##1}
5377
5378   \stex_add_constant_to_current_module:n {\l_tmpa_str}
5379   \stex_execute_in_module:x {
5380     \prop_set_from_keyval:cn { l_stex_symdecl_ \l_stex_current_module_str?\l_tmpa_str _p
5381     name   = \l_tmpa_str ,
5382     args   = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
5383     arity  = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
5384     assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs} ,
5385     argnames = {\prop_item:cn {l_stex_symdecl_##1_prop}{argnames}}
5386   }
5387   \seq_clear:c {l_stex_symdecl_ \l_stex_current_module_str?\l_tmpa_str _notations}
5388 }
5389
5390 \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
5391   \stex_find_notation:nn{##1}{}
5392   \stex_execute_in_module:x {
5393     \seq_put_right:cn {l_stex_symdecl_ \l_stex_current_module_str?\l_tmpa_str _notation
5394   }
5395
5396   \stex_copy_control_sequence_ii:ccN
5397   {stex_notation_ \l_stex_current_module_str?\l_tmpa_str \c_hash_str \l_stex_notation_
5398   {stex_notation_##1 \c_hash_str \l_stex_notation_variant_str _cs}
5399   \l_tmpa_tl
5400   \exp_args:No \stex_execute_in_module:n \l_tmpa_tl
5401
5402
5403   \cs_if_exist:cT{stex_op_notation_##1 \c_hash_str \l_stex_notation_variant_str _cs}{
5404     \tl_set_eq:Nc \l_tmpa_cs {stex_op_notation_##1 \c_hash_str \l_stex_notation_variant
5405     \stex_execute_in_module:x {
5406       \tl_set:cn
5407       {stex_op_notation_ \l_stex_current_module_str?\l_tmpa_str \c_hash_str \l_stex_not
5408       { \exp_args:No \exp_not:n \l_tmpa_cs}
5409     }
5410   }
5411
5412 }
5413
5414 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\l_stex_cur
5415 }
5416
5417 \stex_execute_in_module:x {
5418   \prop_set_from_keyval:cn {l_stex_instance_ \l_stex_current_module_str?\l__stex_structur
5419   domain = \l_stex_get_structure_module_str ,
5420   \prop_to_keyval:N \l_tmpa_prop
5421 }
5422 \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
5423 }
5424 \stex_debug:nn{instantiate}{
5425   Instance~\l_stex_current_module_str?\l__stex_structures_name_str \
5426   \prop_to_keyval:N \l_tmpa_prop
5427 }

```

```

5428 \exp_args:Nxx \stex_symdecl_do:nn {
5429   type={\STEXsymbol{module-type}}{
5430     \STEXInternalTermMathOMSiiii {
5431       \l_stex_get_structure_module_str
5432     }{}{0}{}
5433   }}
5434 }{\l__stex_structures_name_str}
5435 % {
5436   \str_set:Nx \l_stex_get_symbol_uri_str {\l_stex_current_module_str?\l__stex_structures
5437   \tl_set:Nn \l_stex_notation_after_do_tl {\_stex_notation_final:}
5438   \stex_notation_do:nnnnn{}{}{}{}{\comp{#4}}
5439 % }
5440 %\exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}
5441 \endgroup
5442 \stex_smsmode_do:\ignorespacesandpars
5443 }
5444
5445 \cs_new_protected:Nn \stex_symbol_or_var:n {
5446   \cs_if_exist:cTF{#1}{
5447     \cs_set_eq:Nc \l_tmpa_tl { #1 }
5448     \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
5449     \str_if_empty:NTF \l_tmpa_str {
5450       \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
5451       \stex_invoke_variable:n {
5452         \bool_set_true:N \l_stex_symbol_or_var_bool
5453         \bool_set_false:N \l_stex_instance_or_symbol_bool
5454         \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
5455         \tl_set:Nx \l_tmpa_tl {\exp_after:wN \use:n \l_tmpa_tl}
5456         \str_set:Nx \l_stex_get_symbol_uri_str {
5457           \exp_after:wN \use:n \l_tmpa_tl
5458         }
5459       }{ % TODO \stex_invoke_varinstance:n
5460         \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl } \stex_invoke_varinstance:n {
5461           \bool_set_true:N \l_stex_symbol_or_var_bool
5462           \bool_set_true:N \l_stex_instance_or_symbol_bool
5463           \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
5464           \tl_set:Nx \l_tmpa_tl {\exp_after:wN \use:n \l_tmpa_tl}
5465           \str_set:Nx \l_stex_get_symbol_uri_str {
5466             \exp_after:wN \use:n \l_tmpa_tl
5467           }
5468         }{
5469           \bool_set_false:N \l_stex_symbol_or_var_bool
5470           \stex_get_symbol:n{#1}
5471         }
5472       }
5473     }{
5474       \__stex_structures_symbolorvar_from_string:n{ #1 }
5475     }
5476   }{
5477     \__stex_structures_symbolorvar_from_string:n{ #1 }
5478   }
5479 }
5480
5481 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {

```

```

5482 \prop_if_exist:cTF {l_stex_symdecl_var://#1 _prop}{
5483   \bool_set_true:N \l_stex_symbol_or_var_bool
5484   \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
5485 }{
5486   \bool_set_false:N \l_stex_symbol_or_var_bool
5487   \stex_get_symbol:n{#1}
5488 }
5489 }
5490
5491 \keys_define:nn { stex / varinstantiate } {
5492   name          .str_set_x:N = \l__stex_structures_name_str,
5493   bind          .choices:nn =
5494     {forall,exists}
5495     {\str_set:Nx \l__stex_structures_bind_str {\l_keys_choice_tl}}
5496 }
5497 }
5498 \cs_new_protected:Nn \__stex_structures_varinstantiate_args:n {
5499   \str_clear:N \l__stex_structures_name_str
5500   \str_clear:N \l__stex_structures_bind_str
5501   \keys_set:nn { stex / varinstantiate } { #1 }
5502 }
5503
5504 \NewDocumentCommand \varinstantiate {m O{} m m O{}}{
5505   \beginingroup
5506     \stex_get_structure:n {#3}
5507     \__stex_structures_varinstantiate_args:n { #2 }
5508     \str_if_empty:NT \l__stex_structures_name_str {
5509       \str_set:Nn \l__stex_structures_name_str { #1 }
5510     }
5511     \stex_if_do_html:TF{
5512       \stex_annotate:nnn{varinstance}{\l__stex_structures_name_str}
5513     }{\use:n}
5514     {
5515       \stex_if_do_html:T{
5516         \stex_annotate_invisible:nnn{domain}{\l_stex_get_structure_module_str}{}
5517       }
5518       \seq_clear:N \l__stex_structures_fields_seq
5519       \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
5520       \seq_map_inline:Nn \l_stex_collect_imports_seq {
5521         \seq_map_inline:cn {c_stex_module_##1_constants}{
5522           \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
5523         }
5524       }
5525       \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5526       \prop_clear:N \l_tmpa_prop
5527       \tl_if_empty:nF {#5} {
5528         \seq_set_split:Nnn \l_tmpa_seq , {#5}
5529         \seq_map_inline:Nn \l_tmpa_seq {
5530           \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
5531           \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
5532             \msg_error:nnn{stex}{error/keyval}{##1}
5533           }
5534           \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_stru
5535           \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str

```



```

5536 \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol
5537 \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
5538 \stex_if_do_html:T{
5539 \stex_annotate:nnn{assign}{\l__stex_structures_dom_str,
5540 \bool_if:NTF\l_stex_symbol_or_var_bool{var://}{\l_stex_get_symbol_uri_str}{}}
5541 }
5542 \bool_if:NTF \l_stex_symbol_or_var_bool {
5543 \exp_args:Nxx \str_if_eq:nnF
5544 {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
5545 {\prop_item:cn{l_stex_symdecl_var://\l_stex_get_symbol_uri_str _prop}{args}}{
5546 \msg_error:nnxxx{stex}{error/incompatible}
5547 {\l__stex_structures_dom_str}
5548 {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
5549 {\l_stex_get_symbol_uri_str}
5550 {\prop_item:cn{l_stex_symdecl_var://\l_stex_get_symbol_uri_str _prop}{args}}}
5551 }
5552 \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n
5553 }{
5554 \exp_args:Nxx \str_if_eq:nnF
5555 {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
5556 {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}{
5557 \msg_error:nnxxx{stex}{error/incompatible}
5558 {\l__stex_structures_dom_str}
5559 {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
5560 {\l_stex_get_symbol_uri_str}
5561 {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}}
5562 }
5563 \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n
5564 }
5565 }
5566 }
5567 \tl_gclear:N \g__stex_structures_aftergroup_tl
5568 \seq_map_inline:Nn \l__stex_structures_fields_seq {
5569 \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdecl
5570 \stex_debug:nn{varinstantiate}{Field~\l_tmpa_str :~##1}
5571 \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
5572 \stex_find_notation:nn{##1}{}
5573 \cs_gset_eq:cc{g__stex_structures_tmpa\l_tmpa_str _cs}
5574 {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5575 \stex_debug:nn{varinstantiate}{Notation:~\cs_meaning:c{g__stex_structures_tmpa\l
5576 \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
5577 \cs_gset_eq:cc {g__stex_structures_tmpa_op\l_tmpa_str _cs}
5578 {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5579 \stex_debug:nn{varinstantiate}{Operator~Notation:~\cs_meaning:c{g__stex_struct
5580 }
5581 }
5582 }
5583 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
5584 \prop_set_from_keyval:cn { l_stex_symdecl_ var://\l_tmpa_str _prop}{
5585 name = \l_tmpa_str ,
5586 args = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
5587 arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
5588 assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs} ,
5589 argnames = {\prop_item:cn {l_stex_symdecl_##1_prop}{argnames}} ,

```

```

5590     }
5591     \cs_set_eq:cc {stex_var_notation_\l_tmpa_str_cs}
5592     {g__stex_structures_tmpa_\l_tmpa_str_cs}
5593     \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str_cs}
5594     {g__stex_structures_tmpa_op_\l_tmpa_str_cs}
5595   }
5596   \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\stex_inv
5597 }
5598 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
5599   \prop_set_from_keyval:cn {l_stex_varinstance_\l__stex_structures_name_str_prop }{
5600     domain = \l_stex_get_structure_module_str ,
5601     \prop_to_keyval:N \l_tmpa_prop
5602   }
5603   \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
5604   \tl_set:cn {l_stex_varinstance_\l__stex_structures_name_str_op_tl}{
5605     \exp_args:Nnx \exp_not:N \use:nn {
5606       \str_set:Nn \exp_not:N \STEXInternalCurrentSymbolStr {var://\l__stex_structures_
5607       \_stex_term_omv:nn {var://\l__stex_structures_name_str}{
5608         \exp_not:n{
5609           \_varcomp{#4}
5610         }
5611       }
5612     }{
5613       \exp_not:n{\_stex_reset:N \STEXInternalCurrentSymbolStr}
5614     }
5615   }
5616 }
5617 }
5618 \stex_debug:nn{varinstantiate}{\expandafter\detokenize\expandafter{\g__stex_structures_a
5619 \aftergroup\g__stex_structures_aftergroup_tl
5620 \endgroup
5621 \stex_smsmode_do:\ignorespacesandpars
5622 }
5623
5624 \cs_new_protected:Nn \stex_invoke_instance:n {
5625   \peek_charcode_remove:NTF ! {
5626     \stex_invoke_symbol:n{#1}
5627   }{
5628     \_stex_invoke_instance:nn {#1}
5629   }
5630 }
5631
5632
5633 \cs_new_protected:Nn \stex_invoke_varinstance:n {
5634   \peek_charcode_remove:NTF ! {
5635     \exp_args:Nnx \use:nn {
5636       \def\comp{\_varcomp}
5637       \use:c{l_stex_varinstance_#1_op_tl}
5638     }{
5639       \_stex_reset:N \comp
5640     }
5641   }{
5642     \_stex_invoke_varinstance:nn {#1}
5643   }

```

```

5644 }
5645
5646 \cs_new_protected:Nn \stex_invoke_instance:nn {
5647   \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
5648     \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
5649   }{
5650     \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
5651     \msg_error:nnxxx{stex}{error/unknownfield}{#2}{#1}{
5652       \prop_to_keyval:N \l_tmpa_prop
5653     }
5654   }
5655 }
5656
5657 \cs_new_protected:Nn \stex_invoke_varinstance:nn {
5658   \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
5659     \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
5660     \l_tmpa_tl
5661   }{
5662     \msg_error:nnnnn{stex}{error/unknownfield}{#2}{#1}{}
5663   }
5664 }

```

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

\stex_invoke_structure:nnn

```

5665 % #1: URI of the instance
5666 % #2: URI of the instantiated module
5667 \cs_new_protected:Nn \stex_invoke_structure:nnn {
5668   \tl_if_empty:nTF{ #3 }{
5669     \prop_set_eq:Nc \l__stex_structures_structure_prop {
5670       c_stex_feature_ #2 _prop
5671     }
5672     \tl_clear:N \l_tmpa_tl
5673     \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
5674     \seq_map_inline:Nn \l_tmpa_seq {
5675       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
5676       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
5677       \cs_if_exist:cT {
5678         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
5679       }{
5680         \tl_if_empty:NF \l_tmpa_tl {
5681           \tl_put_right:Nn \l_tmpa_tl {,}
5682         }
5683         \tl_put_right:Nx \l_tmpa_tl {
5684           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
5685         }
5686       }
5687     }
5688     \exp_args:No \mathstruct \l_tmpa_tl
5689   }{
5690     \stex_invoke_symbol:n{#1/#3}
5691   }
5692 }

```

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

5693 `\endpackage`

Chapter 34

STEX -Statements Implementation

```
5694 <*package>
5695
5696 %%%%%%%%%%% features.dtx %%%%%%%%%%%
5697
5698 <@@=stex_statements>
5699
5700 Warnings and error messages
5701
5702 \titleemph
5703 \def\titleemph#1{\textbf{#1}}
5704
5705 (End definition for \titleemph. This function is documented on page ??.)
```

34.1 Definitions

definiendum

```
5701 \keys_define:nn {stex / definiendum }{
5702   pre      .tl_set:N      = \l__stex_statements_definiendum_pre_tl,
5703   post     .tl_set:N      = \l__stex_statements_definiendum_post_tl,
5704   root     .str_set_x:N    = \l__stex_statements_definiendum_root_str,
5705   gfa      .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
5706 }
5707 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
5708   \str_clear:N \l__stex_statements_definiendum_root_str
5709   \tl_clear:N \l__stex_statements_definiendum_post_tl
5710   \str_clear:N \l__stex_statements_definiendum_gfa_str
5711   \keys_set:nn { stex / definiendum }{ #1 }
5712 }
5713 \NewDocumentCommand \definiendum { O{} m m } {
5714   \__stex_statements_definiendum_args:n { #1 }
5715   \stex_get_symbol:n { #2 }
5716   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5717   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
5718     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

```

5719     \tl_set:Nn \l_tmpa_tl { #3 }
5720   } {
5721     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
5722     \tl_set:Nn \l_tmpa_tl {
5723       \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
5724     }
5725   }
5726 } {
5727   \tl_set:Nn \l_tmpa_tl { #3 }
5728 }
5729
5730 % TODO root
5731 \stex_html_backend:TF {
5732   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
5733 } {
5734   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
5735 }
5736 }
5737 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

(End definition for definiendum. This function is documented on page 48.)

definame

```

5738
5739 \NewDocumentCommand \definame { 0{ } m } {
5740   \__stex_statements_definiendum_args:n { #1 }
5741   % TODO: root
5742   \stex_get_symbol:n { #2 }
5743   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5744   \str_set:Nx \l_tmpa_str {
5745     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5746   }
5747   \str_replace_all:Nnn \l_tmpa_str {-} {~}
5748   \stex_html_backend:TF {
5749     \stex_if_do_html:T {
5750       \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5751         \l_tmpa_str\l__stex_statements_definiendum_post_tl
5752       }
5753     }
5754   } {
5755     \exp_args:Nnx \defemph@uri {
5756       \l_tmpa_str\l__stex_statements_definiendum_post_tl
5757     } { \l_stex_get_symbol_uri_str }
5758   }
5759 }
5760 \stex_deactivate_macro:Nn \definame {definition~environments}
5761
5762 \NewDocumentCommand \Definame { 0{ } m } {
5763   \__stex_statements_definiendum_args:n { #1 }
5764   \stex_get_symbol:n { #2 }
5765   \str_set:Nx \l_tmpa_str {
5766     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5767   }
5768   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}

```

```

5769 \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5770 \stex_html_backend:TF {
5771   \stex_if_do_html:T {
5772     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5773       \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5774     }
5775   }
5776 } {
5777   \exp_args:Nnx \defemph@uri {
5778     \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5779   } { \l_stex_get_symbol_uri_str }
5780 }
5781 }
5782 \stex_deactivate_macro:Nn \Definame {definition-environments}
5783
5784 \NewDocumentCommand \premise { m }{
5785   \noindent\stex_annotate:nnn{ premise }{}{\ignorespaces #1 }
5786 }
5787 \NewDocumentCommand \conclusion { m }{
5788   \noindent\stex_annotate:nnn{ conclusion }{}{\ignorespaces #1 }
5789 }
5790 \NewDocumentCommand \definiens { 0{} m }{
5791   \str_clear:N \l_stex_get_symbol_uri_str
5792   \tl_if_empty:nF {#1} {
5793     \stex_get_symbol:n { #1 }
5794   }
5795   \str_if_empty:NT \l_stex_get_symbol_uri_str {
5796     \int_compare:nNnTF {\clist_count:N \l__stex_statements_sdefinition_for_clist} = 1 {
5797       \str_set:Nx \l_stex_get_symbol_uri_str {\clist_item:Nn \l__stex_statements_sdefinition
5798     }{
5799       % TODO throw error
5800     }
5801   }
5802   \str_if_eq:eeT {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{module}}
5803   {\l_stex_current_module_str}{
5804     \str_if_eq:eeF {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defin
5805   }{true}{
5806     \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5807     \exp_args:Nx \stex_add_to_current_module:n {
5808       \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5809     }
5810   }
5811 }
5812 \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
5813 }
5814
5815 \NewDocumentCommand \varbindforall {m}{
5816   \stex_symbol_or_var:n {#1}
5817   \bool_if:NTF\l_stex_symbol_or_var_bool{
5818     \stex_if_do_html:T {
5819       \stex_annotate_invisible:nnn {bindtype}{forall,\l_stex_get_symbol_uri_str}{}
5820     }
5821   }{
5822     % todo throw error

```

```

5823 }
5824 }
5825
5826 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
5827 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
5828 \stex_deactivate_macro:Nn \definiens {definition~environments}
5829 \stex_deactivate_macro:Nn \varbindforall {definition~or~assertion~environments}
5830

```

(End definition for definame. This function is documented on page 48.)

sdefinition (env.)

