

The sTeX3 Package Collection *

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

2022-09-27

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 2022-09-27)

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	13
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 <code>smodule</code> -Environment	18
5.3.2	Declaring New Symbols and Notations	20
	Operator Notations	24
5.3.3	Argument Modes	24
	Mode- <code>b</code> Arguments	24
	Mode- <code>a</code> Arguments	25
	Mode- <code>B</code> 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 <code>mathstructure</code> Environment	36
5.4.4	The <code>copymodule</code> 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	sTeX-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	sTeX-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	sTeX-References	81
12.1	Macros and Environments	81
12.1.1	Setting Reference Targets	81
12.1.2	Using References	82
13	sTeX-Modules	83
13.1	Macros and Environments	83
13.1.1	The <code>smodule</code> environment	85
14	sTeX-Module Inheritance	87
14.1	Macros and Environments	87
14.1.1	SMS Mode	87
14.1.2	Imports and Inheritance	88
15	sTeX-Symbols	90
15.1	Macros and Environments	90
16	sTeX-Terms	92
16.1	Macros and Environments	92
17	sTeX-Structural Features	94
17.1	Macros and Environments	94
17.1.1	Structures	94
18	sTeX-Statements	95
18.1	Macros and Environments	95
19	sTeX-Proofs: Structural Markup for Proofs	96
20	sTeX-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 L^AT_EX	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	154
30.1	SMS Mode	154
30.2	Inheritance	159
31	STEX-Symbols Implementation	164
31.1	Symbol Declarations	164
31.2	Notations	172
31.3	Variables	182
32	STEX-Terms Implementation	191
32.1	Symbol Invocations	191
32.2	Terms	199
32.3	Notation Components	204
32.4	Variables	206
32.5	Sequences	209

33	STEX-Structural Features Implementation	210
33.1	Imports with modification	211
33.2	The feature environment	219
33.3	Structure	219
34	STEX-Statements Implementation	231
34.1	Definitions	231
34.2	Assertions	237
34.3	Examples	240
34.4	Logical Paragraphs	243
35	The Implementation	248
35.1	Proofs	248
36	STEX-Others Implementation	257
37	STEX-Metatheory Implementation	259
38	Tikzinput Implementation	262
39	document-structure.sty Implementation	265
39.1	Package Options	265
39.2	Document Structure	266
39.3	Front and Backmatter	270
39.4	Global Variables	272
40	NotesSlides – Implementation	273
40.1	Class and Package Options	273
40.2	Notes and Slides	275
40.3	Header and Footer Lines	279
40.4	Frame Images	281
40.5	Sectioning	282
40.6	Excursions	285
41	The Implementation	287
41.1	Package Options	287
41.2	Problems and Solutions	288
41.3	Markup for Added Value Services	295
41.4	Multiple Choice Blocks	295
41.5	Filling in Concrete Solutions	296
41.6	Including Problems	297
41.7	Reporting Metadata	298
41.8	Testing and Spacing	299
42	Implementation: The hwexam Package	301
42.1	Package Options	301
42.2	Assignments	302
42.3	Including Assignments	305
42.4	Typesetting Exams	306
42.5	Leftovers	308

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 L^AT_EX 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 T_EXLive on your system as a L^AT_EX enthusiast. If not now is the time to install it; see [TL]. You can usually update T_EXLive via a package manager or the T_EXLive manager **tlmgr**. sTeX requires a T_EX kernel newer than February 2022.

Alternatively, you can install sTeX from CTAN, the Comprehensive T_EX 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.

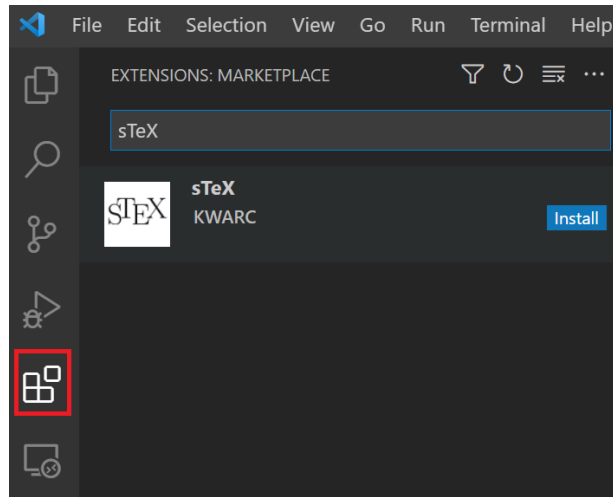


Figure 1: Installing the sTeX extension for VSCode

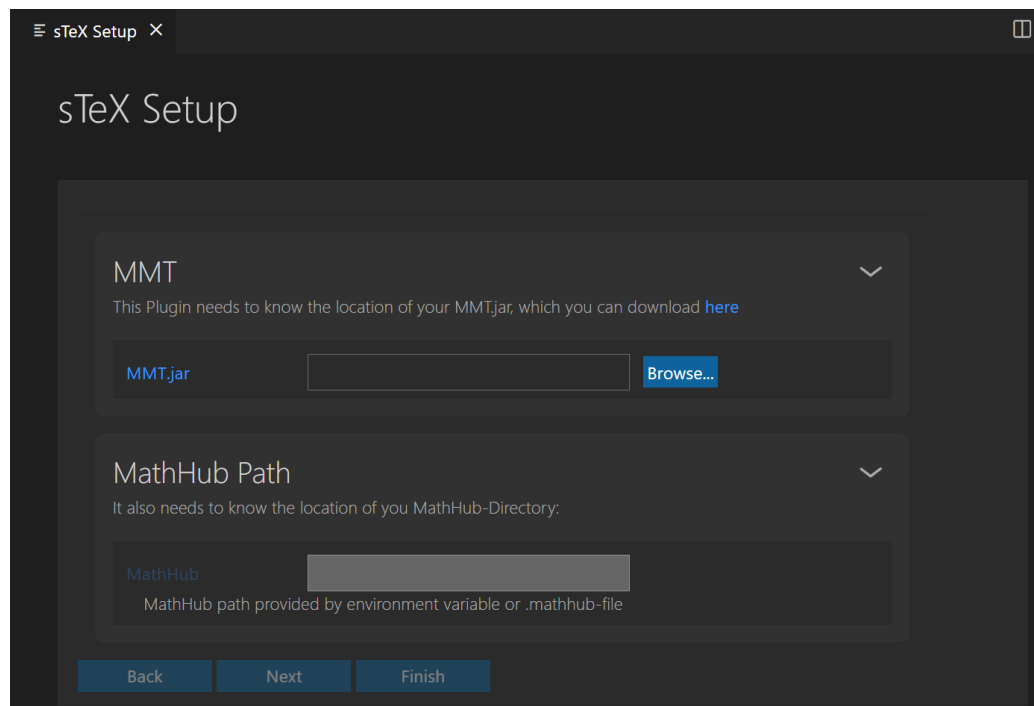


Figure 2: sTeX Setup Routine

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/smgloom>.

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/smgloom/<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 smgloom` will download all `smgloom` 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 \definame{geometricSeries} is the \symname{series}
13     \[\defeq{\geometricSeries}{\definiens{
14       \infinitesum{\svar{n}}{1}{
15         \realdivide[frac]{1}{
16           \realpower{2}{\svar{n}}
17         }
18       }}
19     \end{sdefinition}
20
21     \begin{sassertion}[name=geometricSeriesConverges,type=theorem]
22       The \symname{geometricSeries} \symname{converges} towards $1$.
23     \end{sassertion}
24 \end{smodule}
25 \end{document}
```

Compiling this document with pdf_lat_{ex} 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[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 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 `\symsdecl{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
```

teaser: Terminology for the mathematical study of change. description: desc.html

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

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

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

Additionally to translating the informal content of a module file to different languages, it also allows for customizing notations between languages. For example, the *least common multiple* of two numbers is often denoted as `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     \renamedekl[name=universe]{universe}{runiverse}
4     \renamedekl[name=plus]{operation}{rplus}
5     \renamedekl[name=zero]{unit}{rzero}
6     \renamedekl[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    \renamedekl[name=times]{operation}{rtimes}
14    \renamedekl[name=one]{unit}{rone}
15  \end{copymodule}
16  \notation*{rtimes}[cdot,op=\cdot,prec=50]{#1 \comp\cdot #2}
17  \notation*{rone}[one]{\comp1}
18  Test: $\rtimes a\{rplus c\{rtimes de\}}$
19 \end{smodule}

```

Output:

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

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}]
```

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

ST_EX 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   $\addition{2,3}$ is $5$, $\multiplication{2,3}$ 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 `\Definame` 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   \syndef{universe}[type=\set]{\comp{U}}
3   \syndef{op}[
4     args=2,
5     type=\funtype{\universe,\universe}{\universe},
6     op=\circ
7   ]{#1 \comp{\circ} #2}
8   \syndef{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-

sulting 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{spfblock}\end{subproof}
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.

<code>\sproofend</code>	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: <div style="margin-left: 20px;">The <code>stex-proofs</code> package provides the <code>\sproofend</code> macro for this.</div>
-------------------------	---

<code>\sProofEndSymbol</code>	If a different symbol for the proof end is to be used (e.g. <i>q.e.d</i>), then this can be obtained by specifying it using the <code>\sProofEndSymbol</code> configuration macro (e.g. by specifying <code>\sProofEndSymbol{q.e.d}</code>). 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 <code>proofend={}</code> in them or use <code>\sProofEndSymbol{}</code> .
-------------------------------	---

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 L^AT_EX 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 S_TE_X 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*`.

<code>\stexpatchmodule</code> <code>\stexpatchdefinition</code> <code>\stexpatchassertion</code> <code>\stexpatchexample</code> <code>\stexpatchparagraph</code> <code>\stexpatchproof</code>	All of these commands take one optional and two proper arguments, i.e. <code>\stexpatch* [<type>] {<begin-code>} {<end-code>}</code> . After S _T E _X reads and processes the optional arguments for these environments, (some of) their values are stored in the macros <code>\s*<field></code> (i.e. <code>sexampleid</code> , <code>\sassertionname</code> , etc.). It then checks for all the values <code><type></code> in the <code>type=</code> -list, whether an <code>\stexpatch* [<type>]</code> for the current environment has been called. If it finds one, it uses the patches <code><begin-code></code> and <code><end-code></code> to mark up the current environment. If no patch for (any of) the type(s) is found, it checks whether and <code>\stexpatch*</code> 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 \sTEX3 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 `gTEX3` documentation” everywhere else. This is achieved using

```
\sref[file=./stex-doc]{part:extends}[in=./stex-doc,title={the full \sTEX3 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 `ikzinput` package is determined by whether the `image` option is given. If it is not, then the `tikz` package is loaded, all other options are passed on to it and `\tikzinput{⟨file⟩}` inputs the TIKZ file `⟨file⟩.tex`; if not, only the `graphicx` package is loaded and `\tikzinput{⟨file⟩}` loads an image file `⟨file⟩.⟨ext⟩` generated from `⟨file⟩.tex`.

The selective input functionality of the `tikzinput` package assumes that the TIKZ pictures are externalized into a standalone picture file, such as the following one

```
1 \documentclass{standalone}
2 \usepackage{tikz}
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>\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> <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.
--	--

9.2.4 Ending Documents Prematurely

<hr/> <code>\prematurestop</code> <code>\afterprematurestop</code> <hr/>	For prematurely stopping the formatting of a document, <code>TeX</code> 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.
---	---

`\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> <code>\useSGvar</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.
--	---

<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> <code>notes</code>	The options <code>slides</code> and <code>notes</code> switch between slides mode and notes mode (see subsection 9.3.3).
<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> <code>fiboxed</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.

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

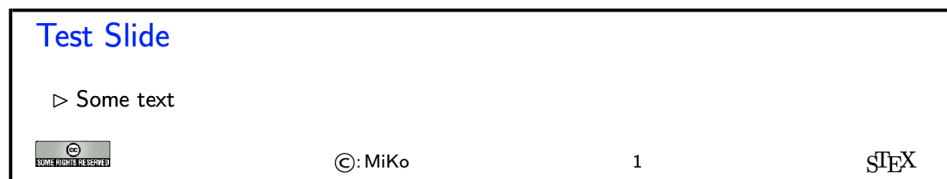
`nexample` (*env.*) level of source indenting). Similarly, we have the `nfragment`, `ndefinition`, `nexample`,

`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 `beamterthemesTeX`. It is assumed as the default theme for `sTeX`-based notes and slides. The result in `notes` mode (which is like the `slides` version except that the slide 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 problem package takes the options solutions (should solutions be output?), notes
notes	(should the problem notes be presented?), hints (do we give the hints?), gnotes (do we
hints	show grading notes?), pts (do we display the points awarded for solving the problem?),
gnotes	min (do we display the estimated minutes for problem soling). If theses are specified, then
pts	the corresponding auxiliary parts of the problems are output, otherwise, they remain
min	invisible.
boxed	The boxed option specifies that problems should be formatted in framed boxes so
test	that they are more visible in the text. Finally, the test option signifies that we are in

a test situation, so this option does not show the solutions (of course), but leaves space for the students to solve them.

problem (*env.*) The main environment provided by the **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`,
`notes` `min`, and `boxed` that are just passed on to the `problems` package (cf. its documentation
`hints` for a description of the intended behavior).
`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)

2022-09-27

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
<code>\stex_path_to_string:N</code>	leaves it in the input stream.

<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> *	
<code>\stex_path_if_absolute:N\underline{T}</code> *	

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

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

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

`name` The *name* of the module,

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

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

`lang` the module's *language*,

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

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

`meta` the metatheory of the module.

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

`\c_stex_module_<URI>_constants`

The names of all constants declared in the module

`\c_stex_module_<URI>_constants`

The full URIs of all modules imported in this module

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

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

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

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

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

`\stex_add_to_current_module:n`
`\STEXexport`

Adds the provided tokens to the `_code` control sequence of the current module.

`\stex_add_to_current_module:n` is used internally, `\STEXexport` is intended for users and additionally executes the provided code immediately.

`\stex_add_constant_to_current_module:n`

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

`\stex_add_import_to_current_module:n`

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

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

`\stex_do_up_to_module:n` Code that is *exported* from module (such as symbol declarations) should be local *to the current module*. For that reason, ideally all symbol declarations and similar commands should be called directly in the module environment, however, that is not always feasible, e.g. in structural features or `sparapraphs`. `\stex_do_up_to_module` therefore executes the provided code repeatedly in an `\aftergroup` up until the group level is equal to that of the innermost smodule environment.

\stex_modules_current_namespace:

Computes the current namespace as follows:

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

The result is stored in `\l_stex_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.

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

Chapter 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

<code>\stex_import_module_uri:nn</code>	<code>\stex_import_module_uri:nn {<archive-ID>} {<module-path>}</code>
---	--

Determines the URI of a module by splitting `<module-path>` into `<path>?<name>`. If `<module-path>` does *not* contain a `?`-character, we consider it to be the `<name>`, and `<path>` to be empty.

If `<archive-ID>` is empty, it is automatically set to the ID of the current archive (if one exists).

1. If `<archive-ID>` is empty:

- (a) If `<path>` is empty, then `<name>` must have been declared earlier in the same file and retrievable from `\g_stex_modules_in_file_seq`, or a file with name `<name>.<lang>.tex` must exist in the same folder, containing a module `<name>`.

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

- (b) If `<path>` is not empty, it must point to the relative path of the containing file as well as the namespace.

2. Otherwise:

- (a) If `<path>` is empty, then `<name>` must have been declared earlier in the same file and retrievable from `\g_stex_modules_in_file_seq`, or a file with name `<name>.<lang>.tex` must exist in the top `source` folder of the archive, containing a module `<name>`.

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

- (b) If `<path>` is not empty, it must point to the path of the containing file as well as the namespace, relative to the namespace of the archive.

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

<code>\l_stex_import_name_str</code>	stores the result in these four variables.
<code>\l_stex_import_archive_str</code>	
<code>\l_stex_import_path_str</code>	
<code>\l_stex_import_ns_str</code>	

<code>\stex_import_require_module:nnnn</code>	<code>{<ns>} {<archive-ID>} {<path>} {<name>}</code>
---	--

Checks whether a module with URI `<ns>?<name>` already exists. If not, it looks for a plausible file that declares a module with that URI.

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

Chapter 15

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

Code related to symbol declarations and notations

15.1 Macros and Environments

$\backslash\text{symdecl}$	$\backslash\text{symdecl}\{\langle\text{macroname}\rangle\}[\langle\text{args}\rangle]$
----------------------------	---

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

- **name**: An (OMDOC) name. By default equal to $\langle\text{macroname}\rangle$.
- **type**: An (ideally semantic) term, representing a *type*. Not used by $\text{\texttt{S}\text{T}\text{E}\text{X}}$, but passed on to MMT for semantic services.
- **def**: An (ideally semantic) term, representing a *definiens*. Not used by $\text{\texttt{S}\text{T}\text{E}\text{X}}$, 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. $\backslash\text{symdecl}\{\text{plus}\}[\text{args=ii}]$ allows for $\backslash\text{plus}\{2\}\{2\}$.
 - a** an *associative* argument; i.e. a sequence of arbitrarily many arguments provided as a comma-separated list, e.g. $\backslash\text{symdecl}\{\text{plus}\}[\text{args=a}]$ allows for $\backslash\text{plus}\{2,2,2\}$.
 - b** a *variable* argument. Is treated by $\text{\texttt{S}\text{T}\text{E}\text{X}}$ like an **i**-argument, but an application is turned into an $\text{\texttt{O}\text{M}\text{B}\text{ind}}$ in OMDOC, binding the provided variable in the subsequent arguments of the operator; e.g. $\backslash\text{symdecl}\{\text{forall}\}[\text{args=bi}]$ allows for $\backslash\text{forall}\{x\}\text{in}\backslash\text{Nat}\}\{x\geq 0\}$.