```

5831
5832 \keys_define:nn {stex / sdefinition }{
5833   type      .str_set_x:N = \sdefinitiontype,
5834   id        .str_set_x:N = \sdefinitionid,
5835   name      .str_set_x:N = \sdefinitionname,
5836   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5837   title     .tl_set:N     = \sdefinitiontitle
5838 }
5839 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
5840   \str_clear:N \sdefinitiontype
5841   \str_clear:N \sdefinitionid
5842   \str_clear:N \sdefinitionname
5843   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5844   \tl_clear:N \sdefinitiontitle
5845   \keys_set:nn { stex / sdefinition }{ #1 }
5846 }
5847
5848 \NewDocumentEnvironment{sdefinition}{0{}}{
5849   \__stex_statements_sdefinition_args:n{ #1 }
5850   \stex_reactivate_macro:N \definiendum
5851   \stex_reactivate_macro:N \definame
5852   \stex_reactivate_macro:N \Definame
5853   \stex_reactivate_macro:N \premise
5854   \stex_reactivate_macro:N \definiens
5855   \stex_reactivate_macro:N \varbindforall
5856   \stex_if_smsmode:F{
5857     \seq_clear:N \l_tmpb_seq
5858     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5859       \tl_if_empty:nF{ ##1 }{
5860         \stex_get_symbol:n { ##1 }
5861         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5862           \l_stex_get_symbol_uri_str
5863         }
5864       }
5865     }
5866     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5867     \exp_args:Nnnx
5868     \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpb_seq {,}}
5869     \str_if_empty:NF \sdefinitiontype {
5870       \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{ }
5871     }
5872     \str_if_empty:NF \sdefinitionname {

```



```

5873     \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5874   }
5875   \clist_set:No \l_tmpa_clist \sdefinitiontype
5876   \tl_clear:N \l_tmpa_tl
5877   \clist_map_inline:Nn \l_tmpa_clist {
5878     \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
5879       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
5880     }
5881   }
5882   \tl_if_empty:NTF \l_tmpa_tl {
5883     \__stex_statements_sdefinition_start:
5884   }{
5885     \l_tmpa_tl
5886   }
5887 }
5888 \stex_ref_new_doc_target:n \sdefinitionid
5889 \stex_smsmode_do:
5890 }{
5891   \stex_suppress_html:n {
5892     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5893   }
5894   \stex_if_smsmode:F {
5895     \clist_set:No \l_tmpa_clist \sdefinitiontype
5896     \tl_clear:N \l_tmpa_tl
5897     \clist_map_inline:Nn \l_tmpa_clist {
5898       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
5899         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
5900       }
5901     }
5902     \tl_if_empty:NTF \l_tmpa_tl {
5903       \__stex_statements_sdefinition_end:
5904     }{
5905       \l_tmpa_tl
5906     }
5907     \end{stex_annotate_env}
5908   }
5909 }

```

\stexpatchdefinition

```

5910 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
5911   \stex_par:\noindent\titllemph{Definition\tl_if_empty:NF \sdefinitiontitle {
5912     ~(\sdefinitiontitle)
5913   }~}
5914 }
5915 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\stex_par:\medskip}
5916
5917 \newcommand\stexpatchdefinition[3] [] {
5918   \str_set:Nx \l_tmpa_str{ #1 }
5919   \str_if_empty:NTF \l_tmpa_str {
5920     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
5921     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
5922   }{
5923     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
5924     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }

```

```

5925     }
5926 }

```

(End definition for `\stexpatchdefinition`. This function is documented on page 55.)

`\inlinedef` inline:

```

5927 \keys_define:nn {stex / inlinedef }{
5928   type      .str_set_x:N = \sdefinitiontype,
5929   id        .str_set_x:N = \sdefinitionid,
5930   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5931   name      .str_set_x:N = \sdefinitionname
5932 }
5933 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
5934   \str_clear:N \sdefinitiontype
5935   \str_clear:N \sdefinitionid
5936   \str_clear:N \sdefinitionname
5937   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5938   \keys_set:nn { stex / inlinedef }{ #1 }
5939 }
5940 \NewDocumentCommand \inlinedef { 0{} m } {
5941   \begingroup
5942   \__stex_statements_inlinedef_args:n{ #1 }
5943   \stex_reactivate_macro:N \definiendum
5944   \stex_reactivate_macro:N \definame
5945   \stex_reactivate_macro:N \Definame
5946   \stex_reactivate_macro:N \premise
5947   \stex_reactivate_macro:N \definiens
5948   \stex_reactivate_macro:N \varbindforall
5949   \stex_ref_new_doc_target:n \sdefinitionid
5950   \stex_if_smsmode:TF{\stex_suppress_html:n {
5951     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5952   }}{
5953     \seq_clear:N \l_tmpb_seq
5954     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5955       \tl_if_empty:nF{ ##1 }{
5956         \stex_get_symbol:n { ##1 }
5957         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5958           \l_stex_get_symbol_uri_str
5959         }
5960       }
5961     }
5962     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5963     \ifvmode\noindent\fi
5964     \exp_args:Nnx
5965     \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpb_seq {,}}{
5966       \str_if_empty:NF \sdefinitiontype {
5967         \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5968       }
5969       #2
5970       \str_if_empty:NF \sdefinitionname {
5971         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sdefinitionname}}
5972         \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5973       }
5974     }

```

```

5975 }
5976 \endgroup
5977 \stex_smsmode_do:
5978 }

```

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

34.2 Assertions

`sassertion (env.)`

```

5979
5980 \keys_define:nn {stex / sassertion }{
5981   type      .str_set_x:N = \sassertiontype,
5982   id        .str_set_x:N = \sassertionid,
5983   title     .tl_set:N    = \sassertiontitle ,
5984   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
5985   name      .str_set_x:N = \sassertionname
5986 }
5987 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
5988   \str_clear:N \sassertiontype
5989   \str_clear:N \sassertionid
5990   \str_clear:N \sassertionname
5991   \clist_clear:N \l__stex_statements_sassertion_for_clist
5992   \tl_clear:N \sassertiontitle
5993   \keys_set:nn { stex / sassertion }{ #1 }
5994 }
5995
5996 %\tl_new:N \g__stex_statements_aftergroup_tl
5997
5998 \NewDocumentEnvironment{sassertion}{0{}}{
5999   \__stex_statements_sassertion_args:n{ #1 }
6000   \stex_reactivate_macro:N \premise
6001   \stex_reactivate_macro:N \conclusion
6002   \stex_reactivate_macro:N \varbindforall
6003   \stex_if_smsmode:F {
6004     \seq_clear:N \l_tmpb_seq
6005     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
6006       \tl_if_empty:nF{ ##1 }{
6007         \stex_get_symbol:n { ##1 }
6008         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6009           \l_stex_get_symbol_uri_str
6010         }
6011       }
6012     }
6013     \exp_args:Nnnx
6014     \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
6015     \str_if_empty:NF \sassertiontype {
6016       \stex_annotate_invisible:nnn{type}{\sassertiontype}{\sassertiontype}{}
6017     }
6018     \str_if_empty:NF \sassertionname {
6019       \stex_annotate_invisible:nnn{statementname}{\sassertionname}{\sassertionname}{}
6020     }
6021     \clist_set:Nn \l_tmpa_clist \sassertiontype

```

```

6022 \tl_clear:N \l_tmpa_tl
6023 \clist_map_inline:Nn \l_tmpa_clist {
6024   \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
6025     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
6026   }
6027 }
6028 \tl_if_empty:NTF \l_tmpa_tl {
6029   \__stex_statements_sassertion_start:
6030 }{
6031   \l_tmpa_tl
6032 }
6033 }
6034 \str_if_empty:NTF \sassertionid {
6035   \str_if_empty:NF \sassertionname {
6036     \stex_ref_new_doc_target:n {}
6037   }
6038 } {
6039   \stex_ref_new_doc_target:n \sassertionid
6040 }
6041 \stex_smsmode_do:
6042 ){
6043   \str_if_empty:NF \sassertionname {
6044     \stex_suppress_html:n{\stex_symdecl_do:nn}{\sassertionname}}
6045     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
6046   }
6047   \stex_if_smsmode:F {
6048     \clist_set:Nn \l_tmpa_clist \sassertiontype
6049     \tl_clear:N \l_tmpa_tl
6050     \clist_map_inline:Nn \l_tmpa_clist {
6051       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
6052         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
6053       }
6054     }
6055     \tl_if_empty:NTF \l_tmpa_tl {
6056       \__stex_statements_sassertion_end:
6057     }{
6058       \l_tmpa_tl
6059     }
6060     \end{stex_annotate_env}
6061   }
6062 }

```

\stexpatchassertion

```

6063
6064 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
6065   \stex_par:\noindent\titllemph{Assertion~\tl_if_empty:NF \sassertiontitle {
6066     (\sassertiontitle)
6067   }~}
6068 }
6069 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\stex_par:\medskip}
6070
6071 \newcommand\stexpatchassertion[3] [] {
6072   \str_set:Nx \l_tmpa_str{ #1 }
6073   \str_if_empty:NTF \l_tmpa_str {

```

```

6074     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
6075     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
6076   }{
6077     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
6078     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
6079   }
6080 }

```

(End definition for `\stexpatchassertion`. This function is documented on page 55.)

`\inlineass` inline:

```

6081 \keys_define:nn {stex / inlineass }{
6082   type      .str_set_x:N = \sassertiontype,
6083   id        .str_set_x:N = \sassertionid,
6084   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
6085   name      .str_set_x:N = \sassertionname
6086 }
6087 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
6088   \str_clear:N \sassertiontype
6089   \str_clear:N \sassertionid
6090   \str_clear:N \sassertionname
6091   \clist_clear:N \l__stex_statements_sassertion_for_clist
6092   \keys_set:nn { stex / inlineass }{ #1 }
6093 }
6094 \NewDocumentCommand \inlineass { 0{} m } {
6095   \begingroup
6096   \stex_reactivate_macro:N \premise
6097   \stex_reactivate_macro:N \conclusion
6098   \stex_reactivate_macro:N \varbindforall
6099   \__stex_statements_inlineass_args:n{ #1 }
6100   \str_if_empty:NTF \sassertionid {
6101     \str_if_empty:NF \sassertionname {
6102       \stex_ref_new_doc_target:n {}
6103     }
6104   } {
6105     \stex_ref_new_doc_target:n \sassertionid
6106   }
6107
6108   \stex_if_smsmode:TF{
6109     \str_if_empty:NF \sassertionname {
6110       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sassertionname}}
6111     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
6112   }
6113 }{
6114   \seq_clear:N \l_tmpb_seq
6115   \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
6116     \tl_if_empty:nF{ ##1 }{
6117       \stex_get_symbol:n { ##1 }
6118       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6119         \l_stex_get_symbol_uri_str
6120       }
6121     }
6122   }
6123   \ifvmode\noindent\fi

```

```

6124 \exp_args:Nnx
6125 \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpb_seq {,}}{
6126   \str_if_empty:NF \sassertiontype {
6127     \stex_annotate_invisible:nnn{typestrings}{\sassertiontype}{}
6128   }
6129   #2
6130   \str_if_empty:NF \sassertionname {
6131     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
6132     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
6133     \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
6134   }
6135 }
6136 }
6137 \endgroup
6138 \stex_smsmode_do:
6139 }

```

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

34.3 Examples

`sexample (env.)`

```

6140
6141 \keys_define:nn {stex / sexample }{
6142   type      .str_set_x:N = \exampletype,
6143   id        .str_set_x:N = \sexampleid,
6144   title     .tl_set:N     = \sexampletitle,
6145   name      .str_set_x:N = \sexamplename ,
6146   for       .clist_set:N  = \l__stex_statements_sexample_for_clist,
6147 }
6148 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
6149   \str_clear:N \sexampletype
6150   \str_clear:N \sexampleid
6151   \str_clear:N \sexamplename
6152   \tl_clear:N \sexampletitle
6153   \clist_clear:N \l__stex_statements_sexample_for_clist
6154   \keys_set:nn { stex / sexample }{ #1 }
6155 }
6156
6157 \NewDocumentEnvironment{sexample}{0{}}{
6158   \__stex_statements_sexample_args:n{ #1 }
6159   \stex_reactivate_macro:N \premise
6160   \stex_reactivate_macro:N \conclusion
6161   \stex_if_smsmode:F {
6162     \seq_clear:N \l_tmpb_seq
6163     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
6164       \tl_if_empty:nF{ ##1 }{
6165         \stex_get_symbol:n { ##1 }
6166         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6167           \l_stex_get_symbol_uri_str
6168         }
6169       }
6170     }

```

```

6171 \exp_args:Nnnx
6172 \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpb_seq {,}}
6173 \str_if_empty:NF \sexamplotype {
6174   \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{}}
6175 }
6176 \str_if_empty:NF \sexamplename {
6177   \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}}
6178 }
6179 \clist_set:No \l_tmpa_clist \sexamplotype
6180 \tl_clear:N \l_tmpa_tl
6181 \clist_map_inline:Nn \l_tmpa_clist {
6182   \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
6183     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
6184   }
6185 }
6186 \tl_if_empty:NTF \l_tmpa_tl {
6187   \__stex_statements_sexample_start:
6188 }{
6189   \l_tmpa_tl
6190 }
6191 }
6192 \str_if_empty:NF \sexampleid {
6193   \stex_ref_new_doc_target:n \sexampleid
6194 }
6195 \stex_smsmode_do:
6196 }{
6197   \str_if_empty:NF \sexamplename {
6198     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
6199   }
6200   \stex_if_smsmode:F {
6201     \clist_set:No \l_tmpa_clist \sexamplotype
6202     \tl_clear:N \l_tmpa_tl
6203     \clist_map_inline:Nn \l_tmpa_clist {
6204       \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
6205         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
6206       }
6207     }
6208     \tl_if_empty:NTF \l_tmpa_tl {
6209       \__stex_statements_sexample_end:
6210     }{
6211       \l_tmpa_tl
6212     }
6213     \end{stex_annotate_env}
6214   }
6215 }

```

\stexpatchexample

```

6216 \cs_new_protected:Nn \__stex_statements_sexample_start: {
6217   \stex_par:\noindent\titllemph{Example~\tl_if_empty:NF \sexampltitle {
6218     (\sexampltitle)
6219   }}~}
6220 }
6221 }
6222 \cs_new_protected:Nn \__stex_statements_sexample_end: {\stex_par:\medskip}

```

```

6223
6224 \newcommand\stexpatchexample[3][] {
6225   \str_set:Nx \l_tmpa_str{ #1 }
6226   \str_if_empty:NTF \l_tmpa_str {
6227     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
6228     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
6229   }{
6230     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
6231     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
6232   }
6233 }

```

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

\inlineex inline:

```

6234 \keys_define:nn {stex / inlineex }{
6235   type      .str_set_x:N = \sexamplotype,
6236   id        .str_set_x:N = \sexampleid,
6237   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
6238   name      .str_set_x:N = \sexamplename
6239 }
6240 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
6241   \str_clear:N \sexamplotype
6242   \str_clear:N \sexampleid
6243   \str_clear:N \sexamplename
6244   \clist_clear:N \l__stex_statements_sexample_for_clist
6245   \keys_set:nn { stex / inlineex }{ #1 }
6246 }
6247 \NewDocumentCommand \inlineex { 0{} m } {
6248   \begingroup
6249   \stex_reactivate_macro:N \premise
6250   \stex_reactivate_macro:N \conclusion
6251   \__stex_statements_inlineex_args:n{ #1 }
6252   \str_if_empty:NF \sexampleid {
6253     \stex_ref_new_doc_target:n \sexampleid
6254   }
6255   \stex_if_smsmode:TF{
6256     \str_if_empty:NF \sexamplename {
6257       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
6258     }
6259   }{
6260     \seq_clear:N \l_tmpb_seq
6261     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
6262       \tl_if_empty:NF{ ##1 }{
6263         \stex_get_symbol:n { ##1 }
6264         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6265           \l_stex_get_symbol_uri_str
6266         }
6267       }
6268     }
6269     \ifvmode\noindent\fi
6270     \exp_args:Nnx
6271     \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpb_seq {},}}{
6272     \str_if_empty:NF \sexamplotype {

```



```

6273 \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{}
6274 }
6275 #2
6276 \str_if_empty:NF \sexamplename {
6277   \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sexamplename}}
6278   \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}
6279 }
6280 }
6281 }
6282 \endgroup
6283 \stex_smsmode_do:
6284 }

```

34.4 Logical Paragraphs

```

6285 \keys_define:nn { stex / spargraph } {
6286   id      .str_set:x:N = \sparagraphid ,
6287   title   .tl_set:N     = \l_stex_sparagraph_title_tl ,
6288   type    .str_set:x:N = \sparagraphtype ,
6289   for     .clist_set:N  = \l__stex_statements_sparagraph_for_clist ,
6290   from    .tl_set:N     = \sparagraphfrom ,
6291   to      .tl_set:N     = \sparagraphto ,
6292   start   .tl_set:N     = \l_stex_sparagraph_start_tl ,
6293   name    .str_set:N     = \sparagraphname ,
6294   imports .tl_set:N     = \l__stex_statements_sparagraph_imports_tl
6295 }
6296
6297 \cs_new_protected:Nn \stex_sparagraph_args:n {
6298   \tl_clear:N \l_stex_sparagraph_title_tl
6299   \tl_clear:N \sparagraphfrom
6300   \tl_clear:N \sparagraphto
6301   \tl_clear:N \l_stex_sparagraph_start_tl
6302   \tl_clear:N \l__stex_statements_sparagraph_imports_tl
6303   \str_clear:N \sparagraphid
6304   \str_clear:N \sparagraphtype
6305   \clist_clear:N \l__stex_statements_sparagraph_for_clist
6306   \str_clear:N \sparagraphname
6307   \keys_set:nn { stex / spargraph } { #1 }
6308 }
6309 \newif\if@in@omtext\@in@omtextfalse
6310
6311 \NewDocumentEnvironment {sparagraph} { 0{} } {
6312   \stex_sparagraph_args:n { #1 }
6313   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
6314     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
6315   }{
6316     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
6317   }
6318   \@in@omtexttrue
6319   \stex_if_smsmode:F {

```

```

6320 \seq_clear:N \l_tmpb_seq
6321 \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
6322   \tl_if_empty:nF{ ##1 }{
6323     \stex_get_symbol:n { ##1 }
6324     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6325       \l_stex_get_symbol_uri_str
6326     }
6327   }
6328 }
6329 \exp_args:Nnnx
6330 \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}
6331 \str_if_empty:NF \sparagraphtype {
6332   \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
6333 }
6334 \str_if_empty:NF \sparagraphfrom {
6335   \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
6336 }
6337 \str_if_empty:NF \sparagraphto {
6338   \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
6339 }
6340 \str_if_empty:NF \sparagraphname {
6341   \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
6342 }
6343 \clist_set:No \l_tmpa_clist \sparagraphtype
6344 \tl_clear:N \l_tmpa_tl
6345 \clist_map_inline:Nn \sparagraphtype {
6346   \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
6347     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
6348   }
6349 }
6350 \stex_csl_to_imports:No \usemodule \l__stex_statements_sparagraph_imports_tl
6351 \tl_if_empty:NTF \l_tmpa_tl {
6352   \__stex_statements_sparagraph_start:
6353 }{
6354   \l_tmpa_tl
6355 }
6356 }
6357 \clist_set:No \l_tmpa_clist \sparagraphtype
6358 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
6359 {
6360   \stex_reactivate_macro:N \definiendum
6361   \stex_reactivate_macro:N \definame
6362   \stex_reactivate_macro:N \Definame
6363   \stex_reactivate_macro:N \premise
6364   \stex_reactivate_macro:N \definiens
6365 }
6366 \str_if_empty:NTF \sparagraphid {
6367   \str_if_empty:NTF \sparagraphname {
6368     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
6369       \stex_ref_new_doc_target:n {}
6370     }
6371   } {
6372     \stex_ref_new_doc_target:n {}
6373   }

```

```

6374 } {
6375   \stex_ref_new_doc_target:n \sparagraphid
6376 }
6377 \exp_args:NNx
6378 \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
6379   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
6380     \tl_if_empty:nF{ ##1 }{
6381       \stex_get_symbol:n { ##1 }
6382       \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
6383     }
6384   }
6385 }
6386 \stex_smsmode_do:
6387 \ignorespacesandpars
6388 }{
6389   \str_if_empty:NF \sparagraphname {
6390     \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
6391     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
6392   }
6393   \stex_if_smsmode:F {
6394     \clist_set:No \l_tmpa_clist \sparagraphtype
6395     \tl_clear:N \l_tmpa_tl
6396     \clist_map_inline:Nn \l_tmpa_clist {
6397       \tl_if_exist:cT {\__stex_statements_sparagraph_##1_end:}{
6398         \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_statements_sparagraph_##1_end:}}
6399       }
6400     }
6401     \tl_if_empty:NTF \l_tmpa_tl {
6402       \__stex_statements_sparagraph_end:
6403     }{
6404       \l_tmpa_tl
6405     }
6406     \end{stex_annotate_env}
6407   }
6408 }

```

\stexpatchparagraph

```

6409
6410 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
6411   \stex_par:\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
6412     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
6413       \titleemph{\l_stex_sparagraph_title_tl}:~
6414     }
6415   }{
6416     \titleemph{\l_stex_sparagraph_start_tl}~
6417   }
6418 }
6419 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\stex_par:\medskip}
6420
6421 \newcommand\stexpatchparagraph[3]{} {
6422   \str_set:Nx \l_tmpa_str{ #1 }
6423   \str_if_empty:NTF \l_tmpa_str {
6424     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
6425     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }

```

```

6426     }{
6427         \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
6428         \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
6429     }
6430 }
6431
6432 \keys_define:nn { stex / inlinepara} {
6433     id      .str_set_x:N = \sparagraphid ,
6434     type    .str_set_x:N = \sparagraphtype ,
6435     for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
6436     from    .tl_set:N    = \sparagraphfrom ,
6437     to      .tl_set:N    = \sparagraphto ,
6438     name    .str_set:N   = \sparagraphname
6439 }
6440 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
6441     \tl_clear:N \sparagraphfrom
6442     \tl_clear:N \sparagraphto
6443     \str_clear:N \sparagraphid
6444     \str_clear:N \sparagraphtype
6445     \clist_clear:N \l__stex_statements_sparagraph_for_clist
6446     \str_clear:N \sparagraphname
6447     \keys_set:nn { stex / inlinepara }{ #1 }
6448 }
6449 \NewDocumentCommand \inlinepara { 0{} m } {
6450     \begingroup
6451     \__stex_statements_inlinepara_args:n{ #1 }
6452     \clist_set:Nn \l_tmpa_clist \sparagraphtype
6453     \str_if_empty:NTF \sparagraphid {
6454         \str_if_empty:NTF \sparagraphname {
6455             \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
6456                 \stex_ref_new_doc_target:n {}
6457             }
6458         } {
6459             \stex_ref_new_doc_target:n {}
6460         }
6461     } {
6462         \stex_ref_new_doc_target:n \sparagraphid
6463     }
6464     \stex_if_smsmode:TF{
6465         \str_if_empty:NF \sparagraphname {
6466             \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
6467             \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
6468         }
6469     }{
6470         \seq_clear:N \l_tmpb_seq
6471         \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
6472             \tl_if_empty:nF{ ##1 }{
6473                 \stex_get_symbol:n { ##1 }
6474                 \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6475                     \l_stex_get_symbol_uri_str
6476                 }
6477             }
6478         }
6479         \ifvmode\noindent\fi

```

```

6480 \exp_args:Nnx
6481 \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}{
6482   \str_if_empty:NF \sparagraphtype {
6483     \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
6484   }
6485   \str_if_empty:NF \sparagraphfrom {
6486     \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
6487   }
6488   \str_if_empty:NF \sparagraphto {
6489     \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
6490   }
6491   \str_if_empty:NF \sparagraphname {
6492     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
6493     \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
6494     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
6495   }
6496   \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
6497     \clist_map_inline:Nn \l_tmpb_seq {
6498       \stex_ref_new_sym_target:n {##1}
6499     }
6500   }
6501   #2
6502 }
6503 }
6504 \endgroup
6505 \stex_smsmode_do:
6506 }
6507

```

(End definition for `\stexpatchparagraph`. This function is documented on page 55.)

```

6508 </package>

```

Chapter 35

The Implementation

```
6509 <*package>
6510 <@@=stex_sproof>
6511
6512 %%%%%%%%%%    sproof.dtx    %%%%%%%%%%
6513
```

35.1 Proofs

We first define some keys for the proof environment.

```
6514 \keys_define:nn { stex / spf } {
6515   id          .str_set_x:N = \spfid,
6516   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,
6517   from        .tl_set:N    = \l__stex_sproof_spf_from_tl ,
6518   proofend    .tl_set:N    = \l__stex_sproof_spf_proofend_tl,
6519   type        .str_set_x:N = \spftype,
6520   title       .tl_set:N    = \spftitle,
6521   continues   .tl_set:N    = \l__stex_sproof_spf_continues_tl,
6522   functions   .tl_set:N    = \l__stex_sproof_spf_functions_tl,
6523   term        .tl_set:N    = \l__stex_sproof_spf_term_tl,
6524   method      .tl_set:N    = \l__stex_sproof_spf_method_tl,
6525   hide        .bool_set:N  = \l__stex_sproof_spf_hide_bool
6526 }
6527 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
6528   \str_clear:N \spfid
6529   \tl_clear:N \l__stex_sproof_spf_for_tl
6530   \tl_clear:N \l__stex_sproof_spf_from_tl
6531   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
6532   \str_clear:N \spftype
6533   \tl_clear:N \spftitle
6534   \tl_clear:N \l__stex_sproof_spf_continues_tl
6535   \tl_clear:N \l__stex_sproof_spf_term_tl
6536   \tl_clear:N \l__stex_sproof_spf_functions_tl
6537   \tl_clear:N \l__stex_sproof_spf_method_tl
6538   \bool_set_false:N \l__stex_sproof_spf_hide_bool
6539   \keys_set:nn { stex / spf }{ #1 }
6540 }
6541 \bool_set_true:N \l__stex_sproof_inc_counter_bool
```

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

```
6542 \str_set:Nn\c__stex_sproof_flow_str{inline}
```

(End definition for `\c__stex_sproof_flow_str`.)

For proofs, we will have to have deeply nested structures of enumerated list-like environments. However, L^AT_EX 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.

```
6543 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
6544 \cs_new_protected:Npn \sproofnumber {
6545   \int_set:Nn \l_tmpa_int {1}
6546   \bool_while_do:nn {
6547     \int_compare_p:nNn {
6548       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6549     } > 0
6550   }{
6551     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
6552     \int_incr:N \l_tmpa_int
6553   }
6554 }
6555 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
6556   \int_set:Nn \l_tmpa_int {1}
6557   \bool_while_do:nn {
6558     \int_compare_p:nNn {
6559       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6560     } > 0
6561   }{
6562     \int_incr:N \l_tmpa_int
6563   }
6564   \int_compare:nNnF \l_tmpa_int = 1 {
6565     \int_decr:N \l_tmpa_int
6566   }
6567   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
6568     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1
6569   }
6570 }
6571
6572 \cs_new_protected:Npn \__stex_sproof_add_counter: {
6573   \int_set:Nn \l_tmpa_int {1}
6574   \bool_while_do:nn {
6575     \int_compare_p:nNn {
6576       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6577     } > 0
6578   }{
6579     \int_incr:N \l_tmpa_int
6580   }
6581   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 1 }
6582 }
6583
```

```

6584 \cs_new_protected:Npn \__stex_sproof_remove_counter: {
6585   \int_set:Nn \l_tmpa_int {1}
6586   \bool_while_do:nn {
6587     \int_compare_p:nNn {
6588       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6589     } > 0
6590   }{
6591     \int_incr:N \l_tmpa_int
6592   }
6593   \int_decr:N \l_tmpa_int
6594   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 0 }
6595 }

```

\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

```

6596 \def\sproof@box{
6597   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
6598 }
6599 \def\sproofend{
6600   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
6601     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
6602   }
6603 }

```

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

spf@*kw

```

6604 \def\spf@proofsketch@kw{Proof~Sketch}
6605 \def\spf@proof@kw{Proof}
6606 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

6607 \AddToHook{begindocument}{
6608   \ltx@ifpackageloaded{babel}{
6609     \makeatletter
6610     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6611     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6612       \input{sproof-ngerman.ldf}
6613     }
6614     \clist_if_in:NnT \l_tmpa_clist {finnish}{
6615       \input{sproof-finnish.ldf}
6616     }
6617     \clist_if_in:NnT \l_tmpa_clist {french}{
6618       \input{sproof-french.ldf}
6619     }
6620     \clist_if_in:NnT \l_tmpa_clist {russian}{
6621       \input{sproof-russian.ldf}
6622     }
6623     \makeatother
6624   }{}
6625 }

```


spfsketch

```

6626 \newcommand\spfsketch[2] [] {
6627   \begin{group}
6628   \let \premise \stex_proof_premise:
6629   \_stex_sproof_spf_args:n{#1}
6630   \stex_if_smsmode:TF {
6631     \str_if_empty:NF \spfid {
6632       \stex_ref_new_doc_target:n \spfid
6633     }
6634   }{
6635     \seq_clear:N \l_tmpa_seq
6636     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6637       \tl_if_empty:nF{ ##1 }{
6638         \stex_get_symbol:n { ##1 }
6639         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6640           \l_stex_get_symbol_uri_str
6641         }
6642       }
6643     }
6644     \exp_args:Nnx
6645     \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
6646       \str_if_empty:NF \spftype {
6647         \stex_annotate_invisible:nnn{type}{\spftype}{ }
6648       }
6649       \clist_set:Nn \l_tmpa_clist \spftype
6650       \tl_set:Nn \l_tmpa_tl {
6651         \titleemph{
6652           \tl_if_empty:NTF \spftitle {
6653             \spf@proofsketch@kw
6654           }{
6655             \spftitle
6656           }
6657         }:-
6658       }
6659       \clist_map_inline:Nn \l_tmpa_clist {
6660         \exp_args:Nn \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6661           \tl_clear:N \l_tmpa_tl
6662         }
6663       }
6664       \str_if_empty:NF \spfid {
6665         \stex_ref_new_doc_target:n \spfid
6666       }
6667       \l_tmpa_tl #2 \sproofend
6668     }
6669   }
6670   \endgroup
6671   \stex_smsmode_do:
6672 }
6673

```

(End definition for *spfsketch*. This function is documented on page 54.)

```

\_stex_sproof_maybe_comment:
\_stex_sproof_maybe_comment_end:
\_stex_sproof_start_comment:
6674 \bool_set_false:N \l__stex_sproof_in_spfblock_bool

```

```

6675
6676 \cs_new_protected:Nn \__stex_sproof_maybe_comment: {
6677   \bool_if:NF \l__stex_sproof_in_spfblock_bool {
6678     \par \setbox \l_tmpa_box \vbox \bgroup \everypar{\__stex_sproof_start_comment:}
6679   }
6680 }
6681 \cs_new_protected:Nn \__stex_sproof_maybe_comment_end: {
6682   \bool_if:NF \l__stex_sproof_in_spfblock_bool { \egroup }
6683 }
6684 \cs_new_protected:Nn \__stex_sproof_start_comment: {
6685   \csname @ @ par\endcsname\egroup\item[]\bgroup\stexcommentfont
6686 }
6687

```

(End definition for __stex_sproof_maybe_comment:, __stex_sproof_maybe_comment_end:, and __stex_sproof_start_comment:.)

\stexcommentfont

```

6688 \cs_new_protected:Npn \stexcommentfont {
6689   \small\itshape
6690 }

```

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

sproof (env.) In this environment, we initialize the proof depth counter \count10 to 10, and set up the description environment that will take the proof steps. At the end of the proof, we position the proof end into the last line.

```

6691 \cs_new_protected:Nn \__stex_sproof_start_env:nnn {
6692   \seq_clear:N \l_tmpa_seq
6693   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6694     \tl_if_empty:NF{ ##1 }{
6695       \stex_get_symbol:n { ##1 }
6696       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6697         \l_stex_get_symbol_uri_str
6698       }
6699     }
6700   }
6701   \exp_args:Nnnx
6702   \begin{stex_annotate_env}{#1}{\seq_use:Nn \l_tmpa_seq {,}}
6703   \str_if_empty:NF \spftype {
6704     \stex_annotate_invisible:nnn{type}{\spftype}{}
6705   }
6706   #3 {\~\stex_annotate:nnn{spftitle}{}{#2}}
6707   \str_if_empty:NF \spfid {
6708     \stex_ref_new_doc_target:n \spfid
6709   }
6710   \begin{stex_annotate_env}{spfbody}{\bool_if:NTF \l__stex_sproof_spf_hide_bool {false}{true}
6711   \bool_if:NT \l__stex_sproof_spf_hide_bool{
6712     \stex_html_backend:F{\setbox\l_tmpa_box\vbox\bgroup}
6713   }
6714   \begin{list}{}{
6715     \setlength\topsep{0pt}
6716     \setlength\parsep{0pt}
6717     \setlength\rightmargin{0pt}

```

```

6718 } \_stex_sproof_maybe_comment:
6719 }
6720 \cs_new_protected:Nn \_stex_sproof_end_env:n {
6721   \stex_if_smsmode:F{
6722     \_stex_sproof_maybe_comment_end:
6723     \end{list}
6724     \bool_if:NT \l__stex_sproof_spf_hide_bool{
6725       \stex_html_backend:F{\egroup}
6726     }
6727     \clist_set:No \l_tmpa_clist \spftype
6728     #1
6729     \end{stex_annotate_env}
6730     \end{stex_annotate_env}
6731   }
6732 }
6733 }
6734 \NewDocumentEnvironment{sproof}{s O{} m}{
6735   \intarray_gzero:N \l__stex_sproof_counter_intarray
6736   \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
6737   \stex_reactivate_macro:N \yield
6738   \stex_reactivate_macro:N \eqstep
6739   \stex_reactivate_macro:N \assumption
6740   \stex_reactivate_macro:N \conclude
6741   \stex_reactivate_macro:N \spfstep
6742   \_stex_sproof_spf_args:n{#2}
6743   \stex_if_smsmode:TF {
6744     \str_if_empty:NF \spfid {
6745       \stex_ref_new_doc_target:n \spfid
6746     }
6747   }{
6748     \_stex_sproof_start_env:nnn{sproof}{#3}{
6749       \clist_set:No \l_tmpa_clist \spftype
6750       \tl_clear:N \l_tmpa_tl
6751       \clist_map_inline:Nn \l_tmpa_clist {
6752         \tl_if_exist:cT {\_stex_sproof_sproof_##1_start:}{
6753           \tl_set:Nn \l_tmpa_tl {\use:c{\_stex_sproof_sproof_##1_start:}}
6754         }
6755         \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6756           \tl_set:Nn \l_tmpa_tl {\use:n{}}
6757         }
6758       }
6759       \tl_if_empty:NTF \l_tmpa_tl {
6760         \_stex_sproof_sproof_start:
6761       }{
6762         \l_tmpa_tl
6763       }
6764     }
6765   }
6766   \stex_smsmode_do:
6767 }{\_stex_sproof_end_env:n{
6768   \tl_clear:N \l_tmpa_tl
6769   \clist_map_inline:Nn \l_tmpa_clist {
6770     \tl_if_exist:cT {\_stex_sproof_sproof_##1_end:}{
6771       \tl_set:Nn \l_tmpa_tl {\use:c{\_stex_sproof_sproof_##1_end:}}

```

```

6772     }
6773   }
6774   \tl_if_empty:NTF \l_tmpa_tl {
6775     \__stex_sproof_sproof_end:
6776   }{
6777     \l_tmpa_tl
6778   }
6779 }
6780 \NewDocumentEnvironment{subproof}{s O{} m}{
6781   \__stex_sproof_spf_args:n{#2}
6782   \stex_if_smsmode:TF {
6783     \str_if_empty:NF \spfid {
6784       \stex_ref_new_doc_target:n \spfid
6785     }
6786   }{
6787     \__stex_sproof_start_env:nnn{subproof}{\item[\sproofnumber]\ignorespacesandpars #3}{}
6788   }
6789   \__stex_sproof_add_counter:
6790   \stex_smsmode_do:
6791 }{\__stex_sproof_remove_counter:\__stex_sproof_end_env:n{
6792   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6793     \__stex_sproof_inc_counter:
6794   }
6795   \aftergroup\__stex_sproof_maybe_comment:
6796 }
6797 \AddToHook{env/subproof/before}{\__stex_sproof_maybe_comment_end:}
6798
6799 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
6800   \par\noindent\titleemph{
6801     \tl_if_empty:NTF \spftype {
6802       \spf@proof@kw
6803     }{
6804       \spftype
6805     }
6806   }:
6807 }
6808 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
6809
6810 \newcommand\stexpatchproof[3] [] {
6811   \str_set:Nx \l_tmpa_str{ #1 }
6812   \str_if_empty:NTF \l_tmpa_str {
6813     \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
6814     \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
6815   }{
6816     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
6817     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
6818   }
6819 }

\pstep
\conclude
\assumption
\have
\eqstep

```

```

6824 type .str_set_x:N = \spftype,
6825 title .tl_set:N = \spftitle,
6826 method .tl_set:N = \l__stex_sproof_spf_method_tl,
6827 term .tl_set:N = \l__stex_sproof_spf_term_tl
6828 }
6829 \cs_new_protected:Nn \__stex_sproof_spfstep_args:n {
6830 \str_clear:N \spfstepid
6831 \clist_clear:N \l__stex_sproof_spf_for_clist
6832 \str_clear:N \spftype
6833 \tl_clear:N \l__stex_sproof_spf_method_tl
6834 \tl_clear:N \l__stex_sproof_spf_term_tl
6835 %\bool_set_false:N \l__stex_sproof_inc_counter_bool
6836 \keys_set:nn { stex / spfsteps }{ #1 }
6837 }
6838
6839 \cs_new_protected:Nn \__stex_sproof_make_step_macro:Nnnnn {
6840 \NewDocumentCommand #1 {s O{} +m} {
6841 \__stex_sproof_maybe_comment_end:
6842
6843 \__stex_sproof_spfstep_args:n{##2}
6844 \stex_annotate:nnn{spfstep}{#2}{
6845 \tl_if_empty:NF \l__stex_sproof_spf_term_tl {
6846 \stex_annotate_invisible:nnn{spfyield}{}{\l__stex_sproof_spf_term_tl$}
6847 }
6848 \bool_if:NTF \l__stex_sproof_in_spfblock_bool {
6849 #4
6850 }{
6851 \item[\IfBooleanTF ##1 {}{#3}]
6852 }
6853 \ignorespacesandpars ##3
6854 }
6855 \bool_if:NF \l__stex_sproof_in_spfblock_bool { \IfBooleanTF ##1 {}{ #5 } }
6856 \__stex_sproof_maybe_comment:
6857 }
6858 \stex_deactivate_macro:Nn #1 {sproof~environments}
6859 }
6860
6861 \__stex_sproof_make_step_macro:Nnnnn \assumption {assumption} \sproofnumber {} \__stex_sproof
6862 \__stex_sproof_make_step_macro:Nnnnn \conclude {conclusion} {\$ \Rightarrow$} {} {}
6863 \__stex_sproof_make_step_macro:Nnnnn \spfstep {} \sproofnumber {} \__stex_sproof_inc_counter
6864
6865 \NewDocumentCommand \eqstep {s m}{
6866 \__stex_sproof_maybe_comment_end:
6867 \bool_if:NTF \l__stex_sproof_in_spfblock_bool {
6868 $=$
6869 }{
6870 \item[$=$]
6871 }
6872 $\stex_annotate:nnn{spfstep}{eq}{ #2 }$
6873 \__stex_sproof_maybe_comment:
6874 }
6875 \stex_deactivate_macro:Nn \eqstep {sproof~environments}
6876
6877 \NewDocumentCommand \yield {+m}{

```

```

6878 \stex_annotate:nnn{spfyield}{\}{ #1 }
6879 }
6880 \stex_deactivate_macro:Nn \yield {sproof~environments}
6881
6882 \NewDocumentEnvironment{spfblock}{\}{\{
6883 \item[]
6884 \bool_set_true:N \l__stex_sproof_in_spfblock_bool
6885 }{
6886 \aftergroup\__stex_sproof_maybe_comment:
6887 }
6888 \AddToHook{env/spfblock/before}{\__stex_sproof_maybe_comment_end:}
6889

```

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

`\spfidea`

```

6890 \NewDocumentCommand\spfidea{0{\} +m}{
6891 \__stex_sproof_spf_args:n{#1}
6892 \titleemph{
6893 \tl_if_empty:NTF \spftype {Proof~Idea}{
6894 \spftype
6895 }:
6896 }~#2
6897 \sproofend
6898 }

```

(End definition for `\spfidea`. This function is documented on page 54.)