<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>. 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> 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> Implements the core functionality of <code>\notation</code>, and is called by <code>\notation</code> and <code>\symdef</code>. 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> 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

<code>\STEXsymbol</code>	Uses <code>\stex_get_symbol:n</code> to find the symbol denoted by the first argument and passes the result on to <code>\stex_invoke_symbol:n</code>
--------------------------	--

<code>\symref</code>	<code>\symref{<symbol>}{<text>}</code> shortcut for <code>\STEXsymbol{<symbol>}! [<text>]</code>
----------------------	---

<code>\stex_invoke_symbol:n</code>	Executes a semantic macro. Outside of math mode or if followed by <code>*</code> , it continues to <code>\stex_term_custom:nn</code> . 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}`.

<code>\STEXInternalTermMathOMSiiii</code>	<code><URI><fragment><precedence><body></code>
<code>\STEXInternalTermMathOMAiiai</code>	
<code>\STEXInternalTermMathOMBiiii</code>	

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.

<code>\STEXInternalTermMathArgiii</code>	<code>\stex_term_arg:nnn<int><prec><body></code>
--	--

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

ST_EX-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}{2022/09/14}{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   \exp_args:NNo \stex_set_language:Nn \l_tmpa_str \l_tmpa_str
40 }
41 }
42 }
43 </cls>

```

26.2 Preliminaries

```

44 <*package>
45
46 %%%%%%%%% basics.dtx %%%%%%%%%
47
48 \RequirePackage{expl3,l3keys2e,ltxcmds}
49 \ProvidesExplPackage{stex}{2022/09/14}{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 \let\STEXInternalAnnotate\stex_annotate:nnn
387
388 \cs_new_protected:Nn \stex_par: {
389   \mode_if_vertical:F{
390     \if@minipage\else\if@nobreak\else\par\fi\fi
391   }
392 }
393
394 \cs_new_protected:Nn \__stex_persist_patchcounter:n{
395   \cs_set_eq:cc{__stex_persist_tmp_#1}{@#1}
396   \cs_set:cpn {@#1} ##1 {
397     \STEXInternalAnnotate{counter}{
398       \expandafter\expandafter\expandafter
399       \expandafter\expandafter\expandafter
400       \expandafter\@gobble
401       \expandafter\expandafter\expandafter\@gobble
402       \expandafter\@gobble\detokenize{##1},
403       #1,\number##1}{\use:c{__stex_persist_tmp_#1}{##1}}
404   }
405 }
406
407 \cs_new_protected:Nn \stex_patch_counters: {
408   \__stex_persist_patchcounter:n{arabic}
409   \__stex_persist_patchcounter:n{roman}
410   \__stex_persist_patchcounter:n{Roman}
411   \__stex_persist_patchcounter:n{alph}
412   \__stex_persist_patchcounter:n{Alph}
413   \__stex_persist_patchcounter:n{fnsymbol}
414   \let\__stex_persist_tmp_refstepcounter\refstepcounter
415   \cs_set:Npn\refstepcounter##1{
416     \__stex_persist_tmp_refstepcounter{##1}
417     \STEXInternalAnnotate{stepcounter}{##1}{ }
418   }
419 }
420
421 \cs_new_protected:Nn \stex_unpatch_counters: {
422   \let\@arabic\__stex_persist_tmp_arabic
423   \let\@roman\__stex_persist_tmp_roman
424   \let\@Roman\__stex_persist_tmp_Roman
425   \let\@alph\__stex_persist_tmp_alph
426   \let\@Alph\__stex_persist_tmp_Alph
427   \let\@fnsymbol\__stex_persist_tmp_fnsymbol
428   \let\refstepcounter\__stex_persist_tmp_refstepcounter
429 }
430
431 %\AtBeginDocument{
432 %}
433
434 </package>

```

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

Chapter 27

STEX -MathHub Implementation

```
435 <*package>
436
437 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
438
439 <@@=stex_path>
440
441 Warnings and error messages
442 \msg_new:nnn{stex}{error/norepository}{
443   No~archive~#1~found~in~#2
444 }
445 \msg_new:nnn{stex}{error/notinarchive}{
446   Not~currently~in~an~archive,~but~\detokenize{#1}~
447   needs~one!
448 }
449 \msg_new:nnn{stex}{error/nofile}{
450   \detokenize{#1}~could~not~find~file~#2
451 }
452 \msg_new:nnn{stex}{error/twofiles}{
453   \detokenize{#1}~found~two~candidates~for~#2
454 }
```

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`

```
453 \cs_new_protected:Nn \stex_path_from_string:Nn {
454   \stex_debug:nn{files}{#2}
455   \str_set:Nx \l_tmpa_str { #2 }
456   \str_if_empty:NTF \l_tmpa_str {
457     \seq_clear:N #1
458   }{
459     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
460     \sys_if_platform_windows:T{
```

```

461     \seq_clear:N \l_tmpa_tl
462     \seq_map_inline:Nn #1 {
463         \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
464         \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
465     }
466     \seq_set_eq:NN #1 \l_tmpa_tl
467 }
468 \stex_path_canonicalize:N #1
469 }
470 \stex_debug:nn{files}{Yields: \stex_path_to_string:N#1}
471 }
472

```

(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`

```

473 \cs_new_protected:Nn \stex_path_to_string:NN {
474     \exp_args:NNe \str_set:Nn #2 { \seq_use:Nn #1 / }
475 }
476
477 \cs_new:Nn \stex_path_to_string:N {
478     \seq_use:Nn #1 /
479 }

```

(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`

```

480 \str_const:Nn \c__stex_path_dot_str {.}
481 \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.

```

482 \cs_new_protected:Nn \stex_path_canonicalize:N {
483     \stex_debug:nn{paths}{canonicalizing~\seq_use:Nn #1 /}
484     \bool_set_false:N \l__stex_path_in_path_bool
485     \seq_if_empty:NF #1 {
486         \seq_clear:N \l_tmpa_seq
487         \seq_get_left:NN #1 \l_tmpa_tl
488         \str_if_empty:NT \l_tmpa_tl {
489             \seq_put_right:Nn \l_tmpa_seq {}
490         }
491         \seq_map_inline:Nn #1 {
492             \str_set:Nn \l_tmpa_tl { ##1 }
493             \str_if_eq:NnF \l_tmpa_tl \c__stex_path_dot_str {
494                 \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
495                     \bool_set_true:N \l__stex_path_in_path_bool
496                     \seq_if_empty:NNTF \l_tmpa_seq {
497                         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
498                             \c__stex_path_up_str
499                         }
500                     }{
501                         \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl

```

```

502         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
503             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
504                 \c__stex_path_up_str
505             }
506         }{
507             \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
508         }
509     }
510 }{
511     \str_if_empty:NNTF \l_tmpa_tl {
512         \bool_if:NT \l__stex_path_in_path_bool {
513             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
514         }
515     } {
516         \bool_set_true:N \l__stex_path_in_path_bool
517         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
518     }
519 }
520 }
521 }
522 \seq_gset_eq:NN #1 \l_tmpa_seq
523 \stex_debug:nn{paths}{...returns~\seq_use:Nn #1 /}
524 }
525 }

```

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

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

```

526 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
527     \seq_if_empty:NNTF #1 {
528         \prg_return_false:
529     }{
530         \seq_get_left:NN #1 \l_tmpa_tl
531         \sys_if_platform_windows:TF{
532             \str_if_in:NnTF \l_tmpa_tl {:}{
533                 \prg_return_true:
534             }{
535                 \prg_return_false:
536             }
537         }{
538             \str_if_empty:NNTF \l_tmpa_tl {
539                 \prg_return_true:
540             }{
541                 \prg_return_false:
542             }
543         }
544     }
545 }

```

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

27.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

546 \str_new:N\l_stex_kpsewhich_return_str
547 \cs_new_protected:Nn \stex_kpsewhich:n {\begingroup
548   \catcode'\ =12
549   \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
550   \tl_gset_eq:NN \l_tmpa_tl \l_tmpa_tl
551   \endgroup
552   \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
553   \tl_trim_spaces:N \l_stex_kpsewhich_return_str
554 }

```

(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`

```

555 \sys_if_platform_windows:TF{
556   \begingroup\escapechar=-1\catcode'\ =12
557   \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
558   \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
559   \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_
560   }}{
561     \stex_kpsewhich:n{-var-value~PWD}
562   }
563
564   \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
565   \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
566   \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

```

567 <@@=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

```

568 \seq_gclear_new:N\g__stex_files_stack

```

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`
`\c_stex_mainfile_str`

```

569 \str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
570 \stex_path_from_string:Nn \c_stex_mainfile_seq
571   \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`

```

572 \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`

```

573 \cs_new_protected:Nn \stex_filestack_push:n {
574   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
575   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
576     \stex_path_from_string:Nn\g_stex_currentfile_seq{
577       \c_stex_pwd_str/#1
578     }
579   }
580   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
581   \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
582   \stex_get_document_uri:
583 }

```

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

`\stex_filestack_pop:`

```

584 \cs_new_protected:Nn \stex_filestack_pop: {
585   \seq_if_empty:NF\g__stex_files_stack{
586     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
587   }
588   \seq_if_empty:NTF\g__stex_files_stack{
589     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
590   }{
591     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
592     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
593   }
594   \stex_get_document_uri:
595 }

```

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

Hooks for the current file:

```

596 \AddToHook{file/before}{
597   \tl_if_empty:NTF\CurrentFilePath{
598     \stex_filestack_push:n{\CurrentFile}
599   }{
600     \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
601   }
602 }
603 \AddToHook{file/after}{
604   \stex_filestack_pop:
605 }

```

27.4 MathHub Repositories

606 `<@=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.

```

607 \str_if_empty:NTF\mathhub{
608   \sys_if_platform_windows:TF{
609     \begingroup\escapechar=-1\catcode'\=12

```

```

610 \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str MATHHUB\c_percent_str}
611 \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
612 \exp_args:NNx\str_if_eq:ont\l_stex_kpsewhich_return_str{\c_percent_str MATHHUB\c_percent
613 \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_st
614 }{
615 \stex_kpsewhich:n{-var-value-MATHHUB}
616 }
617 \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
618
619 \str_if_empty:NT \c_stex_mathhub_str {
620 \sys_if_platform_windows:TF{
621 \begingroup\escapechar=-1\catcode'\=12
622 \exp_args:Nx\stex_kpsewhich:n{-var-value-HOME}
623 \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
624 \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_s
625 }{
626 \stex_kpsewhich:n{-var-value-HOME}
627 }
628 \ior_open:NnT \g_tmpa_ior{\l_stex_kpsewhich_return_str / .stex / mathhub.path}{
629 \begingroup\escapechar=-1\catcode'\=12
630 \ior_str_get:NN \g_tmpa_ior \l_tmpa_str
631 \sys_if_platform_windows:T{
632 \exp_args:NNx\str_replace_all:Nnn\l_tmpa_str{\c_backslash_str}/
633 }
634 \str_gset_eq:NN \c_stex_mathhub_str\l_tmpa_str
635 \endgroup
636 \ior_close:N \g_tmpa_ior
637 }
638 }
639 \str_if_empty:NTF\c_stex_mathhub_str{
640 \msg_warning:nn{stex}{warning/nomathhub}
641 }{
642 \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
643 \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
644 }
645 }{
646 \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
647 \stex_path_if_absolute:NF \c_stex_mathhub_seq {
648 \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
649 \c_stex_pwd_str/\mathhub
650 }
651 }
652 \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
653 \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
654 }

```

(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

```

655 \cs_new_protected:Nn \_stex_mathhub_do_manifest:n {
656 \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
657 \str_set:Nx \l_tmpa_str { #1 }

```

```

658 \prop_new:c { c_stex_mathhub_#1_manifest_prop }
659 \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
660 \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
661 \__stex_mathhub_find_manifest:N \l_tmpa_seq
662 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
663   \msg_error:nnxx{stex}{error/norepository}{#1}{
664     \stex_path_to_string:N \c_stex_mathhub_str
665   }
666   \input{Fatal-Error!}
667 } {
668   \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
669 }
670 }
671 }

```

(End definition for __stex_mathhub_do_manifest:n.)

\l_stex_mathhub_manifest_file_seq

```

672 \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:

```

673 \cs_new_protected:Nn \__stex_mathhub_find_manifest:N {
674   \seq_set_eq:NN\l_tmpa_seq #1
675   \bool_set_true:N\l_tmpa_bool
676   \bool_while_do:Nn \l_tmpa_bool {
677     \seq_if_empty:NTF \l_tmpa_seq {
678       \bool_set_false:N\l_tmpa_bool
679     }{
680       \file_if_exist:nTF{
681         \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
682       }{
683         \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
684         \bool_set_false:N\l_tmpa_bool
685       }{
686         \file_if_exist:nTF{
687           \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
688         }{
689           \seq_put_right:Nn\l_tmpa_seq{META-INF}
690           \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
691           \bool_set_false:N\l_tmpa_bool
692         }{
693           \file_if_exist:nTF{
694             \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
695           }{
696             \seq_put_right:Nn\l_tmpa_seq{meta-inf}
697             \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
698             \bool_set_false:N\l_tmpa_bool
699           }{
700             \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
701           }
702         }

```



```

703     }
704   }
705 }
706 \seq_set_eq:NN \l__stex_mathhub_manifest_file_seq \l_tmpa_seq
707 }

```

(End definition for __stex_mathhub_find_manifest:N.)

\c__stex_mathhub_manifest_ior File variable used for MANIFEST-files

```

708 \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:

```

709 \cs_new_protected:Nn \__stex_mathhub_parse_manifest:n {
710   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
711   \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
712   \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
713     \str_set:Nn \l_tmpa_str {##1}
714     \exp_args:NNoo \seq_set_split:Nnn
715       \l_tmpb_seq \c_colon_str \l_tmpa_str
716     \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
717       \exp_args:NNe \str_set:Nn \l_tmpb_tl {
718         \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
719       }
720       \exp_args:No \str_case:nnTF \l_tmpa_tl {
721         {id} {
722           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
723             { id } \l_tmpb_tl
724         }
725         {narration-base} {
726           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
727             { narr } \l_tmpb_tl
728         }
729         {url-base} {
730           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
731             { docurl } \l_tmpb_tl
732         }
733         {source-base} {
734           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
735             { ns } \l_tmpb_tl
736         }
737         {ns} {
738           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
739             { ns } \l_tmpb_tl
740         }
741         {dependencies} {
742           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
743             { deps } \l_tmpb_tl
744         }
745       }{}{}
746     }{}
747   }
748   \ior_close:N \c__stex_mathhub_manifest_ior

```

```

749 \stex_persist:x {
750   \prop_set_from_keyval:cn{ c_stex_mathhub_#1_manifest_prop }{
751     \exp_after:wN \prop_to_keyval:N \csname c_stex_mathhub_#1_manifest_prop\endcsname
752   }
753 }
754 }

```

(End definition for `__stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

755 \cs_new_protected:Nn \stex_set_current_repository:n {
756   \stex_require_repository:n { #1 }
757   \prop_set_eq:Nc \l_stex_current_repository_prop {
758     c_stex_mathhub_#1_manifest_prop
759   }
760 }

```

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

`\stex_require_repository:n`

```

761 \cs_new_protected:Nn \stex_require_repository:n {
762   \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
763     \stex_debug:nn{mathhub}{Opening~archive:~#1}
764     \__stex_mathhub_do_manifest:n { #1 }
765   }
766 }

```

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

`\l_stex_current_repository_prop` Current MathHub repository

```

767 %\prop_new:N \l_stex_current_repository_prop
768 \bool_if:NF \c_stex_persist_mode_bool {
769   \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
770   \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
771     \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
772   } {
773     \__stex_mathhub_parse_manifest:n { main }
774     \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
775     \l_tmpa_str
776     \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str_manifest_prop }
777     \c_stex_mathhub_main_manifest_prop
778     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
779     \stex_debug:nn{mathhub}{Current~repository:~
780       \prop_item:Nn \l_stex_current_repository_prop {id}
781     }
782   }
783 }

```

(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.

```

784 \cs_new_protected:Nn \stex_in_repository:nn {
785   \str_set:Nx \l_tmpa_str { #1 }
786   \cs_set:Npn \l_tmpa_cs ##1 { #2 }