```

6899 \newcommand\spfjust[1]{
6900 #1
6901 }
6902 \</package>

```

Some auxiliary code, and clean up to be executed at the end of the package.

Chapter 36

STEX -Others Implementation

```
6903 <*package>
6904
6905 %%%%%%%%%% others.dtx %%%%%%%%%%
6906
6907 <@@=stex_others>
        Warnings and error messages
6908 % None

\MSC Math subject classifier

6909 \NewDocumentCommand \MSC {m} {
6910 % TODO
6911 }

(End definition for \MSC. This function is documented on page ??.)
        Patching tikzinput, if loaded

6912 \@ifpackageloaded{tikzinput}{
6913 \RequirePackage{stex-tikzinput}
6914 }{}
6915
6916 \bool_if:NT \c_stex_persist_mode_bool {
6917 \let__stex_notation_restore_notation_old:nnnnn
6918 \__stex_notation_restore_notation:nnnnn
6919 \def__stex_notation_restore_notation_new:nnnnn#1#2#3#4#5{
6920 \__stex_notation_restore_notation_old:nnnnn{#1}{#2}{#3}{#4}{#5}
6921 \ExplSyntaxOn
6922 }
6923 \def__stex_notation_restore_notation:nnnnn{
6924 \ExplSyntaxOff
6925 \catcode'\sim10
6926 \__stex_notation_restore_notation_new:nnnnn
6927 }
6928 \input{\jobname.sms}
6929 \let__stex_notation_restore_notation:nnnnn
6930 \__stex_notation_restore_notation_old:nnnnn
6931 \prop_if_exist:NT\c_stex_mathhub_main_manifest_prop{
```

```

6932     \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
6933     \l_tmpa_str
6934     \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str_manifest_prop }
6935     \c_stex_mathhub_main_manifest_prop
6936     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
6937   }
6938 }
6939
6940 \stex_get_document_uri:
6941 </package>

```


Chapter 37

STEX -Metatheory Implementation

```
6942 <*package>
6943 <@@=stex_modules>
6944
6945 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
6946
6947 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX/meta}
6948 \begingroup
6949 \stex_module_setup:nn{
6950   ns=\c_stex_metatheory_ns_str,
6951   meta=NONE
6952 }{Metatheory}
6953 \stex_reactivate_macro:N \symdecl
6954 \stex_reactivate_macro:N \notation
6955 \stex_reactivate_macro:N \symdef
6956 \ExplSyntaxOff
6957 \csname stex_suppress_html:n\endcsname{
6958   % is-a (a:A, a \in A, a is an A, etc.)
6959   \symdecl{isa}[args=ai]
6960   \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
6961   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
6962   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
6963
6964   % bind (\forall, \Pi, \lambda etc.)
6965   \symdecl{bind}[args=Bi,assoc=pre]
6966   \notation{bind}[depfun,prec=nobrackets,op={(\cdot)\;\to\;\cdot}]{\comp( #1 \comp{}\;\to\;)}
6967   \notation{bind}[forall]{\comp\forall #1.\;#2}{##1 \comp, ##2}
6968   \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6969
6970   % implicit bind
6971   \symdecl{implicitbind}[args=Bi,assoc=pre]
6972   \notation{implicitbind}[braces,prec=nobrackets,op={(\cdot\_I\;\cdot)}]{\comp\{ #1 \comp{
6973     \notation{implicitbind}[depfun,prec=nobrackets]{\comp( #1 \comp{}\;\to\_I\; } #2}{##1 \comp,
6974     \notation{implicitbind}[Pi]{\comp\prod^I_{#1}#2}{##1\comp,##2}
6975
6976   % dummy variable
```

```

6977 \symdecl{dummyvar}
6978 \notation{dummyvar}[underscore]{\comp\_}
6979 \notation{dummyvar}[dot]{\comp\cdot}
6980 \notation{dummyvar}[dash]{\comp{\rm --}}
6981
6982 %fromto (function space, Hom-set, implication etc.)
6983 \symdecl{fromto}[args=ai]
6984 \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
6985 \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
6986
6987 % mapto (lambda etc.)
6988 \symdecl{mapto}[args=Bi]
6989 %\notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
6990 %\notation{mapto}[lambda]{\comp\lambda #1 \comp.\; #2}{#1 \comp, #2}
6991 %\notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.\; #2}{#1 \comp, #2}
6992
6993 % function/operator application
6994 \symdecl{apply}[args=ia]
6995 \notation{apply}[prec=0;0x\infpres,parens,op=\cdot(\cdot)]{#1 \comp( #2 \comp)}{##1 \comp,
6996 \notation{apply}[prec=0;0x\infpres,lambda]{#1 \; #2 }{##1 \; ; ##2}
6997
6998 % collection of propositions/booleans/truth values
6999 \symdecl{prop}[name=proposition]
7000 \notation{prop}[prop]{\comp{\rm prop}}
7001 \notation{prop}[BOOL]{\comp{\rm BOOL}}
7002
7003 \symdecl{judgmentholds}[args=1]
7004 \notation{judgmentholds}[vdash,op=\vdash]{\comp\vdash\; #1}
7005
7006 % sequences
7007 \symdecl{seqtype}[args=1]
7008 \notation{seqtype}[kleene]{#1^{\comp\ast}}
7009
7010 \symdecl{seqexpr}[args=a]
7011 \notation{seqexpr}[angle,prec=nobrackets]{\comp\langle #1\comp\rangle}{##1\comp,##2}
7012
7013 \symdef{seqmap}[args=abi,setlike]{\comp\{#3 \comp| #2\comp\in \dobrackets{#1} \comp\}}{##1
7014 \symdef{seqprepend}[args=ia]{#1 \comp{::} #2}{##1 \comp, ##2}
7015 \symdef{seqappend}[args=ai]{#1 \comp{::} #2}{##1 \comp, ##2}
7016 \symdef{seqfoldleft}[args=iabbi]{ \comp{foldl}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\comp}
7017 \symdef{seqfoldright}[args=iabbi,op=foldr]{ \comp{foldr}\dobrackets{#1,#2}\dobrackets{#3\comp
7018 \symdef{seqhead}[args=a]{\comp{head}\dobrackets{#1}}{##1 \comp, ##2}
7019 \symdef{seqtail}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
7020 \symdef{seqlast}[args=a]{\comp{last}\dobrackets{#1}}{##1 \comp, ##2}
7021 \symdef{seqinit}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
7022
7023 \symdef{sequence-index}[args=2,li,prec=nobrackets]{\comp{#1}_{#2}}
7024 \notation{sequence-index}[ui,prec=nobrackets]{\comp{#1}^{\comp{#2}}}
7025
7026 \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{\,\ellipses\,}}{##1\comp,##2}
7027 \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{\,\ellipses\,}#2}{##1\comp,##2}
7028 \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{\,\ellipses\,}#2\comp{\,\ellipses\,}#3}{##1\comp,##2,##3}
7029
7030 % nat literals

```

```

7031 \symdef{natliteral}{\comp{\mathtt{Ord}}}
7032
7033 % letin (''let'', local definitions, variable substitution)
7034 \symdecl{letin}[args=bii]
7035 \notation{letin}{let}{\comp{\rm let}}\;#1\comp{=}\;#2\;\comp{\rm in}}\;#3}
7036 \notation{letin}{subst}{#3 \comp[ #1 \comp/ #2 \comp]}
7037 \notation{letin}{frac}{#3 \comp[ \frac{#2}{#1} \comp]}
7038
7039 % structures
7040 \symdecl*{module-type}[args=1]
7041 \notation{module-type}{\comp{\mathtt{MOD}}}\;#1}
7042 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
7043 \notation{mathstruct}[angle,prec=nobrackets]{\comp\angle #1 \comp\angle}{##1 \comp, ##2}
7044
7045 % objects
7046 \symdecl{object}
7047 \notation{object}{\comp{\mathtt{OBJECT}}}
7048
7049 }
7050
7051 % The following are abbreviations in the sTeX corpus that are left over from earlier
7052 % developments. They will eventually be phased out.
7053
7054 \ExplSyntaxOn
7055 \stex_add_to_current_module:n{
7056   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
7057   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
7058   \def\livar{\csname sequence-index\endcsname[li]}
7059   \def\uivar{\csname sequence-index\endcsname[ui]}
7060   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
7061   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
7062 }
7063 \__stex_modules_end_module:
7064 \endgroup
7065 \</package>

```

Chapter 38

Tikzinput Implementation

```
7066 <@@=tikzinput>
7067 <*package>
7068
7069 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
7070
7071 \ProvidesExplPackage{tikzinput}{2022/08/08}{3.2.0}{tikzinput package}
7072 \RequirePackage{l3keys2e}
7073
7074 \keys_define:nn { tikzinput } {
7075   image .bool_set:N = \c_tikzinput_image_bool,
7076   image .default:n = false ,
7077   unknown .code:n = {}
7078 }
7079
7080 \ProcessKeysOptions { tikzinput }
7081
7082 \bool_if:NTF \c_tikzinput_image_bool {
7083   \RequirePackage{graphicx}
7084
7085   \providecommand\usetikzlibrary[]{}
7086   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
7087 }{
7088   \RequirePackage{tikz}
7089   \RequirePackage{standalone}
7090
7091   \newcommand \tikzinput [2] [] {
7092     \setkeys{Gin}{#1}
7093     \ifx \Gin@ewidth \Gin@exclamation
7094       \ifx \Gin@eheight \Gin@exclamation
7095         \input { #2 }
7096       \else
7097         \resizebox{!}{ \Gin@eheight }{
7098           \input { #2 }
7099         }
7100       \fi
7101     \else
7102       \ifx \Gin@eheight \Gin@exclamation
7103         \resizebox{ \Gin@ewidth }{!}{
```

```

7104         \input { #2 }
7105     }
7106     \else
7107         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
7108             \input { #2 }
7109         }
7110     \fi
7111 \fi
7112 }
7113 }
7114
7115 \newcommand \ctikzinput [2] [] {
7116     \begin{center}
7117         \tikzinput [#1] {#2}
7118     \end{center}
7119 }
7120
7121 \@ifpackageloaded{stex}{
7122     \RequirePackage{stex-tikzinput}
7123 }{}
7124
7125 </package>
7126 <*stex>
7127
7128 \ProvidesExplPackage{stex-tikzinput}{2022/08/08}{3.2.0}{stex-tikzinput}
7129 \RequirePackage{stex}
7130 \RequirePackage{tikzinput}
7131
7132 \newcommand\mhtikzinput[2] []{%
7133     \def\Gin@mhrepos{ }\setkeys{Gin}{#1}%
7134     \stex_in_repository:nn\Gin@mhrepos{
7135         \tikzinput[#1]{\mhp@hpath{##1}{#2}}
7136     }
7137 }
7138
7139 \newcommand\cmhtikzinput[2] [] {\begin{center}\mhtikzinput[#1]{#2}\end{center}}
7140
7141 \cs_new_protected:Nn \__tikzinput_usetikzlibrary:nn {
7142     \pgfkeys@spdef\pgf@temp{#1}
7143     \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
7144     \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgf@temp
7145     \expandafter\edef\csname tikz@library@#1@atcode\endcsname{\the\catcode'\@}
7146     \expandafter\edef\csname tikz@library@#1@barcode\endcsname{\the\catcode'\|}
7147     \expandafter\edef\csname tikz@library@#1@dollarcode\endcsname{\the\catcode'\$}
7148     \catcode'\@=11
7149     \catcode'\|=12
7150     \catcode'\$=3
7151     \pgfutil@InputIfFileExists{#2}{-}{-}
7152     \catcode'\@=\csname tikz@library@#1@atcode\endcsname
7153     \catcode'\|=\csname tikz@library@#1@barcode\endcsname
7154     \catcode'\$=\csname tikz@library@#1@dollarcode\endcsname
7155 }
7156
7157 \newcommand\libusetikzlibrary[1]{

```

```

7157 \prop_if_exist:NF \l_stex_current_repository_prop {
7158   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
7159 }
7160 \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
7161   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
7162 }
7163 \seq_clear:N \l__tikzinput_libinput_files_seq
7164 \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
7165 \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
7166
7167 \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
7168   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / tikzlibrary
7169   \IfFileExists{ \l_tmpa_str }{
7170     \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
7171   }{}
7172   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
7173   \seq_put_right:No \l_tmpa_seq \l_tmpa_str
7174 }
7175
7176 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / tikzlibrary #1 .code.t
7177 \IfFileExists{ \l_tmpa_str }{
7178   \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
7179 }{}
7180
7181 \seq_if_empty:NTF \l__tikzinput_libinput_files_seq {
7182   \msg_error:nxxx{stex}{error/nofile}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .code.t
7183 }{
7184   \int_compare:nNnTF {\seq_count:N \l__tikzinput_libinput_files_seq} = 1 {
7185     \seq_map_inline:Nn \l__tikzinput_libinput_files_seq {
7186       \__tikzinput_usetikzlibrary:nn{#1}{ ##1 }
7187     }
7188   }{
7189     \msg_error:nxxx{stex}{error/twofiles}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .co
7190   }
7191 }
7192 }
7193 </stex>

```

Chapter 39

document-structure.sty Implementation

```
7194 <*package>
7195 <@@=document_structure>
7196 \ProvidesExplPackage{document-structure}{2022/08/08}{3.2.0}{Modular Document Structure}
7197 \RequirePackage{13keys2e}
```

39.1 Package Options

We declare some switches which will modify the behavior according to the package options. Generally, an option xxx will just set the appropriate switches to true (otherwise they stay false).

```
7198
7199 \keys_define:nn{ document-structure }{
7200   class      .str_set_x:N = \c_document_structure_class_str,
7201   topsect    .str_set_x:N = \c_document_structure_topsect_str,
7202   unknown    .code:n      = {
7203     \PassOptionsToClass{\CurrentOption}{stex}
7204     \PassOptionsToClass{\CurrentOption}{tikzinput}
7205   }
7206   % showignores .bool_set:N = \c_document_structure_showignores_bool,
7207 }
7208 \ProcessKeysOptions{ document-structure }
7209 \str_if_empty:NT \c_document_structure_class_str {
7210   \str_set:Nn \c_document_structure_class_str {article}
7211 }
7212 \str_if_empty:NT \c_document_structure_topsect_str {
7213   \str_set:Nn \c_document_structure_topsect_str {section}
7214 }
```

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

```
7215 \RequirePackage{xspace}
7216 \RequirePackage{comment}
7217 \RequirePackage{stex}
7218 \AddToHook{begindocument}{
```

```

7219 \ltx@ifpackageloaded{babel}{
7220   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7221   \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
7222     \makeatletter\input{document-structure-ngerman.ldf}\makeatother
7223   }
7224 }{}
7225 }

```

`\section@level` Finally, we set the `\section@level` macro that governs sectioning. The default is two (corresponding to the `article` class), then we set the defaults for the standard classes `book` and `report` and then we take care of the levels passed in via the `topsect` option.

```

7226 \int_new:N \l_document_structure_section_level_int
7227 \str_case:NnF \c_document_structure_topsect_str {
7228   {part}}{
7229     \int_set:Nn \l_document_structure_section_level_int {0}
7230   }
7231   {chapter}}{
7232     \int_set:Nn \l_document_structure_section_level_int {1}
7233   }
7234 }{
7235   \str_case:NnF \c_document_structure_class_str {
7236     {book}}{
7237       \int_set:Nn \l_document_structure_section_level_int {0}
7238     }
7239     {report}}{
7240       \int_set:Nn \l_document_structure_section_level_int {0}
7241     }
7242   }{
7243     \int_set:Nn \l_document_structure_section_level_int {2}
7244   }
7245 }

```

39.2 Document Structure

The structure of the document is given by the `sfragment` environment. The hierarchy is adjusted automatically according to the L^AT_EX class in effect.

`\currentsectionlevel` For the `\currentsectionlevel` and `\Currentsectionlevel` macros we use an internal macro `\current@section@level` that only contains the keyword (no markup). We initialize it with “document” as a default. In the generated OMDoc, we only generate a text element of class `omdoc_currentsectionlevel`, which will be instantiated by CSS later.⁹

EdN:9

```

7246 \def\current@section@level{document}%
7247 \newcommand\currentsectionlevel{\lowercase\expandafter\current@section@level\xspace}%
7248 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%

```

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

`\skipfragment`

```

7249 \cs_new_protected:Npn \skipfragment {

```

⁹EdNOTE: MK: we may have to experiment with the more powerful uppercasing macro from `mfirstuc.sty` once we internationalize.


```

7250 \ifcase\l_document_structure_section_level_int
7251 \or\stepcounter{part}
7252 \or\stepcounter{chapter}
7253 \or\stepcounter{section}
7254 \or\stepcounter{subsection}
7255 \or\stepcounter{subsubsection}
7256 \or\stepcounter{paragraph}
7257 \or\stepcounter{subparagraph}
7258 \fi
7259 }

```

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

`blindfragment (env.)`

```

7260 \newcommand\at@begin@blindsfragment[1]{
7261 \newenvironment{blindfragment}
7262 {
7263 \int_incr:N\l_document_structure_section_level_int
7264 \at@begin@blindsfragment\l_document_structure_section_level_int
7265 }{}

```

`\sfragment@nonum` convenience macro: `\sfragment@nonum{<level>}{<title>}` makes an unnumbered sectioning with title `<title>` at level `<level>`.

```

7266 \newcommand\sfragment@nonum[2]{
7267 \ifx\hyper@anchor\@undefined\else\phantomsection\fi
7268 \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
7269 }

```

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

`\sfragment@num` convenience macro: `\sfragment@num{<level>}{<title>}` makes numbered sectioning with title `<title>` at level `<level>`. We have to check the `short` key was given in the `sfragment` environment and – if it is use it. But how to do that depends on whether the `rdmeta` package has been loaded. In the end we call `\sref@label@id` to enable crossreferencing.

```

7270 \newcommand\sfragment@num[2]{
7271 \tl_if_empty:NTF \l__document_structure_sfragment_short_tl {
7272 \@nameuse{#1}{#2}
7273 }{
7274 \cs_if_exist:NTF\rdmeta@sectioning{
7275 \@nameuse{rdmeta@#1@old}[\l__document_structure_sfragment_short_tl]{#2}
7276 }{
7277 \@nameuse{#1}[\l__document_structure_sfragment_short_tl]{#2}
7278 }
7279 }
7280 %\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\sfragment@id
7281 }

```

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

`sfragment (env.)`

```

7282 \keys_define:nn { document-structure / sfragment }{
7283 id .str_set_x:N = \l__document_structure_sfragment_id_str,
7284 date .str_set_x:N = \l__document_structure_sfragment_date_str,

```

```

7285 creators      .clist_set:N = \l__document_structure_sfragment_creators_clist,
7286 contributors   .clist_set:N = \l__document_structure_sfragment_contributors_clist,
7287 srccite        .tl_set:N     = \l__document_structure_sfragment_srccite_tl,
7288 type           .tl_set:N     = \l__document_structure_sfragment_type_tl,
7289 short          .tl_set:N     = \l__document_structure_sfragment_short_tl,
7290 intro          .tl_set:N     = \l__document_structure_sfragment_intro_tl,
7291 imports        .tl_set:N     = \l__document_structure_sfragment_imports_tl,
7292 loadmodules    .bool_set:N   = \l__document_structure_sfragment_loadmodules_bool
7293 }
7294 \cs_new_protected:Nn \__document_structure_sfragment_args:n {
7295   \str_clear:N \l__document_structure_sfragment_id_str
7296   \str_clear:N \l__document_structure_sfragment_date_str
7297   \clist_clear:N \l__document_structure_sfragment_creators_clist
7298   \clist_clear:N \l__document_structure_sfragment_contributors_clist
7299   \tl_clear:N \l__document_structure_sfragment_srccite_tl
7300   \tl_clear:N \l__document_structure_sfragment_type_tl
7301   \tl_clear:N \l__document_structure_sfragment_short_tl
7302   \tl_clear:N \l__document_structure_sfragment_imports_tl
7303   \tl_clear:N \l__document_structure_sfragment_intro_tl
7304   \bool_set_false:N \l__document_structure_sfragment_loadmodules_bool
7305   \keys_set:nn { document-structure / sfragment } { #1 }
7306 }

```

we define a switch for numbering lines and a hook for the beginning of groups: The `\at@begin@sfragment` macro allows customization. It is run at the beginning of the `sfragment`, i.e. after the section heading.

```

7307 \newif\if@mainmatter\@mainmattertrue
7308 \newcommand\at@begin@sfragment[3][]{ }

```

Then we define a helper macro that takes care of the sectioning magic. It comes with its own key/value interface for customization.

```

7309 \keys_define:nn { document-structure / sectioning }{
7310   name      .str_set_x:N = \l__document_structure_sect_name_str  ,
7311   ref       .str_set_x:N = \l__document_structure_sect_ref_str   ,
7312   clear     .bool_set:N  = \l__document_structure_sect_clear_bool ,
7313   clear     .default:n   = {true}                                ,
7314   num       .bool_set:N  = \l__document_structure_sect_num_bool   ,
7315   num       .default:n   = {true}
7316 }
7317 \cs_new_protected:Nn \__document_structure_sect_args:n {
7318   \str_clear:N \l__document_structure_sect_name_str
7319   \str_clear:N \l__document_structure_sect_ref_str
7320   \bool_set_false:N \l__document_structure_sect_clear_bool
7321   \bool_set_false:N \l__document_structure_sect_num_bool
7322   \keys_set:nn { document-structure / sectioning } { #1 }
7323 }
7324 \newcommand\omdoc@sectioning[3][]{
7325   \__document_structure_sect_args:n {#1 }
7326   \let\omdoc@sect@name\l__document_structure_sect_name_str
7327   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
7328   \if@mainmatter% numbering not overridden by frontmatter, etc.
7329     \bool_if:NTF \l__document_structure_sect_num_bool {
7330       \sfragment@num{#2}{#3}
7331     }{

```

```

7332     \sfragment@nonum{#2}{#3}
7333   }
7334   \def\current@section@level{\omdoc@sect@name}
7335   \else
7336     \sfragment@nonum{#2}{#3}
7337   \fi
7338 }% if@mainmatter

```

and another one, if redefines the `\addtocontentsline` macro of L^AT_EX to import the respective macros. It takes as an argument a list of module names.

```

7339 \newcommand\sfragment@redefine@addtocontents[1]{%
7340 %\edef\__document_structureimport{#1}%
7341 %\@for\@I:=\__document_structureimport\do{%
7342 %\edef\@path{\csname module@\@I @path\endcsname}%
7343 %\@ifundefined{tf@toc}\relax%
7344 %    {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
7345 %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
7346 %\def\addcontentsline##1##2##3{%
7347 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
7348 %\else% hyperref.sty not loaded
7349 %\def\addcontentsline##1##2##3{%
7350 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
7351 %\fi
7352 }% hypreref.sty loaded?

```

now the `sfragment` environment itself. This takes care of the table of contents via the helper macro above and then selects the appropriate sectioning command from `article.cls`. It also registers the current level of sfragments in the `\sfragment@level` counter.

```

7353 \newenvironment{sfragment}[2][ ]% keys, title
7354 {
7355   \__document_structure_sfragment_args:n { #1 }%\sref@target%

```

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

```

7356   \stex_csl_to_imports:No \usemodule \l__document_structure_sfragment_imports_tl
7357
7358   \bool_if:NT \l__document_structure_sfragment_loadmodules_bool {
7359     \sfragment@redefine@addtocontents{
7360       %\@ifundefined{module@id}\used@modules%
7361       %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}
7362     }
7363   }

```

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

```

7364
7365   \stex_document_title:n { #2 }
7366
7367   \int_incr:N\l__document_structure_section_level_int
7368   \ifcase\l__document_structure_section_level_int
7369     \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
7370     \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
7371     \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
7372     \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}

```

```

7373 \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
7374 \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#1}
7375 \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#1}
7376 \fi
7377 \at@begin@sfragment[#1]\l__document_structure_section_level_int{#2}
7378 \str_if_empty:NF \l__document_structure_sfragment_id_str {
7379   \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
7380 }
7381 }% for customization
7382 {}

```

and finally, we localize the sections

```

7383 \newcommand\omdoc@part@kw{Part}
7384 \newcommand\omdoc@chapter@kw{Chapter}
7385 \newcommand\omdoc@section@kw{Section}
7386 \newcommand\omdoc@subsection@kw{Subsection}
7387 \newcommand\omdoc@subsubsection@kw{Subsubsection}
7388 \newcommand\omdoc@paragraph@kw{paragraph}
7389 \newcommand\omdoc@subparagraph@kw{subparagraph}

```

39.3 Front and Backmatter

Index markup is provided by the `omtext` package [Kohlhase:smmtf:git], so in the `document-structure` package we only need to supply the corresponding `\printindex` command, if it is not already defined

`\printindex`

```

7390 \providecommand\printindex{\IfFileExists{\jobname.ind}{\input{\jobname.ind}}{}}

```

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

some classes (e.g. `book.cls`) already have `\frontmatter`, `\mainmatter`, and `\backmatter` macros. As we want to define `frontmatter` and `backmatter` environments, we save their behavior (possibly defining it) in `orig*matter` macros and make them undefined (so that we can define the environments).

```

7391 \cs_if_exist:NTF\frontmatter{
7392   \let\__document_structure_orig_frontmatter\frontmatter
7393   \let\frontmatter\relax
7394 }{
7395   \tl_set:Nn\__document_structure_orig_frontmatter{
7396     \clearpage
7397     \@mainmatterfalse
7398     \pagenumbering{roman}
7399   }
7400 }
7401 \cs_if_exist:NTF\backmatter{
7402   \let\__document_structure_orig_backmatter\backmatter
7403   \let\backmatter\relax
7404 }{
7405   \tl_set:Nn\__document_structure_orig_backmatter{
7406     \clearpage
7407     \@mainmatterfalse
7408     \pagenumbering{roman}
7409   }

```

7410 }

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

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

```
7411 \newenvironment{frontmatter}{
7412   \__document_structure_orig_frontmatter
7413 }{
7414   \cs_if_exist:NTF\mainmatter{
7415     \mainmatter
7416   }{
7417     \clearpage
7418     \@mainmattertrue
7419     \pagenumbering{arabic}
7420   }
7421 }
```

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

```
7422 \newenvironment{backmatter}{
7423   \__document_structure_orig_backmatter
7424 }{
7425   \cs_if_exist:NTF\mainmatter{
7426     \mainmatter
7427   }{
7428     \clearpage
7429     \@mainmattertrue
7430     \pagenumbering{arabic}
7431   }
7432 }
```

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

```
7433 \@mainmattertrue\pagenumbering{arabic}
```

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

```
7434 \def \c__document_structure_document_str{document}
7435 \newcommand\afterprematurestop{}
7436 \def\prematurestop@endsfragment{
7437   \unless\ifx\@currenvir\c__document_structure_document_str
7438     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
7439     \expandafter\prematurestop@endsfragment
7440   \fi
7441 }
7442 \providecommand\prematurestop{
7443   \message{Stopping~sTeX~processing~prematurely}
7444   \prematurestop@endsfragment
7445   \afterprematurestop
7446   \end{document}
7447 }
```

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

39.4 Global Variables

\setSGvar set a global variable

```
7448 \RequirePackage{etoolbox}
7449 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}
```

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

\useSGvar use a global variable

```
7450 \newrobustcmd\useSGvar[1]{%
7451   \@ifundefined{sTeX@Gvar@#1}
7452   {\PackageError{document-structure}
7453    {The sTeX Global variable #1 is undefined}
7454    {set it with \protect\setSGvar}}
7455   \@nameuse{sTeX@Gvar@#1}}
```

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

\ifSGvar execute something conditionally based on the state of the global variable.

```
7456 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
7457   \@ifundefined{sTeX@Gvar@#1}
7458   {\PackageError{document-structure}
7459    {The sTeX Global variable #1 is undefined}
7460    {set it with \protect\setSGvar}}
7461   {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}
```

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

Chapter 40

NotesSlides – Implementation

40.1 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.

```
7462 \*cls)
7463 \@@=notesslides)
7464 \ProvidesExplClass{notesslides}{2022/08/08}{3.2.0}{notesslides Class}
7465 \RequirePackage{13keys2e}
7466
7467 \keys_define:nn{notesslides / cls}{
7468   class .str_set_x:N = \c__notesslides_class_str,
7469   notes .bool_set:N = \c__notesslides_notes_bool ,
7470   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
7471   docopt .str_set_x:N = \c__notesslides_docopt_str,
7472   unknown .code:n = {
7473     \PassOptionsToPackage{\CurrentOption}{document-structure}
7474     \PassOptionsToClass{\CurrentOption}{beamer}
7475     \PassOptionsToPackage{\CurrentOption}{notesslides}
7476     \PassOptionsToPackage{\CurrentOption}{stex}
7477   }
7478 }
7479 \ProcessKeysOptions{ notesslides / cls }
7480
7481 \str_if_empty:NF \c__notesslides_class_str {
7482   \PassOptionsToPackage{class=\c__notesslides_class_str}{document-structure}
7483 }
7484
7485 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{book}{
7486   \PassOptionsToPackage{defaulttopsect=part}{notesslides}
7487 }
7488 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{report}{
7489   \PassOptionsToPackage{defaulttopsect=part}{notesslides}
7490 }
7491
7492 \RequirePackage{stex}
```

```

7493 \stex_html_backend:T {
7494   \bool_set_true:N\c__notesslides_notes_bool
7495 }
7496
7497 \bool_if:NTF \c__notesslides_notes_bool {
7498   \PassOptionsToPackage{notes=true}{notesslides}
7499   \message{notesslides.cls:~Formatting~course~materials~in~notes~mode}
7500 }{
7501   \PassOptionsToPackage{notes=false}{notesslides}
7502   \message{notesslides.cls:~Formatting~course~materials~in~slides~mode}
7503 }
7504 \</cls>

```

now we do the same for the notesslides package.

```

7505 <*package>
7506 \ProvidesExplPackage{notesslides}{2022/08/08}{3.2.0}{notesslides Package}
7507 \RequirePackage{l3keys2e}
7508
7509 \keys_define:nn{notesslides / pkg}{
7510   topsect          .str_set_x:N = \c__notesslides_topsect_str,
7511   defaulttopsect   .str_set_x:N = \c__notesslides_defaulttopsec_str,
7512   notes            .bool_set:N = \c__notesslides_notes_bool ,
7513   slides           .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
7514   sectocframes     .bool_set:N = \c__notesslides_sectocframes_bool ,
7515   frameimages      .bool_set:N = \c__notesslides_frameimages_bool ,
7516   fiboxed          .bool_set:N = \c__notesslides_fiboxed_bool ,
7517   nopproblems      .bool_set:N = \c__notesslides_nopproblems_bool,
7518   unknown          .code:n      = {
7519     \PassOptionsToClass{\CurrentOption}{stex}
7520     \PassOptionsToClass{\CurrentOption}{tikzinput}
7521   }
7522 }
7523 \ProcessKeysOptions{ notesslides / pkg }
7524
7525 \RequirePackage{stex}
7526 \stex_html_backend:T {
7527   \bool_set_true:N\c__notesslides_notes_bool
7528 }
7529
7530 \newif\ifnotes
7531 \bool_if:NTF \c__notesslides_notes_bool {
7532   \notesttrue
7533 }{
7534   \notestfalse
7535 }
7536

```

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

```

7537 \str_if_empty:NTF \c__notesslides_topsect_str {
7538   \str_set_eq:NN \__notesslides_topsect \c__notesslides_defaulttopsec_str
7539 }{
7540   \str_set_eq:NN \__notesslides_topsect \c__notesslides_topsect_str
7541 }
7542 \PassOptionsToPackage{topsect=\__notesslides_topsect}{document-structure}

```



```
7543 </package>
```

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

```
7544 <*cls>
7545 \bool_if:NTF \c__notesslides_notes_bool {
7546   \str_if_empty:NT \c__notesslides_class_str {
7547     \str_set:Nn \c__notesslides_class_str {article}
7548   }
7549   \exp_after:wN\LoadClass\exp_after:wN[\c__notesslides_dcopt_str]
7550   {\c__notesslides_class_str}
7551 }{
7552   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
7553   \newcounter{Item}
7554   \newcounter{paragraph}
7555   \newcounter{subparagraph}
7556   \newcounter{Hfootnote}
7557 }
7558 \RequirePackage{document-structure}
```

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

```
7559 \RequirePackage{notesslides}
7560 </cls>
```

In `notes` mode, we also have to make the `beamer`-specific things available to `article` via the `beamerarticle` package. We use options to avoid loading theorem-like environments, since we want to use our own from the \TeX packages. The first batch of packages we want are loaded on `notesslides.sty`. These are the general ones, we will load the \TeX -specific ones after we have done some work (e.g. defined the counters `m*`). Only the `stex-logo` package is already needed now for the default theme.

```
7561 <*package>
7562 \bool_if:NT \c__notesslides_notes_bool {
7563   \RequirePackage{a4wide}
7564   \RequirePackage{marginnote}
7565   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
7566   \RequirePackage{mdframed}
7567   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
7568   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
7569 }
7570 \RequirePackage{stex-tikzinput}
7571 \RequirePackage{comment}
7572 \RequirePackage{url}
7573 \RequirePackage{graphicx}
7574 \RequirePackage{pgf}
7575 \RequirePackage{bookmark}
```

40.2 Notes and Slides

For the lecture notes cases, we also provide the `\usetheme` macro that would otherwise come from the `beamer` class.

```
7576 \bool_if:NT \c__notesslides_notes_bool {
7577   \renewcommand\usetheme[2][\usepackage{#1}{beamertheme#2}]
7578 }
```

```

7579 \NewDocumentCommand \libusetheme {0{} m} {
7580   \libusepackage[#1]{beamertheme#2}
7581 }
7582

```

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

```

7583 \newcounter{slide}
7584 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
7585 \newlength{\slideheight}\setlength{\slideheight}{9cm}

```

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

```

7586 \bool_if:NTF \c__notesslides_notes_bool {
7587   \renewenvironment{note}{\ignorespaces}{}
7588 }{
7589   \excludecomment{note}
7590 }

```

We first set up the slide boxes in **article** mode. We set up sizes and provide a box register for the frames and a counter for the slides.

```

7591 \bool_if:NT \c__notesslides_notes_bool {
7592   \newlength{\slideframewidth}
7593   \setlength{\slideframewidth}{1.5pt}

```

frame (*env.*) We first define the keys.

```

7594 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
7595   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
7596     \bool_set_true:N #1
7597   }{
7598     \bool_set_false:N #1
7599   }
7600 }
7601 \keys_define:nn{notesslides / frame}{
7602   label .str_set_x:N = \l__notesslides_frame_label_str,
7603   allowframebreaks .code:n = {
7604     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
7605   },
7606   allowdisplaybreaks .code:n = {
7607     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
7608   },
7609   fragile .code:n = {
7610     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
7611   },
7612   shrink .code:n = {
7613     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
7614   },
7615   squeeze .code:n = {
7616     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
7617   },
7618   t .code:n = {

```

```

7619     \_notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
7620   },
7621   unknown    .code:n      = {}
7622 }
7623 \cs_new_protected:Nn \_notesslides_frame_args:n {
7624   \str_clear:N \l__notesslides_frame_label_str
7625   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
7626   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
7627   \bool_set_true:N \l__notesslides_frame_fragile_bool
7628   \bool_set_true:N \l__notesslides_frame_shrink_bool
7629   \bool_set_true:N \l__notesslides_frame_squeeze_bool
7630   \bool_set_true:N \l__notesslides_frame_t_bool
7631   \keys_set:nn { notesslides / frame }{ #1 }
7632 }

```

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

```

7633 \renewenvironment{frame}[1][]{
7634   \_notesslides_frame_args:n{#1}
7635   \sffamily
7636   \stepcounter{slide}
7637   \def\@currentlabel{\theslide}
7638   \str_if_empty:NF \l__notesslides_frame_label_str {
7639     \label{\l__notesslides_frame_label_str}
7640   }

```

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

```

7641   \def\itemize@level{outer}
7642   \def\itemize@outer{outer}
7643   \def\itemize@inner{inner}
7644   \renewcommand\newpage{\addtocounter{framenum}{1}}
7645   %\newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
7646   \renewenvironment{itemize}{
7647     \ifx\itemize@level\itemize@outer
7648       \def\itemize@label{$\rhd$}
7649     \fi
7650     \ifx\itemize@level\itemize@inner
7651       \def\itemize@label{$\scriptstyle\rhd$}
7652     \fi
7653     \begin{list}
7654       {\itemize@label}
7655       {\setlength{\labelsep}{.3em}
7656        \setlength{\labelwidth}{.5em}
7657        \setlength{\leftmargin}{1.5em}
7658       }
7659     \edef\itemize@level{\itemize@inner}
7660   }{
7661     \end{list}
7662   }

```

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

```

7663   \stex_html_backend:TF {
7664     \begin{stex_annotate_env}{frame}{}\vbox\bgroup
7665     \mdf@patchamsthm
7666   }{
7667     \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwid

```

```

7668     }
7669   }{
7670     \stex_html_backend:TF {
7671       \miko@slidelabel\egroup\end{stex_annotate_env}
7672     }\medskip\miko@slidelabel\end{mdframed}}
7673   }

```

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

`\frametitle`

```

7674   \renewcommand{\frametitle}[1]{
7675     \stex_document_title:n { #1 }
7676     {\Large\bf\sf\color{blue}{#1}}\medskip
7677   }
7678 }

```

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

EdN:10

`\pause` 10

```

7679 \bool_if:NT \c__notesslides_notes_bool {
7680   \newcommand\pause{
7681   }

```

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

`nparagraph (env.)`

```

7682 \bool_if:NTF \c__notesslides_notes_bool {
7683   \newenvironment{nparagraph}[1] [] {\begin{sparagraph}[#1]}\end{sparagraph}}
7684 }{
7685   \excludecomment{nparagraph}
7686 }

```

`nfragment (env.)`

```

7687 \bool_if:NTF \c__notesslides_notes_bool {
7688   \newenvironment{nfragment}[2] [] {\begin{sfragment}[#1][#2]}\end{sfragment}}
7689 }{
7690   \excludecomment{nfragment}
7691 }

```

`ndefinition (env.)`

```

7692 \bool_if:NTF \c__notesslides_notes_bool {
7693   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}\end{sdefinition}}
7694 }{
7695   \excludecomment{ndefinition}
7696 }

```

`nassertion (env.)`

```

7697 \bool_if:NTF \c__notesslides_notes_bool {
7698   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}\end{sassertion}}
7699 }{
7700   \excludecomment{nassertion}
7701 }

```

¹⁰EDNOTE: MK: fake it in notes mode for now

`nsproof (env.)`

```
7702 \bool_if:NTF \c__notesslides_notes_bool {
7703   \newenvironment{nproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
7704 }{
7705   \excludecomment{nproof}
7706 }
```

`nexample (env.)`

```
7707 \bool_if:NTF \c__notesslides_notes_bool {
7708   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}{\end{sexample}}
7709 }{
7710   \excludecomment{nexample}
7711 }
```

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

```
7712 \def\inputref@preskip{\smallskip}
7713 \def\inputref@postskip{\medskip}
```

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

`\inputref*`

```
7714 \let\orig@inputref\inputref
7715 \def\inputref{\@ifstar\ninputref\orig@inputref}
7716 \newcommand\ninputref[2] []{
7717   \bool_if:NT \c__notesslides_notes_bool {
7718     \orig@inputref[#1]{#2}
7719   }
7720 }
```

(End definition for `\inputref*`. This function is documented on page 64.)

40.3 Header and Footer Lines

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

`\setslidelogo` The default logo is the `STEX` logo. Customization can be done by `\setslidelogo{<logo name>}`.