```

```

787 \str_if_empty:NTF \l_tmpa_str {
788   \prop_if_exist:NTF \l_stex_current_repository_prop {
789     \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
790     \exp_args:Ne \l_tmpa_cs{
791       \prop_item:Nn \l_stex_current_repository_prop { id }
792     }
793   }{
794     \l_tmpa_cs{}
795   }
796 }{
797   \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}
798   \stex_require_repository:n \l_tmpa_str
799   \str_set:Nx \l_tmpa_str { #1 }
800   \exp_args:Nne \use:nn {
801     \stex_set_current_repository:n \l_tmpa_str
802     \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
803   }{
804     \stex_debug:nn{mathhub}{switching~back~to:~
805     \prop_if_exist:NTF \l_stex_current_repository_prop {
806       \prop_item:Nn \l_stex_current_repository_prop { id }::~
807     \meaning\l_stex_current_repository_prop
808     }{
809       no~repository
810     }
811   }
812   \prop_if_exist:NTF \l_stex_current_repository_prop {
813     \stex_set_current_repository:n {
814       \prop_item:Nn \l_stex_current_repository_prop { id }
815     }
816   }{
817     \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
818   }
819 }
820 }
821 }

```

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

27.5 Using Content in Archives

`\mhpath`

```

822 \def \mhpath #1 #2 {
823   \exp_args:Ne \tl_if_empty:NTF{#1}{
824     \c_stex_mathhub_str /
825     \prop_item:Nn \l_stex_current_repository_prop { id }
826     / source / #2
827   }{
828     \c_stex_mathhub_str / #1 / source / #2
829   }
830 }

```

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

\inputref
\mhinput

```

831 \newif \ifinputref \inputreffalse
832
833 \cs_new_protected:Nn \__stex_mathhub_mhinput:nn {
834   \stex_in_repository:nn {#1} {
835     \ifinputref
836       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
837     \else
838       \inputreftrue
839       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
840       \inputreffalse
841     \fi
842   }
843 }
844 \NewDocumentCommand \mhinput { 0{} m}{
845   \__stex_mathhub_mhinput:nn{ #1 }{ #2 }
846 }
847
848 \cs_new_protected:Nn \__stex_mathhub_inputref:nn {
849   \stex_in_repository:nn {#1} {
850     \stex_html_backend:TF {
851       \str_clear:N \l_tmpa_str
852       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
853         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
854       }
855
856       \tl_if_empty:nTF{ ##1 }{
857         \IfFileExists{#2}{
858           \stex_annotate_invisible:nnn{inputref}{
859             \l_tmpa_str / #2
860           }{}
861         }{
862           \input{#2}
863         }
864       }{
865         \IfFileExists{ \c_stex_mathhub_str / ##1 / source / #2 }{
866           \stex_annotate_invisible:nnn{inputref}{
867             \l_tmpa_str / #2
868           }{}
869         }{
870           \input{ \c_stex_mathhub_str / ##1 / source / #2 }
871         }
872       }
873
874     }{
875       \begingroup
876       \inputreftrue
877       \tl_if_empty:nTF{ ##1 }{
878         \input{#2}
879       }{
880         \input{ \c_stex_mathhub_str / ##1 / source / #2 }
881       }
882       \endgroup
883     }

```

```

884 }
885 }
886 \NewDocumentCommand \inputref { 0{ } m }{
887   \__stex_mathhub_inputref:nn{ #1 }{ #2 }
888 }

```

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

`\addmhbibresource`

```

889 \cs_new_protected:Nn \__stex_mathhub_mhbibresource:nn {
890   \stex_in_repository:nn {#1} {
891     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
892   }
893 }
894 \newcommand\addmhbibresource[2][]{
895   \__stex_mathhub_mhbibresource:nn{ #1 }{ #2 }
896 }

```

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

`\libinput`

```

897 \cs_new_protected:Npn \libinput #1 {
898   \prop_if_exist:NF \l_stex_current_repository_prop {
899     \msg_error:nnn{stex}{error/notinarchive}\libinput
900   }
901   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
902     \msg_error:nnn{stex}{error/notinarchive}\libinput
903   }
904   \seq_clear:N \l__stex_mathhub_libinput_files_seq
905   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
906   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
907
908   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
909     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #1.tex}
910     \IfFileExists{ \l_tmpa_str }{
911       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
912     }{}
913     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
914     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
915   }
916
917   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}
918   \IfFileExists{ \l_tmpa_str }{
919     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
920   }{}
921
922   \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
923     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
924   }{
925     \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
926       \input{ ##1 }
927     }
928   }
929 }

```

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

`\libusepackage`

```

930 \NewDocumentCommand \libusepackage {0{} m} {
931   \prop_if_exist:NF \l_stex_current_repository_prop {
932     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
933   }
934   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
935     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
936   }
937   \seq_clear:N \l__stex_mathhub_libinput_files_seq
938   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
939   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
940
941   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
942     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #2}
943     \IfFileExists{ \l_tmpa_str.sty }{
944       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
945     }{}
946     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
947     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
948   }
949
950   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2}
951   \IfFileExists{ \l_tmpa_str.sty }{
952     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
953   }{}
954
955   \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
956     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
957   }{
958     \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
959       \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
960         \usepackage[#1]{ ##1 }
961       }
962     }{
963       \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
964     }
965   }
966 }
```

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

`\mhgraphics`

`\cmhgraphics`

```

967
968 \AddToHook{begindocument}{
969   \ltx@ifpackageloaded{graphicx}{
970     \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
971     \providecommand\mhgraphics[2][]{%
972       \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
973       \includegraphics[#1]{\mhpath\Gin@mhrepos{#2}}}
974     \providecommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
975   }{}
```

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

`\lstinputmhlisting`
`\clstinputmhlisting`

```
976 \ltx@ifpackageloaded{listings}{  
977   \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}  
978   \newcommand\lstinputmhlisting[2][]{%  
979     \def\lst@mhrepos{}\setkeys{lst}{#1}%  
980     \lstinputlisting[#1]{\mhpath\lst@mhrepos{#2}}}  
981   \newcommand\clstinputmhlisting[2][]{\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}  
982 }{}  
983 }  
984  
985 \end{package}
```

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

Chapter 28

STEX -References Implementation

```
986 <*package>
987
988 %%%%%%%%% stex-references.dtx %%%%%%%%%
989
990 <@@=stex_refs>
991
992   Warnings and error messages
993   \msg_new:nnn{stex}{error/extrefmissing}{
994     Missing~in~or~cite~value~for~\detokenize{\extref}!
995   }
996   \msg_new:nnn{stex}{warning/smsmissing}{
997     .sref~file~#1~doesn't~exist!
998   }
999   \msg_new:nnn{stex}{warning/smslabelmissing}{
1000     No~label~#2~in~.sref~file~#1!
1001   }
```

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

```
1000 \iow_new:N \c__stex_refs_refs_iow
1001 \AtBeginDocument{
1002   \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
1003 }
1004 \AtEndDocument{
1005   \iow_close:N \c__stex_refs_refs_iow
1006 }
```

28.1 Document URIs and URLs

`\l_stex_current_docns_str`

```
1007 \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:`

```
1008 \cs_new_protected:Nn \stex_get_document_uri: {
```



```

1009 \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1010 \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1011 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1012 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1013 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1014
1015 \str_clear:N \l_tmpa_str
1016 \prop_if_exist:NT \l_stex_current_repository_prop {
1017   \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
1018     \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
1019   }
1020 }
1021
1022 \str_if_empty:NTF \l_tmpa_str {
1023   \str_set:Nx \l_stex_current_docns_str {
1024     file:/\stex_path_to_string:N \l_tmpa_seq
1025   }
1026 }{
1027   \bool_set_true:N \l_tmpa_bool
1028   \bool_while_do:Nn \l_tmpa_bool {
1029     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1030     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1031       {source} { \bool_set_false:N \l_tmpa_bool }
1032     }{}{
1033       \seq_if_empty:NT \l_tmpa_seq {
1034         \bool_set_false:N \l_tmpa_bool
1035       }
1036     }
1037   }
1038
1039   \seq_if_empty:NTF \l_tmpa_seq {
1040     \str_gset_eq:NN \l_stex_current_docns_str \l_tmpa_str
1041   }{
1042     \str_gset:Nx \l_stex_current_docns_str {
1043       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
1044     }
1045   }
1046 }
1047 %\stex_get_document_url:
1048 }

```

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

`\l_stex_current_docurl_str`

```

1049 \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:`

```

1050 \cs_new_protected:Nn \stex_get_document_url: {
1051   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1052   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1053   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1054   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1055   \seq_put_right:No \l_tmpa_seq \l_tmpb_str

```

```

1056 \str_clear:N \l_tmpa_str
1057 \prop_if_exist:NT \l_stex_current_repository_prop {
1058   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
1059     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
1060       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
1061     }
1062   }
1063 }
1064 }
1065
1066 \str_if_empty:NTF \l_tmpa_str {
1067   \str_set:Nx \l_stex_current_docurl_str {
1068     file:/\stex_path_to_string:N \l_tmpa_seq
1069   }
1070 }{
1071   \bool_set_true:N \l_tmpa_bool
1072   \bool_while_do:Nn \l_tmpa_bool {
1073     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1074     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1075       {source} { \bool_set_false:N \l_tmpa_bool }
1076     }{}{
1077       \seq_if_empty:NT \l_tmpa_seq {
1078         \bool_set_false:N \l_tmpa_bool
1079       }
1080     }
1081   }
1082
1083   \seq_if_empty:NTF \l_tmpa_seq {
1084     \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
1085   }{
1086     \str_set:Nx \l_stex_current_docurl_str {
1087       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
1088     }
1089   }
1090 }
1091 }

```

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

28.2 Setting Reference Targets

```

1092 \str_const:Nn \c__stex_refs_url_str{URL}
1093 \str_const:Nn \c__stex_refs_ref_str{REF}
1094 \str_new:N \l__stex_refs_curr_label_str
1095 % @currentlabel -> number
1096 % @currentlabelname -> title
1097 % @currentHref -> name.number <- id of some kind
1098 % @currentcounter <- name/id
1099 % \#autorefname <- "Section"
1100 % \theH# -> \arabic{section}
1101 % \the# -> number
1102 % \hyper@makecurrent{#}
1103 \int_new:N \l__stex_refs_unnamed_counter_int

```

Restoring references from .sref-files

\STEXInternalSrefRestoreTarget

1104 \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

```

1105 \seq_new:N \g_stex_ref_files_seq
1106
1107 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
1108   %\stex_get_document_uri:
1109   \str_clear:N \l__stex_refs_curr_label_str
1110   \str_set:Nx \l_tmpa_str { #1 }
1111   \str_if_empty:NT \l_tmpa_str {
1112     \int_gincr:N \l__stex_refs_unnamed_counter_int
1113     \str_set:Nx \l_tmpa_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
1114   }
1115   \str_set:Nx \l__stex_refs_curr_label_str {
1116     \l_stex_current_docns_str?\l_tmpa_str
1117   }
1118
1119   \exp_args:Noo \STEXInternalAuxAddDocRef\l_stex_current_docns_str\l_tmpa_str
1120
1121   %\seq_if_exist:cF{g__stex_refs_labels_\l_tmpa_str_seq}{
1122   %   \seq_new:c {g__stex_refs_labels_\l_tmpa_str_seq}
1123   %}
1124   %\seq_if_in:coF{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str {
1125   %   \seq_gput_right:co{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str
1126   %}
1127
1128
1129   \stex_if_smsmode:TF {
1130     %\stex_get_document_url:
1131     %\str_gset_eq:cN {sref_url_\l__stex_refs_curr_label_str_str}\l_stex_current_docurl_str
1132     %\str_gset_eq:cN {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_url_str
1133   }{
1134     \iow_now:Nx \c__stex_refs_refs_iow {
1135       \STEXInternalSrefRestoreTarget
1136       {\l_stex_current_docns_str}
1137       {\l_tmpa_str}
1138       {\@currentcounter}
1139       {\@currentlabel}
1140       {\tl_if_exist:NT\@currentlabelname{\exp_args:No\unexpanded\@currentlabelname}}
1141     }
1142     %\iow_now:Nx \c__stex_refs_refs_iow {
1143     %   {\l_stex_current_docns_str?\l_tmpa_str}~=={\use:c{\@currentcounter autorefname}~\@currentlabel}
1144     %\stex_debug:nn{sref}{New~label~\l__stex_refs_curr_label_str~at~\use:c{\use:c{\@currentcounter}~\@currentlabel}}
1145     %\exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
1146     %\immediate\write\auxout{\STEXInternalAuxAddDocRef{\l_stex_current_docns_str}{\l_tmpa_str}{\l__stex_refs_curr_label_str_type}\c__stex_refs_ref_str}
1147     %\str_gset:cx {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_ref_str
1148   }
1149 }
1150 \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.

```

1151 \cs_new_protected:Npn \STEXInternalAuxAddDocRef #1 #2 {
1152   \exp_args:NNx \seq_if_in:NnTF \g_stex_ref_files_seq {\detokenize{#1}} {
1153     \exp_args:Nnx \seq_if_in:cnF{g_stex_ref_ #1 _seq}{\detokenize{#2}}{
1154       \exp_args:Nnx \seq_gput_left:cn{g_stex_ref_ #1 _seq}{\detokenize{#2}}
1155     }
1156   }{
1157     \exp_args:NNx \seq_gput_right:Nn \g_stex_ref_files_seq {\detokenize{#1}}
1158     \%seq_if_exist:cF{g_stex_ref_ #1 _seq}{
1159       \seq_new:c{g_stex_ref_ #1 _seq} % <- seq_new throws errors??
1160     }
1161     \exp_args:Nnx \seq_gput_left:cn{g_stex_ref_ #1 _seq}{\detokenize{#2}}
1162   }
1163
1164   \%str_set:Nn \l_tmpa_str {#1?#2}
1165   \%str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
1166   \%seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
1167     \% \seq_new:c {g__stex_refs_labels_#2_seq}
1168   }
1169   \%seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
1170     \% \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
1171   }
1172 }

```

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

```

1173 \AtEndDocument{
1174   \def\STEXInternalAuxAddDocRef#1 #2 {}{}
1175 }

```

`\stex_ref_new_sym_target:n`

```

1176 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
1177
1178   \% \stex_if_smsmode:TF {
1179     \% \str_if_exist:cF{sref_sym_#1_type}{
1180       \% \stex_get_document_url:
1181       \% \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str
1182       \% \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
1183     }
1184   }{
1185     \% \str_if_empty:NF \l__stex_refs_curr_label_str {
1186       \% \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
1187       \% \immediate\write\@auxout{
1188         \% \exp_not:N\expandafter\def\exp_not:N\csname \exp_not:N\detokenize{sref_sym_#1_label
1189           \% \l__stex_refs_curr_label_str
1190         }
1191       }
1192     }
1193   }
1194 }

```

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

28.3 Using References

`\sref` Optional arguments:

```

1195
1196 \keys_define:nn { stex / sref / 1 } {
1197   archive .str_set_x:N = \l__stex_refs_repo_str,
1198   file     .str_set_x:N = \l__stex_refs_file_str,
1199   % TODO get rid of this
1200   fallback .code:n = {},
1201   pre      .code:n = {},
1202   post     .code:n = {}
1203 }
1204 \cs_new_protected:Nn \__stex_refs_args_i:n {
1205   \str_clear:N \l__stex_refs_repo_str
1206   \str_clear:N \l__stex_refs_file_str
1207   \keys_set:nn { stex / sref / 1 } { #1 }
1208 }
1209 \keys_define:nn { stex / sref / 2 } {
1210   in .str_set_x:N = \l__stex_refs_in_str,
1211   archive .str_set_x:N = \l__stex_refs_repob_str,
1212   title .tl_set:N = \l__stex_refs_title_tl
1213 }
1214 \cs_new_protected:Nn \__stex_refs_args_ii:n {
1215   \str_clear:N \l__stex_refs_in_str
1216   \tl_clear:N \l__stex_refs_title_tl
1217   \str_clear:N \l__stex_refs_repob_str
1218   \keys_set:nn { stex / sref / 2 } { #1 }
1219 }

```

The actual macro:

```

1220 \NewDocumentCommand \sref { 0{} m 0{} }{
1221   \__stex_refs_args_i:n{#1}
1222   \__stex_refs_args_ii:n{#3}
1223   \str_clear:N \l__stex_refs_uri_str
1224   \__stex_refs_find_uri:n{#2}
1225   \__stex_refs_do_sref:n{#2}
1226 }
1227 \NewDocumentCommand \extref { 0{} m m }{
1228   \__stex_refs_args_i:n{#1}
1229   \__stex_refs_args_ii:n{#3}
1230   \str_if_empty:NT \l__stex_refs_in_str {
1231     \msg_error:nn{stex}{error/extrefmissing}
1232   }
1233   \str_clear:N \l__stex_refs_uri_str
1234   \__stex_refs_find_uri:n{#2}
1235   \__stex_refs_do_sref_in:n{#2}
1236 }
1237
1238 \cs_new_protected:Nn \__stex_refs_find_uri:n {
1239   \stex_debug:nn{sref}{File:~\l__stex_refs_file_str^^JRepo:\l__stex_refs_repo_str}
1240   \str_if_empty:NTF \l__stex_refs_file_str {
1241     \stex_debug:nn{sref}{Empty.~Checking~current~file~for~#1}
1242     \seq_if_exist:cT{g_stex_ref_\l_stex_current_docns_str_seq}{
1243       \seq_map_inline:cn{g_stex_ref_\l_stex_current_docns_str_seq}{

```