```
7721 \newlength{\slidelogoheight}
7722
7723 \RequirePackage{graphicx}
7724
7725 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7726 \providecommand\mhgraphics[2] []{
7727   \def\Gin@mhrepos{}\setkeys{Gin}{#1}
7728   \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}
7729 }
7730
7731 \bool_if:NTF \c__notesslides_notes_bool {
7732   \setlength{\slidelogoheight}{.4cm}
7733 }{
7734   \setlength{\slidelogoheight}{.25cm}
7735 }
```

```

7736 \ifcsname slidelogo\endcsname\else
7737 \newsavebox{\slidelogo}
7738 \sbox{\slidelogo}{\sTeX}
7739 \fi
7740 \newrobustcmd{\setslidelogo}[2][]{
7741 \tl_if_empty:nTF{#1}{
7742 \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#2}}
7743 }{
7744 \sbox{\slidelogo}{\mhgraphics[height=\slidelogoheight,mhrepos=#1]{#2}}
7745 }
7746 }

```

(End definition for `\setslidelogo`. This function is documented on page 65.)

\author In notes mode, we redefine the `\author` macro so that it does not disregard the optional argument (as `beamerarticle` does). We want to use it to set the source later.

```

7747 \bool_if:NT \c__notesslides_notes_bool {
7748 \def\author{\@dblarg\ns@author}
7749 \long\def\ns@author[#1]#2{%
7750 \def\c__notesslides_shortauthor{#1}%
7751 \def\@author{#2}
7752 }
7753 }

```

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

\setsource `\source` stores the writer's name. By default it is *Michael Kohlhase* since he is the main user and designer of this package. `\setsource{<name>}` can change the writer's name.

```

7754 \newrobustcmd{\setsource}[1]{\def\source{#1}}

```

(End definition for `\setsource`. This function is documented on page 65.)

\setlicensing Now, we set up the copyright and licensing. By default we use the Creative Commons Attribution-ShareAlike license to strengthen the public domain. If package `hyperref` is loaded, then we can attach a hyperlink to the license logo. `\setlicensing[<url>]{<logo name>}` is used for customization, where `<url>` is optional.

```

7755 \def\copyrightnotice{%
7756 \footnotesize\copyright : \hspace{.3ex}%
7757 \ifcsname source\endcsname\source\else%
7758 \ifcsname c__notesslides_shortauthor\endcsname\c__notesslides_shortauthor\else%
7759 \PackageWarning{notesslides}{Author/Source-undefined-in-copyright-notice}%
7760 ?source/author?\fi%
7761 \fi}
7762 \newsavebox{\cclogo}
7763 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{stex-cc_somerights}}
7764 \newif\ifcchref\cchreffalse
7765 \AtBeginDocument{
7766 \@ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
7767 }
7768 \def\licensing{
7769 \ifcchref
7770 \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
7771 \else
7772 {\usebox{\cclogo}}

```

```

7773 \fi
7774 }
7775 \newrobustcmd{\setlicensing}[2][]{
7776   \def\@url{#1}
7777   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
7778   \ifx\@url\@empty
7779     \def\licensing{\usebox{\cclogo}}
7780   \else
7781     \def\licensing{
7782       \ifcchref
7783         \href{#1}{\usebox{\cclogo}}
7784       \else
7785         {\usebox{\cclogo}}
7786       \fi
7787     }
7788   \fi
7789 }

```

(End definition for \setlicensing. This function is documented on page 65.)

EdN:11

```

\slidelabel Now, we set up the slide label for the article mode.11
7790 \newrobustcmd\miko@slidelabel{
7791   \vbox to \slidelogoheight{
7792     \vss\hbox to \slidewidth
7793       {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
7794   }
7795 }

```

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

40.4 Frame Images

\frameimage We have to make sure that the width is overwritten, for that we check the \Gin@ewidth macro from the graphicx package. We also add the label key.

```

7796 \def\Gin@mhrepos{}
7797 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7798 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
7799 \newrobustcmd\frameimage[2][]{
7800   \stepcounter{slide}
7801   \bool_if:NT \c__notesslides_frameimages_bool {
7802     \def\Gin@ewidth{}\setkeys{Gin}{#1}
7803     \bool_if:NF \c__notesslides_notes_bool { \vfill }
7804     \begin{center}
7805       \bool_if:NTF \c__notesslides_fiboxed_bool {
7806         \fbox{
7807           \ifx\Gin@ewidth\@empty
7808             \ifx\Gin@mhrepos\@empty
7809               \mhgraphics[width=\slidewidth,#1]{#2}
7810             \else
7811               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7812             \fi
7813           \else% Gin@ewidth empty

```

¹¹EdNOTE: see that we can use the themes for the slides some day. This is all fake.

```

7814         \ifx\Gin@mhrepos\@empty
7815         \mhgraphics[#1]{#2}
7816     \else
7817         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7818     \fi
7819 \fi% Gin@ewidth empty
7820 }
7821 }{
7822     \ifx\Gin@ewidth\@empty
7823     \ifx\Gin@mhrepos\@empty
7824         \mhgraphics[width=\slidewidth,#1]{#2}
7825     \else
7826         \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7827     \fi
7828     \ifx\Gin@mhrepos\@empty
7829         \mhgraphics[#1]{#2}
7830     \else
7831         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7832     \fi
7833     \fi% Gin@ewidth empty
7834 }
7835 \end{center}
7836 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
7837 \bool_if:NF \c__notesslides_notes_bool { \vfill }
7838 }
7839 } % ifmks@sty@frameimages

```

(End definition for `\frameimage`. This function is documented on page 65.)

40.5 Sectioning

If the `sectocframes` option is set, then we make section frames. We first define counters for `part` and `chapter`, which `beamer.cls` does not have and we make the `section` counter which it does dependent on `chapter`.

```

7840 \stex_html_backend:F {
7841     \bool_if:NT \c__notesslides_sectocframes_bool {
7842         \str_if_eq:VnTF \__notesslidesstopsect{part}{
7843             \newcounter{chapter}\counterwithin*{section}{chapter}
7844         }{
7845             \str_if_eq:VnT\__notesslidesstopsect{chapter}{
7846                 \newcounter{chapter}\counterwithin*{section}{chapter}
7847             }
7848         }
7849     }
7850 }

```

`\section@level` We set the `\section@level` counter that governs sectioning according to the class options. We also introduce the sectioning counters accordingly.

`\section@level`

```

7851 \def\part@prefix{}
7852 \@ifpackageloaded{document-structure}{}{
7853     \str_case:VnF \__notesslidesstopsect {

```



```

7854 {part}{
7855   \int_set:Nn \l_document_structure_section_level_int {0}
7856   \def\thesection{\arabic{chapter}.\arabic{section}}
7857   \def\part@prefix{\arabic{chapter}.}
7858 }
7859 {chapter}{
7860   \int_set:Nn \l_document_structure_section_level_int {1}
7861   \def\thesection{\arabic{chapter}.\arabic{section}}
7862   \def\part@prefix{\arabic{chapter}.}
7863 }
7864 }{
7865   \int_set:Nn \l_document_structure_section_level_int {2}
7866   \def\part@prefix{}
7867 }
7868 }
7869
7870 \bool_if:NF \c__notesslides_notes_bool { % only in slides

```

(End definition for \section@level. This function is documented on page ??.)

The new counters are used in the `sfragment` environment that chooses the L^AT_EX sectioning macros according to `\section@level`.

`sfragment (env.)`

```

7871 \renewenvironment{sfragment}[2][]{
7872   \__document_structure_sfragment_args:n { #1 }
7873   \int_incr:N \l_document_structure_section_level_int
7874   \bool_if:NT \c__notesslides_sectocframes_bool {
7875     \stepcounter{slide}
7876     \begin{frame}[noframenumbering]
7877     \vfill\Large\centering
7878     \red{
7879       \ifcase\l_document_structure_section_level_int\or
7880         \stepcounter{part}
7881         \def\__notesslideslabel{\omdoc@part@kw}~\Roman{part}}
7882         \addcontentsline{toc}{part}{\protect\numberline{\thepart}#2}
7883         \pdfbookmark[0]{\thepart\ #2}{part.\thepart}
7884         \def\currentsectionlevel{\omdoc@part@kw}
7885       \or
7886         \stepcounter{chapter}
7887         \def\__notesslideslabel{\omdoc@chapter@kw}~\arabic{chapter}}
7888         \addcontentsline{toc}{chapter}{\protect\numberline{\thechapter}#2}
7889         \pdfbookmark[1]{\thechapter\ #2}{chapter.\cs_if_exist:cT{\thepart}\thepart.\thechapter}
7890         \def\currentsectionlevel{\omdoc@chapter@kw}
7891       \or
7892         \stepcounter{section}
7893         \def\__notesslideslabel{\part@prefix\arabic{section}}
7894         \addcontentsline{toc}{section}{\protect\numberline{\thesection}#2}
7895         \pdfbookmark[2]{\cs_if_exist:cT{\thechapter}{\thechapter}.\thesection\ #2}
7896         {section.\cs_if_exist:cT{\thepart}{\thepart}.\cs_if_exist:cT{\thechapter}{\thechapter}.\thepart}
7897         \def\currentsectionlevel{\omdoc@section@kw}
7898       \or
7899         \stepcounter{subsection}
7900         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
7901         \addcontentsline{toc}{subsection}{\protect\numberline{\thesubsection}#2}

```

```

7902         \pdfbookmark[3]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection
7903         {subsection.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection}
7904         \def\currentsectionlevel{\omdoc@subsection@kw}
7905     \or
7906         \stepcounter{subsubsection}
7907         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
7908         \addcontentsline{toc}{subsubsection}{\protect\numberline{\thesubsubsection}#2}
7909         \pdfbookmark[4]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection.\thesubsubsection}
7910         {subsubsection.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsubsection}
7911         \def\currentsectionlevel{\omdoc@subsubsection@kw}
7912     \or
7913         \stepcounter{paragraph}
7914         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{paragraph}}
7915         \addcontentsline{toc}{paragraph}{\protect\numberline{\theparagraph}#2}
7916         \pdfbookmark[5]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection.\theparagraph}
7917         {paragraph.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\theparagraph}
7918         \def\currentsectionlevel{\omdoc@paragraph@kw}
7919     \else
7920         \def\__notesslideslabel{}
7921         \def\currentsectionlevel{\omdoc@paragraph@kw}
7922     \fi% end ifcase
7923     \__notesslideslabel\quad #2%
7924 }%
7925 \vfill%
7926 \end{frame}%
7927 }
7928 \str_if_empty:NF \l__document_structure_sfragment_id_str {
7929     \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
7930 }
7931 }{}
7932 }

```

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

```

7933 \def\inserttheorembodyfont{\normalfont}
7934 %\bool_if:NF \c__notesslides_notes_bool {
7935 %   \defbeamertemplate{theorem begin}{miko}
7936 %   {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
7937 %    \ifx\inserttheoremaddition@empty\else\ (\inserttheoremaddition)\fi%
7938 %    \inserttheorempunctuation\inserttheorembodyfont\xspace}
7939 %   \defbeamertemplate{theorem end}{miko}{}}

```

and we set it as the default one.

```

7940 % \setbeamertemplate{theorems}[miko]

```

The following fixes an error I do not understand, this has something to do with beamer compatibility, which has similar definitions but only up to 1.

```

7941 % \expandafter\def\csname Parent2\endcsname{}
7942 %}
7943
7944 \AddToHook{begindocument}{% this does not work for some reason
7945     \setbeamertemplate{theorems}[ams style]
7946 }
7947 \bool_if:NT \c__notesslides_notes_bool {
7948     \renewenvironment{columns}[1][{}]{%

```

```

7949     \par\noindent%
7950     \begin{minipage}%
7951     \slidewidth\centering\leavevmode%
7952   }{%
7953     \end{minipage}\par\noindent%
7954   }%
7955   \newsavebox\columnbox%
7956   \renewenvironment<>{column}[2][]{%
7957     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
7958   }{%
7959     \end{minipage}\end{lrbox}\usebox\columnbox%
7960   }%
7961 }

7962 \bool_if:NTF \c__notesslides_noproblems_bool {
7963   \newenvironment{problems}{}{}
7964 }{
7965   \excludacomment{problems}
7966 }

```

40.6 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.

```

7967 \gdef\printexcursions{}
7968 \newcommand\excursionref[2]{% label, text
7969   \bool_if:NT \c__notesslides_notes_bool {
7970     \begin{sparagraph}[title=Excursion]
7971       #2 \sref[fallback=the appendix]{#1}.
7972     \end{sparagraph}
7973   }
7974 }
7975 \newcommand\activate@excursion[2][{}{
7976   \gappto\printexcursions{\inputref{#1}{#2}}
7977 }
7978 \newcommand\excursion[4][{}{ repos, label, path, text
7979   \bool_if:NT \c__notesslides_notes_bool {
7980     \activate@excursion{#1}{#3}\excursionref{#2}{#4}
7981   }
7982 }

```

(End definition for `\excursion`. This function is documented on page 66.)

\excursiongroup

```

7983 \keys_define:nn{notesslides / excursiongroup }{
7984   id          .str_set_x:N = \l__notesslides_excursion_id_str,
7985   intro       .tl_set:N    = \l__notesslides_excursion_intro_tl,
7986   mhrepos     .str_set_x:N = \l__notesslides_excursion_mhrepos_str
7987 }
7988 \cs_new_protected:Nn \__notesslides_excursion_args:n {
7989   \tl_clear:N \l__notesslides_excursion_intro_tl
7990   \str_clear:N \l__notesslides_excursion_id_str

```

```

7991 \str_clear:N \l__notesslides_excursion_mhrepos_str
7992 \keys_set:nn {notesslides / excursionsgroup }{ #1 }
7993 }
7994 \newcommand\excursionsgroup[1][]{
7995   \__notesslides_excursion_args:n{ #1 }
7996   \ifdefempty\printexcursions{}% only if there are excursions
7997   {\begin{note}
7998     \begin{sfragment}[#1]{Excursions}%
7999     \ifdefempty\l__notesslides_excursion_intro_tl}{
8000       \inputref[\l__notesslides_excursion_mhrepos_str]{
8001         \l__notesslides_excursion_intro_tl
8002       }
8003     }
8004     \printexcursions%
8005     \end{sfragment}
8006   \end{note}}
8007 }
8008 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
8009 \end{package}

```

(End definition for \excursionsgroup. This function is documented on page 66.)

Chapter 41

The Implementation

41.1 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.

```
8010 <*package>
8011 <@@=problems>
8012 \ProvidesExplPackage{problem}{2022/08/08}{3.2.0}{Semantic Markup for Problems}
8013 \RequirePackage{13keys2e}
8014 \RequirePackage{amssymb}% for \Box
8015
8016 \keys_define:nn { problem / pkg }{
8017   notes      .default:n    = { true },
8018   notes      .bool_set:N    = \c__problems_notes_bool,
8019   gnotes     .default:n    = { true },
8020   gnotes     .bool_set:N    = \c__problems_gnotes_bool,
8021   hints      .default:n    = { true },
8022   hints      .bool_set:N    = \c__problems_hints_bool,
8023   solutions  .default:n    = { true },
8024   solutions  .bool_set:N    = \c__problems_solutions_bool,
8025   pts        .default:n    = { true },
8026   pts        .bool_set:N    = \c__problems_pts_bool,
8027   min        .default:n    = { true },
8028   min        .bool_set:N    = \c__problems_min_bool,
8029   boxed      .default:n    = { true },
8030   boxed      .bool_set:N    = \c__problems_boxed_bool,
8031   test       .default:n    = { true },
8032   test       .bool_set:N    = \c__problems_test_bool,
8033   unknown    .code:n       = {
8034     \PassOptionsToPackage{\CurrentOption}{stex}
8035   }
8036 }
8037 \newif\ifsolutions
8038
8039 \ProcessKeysOptions{ problem / pkg }
8040 \bool_if:NTF \c__problems_solutions_bool {
8041   \solutionstrue
```

```

8042 }{
8043   \solutionsfalse
8044 }
8045 \RequirePackage{stex}

```

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

```

8046 \RequirePackage{comment}

```

The next package relies on the L^AT_EX3 kernel, which L^AT_EXML only partially supports. As it is purely presentational, we only load it when the `boxed` option is given and we run L^AT_EXML.

```

8047 \bool_if:NT \c__problems_boxed_bool { \RequirePackage{mdframed} }

```

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

```

8048 \def\prob@problem@kw{Problem}
8049 \def\prob@solution@kw{Solution}
8050 \def\prob@hint@kw{Hint}
8051 \def\prob@note@kw{Note}
8052 \def\prob@grade@kw{Grading}
8053 \def\prob@pt@kw{pt}
8054 \def\prob@min@kw{min}
8055 \def\prob@correct@kw{Correct}
8056 \def\prob@wrong@kw{Wrong}

```

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

For the other languages, we set up triggers

```

8057 \AddToHook{begindocument}{
8058   \ltx@ifpackageloaded{babel}{
8059     \makeatletter
8060     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
8061     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
8062       \input{problem-ngerman.ldf}
8063     }
8064     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
8065       \input{problem-finnish.ldf}
8066     }
8067     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
8068       \input{problem-french.ldf}
8069     }
8070     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
8071       \input{problem-russian.ldf}
8072     }
8073     \makeatother
8074   }{ }
8075 }

```

41.2 Problems and Solutions

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

```

8076 \keys_define:nn{ problem / problem }{
8077   id .str_set_x:N = \l__problems_prob_id_str,

```

```

8078 pts      .tl_set:N      = \l__problems_prob_pts_tl,
8079 min      .tl_set:N      = \l__problems_prob_min_tl,
8080 title    .tl_set:N      = \l__problems_prob_title_tl,
8081 type     .tl_set:N      = \l__problems_prob_type_tl,
8082 imports  .tl_set:N      = \l__problems_prob_imports_tl,
8083 name     .str_set_x:N     = \l__problems_prob_name_str,
8084 refnum   .int_set:N      = \l__problems_prob_refnum_int
8085 }
8086 \cs_new_protected:Nn \__problems_prob_args:n {
8087   \str_clear:N \l__problems_prob_id_str
8088   \str_clear:N \l__problems_prob_name_str
8089   \tl_clear:N \l__problems_prob_pts_tl
8090   \tl_clear:N \l__problems_prob_min_tl
8091   \tl_clear:N \l__problems_prob_title_tl
8092   \tl_clear:N \l__problems_prob_type_tl
8093   \tl_clear:N \l__problems_prob_imports_tl
8094   \int_zero_new:N \l__problems_prob_refnum_int
8095   \keys_set:nn { problem / problem }{ #1 }
8096   \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
8097     \let\l__problems_prob_refnum_int\undefined
8098   }
8099 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

8100 \newcounter{problem}[section]
8101 \newcommand\numberproblemsin[1]{\@addtoreset{problem}{#1}}
8102 \def\theplainsproblem{\arabic{problem}}
8103 \def\thesproblem{\thesection.\theplainsproblem}

```

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

`\prob@label` We provide the macro `\prob@label` to redefine later to get context involved.