```

1244     \str_if_eq:nnT{#1}{##1}{
1245         \str_set_eq:NN \l__stex_refs_uri_str \l_stex_current_docns_str
1246         \stex_debug:nn{sref}{Found.}
1247         \seq_map_break:
1248     }
1249 }
1250 }
1251 \str_if_empty:NT \l__stex_refs_uri_str {
1252     \stex_debug:nn{sref}{Checking~other~files}
1253     \seq_map_inline:Nn \g_stex_ref_files_seq {
1254         \stex_debug:nn{sref}{##1...}
1255         \seq_map_inline:cn{g_stex_ref_##1_seq}{
1256             \str_if_eq:nnT{#1}{####1}{
1257                 \stex_debug:nn{sref}{Found~##1}
1258                 \str_set:Nn \l__stex_refs_uri_str {##1}
1259                 \seq_map_break:n{\seq_map_break:}
1260             }
1261         }
1262     }
1263 }
1264 }{
1265     \str_if_empty:NTF \l__stex_refs_repo_str {
1266         \prop_if_exist:NTF \l_stex_current_repository_prop {
1267             \stex_debug:nn{sref}{in~archive~\prop_item:Nn \l_stex_current_repository_prop { id }
1268             \prop_get:NnN \l_stex_current_repository_prop { ns } \l__stex_refs_uri_str
1269             \stex_debug:nn{sref}{namespace:~\l__stex_refs_uri_str}
1270             \str_set:Nx \l__stex_refs_uri_str {\l__stex_refs_uri_str / \l__stex_refs_file_str}
1271             \stex_path_from_string:Nn \l_tmpb_seq \l__stex_refs_uri_str
1272             \str_set:Nx \l__stex_refs_uri_str {\stex_path_to_string:N \l_tmpb_seq}
1273             \stex_debug:nn{sref}{Return:~\l__stex_refs_uri_str}
1274         }{
1275             \stex_debug:nn{sref}{Not~in~archive}
1276             \stex_path_from_string:Nn \l_tmpb_seq {
1277                 \stex_path_to_string:N \g_stex_currentfile_seq/ .. / \l__stex_refs_file_str
1278             }
1279             \str_set:Nx \l__stex_refs_uri_str {file:~\stex_path_to_string:N \l_tmpb_seq}
1280         }
1281     }{
1282         \stex_require_repository:n \l__stex_refs_repo_str
1283         \prop_get:cnN { c_stex_mathhub \l__stex_refs_repo_str _manifest_prop } { ns } \l__stex
1284         \str_set:Nx \l__stex_refs_uri_str {\l__stex_refs_uri_str / \l__stex_refs_file_str}
1285         \stex_path_from_string:Nn \l_tmpb_seq \l__stex_refs_uri_str
1286         \str_set:Nx \l__stex_refs_uri_str {\stex_path_to_string:N \l_tmpb_seq}
1287     }
1288 }
1289 }
1290
1291 \cs_new_protected:Nn \__stex_refs_do_autoref:n{
1292     \cs_if_exist:cTF{autoref}{
1293         \exp_args:Nx\autoref{sref_#1}
1294     }{
1295         \exp_args:Nx\ref{sref_#1}
1296     }
1297 }

```

```

1298
1299 \cs_new_protected:Nn \__stex_refs_do_sref:n {
1300   \str_if_empty:NTF \l__stex_refs_uri_str {
1301     \str_if_empty:NTF \l__stex_refs_in_str {
1302       \stex_debug:nn{sref}{autoref~on~#1}
1303       \__stex_refs_do_autoref:n{#1}
1304     }{
1305       \stex_debug:nn{sref}{srefin~on~#1}
1306       \__stex_refs_do_sref_in:n{#1}
1307     }
1308   }{
1309     \exp_args:NNo \seq_if_in:NnTF \g_stex_ref_files_seq \l__stex_refs_uri_str {
1310       \exp_args:Nnx \seq_if_in:cnTF{g_stex_ref\_l__stex_refs_uri_str_seq}{\detokenize{#1}}{
1311         \stex_debug:nn{sref}{Reference~found~in~ref~files;~autoref~on~\l__stex_refs_uri_str?#1}
1312         \__stex_refs_do_autoref:n{\l__stex_refs_uri_str?#1}
1313       }{
1314         \str_if_empty:NTF \l__stex_refs_in_str {
1315           \stex_debug:nn{sref}{in~empty;~autoref~on~\l__stex_refs_uri_str?#1}
1316           \__stex_refs_do_autoref:n{\l__stex_refs_uri_str?#1}
1317         }{
1318           \stex_debug:nn{sref}{in~non~empty;~srefin~on~\l__stex_refs_uri_str?#1}
1319           \__stex_refs_do_sref_in:n{#1}
1320         }
1321       }
1322     }{
1323       \str_if_empty:NTF \l__stex_refs_in_str {
1324         \stex_debug:nn{sref}{in~empty;~autoref~on~\l__stex_refs_uri_str?#1}
1325         \__stex_refs_do_autoref:n{\l__stex_refs_uri_str?#1}
1326       }{
1327         \stex_debug:nn{sref}{in~non~empty;~srefin~on~\l__stex_refs_uri_str?#1}
1328         \__stex_refs_do_sref_in:n{#1}
1329       }
1330     }
1331   }
1332 }
1333
1334 \cs_new_protected:Nn \__stex_refs_restore_target:nnnnn {
1335   \str_if_empty:NTF \l__stex_refs_uri_str {
1336     \exp_args:No \str_if_eq:nnT \l__stex_refs_id_str {#2}{
1337       \tl_set:Nn \l__stex_refs_return_tl {
1338         \use:c{#3autorefname}~#4\tl_if_empty:nF{#5}{~(5)}~in~
1339         \tl_if_empty:NTF\l__stex_refs_title_tl{
1340           ???
1341         }\l__stex_refs_title_tl
1342       }
1343     }
1344   }{
1345     \stex_debug:nn{sref}{\l__stex_refs_uri_str{~ == ~ #1 ~ ?}}
1346     \exp_args:No \str_if_eq:nnT \l__stex_refs_uri_str {#1}{
1347       \stex_debug:nn{sref}{\l__stex_refs_id_str~ == ~ #2 ~ ?}}
1348     \exp_args:No \str_if_eq:nnT \l__stex_refs_id_str {#2}{
1349       \stex_debug:nn{sref}{success!}
1350       \tl_set:Nn \l__stex_refs_return_tl {
1351         \use:c{#3autorefname}~#4\tl_if_empty:nF{#5}{~(5)}~in~

```

```

1352         \tl_if_empty:nTF\l__stex_refs_title_tl{
1353             ???
1354         }\l__stex_refs_title_tl
1355     }
1356     \endinput
1357 }
1358 }
1359 }
1360 }
1361
1362 \cs_new_protected:Nn \__stex_refs_do_sref_in:n {
1363     \stex_debug:nn{sref}{In: \l__stex_refs_in_str^^JRepo:\l__stex_refs_repo_str}
1364     \stex_debug:nn{sref}{URI: \l__stex_refs_uri_str?#1}
1365     %\msg_warning:nnn{stex}{warning/smsmissing}{<filename>}
1366     \begin_group\catcode13=9\relax\catcode10=9\relax
1367     \str_if_empty:NTF \l__stex_refs_repob_str {
1368         \prop_if_exist:NTF \l_stex_current_repository_prop {
1369             \str_set:Nx \l_tmpa_str {
1370                 \c_stex_mathhub_str /
1371                 \prop_item:Nn \l_stex_current_repository_prop { id }
1372                 / source / \l__stex_refs_in_str .sref
1373             }
1374         }{
1375             \str_set:Nx \l_tmpa_str {
1376                 \stex_path_to_string:N \g_stex_currentfile_seq/ .. / \l__stex_refs_in_str . sref
1377             }
1378         }
1379     }{
1380         \str_set:Nx \l_tmpa_str {
1381             \c_stex_mathhub_str / \l__stex_refs_repob_str
1382             / source / \l__stex_refs_in_str . sref
1383         }
1384     }
1385     \stex_path_from_string:Nn \l_tmpb_seq \l_tmpa_str
1386     \stex_path_to_string:NN \l_tmpb_seq \l_tmpa_str
1387     \stex_debug:nn{sref}{File: \l_tmpa_str}
1388     \exp_args:No \IfFileExists \l_tmpa_str {
1389         \tl_clear:N \l__stex_refs_return_tl
1390         \str_set:Nn \l__stex_refs_id_str {#1}
1391         \let\STEXInternalSrefRestoreTarget\__stex_refs_restore_target:nnnnn
1392         \use:c{@ @ input}{\l_tmpa_str}
1393         \exp_args:No \tl_if_empty:nTF \l__stex_refs_return_tl {
1394             \exp_args:Nnno \msg_warning:nnnn{stex}{warning/smslabelmissing}\l_tmpa_str{#1}
1395             \__stex_refs_do_autoref:n{
1396                 \str_if_empty:NF\l__stex_refs_uri_str{\l__stex_refs_uri_str?}#1
1397             }
1398         }{
1399             \l__stex_refs_return_tl
1400         }
1401     }{
1402         \exp_args:Nnno \msg_warning:nnn{stex}{warning/smsmissing}\l_tmpa_str
1403         \__stex_refs_do_autoref:n{
1404             \str_if_empty:NF\l__stex_refs_uri_str{\l__stex_refs_uri_str?}#1
1405         }
1406     }

```



```

1406     }
1407 \endgroup
1408 }
1409
1410 % \__stex_refs_args:n { #1 }
1411 % \str_if_empty:NTF \l__stex_refs_indocument_str {
1412 %   \str_set:Nx \l_tmpa_str { #2 }
1413 %   \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
1414 %   \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
1415 %     \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str_seq}{
1416 %       \seq_get_left:cNF {g__stex_refs_labels_\l_tmpa_str_seq} \l_tmpa_str {
1417 %         \str_clear:N \l_tmpa_str
1418 %       }
1419 %     }{
1420 %       \str_clear:N \l_tmpa_str
1421 %     }
1422 %   }{
1423 %     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1424 %     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1425 %     \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
1426 %     \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str_seq}{
1427 %       \str_set_eq:NN \l_tmpc_str \l_tmpa_str
1428 %       \str_clear:N \l_tmpa_str
1429 %       \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str_seq} {
1430 %         \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
1431 %           \str_range:nnn { ##1 }{-\l_tmpa_int}{ -1 }
1432 %         }{
1433 %           \seq_map_break:n {
1434 %             \str_set:Nn \l_tmpa_str { ##1 }
1435 %           }
1436 %         }
1437 %       }
1438 %     }{
1439 %       \str_clear:N \l_tmpa_str
1440 %     }
1441 %   }
1442 % \str_if_empty:NTF \l_tmpa_str {
1443 %   \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_li
1444 % }{
1445 %   \str_if_eq:cNTF {sref_\l_tmpa_str_type} \c__stex_refs_ref_str {
1446 %     \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1447 %       \cs_if_exist:cTF{autoref}{
1448 %         \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1449 %       }{
1450 %         \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1451 %       }
1452 %     }{
1453 %       \ltx@ifpackageloaded{hyperref}{
1454 %         \hyperref[sref_\l_tmpa_str]\l__stex_refs_linktext_tl
1455 %       }{
1456 %         \l__stex_refs_linktext_tl
1457 %       }
1458 %     }
1459 %   }{

```

```

1460 % \ltx@ifpackageloaded{hyperref}{
1461 % \href{\use:c{sref_url_1\l_tmpa_str_str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl
1462 % }{
1463 % \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_fallback_tl
1464 % }
1465 % }
1466 % }
1467 % }{
1468 % TODO
1469 % }
1470 %}

```

(End definition for `\sref`. This function is documented on page 82.)

`\srefsym`

```

1471 \NewDocumentCommand \srefsym { 0{} m}{
1472 \stex_get_symbol:n { #2 }
1473 \__stex_refs_sym_aux:nn{#1}{\l_stex_get_symbol_uri_str}
1474 }
1475
1476 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1477
1478 % \str_if_exist:cTF {sref_sym_#2_label_str }{
1479 % \sref[#1]{\use:c{sref_sym_#2_label_str}}
1480 % }{
1481 % \__stex_refs_args:n { #1 }
1482 % \str_if_empty:NTF \l__stex_refs_indocument_str {
1483 % \tl_if_exist:cTF{sref_sym_#2_type}{
1484 % % doc uri in \l_tmpb_str
1485 % \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2_type}}
1486 % \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1487 % % reference
1488 % \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1489 % \cs_if_exist:cTF{autoref}{
1490 % \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1491 % }{
1492 % \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1493 % }
1494 % }{
1495 % \ltx@ifpackageloaded{hyperref}{
1496 % \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1497 % }{
1498 % \l__stex_refs_linktext_tl
1499 % }
1500 % }
1501 % }{
1502 % % URL
1503 % \ltx@ifpackageloaded{hyperref}{
1504 % \href{\use:c{sref_sym_url_#2_str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl
1505 % }{
1506 % \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_fallback_tl
1507 % }
1508 % }
1509 % }{

```

```

1510 %          \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_
1511 %          }
1512 %      }{
1513 %          % TODO
1514 %      }
1515 %  }
1516 }

```

(End definition for \srefsym. This function is documented on page 82.)

\srefsymuri

```

1517 \cs_new_protected:Npn \srefsymuri #1 #2 { % TODO
1518     #2%\__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1519 }

```

(End definition for \srefsymuri. This function is documented on page 82.)

```

1520 </package>

```

Chapter 29

STEX -Modules Implementation

```
1521 <*package>
1522
1523 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1524
1525 <@@=stex_modules>
1526
1527 Warnings and error messages
1528 \msg_new:nnn{stex}{error/unknownmodule}{
1529   No~module~#1~found
1530 }
1531 \msg_new:nnn{stex}{error/syntax}{
1532   Syntax~error:~#1
1533 }
1534 \msg_new:nnn{stex}{error/siglanguage}{
1535   Module~#1~declares~signature~#2,~but~does~not~
1536   declare~its~language
1537 }
1538 \msg_new:nnn{stex}{warning/deprecated}{
1539   #1~is~deprecated;~please~use~#2~instead!
1540 }
1541 \msg_new:nnn{stex}{error/conflictingmodules}{
1542   Conflicting~imports~for~module~#1
1543 }
```

`\l_stex_current_module_str` The current module:

```
1543 \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

```
1544 \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
1545 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1546   \str_if_empty:NTF \l_stex_current_module_str
1547   \prg_return_false: \prg_return_true:
1548 }

```

(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
1549 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1550   \prop_if_exist:cTF { c_stex_module_#1_prop }
1551   \prg_return_true: \prg_return_false:
1552 }

```

(End definition for `\stex_if_module_exists:nTF`. This function is documented on page 84.)

`\stex_add_to_current_module:n` Only allowed within modules:

```

\STEXexport
1553 \cs_new_protected:Nn \stex_execute_in_module:n { \stex_if_in_module:T {
1554   \stex_add_to_current_module:n { #1 }
1555   \stex_do_up_to_module:n { #1 }
1556 }}
1557 \cs_generate_variant:Nn \stex_execute_in_module:n {x}
1558
1559 \cs_new_protected:Nn \stex_add_to_current_module:n {
1560   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str_code} { #1 }
1561 }
1562 \cs_generate_variant:Nn \stex_add_to_current_module:n {x}
1563 \cs_new_protected:Npn \STEXexport {
1564   \ExplSyntaxOn
1565   \__stex_modules_export:n
1566 }
1567 \cs_new_protected:Nn \__stex_modules_export:n {
1568   \ignorespacesandpars#1\ExplSyntaxOff
1569   \stex_add_to_current_module:n { \ignorespacesandpars#1}
1570   \stex_smsmode_do:
1571 }
1572 \let \stex_module_export_helper:n \use:n
1573 \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
1574 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1575   \str_set:Nx \l_tmpa_str { #1 }
1576   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str_constants} { \l_tmpa_str }
1577 }

```

(End definition for `\stex_add_constant_to_current_module:n`. This function is documented on page 84.)