```

8104 \newcommand\prob@label[1]{\thesection.#1}

```

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

`\prob@number` We consolidate the problem number into a reusable internal macro

```

8105 \newcommand\prob@number{
8106   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
8107     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
8108   }{
8109     \int_if_exist:NTF \l__problems_prob_refnum_int {
8110       \prob@label{\int_use:N \l__problems_prob_refnum_int }
8111     }{
8112       \prob@label\theplainsproblem
8113     }
8114   }
8115 }
8116 \def\sproblemautorefname{\prob@problem@kw}

```

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

`\prob@title` We consolidate the problem title into a reusable internal macro as well. `\prob@title` takes three arguments the first is the fallback when no title is given at all, the second and third go around the title, if one is given.

```

8117 \newcommand\prob@title[3]{%
8118   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
8119     #2 \l__problems_inclprob_title_tl #3
8120   }{
8121     \tl_if_empty:NTF \l__problems_prob_title_tl {
8122       #1
8123     }{
8124       #2 \l__problems_prob_title_tl #3
8125     }
8126   }
8127 }

```

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

With these the problem header is a one-liner

`\prob@heading` We consolidate the problem header line into a separate internal macro that can be reused in various settings.

```

8128 \def\prob@heading{
8129   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
8130   %\sref@label@id{\prob@problem@kw~\prob@number}{~}
8131 }

```

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

With this in place, we can now define the `problem` environment. It comes in two shapes, depending on whether we are in boxed mode or not. In both cases we increment the problem number and output the points and minutes (depending) on whether the respective options are set.

`sproblem (env.)`

```

8132 \newenvironment{sproblem}[1][]{
8133   \__problems_prob_args:n{#1}%\sref@target%
8134   \@in@omtexttrue% we are in a statement (for inline definitions)
8135   \refstepcounter{sproblem}\record@problem
8136   \def\current@section@level{\prob@problem@kw}
8137
8138   \str_if_empty:NT \l__problems_prob_name_str {
8139     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
8140     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
8141     \seq_get_left:NN \l_tmpa_seq \l__problems_prob_name_str
8142   }
8143
8144   \stex_if_do_html:T{
8145     \tl_if_empty:NF \l__problems_prob_title_tl {
8146       \exp_args:No \stex_document_title:n \l__problems_prob_title_tl
8147     }
8148   }
8149
8150   \exp_args:Nno\stex_module_setup:nn{type=problem}\l__problems_prob_name_str
8151
8152   \stex_reactivate_macro:N \STEXexport
8153   \stex_reactivate_macro:N \importmodule

```



```

8154 \stex_reactivate_macro:N \symdecl
8155 \stex_reactivate_macro:N \notation
8156 \stex_reactivate_macro:N \symdef
8157
8158 \stex_if_do_html:T{
8159   \begin{stex_annotate_env} {problem} {
8160     \l_stex_module_ns_str ? \l_stex_module_name_str
8161   }
8162
8163   \stex_annotate_invisible:nnn{header}{} {
8164     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
8165     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
8166     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
8167       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
8168     }
8169   }
8170 }
8171
8172 \stex_csl_to_imports:No \importmodule \l__problems_prob_imports_tl
8173
8174
8175 \tl_if_exist:NTF \l__problems_inclprob_type_tl {
8176   \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
8177 }{
8178   \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
8179 }
8180 \str_if_exist:NTF \l__problems_inclprob_id_str {
8181   \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
8182 }{
8183   \str_set_eq:NN \sproblemid \l__problems_prob_id_str
8184 }
8185
8186
8187 \stex_if_smsmode:F {
8188   \clist_set:No \l_tmpa_clist \sproblemtype
8189   \tl_clear:N \l_tmpa_tl
8190   \clist_map_inline:Nn \l_tmpa_clist {
8191     \tl_if_exist:cT {__problems_sproblem_##1_start:}{
8192       \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_start:}}
8193     }
8194   }
8195   \tl_if_empty:NTF \l_tmpa_tl {
8196     \__problems_sproblem_start:
8197   }{
8198     \l_tmpa_tl
8199   }
8200 }
8201 \stex_ref_new_doc_target:n \sproblemid
8202 \stex_if_smsmode:TF \stex_smsmode_do: \ignorespacesandpars
8203 }{
8204   \__stex_modules_end_module:
8205   \stex_if_smsmode:F{
8206     \clist_set:No \l_tmpa_clist \sproblemtype
8207     \tl_clear:N \l_tmpa_tl

```

```

8208 \clist_map_inline:Nn \l_tmpa_clist {
8209 \tl_if_exist:cT {__problems_sproblem_##1_end:}{
8210 \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_end:}}
8211 }
8212 }
8213 \tl_if_empty:NTF \l_tmpa_tl {
8214 \__problems_sproblem_end:
8215 }{
8216 \l_tmpa_tl
8217 }
8218 }
8219 \stex_if_do_html:T{
8220 \end{stex_annotate_env}
8221 }
8222
8223 \smallskip
8224 }
8225
8226 \seq_put_right:Nx\g_stex_smsmode_allowedenvs_seq{\tl_to_str:n{sproblem}}
8227
8228
8229
8230 \cs_new_protected:Nn \__problems_sproblem_start: {
8231 \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
8232 }
8233 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
8234
8235 \newcommand\stexpatchproblem[3][] {
8236 \str_set:Nx \l_tmpa_str{ #1 }
8237 \str_if_empty:NTF \l_tmpa_str {
8238 \tl_set:Nn \__problems_sproblem_start: { #2 }
8239 \tl_set:Nn \__problems_sproblem_end: { #3 }
8240 }{
8241 \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
8242 \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
8243 }
8244 }
8245
8246
8247 \bool_if:NT \c__problems_boxed_bool {
8248 \surroundwithmdframed{problem}
8249 }

```

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

```

8250 \def\record@problem{
8251 \protected@write\@auxout{}
8252 {
8253 \string\@problem{\prob@number}
8254 {
8255 \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
8256 \l__problems_inclprob_pts_tl
8257 }{
8258 \l__problems_prob_pts_tl
8259 }

```

```

8260 }%
8261 {
8262   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
8263     \l__problems_inclprob_min_tl
8264   }{
8265     \l__problems_prob_min_tl
8266   }
8267 }
8268 }
8269 }

```

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

`\@problem` This macro acts on a problem's record in the `*.aux` file. It does not have any functionality here, but can be redefined elsewhere (e.g. in the `assignment` package).

```

8270 \def\@problem#1#2#3{}

```

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

`solution (env.)` The `solution` environment is similar to the `problem` environment, only that it is independent of the boxed mode. It also has it's own keys that we need to define first.

```

8271 \keys_define:nn { problem / solution }{
8272   id          .str_set_x:N = \l__problems_solution_id_str ,
8273   for         .str_set_x:N = \l__problems_solution_for_str ,
8274   type       .str_set_x:N = \l__problems_solution_type_str ,
8275   title      .tl_set:N     = \l__problems_solution_title_tl
8276 }
8277 \cs_new_protected:Nn \__problems_solution_args:n {
8278   \str_clear:N \l__problems_solution_id_str
8279   \str_clear:N \l__problems_solution_type_str
8280   \str_clear:N \l__problems_solution_for_str
8281   \tl_clear:N \l__problems_solution_title_tl
8282   \keys_set:nn { problem / solution }{ #1 }
8283 }

```

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

```

8284 \box_new:N \l__problems_solution_box
8285 \newenvironment{solution}[1][{}]{
8286   \__problems_solution_args:n{#1}
8287   \stex_html_backend:TF{
8288     \stex_if_do_html:T{
8289       \begin{stex_annotate_env}{solution}{}
8290       \str_if_empty:NF \l__problems_solution_type_str {
8291         \par\noindent
8292         \stex_annotate_invisible:nnn{typestrings}{\sexampletype}{}
8293       }
8294       \noindent\textbf{Solution}\tl_if_empty:NF\l__problems_solution_title_tl{~(\l__problem
8295     }
8296   }{
8297     \setbox\l__problems_solution_box\vbox\bgroup
8298     \par\smallskip\hrule\smallskip
8299     \noindent\textbf{Solution}\tl_if_empty:NF\l__problems_solution_title_tl{~(\l__problems
8300   }

```

```

8301 }{
8302   \stex_html_backend:TF{
8303     \stex_if_do_html:T{
8304       \end{stex_annotate_env}
8305     }
8306   }{
8307     \smallskip\hrule
8308     \egroup
8309     \bool_if:NT \c__problems_solutions_bool {
8310       \strut\par\noindent
8311       \box\l__problems_solution_box
8312     }
8313   }
8314 }
8315
8316 \newcommand\startsolutions{
8317   \bool_set_true:N \c__problems_solutions_bool
8318   \solutionstrue
8319   % \specialcomment{solution}{\@startsolution}{
8320   %   \bool_if:NF \c__problems_boxed_bool {
8321   %     \hrule\medskip
8322   %   }
8323   %   \end{small}}%
8324   % }
8325   % \bool_if:NT \c__problems_boxed_bool {
8326   %   \surroundwithmdframed{solution}
8327   % }
8328 }

```

(End definition for \startsolutions. This function is documented on page 68.)

\stopsolutions

```

8329 \newcommand\stopsolutions{\bool_set_false:N \c__problems_solutions_bool \solutionsfalse}%\ex

```

(End definition for \stopsolutions. This function is documented on page 68.)

exnote (env.)

```

8330 \bool_if:NTF \c__problems_notes_bool {
8331   \newenvironment{exnote}[1][]{
8332     \par\smallskip\hrule\smallskip
8333     \noindent\textbf{\prob@note@kw :~ }\small
8334   }{
8335     \smallskip\hrule
8336   }
8337 }{
8338   \excludecomment{exnote}
8339 }

```

hint (env.)

```

8340 \bool_if:NTF \c__problems_notes_bool {
8341   \newenvironment{hint}[1][]{
8342     \par\smallskip\hrule\smallskip
8343     \noindent\textbf{\prob@hint@kw :~ }\small
8344   }{

```

```

8345     \smallskip\hrule
8346   }
8347   \newenvironment{exhint}[1][]{
8348     \par\smallskip\hrule\smallskip
8349     \noindent\textbf{\prob@hint@kw :~ }\small
8350   }{
8351     \smallskip\hrule
8352   }
8353   }{
8354     \excludecomment{hint}
8355     \excludecomment{exhint}
8356   }

```

gnote (*env.*)

```

8357   \bool_if:NTF \c__problems_notes_bool {
8358     \newenvironment{gnote}[1][]{
8359       \par\smallskip\hrule\smallskip
8360       \noindent\textbf{\prob@gnote@kw :~ }\small
8361     }{
8362       \smallskip\hrule
8363     }
8364     }{
8365       \excludecomment{gnote}
8366     }

```

41.3 Markup for Added Value Services

41.4 Multiple Choice Blocks

EdN:12

mcb (*env.*)¹²

```

8367   \newenvironment{mcb}{
8368     \begin{enumerate}
8369   }{
8370     \end{enumerate}
8371   }

```

we define the keys for the mcb macro

```

8372   \cs_new_protected:Nn \__problems_do_yes_param:Nn {
8373     \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
8374       \bool_set_true:N #1
8375     }{
8376       \bool_set_false:N #1
8377     }
8378   }
8379   \keys_define:nn { problem / mcb }{
8380     id .str_set_x:N = \l__problems_mcc_id_str ,
8381     feedback .tl_set:N = \l__problems_mcc_feedback_tl ,
8382     T .default:n = { false } ,
8383     T .bool_set:N = \l__problems_mcc_t_bool ,
8384     F .default:n = { false } ,

```

¹²EdNOTE: MK: maybe import something better here from a dedicated MC package

```

8385 F      .bool_set:N    = \l__problems_mcc_f_bool ,
8386 Ttext   .tl_set:N     = \l__problems_mcc_Ttext_tl ,
8387 Ftext   .tl_set:N     = \l__problems_mcc_Ftext_tl
8388 }
8389 \cs_new_protected:Nn \l__problems_mcc_args:n {
8390   \str_clear:N \l__problems_mcc_id_str
8391   \tl_clear:N \l__problems_mcc_feedback_tl
8392   \bool_set_false:N \l__problems_mcc_t_bool
8393   \bool_set_false:N \l__problems_mcc_f_bool
8394   \tl_clear:N \l__problems_mcc_Ttext_tl
8395   \tl_clear:N \l__problems_mcc_Ftext_tl
8396   \str_clear:N \l__problems_mcc_id_str
8397   \keys_set:nn { problem / mcc }{ #1 }
8398 }

```

\mcc

```

8399 \def\mccTrueText{\textbf{\prob@correct@kw!~}}
8400 \def\mccFalseText{\textbf{\prob@wrong@kw!~}}
8401 \newcommand\mcc[2][] {
8402   \l__problems_mcc_args:n{ #1 }
8403   \item[{$\Box$}] #2
8404   \bool_if:NT \c__problems_solutions_bool {
8405     \
8406     \bool_if:NT \l__problems_mcc_t_bool {
8407       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccTrueText\l__problems_mcc_Ttext_tl
8408     }
8409     \bool_if:NT \l__problems_mcc_f_bool {
8410       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccFalseText\l__problems_mcc_Ftext_tl
8411     }
8412     \tl_if_empty:NF \l__problems_mcc_feedback_tl {
8413       \emph{\l__problems_mcc_feedback_tl}
8414     }
8415   }
8416 } %solutions

```

(End definition for \mcc. This function is documented on page 69.)

41.5 Filling in Concrete Solutions

\includeproblem This is embarrassingly simple, but can grow over time.

```

8417 \newcommand\fillinsol[2][] {%
8418   \def\@test{#1}
8419   \quad%
8420   \ifsolutions\textcolor{red}{\#1!}\else%
8421   \fbox{\ifx\@test\@empty\phantom{\huge{21}}\else\hspace{#1}\fi}%
8422   \fi}

```

(End definition for \includeproblem. This function is documented on page 71.)

41.6 Including Problems

\includeproblem The `\includeproblem` command is essentially a glorified `\input` statement, it sets some internal macros first that overwrite the local points. Importantly, it resets the `inclprob` keys after the input.

```

8423
8424 \keys_define:nn{ problem / inclproblem }{
8425   id      .str_set_x:N = \l__problems_inclprob_id_str,
8426   pts     .tl_set:N    = \l__problems_inclprob_pts_tl,
8427   min     .tl_set:N    = \l__problems_inclprob_min_tl,
8428   title   .tl_set:N    = \l__problems_inclprob_title_tl,
8429   refnum  .int_set:N    = \l__problems_inclprob_refnum_int,
8430   type    .tl_set:N    = \l__problems_inclprob_type_tl,
8431   mhrepos .str_set_x:N = \l__problems_inclprob_mhrepos_str
8432 }
8433 \cs_new_protected:Nn \l__problems_inclprob_args:n {
8434   \str_clear:N \l__problems_prob_id_str
8435   \tl_clear:N \l__problems_inclprob_pts_tl
8436   \tl_clear:N \l__problems_inclprob_min_tl
8437   \tl_clear:N \l__problems_inclprob_title_tl
8438   \tl_clear:N \l__problems_inclprob_type_tl
8439   \int_zero_new:N \l__problems_inclprob_refnum_int
8440   \str_clear:N \l__problems_inclprob_mhrepos_str
8441   \keys_set:nn { problem / inclproblem }{ #1 }
8442   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
8443     \let\l__problems_inclprob_pts_tl\undefined
8444   }
8445   \tl_if_empty:NT \l__problems_inclprob_min_tl {
8446     \let\l__problems_inclprob_min_tl\undefined
8447   }
8448   \tl_if_empty:NT \l__problems_inclprob_title_tl {
8449     \let\l__problems_inclprob_title_tl\undefined
8450   }
8451   \tl_if_empty:NT \l__problems_inclprob_type_tl {
8452     \let\l__problems_inclprob_type_tl\undefined
8453   }
8454   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
8455     \let\l__problems_inclprob_refnum_int\undefined
8456   }
8457 }
8458
8459 \cs_new_protected:Nn \l__problems_inclprob_clear: {
8460   \let\l__problems_inclprob_id_str\undefined
8461   \let\l__problems_inclprob_pts_tl\undefined
8462   \let\l__problems_inclprob_min_tl\undefined
8463   \let\l__problems_inclprob_title_tl\undefined
8464   \let\l__problems_inclprob_type_tl\undefined
8465   \let\l__problems_inclprob_refnum_int\undefined
8466   \let\l__problems_inclprob_mhrepos_str\undefined
8467 }
8468 \l__problems_inclprob_clear:
8469
8470 \newcommand\includeproblem[2][ ]{
8471   \l__problems_inclprob_args:n{ #1 }

```

```

8472 \exp_args:No \stex_in_repository:nn\l__problems_inclprob_mhrepos_str{
8473   \stex_html_backend:TF {
8474     \str_clear:N \l_tmpa_str
8475     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
8476       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
8477     }
8478     \stex_annotate_invisible:nnn{includeproblem}{
8479       \l_tmpa_str / #2
8480     }{}
8481   }{
8482     \begingroup
8483     \inputreftrue
8484     \tl_if_empty:nTF{ ##1 }{
8485       \input{#2}
8486     }{
8487       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
8488     }
8489     \endgroup
8490   }
8491 }
8492 \__problems_inclprob_clear:
8493 }

```

(End definition for `\includeproblem`. This function is documented on page 71.)

41.7 Reporting Metadata

For messages it is OK to have them in English as the whole documentation is, and we can therefore assume authors can deal with it.

```

8494 \AddToHook{enddocument}{
8495   \bool_if:NT \c__problems_pts_bool {
8496     \message{Total:~\arabic{pts}~points}
8497   }
8498   \bool_if:NT \c__problems_min_bool {
8499     \message{Total:~\arabic{min}~minutes}
8500   }
8501 }

```

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

```

8502 \def\pts#1{
8503   \bool_if:NT \c__problems_pts_bool {
8504     \marginpar{#1~\prob@pt@kw}
8505   }
8506 }
8507 \def\min#1{
8508   \bool_if:NT \c__problems_min_bool {
8509     \marginpar{#1~\prob@min@kw}
8510   }
8511 }

```

`\show@pts` The `\show@pts` shows the points: if no points are given from the outside and also no points are given locally do nothing, else show and add. If there are outside points then we show them in the margin.