```

\stex_add_import_to_current_module:n
1578 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1579   \str_set:Nx \l_tmpa_str { #1 }
1580   \exp_args:Nno

```

```

1581 \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1582 \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1583 }
1584 }

```

(End definition for `\stex_add_import_to_current_module:n`. This function is documented on page 84.)

`\stex_collect_imports:n`

```

1585 \cs_new_protected:Nn \stex_collect_imports:n {
1586 \seq_clear:N \l_stex_collect_imports_seq
1587 \__stex_modules_collect_imports:n {#1}
1588 }
1589 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1590 \seq_map_inline:cn {c_stex_module_#1_imports} {
1591 \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1592 \__stex_modules_collect_imports:n { ##1 }
1593 }
1594 }
1595 \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1596 \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1597 }
1598 }

```

(End definition for `\stex_collect_imports:n`. This function is documented on page 84.)

`\stex_do_up_to_module:n`

```

1599 \int_new:N \l__stex_modules_group_depth_int
1600 \cs_new_protected:Nn \stex_do_up_to_module:n {
1601 \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1602 #1
1603 }{
1604 #1
1605 \expandafter \tl_gset:Nn
1606 \csname l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1607 \expandafter\expandafter\expandafter\endcsname
1608 \expandafter\expandafter\expandafter { \csname
1609 l__stex_modules_aftergroup_\l_stex_current_module_str _tl\endcsname #1 }
1610 \aftergroup\__stex_modules_aftergroup_do:
1611 }
1612 }
1613 \cs_generate_variant:Nn \stex_do_up_to_module:n {x}
1614 \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1615 \stex_debug:nn{aftergroup}{\cs_meaning:c{
1616 l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1617 }}}
1618 \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1619 \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1620 \tl_gclear:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1621 }{
1622 \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1623 \aftergroup\__stex_modules_aftergroup_do:
1624 }
1625 }
1626 \cs_new_protected:Nn \stex_reset_up_to_module:n {
1627 \expandafter\let\csname l__stex_modules_aftergroup_#1_tl\endcsname\undefined

```

1628 }

(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).

1629

(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.

```
1630 \str_new:N \l_stex_module_ns_str
1631 \str_new:N \l_stex_module_subpath_str
1632 \cs_new_protected:Nn \__stex_modules_compute_namespace:nN {
1633   \seq_set_eq:NN \l_tmpa_seq #2
1634   % split off file extension
1635   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str % <- filename
1636   \exp_args:Nnno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1637   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str % <- filename without suffixes
1638   \seq_put_right:No \l_tmpa_seq \l_tmpb_str % <- file path including name without suffixes
1639
1640   \bool_set_true:N \l_tmpa_bool
1641   \bool_while_do:Nn \l_tmpa_bool {
1642     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1643     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1644       {source} { \bool_set_false:N \l_tmpa_bool }
1645     }{}{
1646       \seq_if_empty:NT \l_tmpa_seq {
1647         \bool_set_false:N \l_tmpa_bool
1648       }
1649     }
1650   }
1651
1652   \stex_path_to_string:NN \l_tmpa_seq \l_stex_module_subpath_str
1653   % \l_tmpa_seq <- sub-path relative to archive
1654   \str_if_empty:NTF \l_stex_module_subpath_str {
1655     \str_set:Nx \l_stex_module_ns_str {#1}
1656   }{
1657     \str_set:Nx \l_stex_module_ns_str {
1658       #1/\l_stex_module_subpath_str
1659     }
1660   }
1661 }
1662
1663 \cs_new_protected:Nn \stex_modules_current_namespace: {
1664   \str_clear:N \l_stex_module_subpath_str
1665   \prop_if_exist:NTF \l_stex_current_repository_prop {
1666     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1667     \__stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1668   }{
1669     % split off file extension
1670     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1671     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
```

```

1672 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1673 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1674 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1675 \str_set:Nx \l_stex_module_ns_str {
1676   file:/\stex_path_to_string:N \l_tmpa_seq
1677 }
1678 }
1679 }

```

(End definition for `\stex_modules_current_namespace:.` This function is documented on page 85.)

29.1 The smodule environment

smodule arguments:

```

1680 \keys_define:nn { stex / module } {
1681   title      .tl_set:N      = \smodulename ,
1682   type       .str_set_x:N   = \smodulename ,
1683   id         .str_set_x:N   = \smoduleid ,
1684   deprecate  .str_set_x:N   = \l_stex_module_deprecate_str ,
1685   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1686   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1687   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1688   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1689   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1690   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1691   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1692 }
1693
1694 \cs_new_protected:Nn \__stex_modules_args:n {
1695   \str_clear:N \smodulename
1696   \str_clear:N \smodulename
1697   \str_clear:N \smoduleid
1698   \str_clear:N \l_stex_module_ns_str
1699   \str_clear:N \l_stex_module_deprecate_str
1700   \str_clear:N \l_stex_module_lang_str
1701   \str_clear:N \l_stex_module_sig_str
1702   \str_clear:N \l_stex_module_creators_str
1703   \str_clear:N \l_stex_module_contributors_str
1704   \str_clear:N \l_stex_module_meta_str
1705   \str_clear:N \l_stex_module_srccite_str
1706   \keys_set:nn { stex / module } { #1 }
1707 }
1708
1709 % module parameters here? In the body?
1710

```

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

```

1711 \cs_new_protected:Nn \stex_module_setup:nn {
1712   \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1713   \str_set:Nx \l_stex_module_name_str { #2 }
1714   \__stex_modules_args:n { #1 }

```


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

Are we in a nested module?

```

1715 \stex_if_in_module:TF {
1716   % Nested module
1717   \prop_get:cnN {c_stex_module\_l_stex_current_module_str _prop}
1718   { ns } \l_stex_module_ns_str
1719   \str_set:Nx \l_stex_module_name_str {
1720     \prop_item:cn {c_stex_module\_l_stex_current_module_str _prop}
1721     { name } / \l_stex_module_name_str
1722   }
1723   \str_if_empty:NT \l_stex_module_lang_str {
1724     \str_set:Nx \l_stex_module_lang_str {
1725       \prop_item:cn {c_stex_module\_l_stex_current_module_str _prop}
1726       { lang }
1727     }
1728   }
1729 }{
1730   % not nested:
1731   \str_if_empty:NT \l_stex_module_ns_str {
1732     \stex_modules_current_namespace:
1733     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1734       / {\l_stex_module_ns_str}
1735     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1736     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1737       \str_set:Nx \l_stex_module_ns_str {
1738         \stex_path_to_string:N \l_tmpa_seq
1739       }
1740     }
1741   }
1742 }

```

Next, we determine the language of the module:

```

1743 \str_if_empty:NT \l_stex_module_lang_str {
1744   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1745   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1746   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1747   \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
1748     \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
1749       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
1750     }
1751   }
1752   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1753   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
1754     \seq_pop_right:NN \l_tmpa_seq \l_stex_module_lang_str
1755     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1756       inferred~from~file~name}
1757   }
1758 }
1759
1760 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
1761   \exp_args:NNo \stex_set_language:Nn \l_tmpa_str \l_stex_module_lang_str
1762 }}

```

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

```

1763 \str_if_empty:NTF \l_stex_module_sig_str {
1764   \exp_args:Nnx \prop_gset_from_keyval:cn {
1765     c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1766   } {
1767     name      = \l_stex_module_name_str ,
1768     ns        = \l_stex_module_ns_str ,
1769     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1770     lang      = \l_stex_module_lang_str ,
1771     sig       = \l_stex_module_sig_str ,
1772     deprecate = \l_stex_module_deprecate_str ,
1773     meta      = \l_stex_module_meta_str
1774   }
1775   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1776   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1777   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _copymodules}
1778   \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1779   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1780 \str_if_empty:NT \l_stex_module_meta_str {
1781   \str_set_eq:NN \l_stex_module_meta_str \l_stex_metatheory_str
1782 }
1783 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1784   \bool_set_true:N \l_stex_in_meta_bool
1785   \exp_args:Nx \stex_add_to_current_module:n {
1786     \bool_set_true:N \l_stex_in_meta_bool
1787     \stex_activate_module:n {\l_stex_module_meta_str}
1788     \bool_set_false:N \l_stex_in_meta_bool
1789   }
1790   \stex_activate_module:n {\l_stex_module_meta_str}
1791   \bool_set_false:N \l_stex_in_meta_bool
1792 }
1793 }{
1794   \str_if_empty:NT \l_stex_module_lang_str {
1795     \msg_error:nnxx{stex}{error/siglanguage}{
1796       \l_stex_module_ns_str?\l_stex_module_name_str
1797     }{\l_stex_module_sig_str}
1798   }
1799   \stex_debug:nn{modules}{Signature~\l_stex_module_sig_str~for~\l_stex_module_ns_str?\l_st
1800   \stex_if_module_exists:nTF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1801     \stex_debug:nn{modules}{(already exists)}
1802   }{
1803     \stex_debug:nn{modules}{(needs loading)}
1804     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1805     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1806     \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1807     \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1808     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1809     \str_set:Nx \l_tmpa_str {
1810       \stex_path_to_string:N \l_tmpa_seq /
1811       \l_tmpa_str . \l_stex_module_sig_str .tex
1812     }

```

```

1813 \IfFileExists \l_tmpa_str {
1814 \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1815 \str_clear:N \l_stex_current_module_str
1816 \seq_clear:N \l_stex_all_modules_seq
1817 \stex_debug:nn{modules}{Loading~signature}
1818 }
1819 }{
1820 \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1821 }
1822 }
1823 \stex_if_smsmode:F {
1824 \stex_activate_module:n {
1825 \l_stex_module_ns_str ? \l_stex_module_name_str
1826 }
1827 }
1828 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1829 }
1830 \str_if_empty:NF \l_stex_module_deprecate_str {
1831 \msg_warning:nnxx{stex}{warning/deprecated}{
1832 Module~\l_stex_current_module_str
1833 }{
1834 \l_stex_module_deprecate_str
1835 }
1836 }
1837 \seq_put_right:Nx \l_stex_all_modules_seq {
1838 \l_stex_module_ns_str ? \l_stex_module_name_str
1839 }
1840 \tl_clear:c{l_stex_modules_aftergroup\l_stex_module_ns_str ? \l_stex_module_name_str _tl
1841 }

```

(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}
1842 \cs_new_protected:Nn \__stex_modules_begin_module: {
1843 \stex_reactivate_macro:N \STEXexport
1844 \stex_reactivate_macro:N \importmodule
1845 \stex_reactivate_macro:N \symdecl
1846 \stex_reactivate_macro:N \notation
1847 \stex_reactivate_macro:N \symdef
1848
1849 \stex_debug:nn{modules}{
1850 New~module:\\
1851 Namespace:~\l_stex_module_ns_str\\
1852 Name:~\l_stex_module_name_str\\
1853 Language:~\l_stex_module_lang_str\\
1854 Signature:~\l_stex_module_sig_str\\
1855 Metatheory:~\l_stex_module_meta_str\\
1856 File:~\stex_path_to_string:N \g_stex_currentfile_seq
1857 }
1858
1859 \stex_if_do_html:T{
1860 \begin{stex_annotate_env} {theory} {

```

```

1861     \l_stex_module_ns_str ? \l_stex_module_name_str
1862   }
1863
1864   \stex_annotate_invisible:nnn{header}{} {
1865     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1866     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1867     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1868       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1869     }
1870     \str_if_empty:NF \smoduletype {
1871       \stex_annotate:nnn{type}{\smoduletype}{}
1872     }
1873   }
1874 }
1875 % TODO: Inherit metatheory for nested modules?
1876 }
1877 \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}

```

1878 \cs_new_protected:Nn \_stex_modules_end_module: {
1879   \stex_debug:nn{modules}{Closing module~\prop_item:cn {c_stex_module_\l_stex_current_module}
1880   \_stex_reset_up_to_module:n \l_stex_current_module_str
1881   \stex_if_smsmode:T {
1882     \stex_persist:x {
1883       \prop_set_from_keyval:cn{c_stex_module_\l_stex_current_module_str _prop}{
1884         \exp_after:wN \prop_to_keyval:N \csname c_stex_module_\l_stex_current_module_str _pr
1885       }
1886       \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _constants}{
1887         \seq_use:cn{c_stex_module_\l_stex_current_module_str _constants},
1888       }
1889       \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _imports}{
1890         \seq_use:cn{c_stex_module_\l_stex_current_module_str _imports},
1891       }
1892       \tl_set:cn {c_stex_module_\l_stex_current_module_str _code}
1893     }
1894     \exp_after:wN \let \exp_after:wN \l_tmpa_tl \csname c_stex_module_\l_stex_current_module
1895     \exp_after:wN \stex_persist:n \exp_after:wN { \exp_after:wN { \l_tmpa_tl } }
1896   }
1897 }

```

(End definition for _stex_modules_end_module:.)

The core environment

```

1898 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1899 \NewDocumentEnvironment { smodule } { 0{} m } {
1900   \stex_module_setup:nn{#1}{#2}
1901   %\par
1902   \stex_if_smsmode:F{
1903     \tl_if_empty:NF \smoduletitle {
1904       \exp_args:No \stex_document_title:n \smoduletitle
1905     }
1906     \tl_clear:N \l_tmpa_tl
1907     \clist_map_inline:Nn \smoduletype {

```

```

1908     \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1909         \tl_set:Nn \l_tmpa_tl {
1910             \stex_patch_counters:
1911             \use:c{__stex_modules_smodule_##1_start:}
1912             \stex_unpatch_counters:
1913         }
1914     }
1915 }
1916 \tl_if_empty:NTF \l_tmpa_tl {
1917     \__stex_modules_smodule_start:
1918 }{
1919     \l_tmpa_tl
1920 }
1921 }
1922 \__stex_modules_begin_module:
1923 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1924     \exp_args:Nx \stex_add_to_current_module:n{
1925         \stex_activate_module:n {\l_stex_module_meta_str}
1926     }
1927 }
1928 \str_if_empty:NF \smoduleid {
1929     \stex_ref_new_doc_target:n \smoduleid
1930 }
1931 \stex_smsmode_do:
1932 } {
1933     \__stex_modules_end_module:
1934     \stex_if_smsmode:F {
1935         \end{stex_annotate_env}
1936         \clist_set:Nn \l_tmpa_clist \smoduletype
1937         \tl_clear:N \l_tmpa_tl
1938         \clist_map_inline:Nn \l_tmpa_clist {
1939             \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1940                 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1941             }
1942         }
1943         \tl_if_empty:NTF \l_tmpa_tl {
1944             \__stex_modules_smodule_end:
1945         }{
1946             \l_tmpa_tl
1947         }
1948     }
1949 }

```

\stexpatchmodule

```

1950 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1951 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1952
1953 \newcommand\stexpatchmodule[3] [] {
1954     \str_set:Nx \l_tmpa_str{ #1 }
1955     \str_if_empty:NTF \l_tmpa_str {
1956         \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1957         \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1958     }{
1959         \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }

```

```

1960     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1961   }
1962 }

```

(End definition for `\stexpatchmodule`. This function is documented on page 85.)

29.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n
1963 \NewDocumentCommand \STEXModule { m } {
1964   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1965   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1966   \tl_set:Nn \l_tmpa_tl {
1967     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1968   }
1969   \seq_map_inline:Nn \l_stex_all_modules_seq {
1970     \str_set:Nn \l_tmpb_str { ##1 }
1971     \str_if_eq:eeT { \l_tmpa_str } {
1972       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1973     } {
1974       \seq_map_break:n {
1975         \tl_set:Nn \l_tmpa_tl {
1976           \stex_invoke_module:n { ##1 }
1977         }
1978       }
1979     }
1980   }
1981   \l_tmpa_tl
1982 }
1983
1984 \cs_new_protected:Nn \stex_invoke_module:n {
1985   \stex_debug:nn{modules}{Invoking~module~#1}
1986   \peek_charcode_remove:NTF ! {
1987     \__stex_modules_invoke_uri:nN { #1 }
1988   } {
1989     \peek_charcode_remove:NTF ? {
1990       \__stex_modules_invoke_symbol:nn { #1 }
1991     } {
1992       \msg_error:nnx{stex}{error/syntax}{
1993         ?~or~!~expected~after~
1994         \c_backslash_str STEXModule{#1}
1995       }
1996     }
1997   }
1998 }
1999
2000 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
2001   \str_set:Nn #2 { #1 }
2002 }
2003
2004 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
2005   \stex_invoke_symbol:n{#1?#2}
2006 }

```

(End definition for `\STEXModule` and `\stex_invoke_module:n`. These functions are documented on page 85.)

`\stex_activate_module:n`

```

2007 \bool_new:N \l_stex_in_meta_bool
2008 \bool_set_false:N \l_stex_in_meta_bool
2009 \cs_new_protected:Nn \stex_activate_module:n {
2010   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
2011     \stex_debug:nn{modules}{Activating module~#1}
2012     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
2013     \use:c{ c_stex_module_#1_code }
2014   }
2015 }
```

(End definition for `\stex_activate_module:n`. This function is documented on page 86.)

`mmtinterface (env.)`

```

2016 \NewDocumentEnvironment { mmtinterface } { 0{} m m } {
2017   \stex_module_setup:nn{#1}{#3}
2018   %\par
2019   \stex_if_smsmode:F{
2020     \tl_if_empty:NF \smoduletitle {
2021       \exp_args:No \stex_document_title:n \smoduletitle
2022     }
2023     \tl_clear:N \l_tmpa_tl
2024     \clist_map_inline:Nn \smoduletype {
2025       \tl_if_exist:cT {__stex_modules_smodule_#1_start:}{
2026         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_#1_start:}}
2027       }
2028     }
2029     \tl_if_empty:NTF \l_tmpa_tl {
2030       \__stex_modules_smodule_start:
2031     }{
2032       \l_tmpa_tl
2033     }
2034   }
2035   \__stex_modules_begin_module:
2036   \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
2037     \exp_args:Nx \stex_add_to_current_module:n{
2038       \stex_activate_module:n {\l_stex_module_meta_str}
2039     }
2040   }
2041   \str_if_empty:NF \smoduleid {
2042     \stex_ref_new_doc_target:n \smoduleid
2043   }
2044   \str_set:Nx \l_stex_module_mmtfor_str {#2}
2045   \MMTinclude{#2}
2046   \stex_reactivate_macro:N \mmtdecl
2047   \stex_reactivate_macro:N \mmtdef
2048   \stex_smsmode_do:
2049 }{
2050   \__stex_modules_end_module:
2051   \stex_if_smsmode:F {
2052     \end{stex_annotate_env}
```

```

2053 \clist_set:No \l_tmpa_clist \smodulotype
2054 \tl_clear:N \l_tmpa_tl
2055 \clist_map_inline:Nn \l_tmpa_clist {
2056   \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
2057     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
2058   }
2059 }
2060 \tl_if_empty:NTF \l_tmpa_tl {
2061   __stex_modules_smodule_end:
2062 }{
2063   \l_tmpa_tl
2064 }
2065 }
2066 }
2067 </package>

```


Chapter 30

STEX -Module Inheritance Implementation

```
2068 <*package>
2069
2070 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
2071
```

30.1 SMS Mode

```
2072 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq
2073 \tl_new:N \g_stex_smsmode_allowedmacros_tl
2074 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
2075 \seq_new:N \g_stex_smsmode_allowedenvs_seq
2076
2077 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
2078   \makeatletter
2079   \makeatother
2080   \ExplSyntaxOn
2081   \ExplSyntaxOff
2082   \rustexBREAK
2083 }
2084
2085 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
2086   \symdef
2087   \importmodule
2088   \notation
2089   \symdecl
2090   \STEXexport
2091   \inlineass
2092   \inlinedef
2093   \inlineex
2094   \endinput
2095   \setnotation
```

```

2096 \copynotation
2097 \assign
2098 \renamedekl
2099 \donotcopy
2100 \instantiate
2101 \textsymdecl
2102 \mmtdef
2103 \setmetatheory
2104 }
2105
2106 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
2107   \tl_to_str:n {
2108     smodule,
2109     copymodule,
2110     interpretmodule,
2111     realization,
2112     sdefinition,
2113     sexample,
2114     sassertion,
2115     sparagraph,
2116     mmtinterface,
2117     mathstructure,
2118     extstructure,
2119     extstructure*
2120   }
2121 }

(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
2122 \bool_new:N \g__stex_smsmode_bool
2123 \bool_set_false:N \g__stex_smsmode_bool
2124 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
2125   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
2126 }

```

(End definition for \stex_if_smsmode:TF. This function is documented on page 87.)

```

\__stex_smsmode_in_smsmode:nn
2127 \cs_new_protected:Nn \__stex_smsmode_in_smsmode:nn { \stex_suppress_html:n {
2128   \vbox_set:Nn \l_tmpa_box {
2129     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
2130     \bool_gset_true:N \g__stex_smsmode_bool
2131     #2
2132     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
2133   }
2134   \box_clear:N \l_tmpa_box
2135 } }

```

(End definition for __stex_smsmode_in_smsmode:nn.)

```

\stex_file_in_smsmode:nn
2136 \quark_new:N \q__stex_smsmode_break
2137

```

```

2138 \NewDocumentCommand \__stex_smsmode_importmodule: { 0{} m} {
2139   \seq_gput_right:Nn \l__stex_smsmode_importmodules_seq {{#1}{#2}}
2140   \stex_smsmode_do:
2141 }
2142
2143 \cs_new_protected:Nn \__stex_smsmode_module:nn {
2144   \__stex_modules_args:n{#1}
2145   \stex_if_in_module:F {
2146     \str_if_empty:NF \l_stex_module_sig_str {
2147       \stex_modules_current_namespace:
2148       \str_set:Nx \l_stex_module_name_str { #2 }
2149       \stex_if_module_exists:nF{\l_stex_module_ns_str?\l_stex_module_name_str}{
2150         \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
2151         \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
2152         \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
2153         \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
2154         \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
2155         \str_set:Nx \l_tmpa_str {
2156           \stex_path_to_string:N \l_tmpa_seq /
2157           \l_tmpa_str . \l_stex_module_sig_str .tex
2158         }
2159         \IfFileExists \l_tmpa_str {
2160           \exp_args:NNx \seq_gput_right:Nn \l__stex_smsmode_sigmodules_seq \l_tmpa_str
2161         }{
2162           \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
2163         }
2164       }
2165     }
2166   }
2167 }
2168
2169 \prg_new_conditional:Nnn \__stex_smsmode_check_import_pair:nn {T,F,TF} {
2170   %\stex_debug:nn{import-pair}{\detokenize{{#1}~{#2}}}}
2171   \tl_if_empty:nTF{#1}{
2172     \prop_if_exist:NTF \l_stex_current_repository_prop
2173     {
2174       %\stex_debug:nn{import-pair}{in repository \prop_item:Nn \l_stex_current_repository_
2175       \prg_return_true:
2176     } {
2177       \seq_set_split:Nnn \l_tmpa_seq ? {#2}
2178       \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
2179       \tl_if_empty:NT \l_tmpa_tl {
2180         \seq_pop_left:NN \l_tmpa_seq \l_tmpa_tl
2181       }
2182       %\stex_debug:nn{import-pair}{\seq_use:Nn \l_tmpa_seq,~of~length~\seq_count:N \l_tmpa_
2183       \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1
2184       \prg_return_true: \prg_return_false:
2185     }
2186   }\prg_return_true:
2187 }
2188
2189 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
2190   \stex_filestack_push:n{#1}
2191   \seq_gclear:N \l__stex_smsmode_importmodules_seq

```

```

2192 \seq_gclear:N \l__stex_smsmode_sigmodules_seq
2193 % ----- new -----
2194 \__stex_smsmode_in_smsmode:nn{#1}{
2195   \let\importmodule\__stex_smsmode_importmodule:
2196   \let\stex_module_setup:nn\__stex_smsmode_module:nn
2197   \let\__stex_modules_begin_module:\relax
2198   \let\__stex_modules_end_module:\relax
2199   \seq_clear:N \g_stex_smsmode_allowedenvs_seq
2200   \exp_args:NNx \seq_put_right:Nn \g_stex_smsmode_allowedenvs_seq {\tl_to_str:n{smodule}}
2201   \tl_clear:N \g_stex_smsmode_allowedmacros_tl
2202   \tl_clear:N \g_stex_smsmode_allowedmacros_escape_tl
2203   \tl_put_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\importmodule}
2204   \everyeof{\q__stex_smsmode_break\noexpand}
2205   \expandafter\expandafter\expandafter
2206   \stex_smsmode_do:
2207   \csname @ @ input\endcsname "#1"\relax
2208
2209   \seq_map_inline:Nn \l__stex_smsmode_sigmodules_seq {
2210     \stex_filestack_push:n{##1}
2211     \expandafter\expandafter\expandafter
2212     \stex_smsmode_do:
2213     \csname @ @ input\endcsname "##1"\relax
2214     \stex_filestack_pop:
2215   }
2216 }
2217 % ----- new -----
2218 \__stex_smsmode_in_smsmode:nn{#1} {
2219   #2
2220   % ----- new -----
2221   \begingroup
2222   %\stex_debug:nn{smsmode}{Here:~\seq_use:Nn\l__stex_smsmode_importmodules_seq, }
2223   \seq_map_inline:Nn \l__stex_smsmode_importmodules_seq {
2224     \__stex_smsmode_check_import_pair:nnT ##1 { \begingroup
2225       \stex_import_module_uri:nn ##1
2226       \stex_import_require_module:nnnn
2227       \l_stex_import_ns_str
2228       \l_stex_import_archive_str
2229       \l_stex_import_path_str
2230       \l_stex_import_name_str \endgroup
2231     }
2232   }
2233   \endgroup
2234   \stex_debug:nn{smsmode}{Actually~loading~file~#1}
2235   % ----- new -----
2236   \everyeof{\q__stex_smsmode_break\noexpand}
2237   \expandafter\expandafter\expandafter
2238   \stex_smsmode_do:
2239   \csname @ @ input\endcsname "#1"\relax
2240 }
2241 \stex_filestack_pop:
2242 }

```

(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:

```

2243 \cs_new_protected:Npn \stex_smsmode_do: {
2244   \stex_if_smsmode:T {
2245     \__stex_smsmode_do:w
2246   }
2247 }
2248 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
2249   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
2250     \expandafter\if\expandafter\relax\noexpand#1
2251     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
2252   \else\expandafter\__stex_smsmode_do:w\fi
2253 }{
2254   \__stex_smsmode_do:w % #1
2255 }
2256 }
2257 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
2258   \cs_if_eq:NNF #1 \q__stex_smsmode_break {
2259     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
2260       #1\__stex_smsmode_do:w
2261     }{
2262       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
2263         #1
2264       }{
2265         \cs_if_eq:NNTF \begin #1 {
2266           \__stex_smsmode_check_begin:n
2267         }{
2268           \cs_if_eq:NNTF \end #1 {
2269             \__stex_smsmode_check_end:n
2270           }{
2271             \__stex_smsmode_do:w
2272           }
2273         }
2274       }
2275     }
2276   }
2277 }
2278
2279 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
2280   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
2281     \begin{#1}
2282   }{
2283     \__stex_smsmode_do:w
2284   }
2285 }
2286 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
2287   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
2288     \end{#1}\__stex_smsmode_do:w
2289   }{
2290     \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
2291   }
2292 }

```

(End definition for `\stex_smsmode_do:.` This function is documented on page 88.)

30.2 Inheritance

2293 <@@=stex_importmodule>

\stex_import_module_uri:nn

```

2294 \cs_new_protected:Nn \stex_import_module_uri:nn {
2295   \str_set:Nx \l_stex_import_archive_str { #1 }
2296   \str_set:Nn \l_stex_import_path_str { #2 }
2297
2298   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
2299   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
2300   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
2301
2302   \stex_modules_current_namespace:
2303   \bool_lazy_all:nTF {
2304     {\str_if_empty_p:N \l_stex_import_archive_str}
2305     {\str_if_empty_p:N \l_stex_import_path_str}
2306     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
2307   }{
2308     \str_set_eq:NN \l_stex_import_path_str \l_stex_module_subpath_str
2309     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
2310   }{
2311     \str_if_empty:NT \l_stex_import_archive_str {
2312       \prop_if_exist:NT \l_stex_current_repository_prop {
2313         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
2314       }
2315     }
2316     \str_if_empty:NTF \l_stex_import_archive_str {
2317       \str_if_empty:NF \l_stex_import_path_str {
2318         \stex_path_from_string:Nn \l_tmpb_seq {
2319           \l_stex_module_ns_str / .. / \l_stex_import_path_str
2320         }
2321         \str_set:Nx \l_stex_import_ns_str {\stex_path_to_string:N \l_tmpb_seq}
2322         \str_replace_once:Nnn \l_stex_import_ns_str {file://} {file://}
2323       }
2324     }{
2325       \stex_require_repository:n \l_stex_import_archive_str
2326       \prop_get:cnN { c_stex_mathhub_\l_stex_import_archive_str _manifest_prop } { ns }
2327       \l_stex_import_ns_str
2328       \str_if_empty:NF \l_stex_import_path_str {
2329         \str_set:Nx \l_stex_import_ns_str {
2330           \l_stex_import_ns_str / \l_stex_import_path_str
2331         }
2332       }
2333     }
2334   }
2335 }
```

(End definition for \stex_import_module_uri:nn. This function is documented on page 89.)

\l_stex_import_name_str
\l_stex_import_archive_str
\l_stex_import_path_str
\l_stex_import_ns_str

Store the return values of \stex_import_module_uri:nn.

```

2336 \str_new:N \l_stex_import_name_str
2337 \str_new:N \l_stex_import_archive_str
2338 \str_new:N \l_stex_import_path_str
2339 \str_new:N \l_stex_import_ns_str
```

(End definition for `\l_stex_import_name_str` and others. These variables are documented on page 89.)

```

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

2340 \cs_new_protected:Nn \stex_import_require_module:nnnn {
2341   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
2342
2343     \stex_debug:nn{requiremodule}{Here:\!\!\sim1:\sim1\!\!\sim2:\sim2\!\!\sim3:\sim3\!\!\sim4:\sim4}
2344
2345     \exp_args:NNxx \seq_set_split:Nnn \l_tmpa_seq {\tl_to_str:n{/}} {#4}
2346     \seq_get_left:NN \l_tmpa_seq \l_tmpc_str
2347
2348     %\stex_debug:nn{requiremodule}{Top-module:\l_tmpc_str}
2349
2350     % archive
2351     \str_set:Nx \l_tmpa_str { #2 }
2352     \str_if_empty:NTF \l_tmpa_str {
2353       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
2354       \seq_put_right:Nn \l_tmpa_seq {...}
2355     } {
2356       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
2357       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
2358       \seq_put_right:Nn \l_tmpa_seq { source }
2359     }
2360
2361     % path
2362     \str_set:Nx \l_tmpb_str { #3 }
2363     \str_if_empty:NTF \l_tmpb_str {
2364       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / \l_tmpc_str }
2365
2366       \ltx@ifpackageloaded{babel} {
2367         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
2368           { \language_name } \l_tmpb_str {
2369           \msg_error:nnx{stex}{error/unknownlanguage}{\language_name}
2370         }
2371       } {
2372         \str_clear:N \l_tmpb_str
2373       }
2374
2375       \stex_debug:nn{modules}{Checking-a1~\l_tmpa_str.\l_tmpb_str.tex}
2376       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2377         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2378       }{
2379         \stex_debug:nn{modules}{Checking-a2~\l_tmpa_str.tex}
2380         \IfFileExists{ \l_tmpa_str.tex }{
2381           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2382         }{
2383           % try english as default
2384           \stex_debug:nn{modules}{Checking-a3~\l_tmpa_str.en.tex}
2385           \IfFileExists{ \l_tmpa_str.en.tex }{
2386             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2387           }{
2388             \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2389           }
2390         }

```

```

2391     }
2392
2393   } {
2394     \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
2395     \seq_concat:NNN \l_tmpb_seq \l_tmpa_seq \l_tmpb_seq
2396
2397     \ltx@ifpackageloaded{babel} {
2398       \exp_args:NnX \prop_get:NnNF \c_stex_language_abbrevs_prop
2399         { \language } \l_tmpb_str {
2400         \msg_error:nnx{stex}{error/unknownlanguage}{\language}
2401       }
2402     } {
2403       \str_clear:N \l_tmpb_str
2404     }
2405
2406     \stex_path_canonicalize:N \l_tmpb_seq
2407     \stex_path_to_string:NN \l_tmpb_seq \l_tmpa_str
2408
2409     \stex_debug:nn{modules}{Checking~b1~\l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex}
2410     \IfFileExists{ \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }{
2411       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }
2412     }{
2413       \stex_debug:nn{modules}{Checking~b2~\l_tmpa_str/\l_tmpc_str.tex}
2414       \IfFileExists{ \l_tmpa_str/\l_tmpc_str.tex }{
2415         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.tex }
2416       }{
2417         % try english as default
2418         \stex_debug:nn{modules}{Checking~b3~\l_tmpa_str/\l_tmpc_str.en.tex}
2419         \IfFileExists{ \l_tmpa_str/\l_tmpc_str.en.tex }{
2420           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.en.tex }
2421         }{
2422           \stex_debug:nn{modules}{Checking~b4~\l_tmpa_str.\l_tmpb_str.tex}
2423           \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2424             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2425           }{
2426             \stex_debug:nn{modules}{Checking~b4~\l_tmpa_str.tex}
2427             \IfFileExists{ \l_tmpa_str.tex }{
2428               \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2429             }{
2430               % try english as default
2431               \stex_debug:nn{modules}{Checking~b5~\l_tmpa_str.en.tex}
2432               \IfFileExists{ \l_tmpa_str.en.tex }{
2433                 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2434               }{
2435                 \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2436               }
2437             }
2438           }
2439         }
2440       }
2441     }
2442   }
2443
2444   \str_if_eq:eeF{\g__stex_importmodule_file_str}{\seq_use:Nn \g_stex_currentfile_seq /}{

```



```

2445 \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
2446 \seq_clear:N \l_stex_all_modules_seq
2447 \str_clear:N \l_stex_current_module_str
2448 \str_set:Nx \l_tmpb_str { #2 }
2449 \str_if_empty:NF \l_tmpb_str {
2450 \stex_set_current_repository:n { #2 }
2451 }
2452 \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
2453 }
2454
2455 \stex_if_module_exists:nF { #1 ? #4 } {
2456 \msg_error:nnx{stex}{error/unknownmodule}{
2457 #1?#4~(in~file~\g__stex_importmodule_file_str)
2458 }
2459 }
2460 }
2461
2462 }
2463 \stex_activate_module:n { #1 ? #4 }
2464 }

```

(End definition for `\stex_import_require_module:nnnn`. This function is documented on page 89.)