```

8512 \newcounter{pts}
8513 \def\show@pts{
8514   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
8515     \bool_if:NT \c__problems_pts_bool {
8516       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
8517       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
8518     }
8519   }{
8520     \tl_if_exist:NT \l__problems_prob_pts_tl {
8521       \bool_if:NT \c__problems_pts_bool {
8522         \tl_if_empty:NT\l__problems_prob_pts_tl{
8523           \tl_set:Nn \l__problems_prob_pts_tl {0}
8524         }
8525         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
8526         \addtocounter{pts}{\l__problems_prob_pts_tl}
8527       }
8528     }
8529   }
8530 }

```

(End definition for \show@pts. This function is documented on page ??.)
and now the same for the minutes

\show@min

```

8531 \newcounter{min}
8532 \def\show@min{
8533   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
8534     \bool_if:NT \c__problems_min_bool {
8535       \marginpar{\l__problems_inclprob_pts_tl\ min}
8536       \addtocounter{min}{\l__problems_inclprob_min_tl}
8537     }
8538   }{
8539     \tl_if_exist:NT \l__problems_prob_min_tl {
8540       \bool_if:NT \c__problems_min_bool {
8541         \tl_if_empty:NT\l__problems_prob_min_tl{
8542           \tl_set:Nn \l__problems_prob_min_tl {0}
8543         }
8544         \marginpar{\l__problems_prob_min_tl\ min}
8545         \addtocounter{min}{\l__problems_prob_min_tl}
8546       }
8547     }
8548   }
8549 }
8550 \</package>

```

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

41.8 Testing and Spacing

\testspace

```

8551 \newcommand\testspace[1]{\bool_if:NT \c__problems_boxed_bool {\vspace*{#1}}}

```

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

`\testnewpage`

```
8552 \newcommand\testnewpage{\bool_if:NT \c__problems_boxed_bool {\newpage}}
```

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

`\testemptypage`

```
8553 \newcommand\testemptypage[1] [] {%
```

```
8554 \bool_if:NT \c__problems_boxed_bool {\begin{center}\hwexam@testemptypage@kw\end{center}\vfil
```

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

`\test*space`

```
8555 \newcommand\testsmallspace{\testspace{1cm}}
```

```
8556 \newcommand\testmedspace{\testspace{2cm}}
```

```
8557 \newcommand\testbigspace{\testspace{3cm}}
```

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

Chapter 42

Implementation: The hwexam Package

42.1 Package Options

The first step is to declare (a few) package options that handle whether certain information is printed or not. Some come with their own conditionals that are set by the options, the rest is just passed on to the `problems` package.

```
8558 \*package>
8559 \ProvidesExplPackage{hwexam}{2022/08/08}{3.2.0}{homework assignments and exams}
8560 \RequirePackage{13keys2e}
8561
8562 \newif\iftest\testfalse
8563 \DeclareOption{test}{\testtrue\PassOptionsToPackage{\CurrentOption}{problem}}
8564 \newif\ifmultiple\multiplefalse
8565 \DeclareOption{multiple}{\multipletrue}
8566 \DeclareOption{lang}{\PassOptionsToPackage{\CurrentOption}{problem}}
8567 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
8568 \ProcessOptions
```

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

```
8569 \RequirePackage{keyval}[1997/11/10]
8570 \RequirePackage{problem}
```

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

```
8571 \newcommand\hwexam@assignment@kw{Assignment}
8572 \newcommand\hwexam@given@kw{Given}
8573 \newcommand\hwexam@due@kw{Due}
8574 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~blank~for~extra~space}
8575 \newcommand\hwexam@minutes@kw{minutes}
8576 \newcommand\correction@probs@kw{prob.}
8577 \newcommand\correction@pts@kw{total}
8578 \newcommand\correction@reached@kw{reached}
8579 \newcommand\correction@sum@kw{Sum}
8580 \newcommand\correction@grade@kw{grade}
8581 \newcommand\correction@forgrading@kw{To~be~used~for~grading,~do~not~write~here}
```

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

For the other languages, we set up triggers

```

8582 \AddToHook{begindocument}{
8583 \ltx@ifpackageloaded{babel}{
8584 \makeatletter
8585 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
8586 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
8587 \input{hwexam-ngerman.ldf}
8588 }
8589 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
8590 \input{hwexam-finnish.ldf}
8591 }
8592 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
8593 \input{hwexam-french.ldf}
8594 }
8595 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
8596 \input{hwexam-russian.ldf}
8597 }
8598 \makeatother
8599 }{}
8600 }
8601

```

42.2 Assignments

Then we set up a counter for problems and make the problem counter inherited from `problem.sty` depend on it. Furthermore, we specialize the `\prob@label` macro to take the assignment counter into account.

```

8602 \newcounter{assignment}
8603 %\numberproblemsin{assignment}

We will prepare the keyval support for the assignment environment.

8604 \keys_define:nn { hwexam / assignment } {
8605 id .str_set:N = \l_@@_assign_id_str,
8606 number .int_set:N = \l_@@_assign_number_int,
8607 title .tl_set:N = \l_@@_assign_title_tl,
8608 type .tl_set:N = \l_@@_assign_type_tl,
8609 given .tl_set:N = \l_@@_assign_given_tl,
8610 due .tl_set:N = \l_@@_assign_due_tl,
8611 loadmodules .code:n = {
8612 \bool_set_true:N \l_@@_assign_loadmodules_bool
8613 }
8614 }
8615 \cs_new_protected:Nn \_@@_assignment_args:n {
8616 \str_clear:N \l_@@_assign_id_str
8617 \int_set:Nn \l_@@_assign_number_int {-1}
8618 \tl_clear:N \l_@@_assign_title_tl
8619 \tl_clear:N \l_@@_assign_type_tl
8620 \tl_clear:N \l_@@_assign_given_tl
8621 \tl_clear:N \l_@@_assign_due_tl
8622 \bool_set_false:N \l_@@_assign_loadmodules_bool
8623 \keys_set:nn { hwexam / assignment }{ #1 }
8624 }

```

The next three macros are intermediate functions that handle the case gracefully, where the respective token registers are undefined.

The `\given@due` macro prints information about the given and due status of the assignment. Its arguments specify the brackets.

```

8625 \newcommand\given@due[2]{
8626 \bool_lazy_all:nF {
8627 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
8628 { \tl_if_empty_p:V \l_@@_assign_given_tl }
8629 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
8630 { \tl_if_empty_p:V \l_@@_assign_due_tl }
8631 }{ #1 }
8632
8633 \tl_if_empty:NTF \l_@@_inclasssign_given_tl {
8634 \tl_if_empty:NF \l_@@_assign_given_tl {
8635 \hwexam@given@kw\xspace\l_@@_assign_given_tl
8636 }
8637 }{
8638 \hwexam@given@kw\xspace\l_@@_inclasssign_given_tl
8639 }
8640
8641 \bool_lazy_or:nnF {
8642 \bool_lazy_and_p:nn {
8643 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
8644 }{
8645 \tl_if_empty_p:V \l_@@_assign_due_tl
8646 }
8647 }{
8648 \bool_lazy_and_p:nn {
8649 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
8650 }{
8651 \tl_if_empty_p:V \l_@@_assign_due_tl
8652 }
8653 }{ ,~ }
8654
8655 \tl_if_empty:NTF \l_@@_inclasssign_due_tl {
8656 \tl_if_empty:NF \l_@@_assign_due_tl {
8657 \hwexam@due@kw\xspace \l_@@_assign_due_tl
8658 }
8659 }{
8660 \hwexam@due@kw\xspace \l_@@_inclasssign_due_tl
8661 }
8662
8663 \bool_lazy_all:nF {
8664 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
8665 { \tl_if_empty_p:V \l_@@_assign_given_tl }
8666 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
8667 { \tl_if_empty_p:V \l_@@_assign_due_tl }
8668 }{ #2 }
8669 }

```

`\assignment@title` This macro prints the title of an assignment, the local title is overwritten, if there is one from the `\inputassignment`. `\assignment@title` takes three arguments the first is the

fallback when no title is given at all, the second and third go around the title, if one is given.

```

8670 \newcommand\assignment@title[3]{
8671 \tl_if_empty:NTF \l_@@_inclasssign_title_tl {
8672 \tl_if_empty:NTF \l_@@_assign_title_tl {
8673 #1
8674 }{
8675 #2\l_@@_assign_title_tl#3
8676 }
8677 }{
8678 #2\l_@@_inclasssign_title_tl#3
8679 }
8680 }

```

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

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

```

8681 \newcommand\assignment@number{
8682 \int_compare:nNnTF \l_@@_inclasssign_number_int = {-1} {
8683 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8684 \arabic{assignment}
8685 } {
8686 \int_use:N \l_@@_assign_number_int
8687 }
8688 }{
8689 \int_use:N \l_@@_inclasssign_number_int
8690 }
8691 }

```

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

With them, we can define the central **assignment** environment. This has two forms (separated by \ifmultiple) in one we make a title block for an assignment sheet, and in the other we make a section heading and add it to the table of contents. We first define an assignment counter

assignment (env.) For the **assignment** environment we delegate the work to the **@assignment** environment that depends on whether **multiple** option is given.

```

8692 \newenvironment{assignment}[1][]{
8693 \_@@_assignment_args:n { #1 }
8694 %\sref@target
8695 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8696 \global\stepcounter{assignment}
8697 }{
8698 \global\setcounter{assignment}{\int_use:N\l_@@_assign_number_int}
8699 }
8700 \setcounter{sproblem}{0}
8701 \renewcommand\prob@label[1]{\assignment@number.##1}
8702 \def\current@section@level{\document@hwexamtype}
8703 %\sref@label{id}{\document@hwexamtype \thesection}
8704 \begin{@assignment}
8705 }{
8706 \end{@assignment}
8707 }

```

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

```

8708 \def\ass@title{
8709 {\protect\document@hwexamtype}\arabic{assignment}
8710 \assignment@title{}\;{}{}\;} -- \given@due{}\}
8711 }
8712 \ifmultiple
8713 \newenvironment{@assignment}{
8714 \bool_if:NTF \l_@@_assign_loadmodules_bool {
8715 \begin{sfragment}[loadmodules]{\ass@title}
8716 }{
8717 \begin{sfragment}{\ass@title}
8718 }
8719 }{
8720 \end{sfragment}
8721 }

```

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

```

8722 \else
8723 \newenvironment{@assignment}{
8724 \begin{center}\bf
8725 \Large@title\strut\
8726 \document@hwexamtype}\arabic{assignment}\assignment@title{}\;{}{}\;{}\\}
8727 \large\given@due{--\;}\;{};--}
8728 \end{center}
8729 }{}
8730 \fi% multiple

```

42.3 Including Assignments

\in*assignment This macro is essentially a glorified `\include` statement, it just sets some internal macros first that overwrite the local points. Importantly, it resets the `inclassig` keys after the input.

```

8731 \keys_define:nn { hwexam / inclassignment } {
8732 %id .str_set_x:N = \l_@@_assign_id_str,
8733 number .int_set:N = \l_@@_inclassign_number_int,
8734 title .tl_set:N = \l_@@_inclassign_title_tl,
8735 type .tl_set:N = \l_@@_inclassign_type_tl,
8736 given .tl_set:N = \l_@@_inclassign_given_tl,
8737 due .tl_set:N = \l_@@_inclassign_due_tl,
8738 mhrepos .str_set_x:N = \l_@@_inclassign_mhrepos_str
8739 }
8740 \cs_new_protected:Nn \l_@@_inclassignment_args:n {
8741 \int_set:Nn \l_@@_inclassign_number_int {-1}
8742 \tl_clear:N \l_@@_inclassign_title_tl
8743 \tl_clear:N \l_@@_inclassign_type_tl
8744 \tl_clear:N \l_@@_inclassign_given_tl
8745 \tl_clear:N \l_@@_inclassign_due_tl
8746 \str_clear:N \l_@@_inclassign_mhrepos_str
8747 \keys_set:nn { hwexam / inclassignment }{ #1 }
8748 }
8749 \l_@@_inclassignment_args:n {}
8750
8751 \newcommand\inputassignment[2][]{

```

```

8752 \_@@_inclassassignment_args:n { #1 }
8753 \str_if_empty:NTF \l_@@_inclasssign_mhrepos_str {
8754   \input{#2}
8755 }{
8756   \stex_in_repository:nn{\l_@@_inclasssign_mhrepos_str}{
8757     \input{\mhpath{\l_@@_inclasssign_mhrepos_str}{#2}}
8758   }
8759 }
8760 \_@@_inclassassignment_args:n {}
8761 }
8762 \newcommand\includeassignment[2][]{
8763   \newpage
8764   \inputassignment[#1]{#2}
8765 }

```

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

42.4 Typesetting Exams

\quizheading

```

8766 \ExplSyntaxOff
8767 \newcommand\quizheading[1]{%
8768   \def\@tas{#1}%
8769   \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
8770   \ifx\@tas\@empty\else%
8771     \noindent TA:~\@for\@I:=\@tas\do{{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
8772   \fi%
8773 }
8774 \ExplSyntaxOn

```

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

\testheading

```

8775
8776 \def\hwexamheader{\input{hwexam-default.header}}
8777
8778 \def\hwexamminutes{
8779   \tl_if_empty:NTF \testheading@duration {
8780     {\testheading@min}~\hwexam@minutes@kw
8781   }{
8782     \testheading@duration
8783   }
8784 }
8785
8786 \keys_define:nn { hwexam / testheading } {
8787   min .tl_set:N = \testheading@min,
8788   duration .tl_set:N = \testheading@duration,
8789   reqpts .tl_set:N = \testheading@reqpts,
8790   tools .tl_set:N = \testheading@tools
8791 }
8792 \cs_new_protected:Nn \_@@_testheading_args:n {
8793   \tl_clear:N \testheading@min
8794   \tl_clear:N \testheading@duration

```



```

8795 \tl_clear:N \testheading@reqpts
8796 \tl_clear:N \testheading@tools
8797 \keys_set:nn { hwexam / testheading }{ #1 }
8798 }
8799 \newenvironment{testheading}[1][ ]{
8800 \_@@_testheading_args:n{ #1 }
8801 \newcount\check@time\check@time=\testheading@min
8802 \advance\check@time by -\theassignment@totalmin
8803 \newif\if@bonuspoints
8804 \tl_if_empty:NTF \testheading@reqpts {
8805 \@bonuspointsfalse
8806 }{
8807 \newcount\bonus@pts
8808 \bonus@pts=\theassignment@totalpts
8809 \advance\bonus@pts by -\testheading@reqpts
8810 \edef\bonus@pts{\the\bonus@pts}
8811 \@bonuspointstrue
8812 }
8813 \edef\check@time{\the\check@time}
8814
8815 \makeatletter\hwexamheader\makeatother
8816 }{
8817 \newpage
8818 }

```

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

\@problem This macro acts on a problem's record in the *.aux file. Here we redefine it (it was defined to do nothing in problem.sty) to generate the correction table.

```

8819 <@@=problems>
8820 \renewcommand\@problem[3]{
8821 \stepcounter{assignment@probs}
8822 \def\__problemspts{#2}
8823 \ifx\__problemspts\empty\else
8824 \addtocounter{assignment@totalpts}{#2}
8825 \fi
8826 \def\__problemsmin{#3}\ifx\__problemsmin\empty\else\addtocounter{assignment@totalmin}{#3}\fi
8827 \xdef\correction@probs{\correction@probs & #1}%
8828 \xdef\correction@pts{\correction@pts & #2}
8829 \xdef\correction@reached{\correction@reached &}
8830 }
8831 <@@=hwexam>

```

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

\correction@table This macro generates the correction table

```

8832 \newcounter{assignment@probs}
8833 \newcounter{assignment@totalpts}
8834 \newcounter{assignment@totalmin}
8835 \def\correction@probs{\correction@probs@kw}
8836 \def\correction@pts{\correction@pts@kw}
8837 \def\correction@reached{\correction@reached@kw}
8838 \stepcounter{assignment@probs}
8839 \newcommand\correction@table{

```

```

8840 \resizebox{\textwidth}{!}{%
8841 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
8842 &\multicolumn{\theassignment@probs}{c|}{%|
8843 {\footnotesize\correction@forgrading@kw} &\\ \hline
8844 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
8845 \correction@pts & \theassignment@totalpts & \\ \hline
8846 \correction@reached & & \[.7cm]\hline
8847 \end{tabular}}
8848 \end{package}

```

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

42.5 Leftovers

at some point, we may want to reactivate the logos font, then we use

```

here we define the logos that characterize the assignment
\font\bierfont=../assignments/bierglas
\font\denkerfont=../assignments/denker
\font\uhrfont=../assignments/uhr
\font\warnschildfont=../assignments/achtung

\newcommand\bierglas{{\bierfont\char65}}
\newcommand\denker{{\denkerfont\char65}}
\newcommand\uhr{{\uhrfont\char65}}
\newcommand\warnschild{{\warnschildfont\char 65}}
\newcommand\hardA{\warnschild}
\newcommand\longA{\uhr}
\newcommand\thinkA{\denker}
\newcommand\discussA{\bierglas}

```

Chapter 43

References

EdN:13

13

- [Bus+04] Stephen Buswell et al. *The Open Math Standard, Version 2.0*. Tech. rep. The OpenMath Society, 2004. URL: <http://www.openmath.org/standard/om20>.
- [CR99] David Carlisle and Sebastian Rathz. *The graphicx package*. Part of the T_EX distribution. The Comprehensive T_EX Archive Network. 1999. URL: <https://www.tug.org/texlive/devsrc/Master/texmf-dist/doc/latex/graphics/graphics.pdf>.
- [DCM03] The DCMI Usage Board. *DCMI Metadata Terms*. DCMI Recommendation. Dublin Core Metadata Initiative, 2003. URL: <http://dublincore.org/documents/dcmi-terms/>.
- [Koh06] Michael Kohlhase. *OMDoc – An open markup format for mathematical documents [Version 1.2]*. LNAI 4180. Springer Verlag, Aug. 2006. URL: <http://omdoc.org/pubs/omdoc1.2.pdf>.
- [LMH] *LMH Scripts*. URL: <https://github.com/sLaTeX/lmhtools>.
- [MMT] *MMT – Language and System for the Uniform Representation of Knowledge*. Project web site. URL: <https://uniformal.github.io/> (visited on 01/15/2019).
- [MRK18] Dennis Müller, Florian Rabe, and Michael Kohlhase. “Theories as Types”. In: *9th International Joint Conference on Automated Reasoning*. Ed. by Didier Galmiche, Stephan Schulz, and Roberto Sebastiani. Springer Verlag, 2018. URL: <https://kwarc.info/kohlhase/papers/ijcar18-records.pdf>.
- [Rab15] Florian Rabe. “The Future of Logic: Foundation-Independence”. In: *Logica Universalis* 10.1 (2015). 10.1007/s11787-015-0132-x; Winner of the Contest “The Future of Logic” at the World Congress on Universal Logic, pp. 1–20.
- [RK13] Florian Rabe and Michael Kohlhase. “A Scalable Module System”. In: *Information & Computation* 0.230 (2013), pp. 1–54. URL: <https://kwarc.info/frabe/Research/mmt.pdf>.
- [RT] *sLaTeX/RusTeX*. URL: <https://github.com/sLaTeX/RusTeX> (visited on 04/22/2022).

¹³EdNOTE: we need an un-numbered version sfragment*

- [ST] *sTeX – An Infrastructure for Semantic Preloading of LaTeX Documents*. URL: <https://ctan.org/pkg/stex> (visited on 04/22/2022).
- [sTeX] *sTeX: A semantic Extension of TeX/LaTeX*. URL: <https://github.com/sLaTeX/sTeX> (visited on 05/11/2020).
- [Tana] Till Tantau. *beamer – A LaTeX class for producing presentations and slides*. URL: <http://ctan.org/pkg/beamer> (visited on 01/07/2014).
- [Tanb] Till Tantau. *User Guide to the Beamer Class*. URL: <http://ctan.org/macros/latex/contrib/beamer/doc/beameruserguide.pdf>.
- [TL] *TeX Live*. URL: <http://www.tug.org/texlive/> (visited on 12/11/2012).