`\importmodule`

```

2465 \NewDocumentCommand \importmodule { 0{} m } {
2466 \stex_import_module_uri:nn { #1 } { #2 }
2467 \stex_debug:nn{modules}{Importing~module:~
2468 \l_stex_import_ns_str ? \l_stex_import_name_str
2469 }
2470 \stex_if_smsmode:F {
2471 \stex_annotate_invisible:nnn
2472 {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
2473 }
2474 \stex_execute_in_module:x {
2475 \stex_import_require_module:nnnn
2476 { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2477 { \l_stex_import_path_str } { \l_stex_import_name_str }
2478 }
2479 \exp_args:Nx \stex_add_import_to_current_module:n {
2480 \l_stex_import_ns_str ? \l_stex_import_name_str
2481 }
2482 \stex_smsmode_do:
2483 \ignorespacesandpars
2484 }
2485 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

(End definition for `\importmodule`. This function is documented on page 88.)

`\usemodule`

```

2486 \NewDocumentCommand \usemodule { 0{} m } {
2487 \stex_if_smsmode:F {
2488 \stex_import_module_uri:nn { #1 } { #2 }
2489 \stex_import_require_module:nnnn
2490 { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2491 { \l_stex_import_path_str } { \l_stex_import_name_str }

```

```

2492 \stex_annotate_invisible:nnn
2493 {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
2494 }
2495 \stex_smsmode_do:
2496 \ignorespacesandpars
2497 }

(End definition for \usemodule. This function is documented on page 88.)

2498 \cs_new_protected:Nn \stex_csl_to_imports:Nn {
2499   \tl_if_empty:nF{#2}{
2500     \clist_set:Nn \l_tmpa_clist {#2}
2501     \clist_map_inline:Nn \l_tmpa_clist {
2502       \tl_if_head_eq_charcode:nNTF {##1}[{
2503         #1 ##1
2504       }{
2505         #1{##1}
2506       }
2507     }
2508   }
2509 }
2510 \cs_generate_variant:Nn \stex_csl_to_imports:Nn {No}
2511
2512
2513 \endpackage

```

Chapter 31

STEX -Symbols Implementation

```
2514 <*package>
2515
2516 %%%%%%%%%%%%% symbols.dtx %%%%%%%%%%%%%
2517
2518 Warnings and error messages
2518 \msg_new:nnn{stex}{error/wrongargs}{
2519   args~value~in~symbol~declaration~for~#1~
2520   needs~to~be~i,~a,~b~or~B,~but~#2~given
2521 }
2522 \msg_new:nnn{stex}{error/unknownsymbol}{
2523   No~symbol~#1~found!
2524 }
2525 \msg_new:nnn{stex}{error/seqlength}{
2526   Expected~#1~arguments;~got~#2!
2527 }
2528 \msg_new:nnn{stex}{error/unknownnotation}{
2529   Unknown~notation~#1~for~#2!
2530 }
```

31.1 Symbol Declarations

```
2531 <@@=stex_symdecl>
\stex_all_symbols:n Map over all available symbols
2532 \cs_new_protected:Nn \stex_all_symbols:n {
2533   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
2534   \seq_map_inline:Nn \l_stex_all_modules_seq {
2535     \seq_map_inline:cn{c_stex_module_##1_constants}{
2536       \__stex_symdecl_all_symbols_cs{##1?####1}
2537     }
2538   }
2539 }
```

(End definition for `\stex_all_symbols:n`. This function is documented on page 91.)

`\STEXsymbol`

```
2540 \NewDocumentCommand \STEXsymbol { m } {
2541   \stex_get_symbol:n { #1 }
2542   \exp_args:No
2543   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
2544 }
```

(End definition for `\STEXsymbol`. This function is documented on page 92.)

`symdecl` arguments:

```
2545 \keys_define:nn { stex / symdecl } {
2546   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2547   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2548   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
2549   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,
2550   align     .str_set:N    = \l_stex_symdecl_align_str , % TODO(?)
2551   gfc       .str_set:N    = \l_stex_symdecl_gfc_str , % TODO(?)
2552   def       .tl_set:N    = \l_stex_symdecl_definiens_tl ,
2553   reorder   .str_set_x:N = \l_stex_symdecl_reorder_str ,
2554   argnames  .clist_set:N = \l_stex_symdecl_argnames_clist ,
2555   assoc     .choices:nn =
2556             {bin,binl,binr,pre,conj,pwconj}
2557             {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}
2558 }
2559
2560 \bool_new:N \l_stex_symdecl_make_macro_bool
2561
2562 \cs_new_protected:Nn \__stex_symdecl_args:n {
2563   \str_clear:N \l_stex_symdecl_name_str
2564   \str_clear:N \l_stex_symdecl_args_str
2565   \str_clear:N \l_stex_symdecl_deprecate_str
2566   \str_clear:N \l_stex_symdecl_reorder_str
2567   \str_clear:N \l_stex_symdecl_assoctype_str
2568   \bool_set_false:N \l_stex_symdecl_local_bool
2569   \tl_clear:N \l_stex_symdecl_type_tl
2570   \tl_clear:N \l_stex_symdecl_definiens_tl
2571   \clist_clear:N \l_stex_symdecl_argnames_clist
2572
2573   \keys_set:nn { stex / symdecl } { #1 }
2574 }
```

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

```
2575
2576 \NewDocumentCommand \symdecl { s m O{} } {
2577   \__stex_symdecl_args:n { #3 }
2578   \IfBooleanTF #1 {
2579     \bool_set_false:N \l_stex_symdecl_make_macro_bool
2580   } {
2581     \bool_set_true:N \l_stex_symdecl_make_macro_bool
2582   }
2583   \stex_symdecl_do:n { #2 }
2584   \stex_smsmode_do:
2585 }
```

```

2586
2587 \cs_new_protected:Nn \stex_symdecl_do:nn {
2588   \__stex_symdecl_args:n{#1}
2589   \bool_set_false:N \l_stex_symdecl_make_macro_bool
2590   \stex_symdecl_do:n{#2}
2591 }
2592
2593 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

(End definition for \symdecl. This function is documented on page 90.)

\stex_symdecl_do:n

```

2594 \cs_new_protected:Nn \stex_symdecl_do:n {
2595   \stex_if_in_module:F {
2596     % TODO throw error? some default namespace?
2597   }
2598
2599   \str_if_empty:NT \l_stex_symdecl_name_str {
2600     \str_set:Nx \l_stex_symdecl_name_str { #1 }
2601   }
2602
2603   \prop_if_exist:cT { l_stex_symdecl_
2604     \l_stex_current_module_str ?
2605     \l_stex_symdecl_name_str
2606     _prop
2607   }{
2608     % TODO throw error (beware of circular dependencies)
2609   }
2610
2611   \prop_clear:N \l_tmpa_prop
2612   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
2613   \seq_clear:N \l_tmpa_seq
2614   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
2615   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
2616
2617   \str_if_empty:NT \l_stex_symdecl_deprecate_str {
2618     \str_if_empty:NF \l_stex_module_deprecate_str {
2619       \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
2620     }
2621   }
2622   \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
2623
2624   \exp_args:No \stex_add_constant_to_current_module:n {
2625     \l_stex_symdecl_name_str
2626   }
2627
2628   % arity/args
2629   \int_zero:N \l_tmpb_int
2630
2631   \bool_set_true:N \l_tmpa_bool
2632   \str_map_inline:Nn \l_stex_symdecl_args_str {
2633     \token_case_meaning:NnF ##1 {
2634       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2635       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }

```

```

2636     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2637     {\tl_to_str:n a} {
2638         \bool_set_false:N \l_tmpa_bool
2639         \int_incr:N \l_tmpb_int
2640     }
2641     {\tl_to_str:n B} {
2642         \bool_set_false:N \l_tmpa_bool
2643         \int_incr:N \l_tmpb_int
2644     }
2645     ){
2646         \msg_error:nnxx{stex}{error/wrongargs}{
2647             \l_stex_current_module_str ?
2648             \l_stex_symdecl_name_str
2649         }{##1}
2650     }
2651 }
2652
2653 \bool_if:NTF \l_tmpa_bool {
2654     % possibly numeric
2655     \str_if_empty:NTF \l_stex_symdecl_args_str {
2656         \prop_put:Nnn \l_tmpa_prop { args } {}
2657         \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2658     }{
2659         \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
2660         \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2661         \str_clear:N \l_tmpa_str
2662         \int_step_inline:nn \l_tmpa_int {
2663             \str_put_right:Nn \l_tmpa_str i
2664         }
2665         \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
2666     }
2667 } {
2668     \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
2669     \prop_put:Nnx \l_tmpa_prop { arity }
2670     { \str_count:N \l_stex_symdecl_args_str }
2671 }
2672 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2673
2674 \tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
2675     \prop_put:Nnx \l_tmpa_prop { defined }{ false }
2676 }{
2677     \prop_put:Nnx \l_tmpa_prop { defined }{ true }
2678 }
2679
2680 % argnames
2681
2682 \clist_clear:N \l_tmpa_clist
2683 \int_step_inline:nn {\prop_item:Nn \l_tmpa_prop {arity}} {
2684     \clist_if_empty:NTF \l_stex_symdecl_argnames_clist {
2685         \clist_put_right:Nn \l_tmpa_clist {##1}
2686     }{
2687         \clist_pop:NN \l_stex_symdecl_argnames_clist \l_tmpa_tl
2688         \exp_args:NNx \clist_put_right:Nn \l_tmpa_clist {\c_dollar_str\l_tmpa_tl}
2689     }

```

```

2690 }
2691 \prop_put:Nn \l_tmpa_prop {argnames} {\clist_use:Nn \l_tmpa_clist ,}
2692
2693 % semantic macro
2694
2695 \bool_if:NT \l_stex_symdecl_make_macro_bool {
2696   \exp_args:Nx \stex_do_up_to_module:n {
2697     \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2698       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2699     }}
2700   }
2701 }
2702
2703 \stex_debug:nn{symbols}{New~symbol:~
2704   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2705   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2706   Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
2707   Definiens:~\exp_not:o { \l_stex_symdecl_definiens_tl}
2708 }
2709
2710 % circular dependencies require this:
2711 \stex_if_do_html:T {
2712   \stex_annotate_invisible:nnn {symdecl} {
2713     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2714   } {
2715     \tl_if_empty:NF \l_stex_symdecl_type_tl {
2716       \stex_annotate_invisible:nnn{type}{\l_stex_symdecl_type_tl$}
2717     }
2718     \stex_annotate_invisible:nnn{args}{\prop_item:Nn \l_tmpa_prop { args }}{}
2719     \stex_annotate_invisible:nnn{macroname}{#1}{}
2720     \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2721       \stex_annotate_invisible:nnn{definiens}{}
2722       {\l_stex_symdecl_definiens_tl$}
2723     }
2724     \str_if_empty:NF \l_stex_symdecl_assoc_type_str {
2725       \stex_annotate_invisible:nnn{assoc_type}{\l_stex_symdecl_assoc_type_str}{}
2726     }
2727     \str_if_empty:NF \l_stex_symdecl_reorder_str {
2728       \stex_annotate_invisible:nnn{reorderargs}{\l_stex_symdecl_reorder_str}{}
2729     }
2730   }
2731 }
2732 \prop_if_exist:cF {
2733   \l_stex_symdecl_
2734   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2735   _prop
2736 } {
2737   \bool_if:NTF \l_stex_symdecl_local_bool \stex_do_up_to_module:x \stex_execute_in_module:
2738     \__stex_symdecl_restore_symbol:nnnnnnnn
2739       {\l_stex_symdecl_name_str}
2740       { \prop_item:Nn \l_tmpa_prop {args} }
2741       { \prop_item:Nn \l_tmpa_prop {arity} }
2742       { \prop_item:Nn \l_tmpa_prop {assoc} }
2743       { \prop_item:Nn \l_tmpa_prop {defined} }

```

```

2744         {\bool_if:NT \l_stex_symdecl_make_macro_bool {#1} }
2745         {\l_stex_current_module_str}
2746         { \prop_item:Nn \l_tmpa_prop {argnames} }
2747     }
2748 }
2749 }
2750 \cs_new_protected:Nn \__stex_symdecl_restore_symbol:nnnnnnnn {
2751     \prop_clear:N \l_tmpa_prop
2752     \prop_put:Nnn \l_tmpa_prop { module } { #7 }
2753     \prop_put:Nnn \l_tmpa_prop { name } { #1 }
2754     \prop_put:Nnn \l_tmpa_prop { args } { #2 }
2755     \prop_put:Nnn \l_tmpa_prop { arity } { #3 }
2756     \prop_put:Nnn \l_tmpa_prop { assocs } { #4 }
2757     \prop_put:Nnn \l_tmpa_prop { defined } { #5 }
2758     \prop_put:Nnn \l_tmpa_prop { argnames } { #8 }
2759     \tl_if_empty:nF{#6}{
2760         \tl_set:cx{#6}{\stex_invoke_symbol:n{\detokenize{#7 ? #1}}}
2761     }
2762     \prop_set_eq:cN{\l_stex_symdecl_ \detokenize{#7 ? #1} _prop}\l_tmpa_prop
2763     \seq_clear:c{\l_stex_symdecl_ \detokenize{#7 ? #1} _notations}
2764 }

```

(End definition for `\stex_symdecl_do:n`. This function is documented on page 91.)

`\textsymdecl`

```

2765
2766 \keys_define:nn { stex / textsymdecl } {
2767     name      .str_set_x:N = \l__stex_symdecl_name_str ,
2768     type      .tl_set:N    = \l__stex_symdecl_type_tl
2769 }
2770
2771 \cs_new_protected:Nn \stex_textsymdecl_args:n {
2772     \str_clear:N \l__stex_symdecl_name_str
2773     \tl_clear:N \l__stex_symdecl_type_tl
2774     \clist_clear:N \l_stex_symdecl_argnames_clist
2775     \keys_set:nn { stex / textsymdecl } { #1 }
2776 }
2777
2778 \NewDocumentCommand \textsymdecl {m O{} m} {
2779     \stex_textsymdecl_args:n { #2 }
2780     \str_if_empty:NTF \l__stex_symdecl_name_str {
2781         \__stex_symdecl_args:n{name=#1,#2}
2782     }{
2783         \__stex_symdecl_args:n{#2}
2784     }
2785     \bool_set_true:N \l_stex_symdecl_make_macro_bool
2786     \stex_symdecl_do:n{#1-sym}
2787     \stex_execute_in_module:n{
2788         \cs_set_nopar:cpn{#1name}{
2789             \ifvmode\hbox_unpack:N\c_empty_box\fi
2790             \ifmmode\hbox{#3}\else#3\fi\hspace
2791         }
2792         \cs_set_nopar:cpn{#1}{
2793             \ifmmode\csname#1-sym\expandafter\endcsname\else

```



```

2794     \ifvmode\hbox_unpack:N\c_empty_box\fi
2795     \symref{#1-sym}{#3}\expandafter\xspace
2796     \fi
2797   }
2798 }
2799 \stex_execute_in_module:x{
2800   \__stex_notation_restore_notation:nnnnn
2801   {\l_stex_current_module_str?\tl_if_empty:NTF\l__stex_symdecl_name_str{#1}\l__stex_symdecl_name_str{#1}}{0}
2802   {\exp_not:n{\STEXInternalTermMathOMSiiii{\STEXInternalCurrentSymbolStr}{}}{\neginfprec}{\neginfprec}}{0}
2803   \comp{\hbox{#3}}\STEXInternalSymbolAfterInvokationTL
2804   }}}
2805   {}
2806 }
2807 }
2808 \stex_smsmode_do:
2809 }

```

(End definition for `\textsymdecl`. This function is documented on page 23.)

`\stex_get_symbol:n`

```

2810 \str_new:N \l_stex_get_symbol_uri_str
2811
2812 \cs_new_protected:Nn \stex_get_symbol:n {
2813   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2814     \tl_set:Nn \l_tmpa_tl { #1 }
2815     \__stex_symdecl_get_symbol_from_cs:
2816   }{
2817     % argument is a string
2818     % is it a command name?
2819     \cs_if_exist:cTF { #1 }{
2820       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2821       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2822       \str_if_empty:NTF \l_tmpa_str {
2823         \exp_args:Nx \cs_if_eq:NNTF {
2824           \tl_head:N \l_tmpa_tl
2825         } \stex_invoke_symbol:n {
2826           \__stex_symdecl_get_symbol_from_cs:
2827         }{
2828           \__stex_symdecl_get_symbol_from_string:n { #1 }
2829         }
2830       } {
2831         \__stex_symdecl_get_symbol_from_string:n { #1 }
2832       }
2833     }{
2834       % argument is not a command name
2835       \__stex_symdecl_get_symbol_from_string:n { #1 }
2836       % \l_stex_all_symbols_seq
2837     }
2838   }
2839   \str_if_eq:eeF {
2840     \prop_item:cn {
2841       \l_stex_symdecl\l_stex_get_symbol_uri_str _prop
2842     }{ deprecate }
2843   }{}{

```

```

2844 \msg_warning:nnxx{stex}{warning/deprecated}{
2845 Symbol~\l_stex_get_symbol_uri_str
2846 }{
2847 \prop_item:cn {l_stex_symdecl_l\l_stex_get_symbol_uri_str _prop}{ deprecate }
2848 }
2849 }
2850 }
2851
2852 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2853 \tl_set:Nn \l_tmpa_tl {
2854 \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2855 }
2856 \str_set:Nn \l_tmpa_str { #1 }
2857
2858 %\int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2859
2860 \str_if_in:NnTF \l_tmpa_str ? {
2861 \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
2862 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
2863 \str_set:Nx \l_tmpb_str {\seq_use:Nn \l_tmpa_seq ?}
2864 }{
2865 \str_clear:N \l_tmpb_str
2866 }
2867 \str_if_empty:NnTF \l_tmpb_str {
2868 \seq_map_inline:Nn \l_stex_all_modules_seq {
2869 \seq_map_inline:cn{c_stex_module_##1_constants}{
2870 \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2871 \seq_map_break:n{\seq_map_break:n{
2872 \tl_set:Nn \l_tmpa_tl {
2873 \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2874 }
2875 }}
2876 }
2877 }
2878 }
2879 }{
2880 \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpb_str }
2881 \seq_map_inline:Nn \l_stex_all_modules_seq {
2882 \str_if_eq:eeT{ \l_tmpb_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2883 \seq_map_inline:cn{c_stex_module_##1_constants}{
2884 \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2885 \seq_map_break:n{\seq_map_break:n{
2886 \tl_set:Nn \l_tmpa_tl {
2887 \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2888 }
2889 }}
2890 }
2891 }
2892 }
2893 }
2894 }
2895
2896 \l_tmpa_tl
2897 }

```

```

2898
2899 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
2900   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2901     { \tl_tail:N \l_tmpa_tl }
2902   \tl_if_single:NTF \l_tmpa_tl {
2903     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2904       \exp_after:wN \str_set:Nn \exp_after:wN
2905         \l_stex_get_symbol_uri_str \l_tmpa_tl
2906     }{
2907       % TODO
2908       % tail is not a single group
2909     }
2910   }{
2911     % TODO
2912     % tail is not a single group
2913   }
2914 }

```

(End definition for `\stex_get_symbol:n`. This function is documented on page 91.)

31.2 Notations

```

2915 <@@=stex_notation>
      notation arguments:
2916 \keys_define:nn { stex / notation } {
2917   % lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2918   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2919   prec     .str_set_x:N = \l__stex_notation_prec_str ,
2920   op       .tl_set:N = \l__stex_notation_op_tl ,
2921   primary .bool_set:N = \l__stex_notation_primary_bool ,
2922   primary .default:n = {true} ,
2923   hints    .str_set_x:N = \l__stex_notation_hints_str,
2924   unknown .code:n = \str_set:Nx
2925     \l__stex_notation_variant_str \l_keys_key_str
2926 }
2927
2928 \cs_new_protected:Nn \_stex_notation_args:n {
2929   % \str_clear:N \l__stex_notation_lang_str
2930   \str_clear:N \l__stex_notation_variant_str
2931   \str_clear:N \l__stex_notation_prec_str
2932   \str_clear:N \l__stex_notation_hints_str
2933   \tl_clear:N \l__stex_notation_op_tl
2934   \bool_set_false:N \l__stex_notation_primary_bool
2935
2936   \keys_set:nn { stex / notation } { #1 }
2937 }

```

\notation

```

2938 \NewDocumentCommand \notation { s m O{}} {
2939   \_stex_notation_args:n { #3 }
2940   \tl_clear:N \l_stex_symdecl_definiens_tl
2941   \stex_get_symbol:n { #2 }
2942   \tl_set:Nn \l_stex_notation_after_do_tl {

```

```

2943 \__stex_notation_final:
2944 \IfBooleanTF#1{
2945   \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2946 }{}
2947 \stex_smsmode_do:\ignorespacesandpars
2948 }
2949 \stex_notation_do:nnnnn
2950 { \prop_item:cn {\l_stex_symdecl\l_stex_get_symbol_uri_str _prop } { args } }
2951 { \prop_item:cn { \l_stex_symdecl\l_stex_get_symbol_uri_str _prop } { arity } }
2952 { \l__stex_notation_variant_str }
2953 { \l__stex_notation_prec_str }
2954 }
2955 \stex_deactivate_macro:Nn \notation {module-environments}

```

(End definition for \notation. This function is documented on page 91.)

\stex_notation_do:nnnnn

```

2956 \seq_new:N \l__stex_notation_precedences_seq
2957 \tl_new:N \l__stex_notation_opprec_tl
2958 \int_new:N \l__stex_notation_currarg_int
2959 \tl_new:N \STEXInternalSymbolAfterInvokationTL
2960
2961 \cs_new_protected:Nn \stex_notation_do:nnnnn {
2962   \let\STEXInternalCurrentSymbolStr\relax
2963   \seq_clear:N \l__stex_notation_precedences_seq
2964   \tl_clear:N \l__stex_notation_opprec_tl
2965   \str_set:Nx \l__stex_notation_args_str { #1 }
2966   \str_set:Nx \l__stex_notation_arity_str { #2 }
2967   \str_set:Nx \l__stex_notation_suffix_str { #3 }
2968   \str_set:Nx \l__stex_notation_prec_str { #4 }
2969
2970   % precedences
2971   \str_if_empty:NTF \l__stex_notation_prec_str {
2972     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2973       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2974     }{
2975       \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2976     }
2977   } {
2978     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2979       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2980       \int_step_inline:nn { \l__stex_notation_arity_str } {
2981         \exp_args:NNo
2982         \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2983       }
2984     }{
2985       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2986       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2987         \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2988         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2989           \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2990             \l_tmpa_seq {\tl_to_str:n{x}} } { \l_tmpa_str }
2991         \seq_map_inline:Nn \l_tmpa_seq {
2992           \seq_put_right:Nn \l__stex_notation_precedences_seq { ##1 }

```

```

2993     }
2994   }
2995   }{
2996     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2997       \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2998     }{
2999       \tl_set:No \l__stex_notation_opprec_tl { 0 }
3000     }
3001   }
3002 }
3003 }
3004
3005 \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
3006 \int_step_inline:nn { \l__stex_notation_arity_str } {
3007   \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
3008     \exp_args:NNo
3009     \seq_put_right:No \l__stex_notation_precedences_seq {
3010       \l__stex_notation_opprec_tl
3011     }
3012   }
3013 }
3014 \tl_clear:N \l_stex_notation_dummyargs_tl
3015
3016 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
3017   \exp_args:NNe
3018   \cs_set:Npn \l_stex_notation_macrocode_cs {
3019     \STEXInternalTermMathOMSiiii { \STEXInternalCurrentSymbolStr }
3020     { \l__stex_notation_suffix_str }
3021     { \l__stex_notation_opprec_tl }
3022     { \exp_not:n { #5 } }
3023   }
3024   \l_stex_notation_after_do_tl
3025 }{
3026   \str_if_in:NnTF \l__stex_notation_args_str b {
3027     \exp_args:Nne \use:nn
3028     {
3029       \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
3030       \cs_set:Npn \l__stex_notation_arity_str } { {
3031         \STEXInternalTermMathOMBiiii { \STEXInternalCurrentSymbolStr }
3032         { \l__stex_notation_suffix_str }
3033         { \l__stex_notation_opprec_tl }
3034         { \exp_not:n { #5 } }
3035       }
3036     }{
3037       \str_if_in:NnTF \l__stex_notation_args_str B {
3038         \exp_args:Nne \use:nn
3039         {
3040           \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
3041           \cs_set:Npn \l__stex_notation_arity_str } { {
3042             \STEXInternalTermMathOMBiiii { \STEXInternalCurrentSymbolStr }
3043             { \l__stex_notation_suffix_str }
3044             { \l__stex_notation_opprec_tl }
3045             { \exp_not:n { #5 } }
3046           } }

```

```

3047   }{
3048     \exp_args:Nne \use:nn
3049     {
3050       \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
3051       \cs_set:Npn \l__stex_notation_arity_str } { {
3052         \STEXInternalTermMathOMAiiii { \STEXInternalCurrentSymbolStr }
3053         { \l__stex_notation_suffix_str }
3054         { \l__stex_notation_opprec_tl }
3055         { \exp_not:n { #5 } }
3056       } }
3057     }
3058   }
3059
3060   \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
3061   \int_zero:N \l__stex_notation_currarg_int
3062   \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
3063   \__stex_notation_arguments:
3064 }
3065 }

```

(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

```

3066 \cs_new_protected:Nn \__stex_notation_arguments: {
3067   \int_incr:N \l__stex_notation_currarg_int
3068   \str_if_empty:NTF \l__stex_notation_remaining_args_str {
3069     \l_stex_notation_after_do_tl
3070   }{
3071     \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
3072     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
3073     \str_if_eq:VnTF \l_tmpa_str a {
3074       \__stex_notation_argument_assoc:nn{a}
3075     }{
3076       \str_if_eq:VnTF \l_tmpa_str B {
3077         \__stex_notation_argument_assoc:nn{B}
3078       }{
3079         \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpb_str
3080         \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
3081           { \STEXInternalTermMathArgiii
3082             { \l_tmpa_str\int_use:N \l__stex_notation_currarg_int }
3083             { \l_tmpb_str }
3084             { ###\int_use:N \l__stex_notation_currarg_int }
3085           }
3086         }
3087         \__stex_notation_arguments:
3088       }
3089     }
3090   }
3091 }

```

(End definition for `__stex_notation_arguments:.`)

`__stex_notation_argument_assoc:nn`

```

3092 \cs_new_protected:Nn \__stex_notation_argument_assoc:nn {

```

```

3093
3094 \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
3095   {\l__stex_notation_arity_str}{
3096     #2
3097   }
3098 \int_zero:N \l_tmpa_int
3099 \tl_clear:N \l_tmpa_tl
3100 \str_map_inline:Nn \l__stex_notation_args_str {
3101   \int_incr:N \l_tmpa_int
3102   \tl_put_right:Nx \l_tmpa_tl {
3103     \str_if_eq:nnTF {##1}{a}{ } {} }{
3104     \str_if_eq:nnTF {##1}{B}{ } {} }{
3105     {\_stex_term_arg:nn{##1}\int_use:N \l_tmpa_int}{##### \int_use:N \l_tmpa_int}
3106   }
3107 }
3108 }
3109 }
3110 \exp_after:wN\exp_after:wN\exp_after:wN \def
3111 \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
3112 \exp_after:wN\exp_after:wN\exp_after:wN ##
3113 \exp_after:wN\exp_after:wN\exp_after:wN 1
3114 \exp_after:wN\exp_after:wN\exp_after:wN ##
3115 \exp_after:wN\exp_after:wN\exp_after:wN 2
3116 \exp_after:wN\exp_after:wN\exp_after:wN {
3117   \exp_after:wN \exp_after:wN \exp_after:wN
3118   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
3119     \exp_after:wN \l_tmpa_cs \l_tmpa_tl
3120   }
3121 }
3122
3123 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
3124 \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
3125   \STEXInternalTermMathAssocArgiiii
3126   { \int_use:N \l__stex_notation_currarg_int }
3127   { \l_tmpa_str }
3128   { ####\int_use:N \l__stex_notation_currarg_int }
3129   { \l_tmpa_cs {####1} {####2} }
3130   {#1}
3131 } }
3132 \__stex_notation_arguments:
3133 }

```

(End definition for _stex_notation_argument_assoc:nn.)

_stex_notation_final: Called after processing all notation arguments

```

3134 \cs_new_protected:Nn \__stex_notation_restore_notation:nnnnn {
3135   \cs_generate_from_arg_count:cNnn{stex_notation_\detokenize{#1} \c_hash_str \detokenize{#2}}
3136   \cs_set_nopar:Npn {#3}{#4}
3137   \tl_if_empty:nF {#5}{
3138     \tl_set:cn{stex_op_notation_\detokenize{#1} \c_hash_str \detokenize{#2}_cs}{\comp{ #5 }
3139   }
3140   \seq_if_exist:cT { l_stex_symdecl_\detokenize{#1} _notations }{
3141     \seq_put_right:cx { l_stex_symdecl_\detokenize{#1} _notations } { \detokenize{#2} }
3142   }

```

```

3143 }
3144
3145 \cs_new_protected:Nn \__stex_notation_final: {
3146
3147   \stex_execute_in_module:x {
3148     \__stex_notation_restore_notation:nnnnn
3149     {\l_stex_get_symbol_uri_str}
3150     {\l__stex_notation_suffix_str}
3151     {\l__stex_notation_arity_str}
3152     {
3153       \exp_after:wN \exp_after:wN \exp_after:wN
3154       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
3155       { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \STEXInt
3156     }
3157     {\exp_args:No \exp_not:n \l__stex_notation_op_tl }
3158   }
3159
3160   \stex_debug:nn{symbols}{
3161     Notation~\l__stex_notation_suffix_str
3162     ~for~\l_stex_get_symbol_uri_str^^J
3163     Operator~precedence:~\l__stex_notation_opprec_tl^^J
3164     Argument~precedences:~
3165     \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
3166     Notation: \cs_meaning:c {
3167       stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
3168       \l__stex_notation_suffix_str
3169       _cs
3170     }
3171   }
3172   % HTML annotations
3173   \stex_if_do_html:T {
3174     \stex_annotate_invisible:nnn { notation }
3175     { \l_stex_get_symbol_uri_str } {
3176       \stex_annotate_invisible:nnn { notationfragment }
3177       { \l__stex_notation_suffix_str }{}
3178       \stex_annotate_invisible:nnn { precedence }
3179       { \l__stex_notation_prec_str }{}
3180
3181       \int_zero:N \l_tmpa_int
3182       \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
3183       \tl_clear:N \l_tmpa_tl
3184       \int_step_inline:nn { \l__stex_notation_arity_str }{
3185         \int_incr:N \l_tmpa_int
3186         \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
3187         \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_rema
3188         \str_if_eq:VnTF \l_tmpb_str a {
3189           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3190             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3191             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3192           } }
3193         }{
3194           \str_if_eq:VnTF \l_tmpb_str B {
3195             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3196               \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,

```



```

3197         \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{\}
3198     } }
3199   }{
3200     \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3201       \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{\}
3202     } }
3203   }
3204 }
3205 }
3206 \stex_annotate_invisible:nnn { notationcomp }{ }{
3207   \str_set:Nx \STEXInternalCurrentSymbolStr {\l_stex_get_symbol_uri_str }
3208   $ \exp_args:Nno \use:nn { \use:c {
3209     stex_notation_ \STEXInternalCurrentSymbolStr
3210     \c_hash_str \l__stex_notation_suffix_str_cs
3211   } } { \l_tmpa_tl } $
3212 }
3213 \tl_if_empty:NF \l__stex_notation_op_tl {
3214   \stex_annotate_invisible:nnn { notationopcomp }{ }{
3215     $\l__stex_notation_op_tl$
3216   }
3217 }
3218 }
3219 }
3220 }

```

(End definition for `_stex_notation_final:`)

`\setnotation`

```

3221 \keys_define:nn { stex / setnotation } {
3222   % lang .tl_set_x:N = \l__stex_notation_lang_str ,
3223   variant .tl_set_x:N = \l__stex_notation_variant_str ,
3224   unknown .code:n = \str_set:Nx
3225     \l__stex_notation_variant_str \l_keys_key_str
3226 }
3227
3228 \cs_new_protected:Nn \_stex_setnotation_args:n {
3229   % \str_clear:N \l__stex_notation_lang_str
3230   \str_clear:N \l__stex_notation_variant_str
3231   \keys_set:nn { stex / setnotation } { #1 }
3232 }
3233
3234 \cs_new_protected:Nn \_stex_notation_setnotation:nn {
3235   \seq_if_exist:cT{l_stex_symdecl_#1_notations}{
3236     \seq_remove_all:cn { l_stex_symdecl_#1_notations }{ #2 }
3237     \seq_put_left:cn { l_stex_symdecl_#1_notations }{ #2 }
3238   }
3239 }
3240
3241 \cs_new_protected:Nn \stex_setnotation:n {
3242   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1_notations }
3243     { \l__stex_notation_variant_str }{
3244     \stex_execute_in_module:x{ \_stex_notation_setnotation:nn {#1}{\l__stex_notation_vari
3245     \stex_debug:nn {notations}{
3246       Setting~default~notation~

```

```

3247         {\l__stex_notation_variant_str }~for~
3248         #1 \\\
3249         \expandafter\meaning\csname
3250         l_stex_symdecl_#1 _notations\endcsname
3251     }
3252     ){
3253         \msg_error:nnxx{stex}{unknownnotation}{\l__stex_notation_variant_str}{#1}
3254     }
3255 }
3256
3257 \NewDocumentCommand \setnotation {m m} {
3258     \stex_get_symbol:n { #1 }
3259     \_stex_setnotation_args:n { #2 }
3260     \stex_setnotation:n{\l_stex_get_symbol_uri_str}
3261     \stex_smsmode_do:\ignorespacesandpars
3262 }
3263
3264 \cs_new_protected:Nn \stex_copy_notations:nn {
3265     \stex_debug:nn {notations}{
3266         Copying~notations~from~#2~to~#1\\
3267         \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
3268     }
3269     \tl_clear:N \l_tmpa_tl
3270     \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
3271         \tl_put_right:Nn \l_tmpa_tl { {##### #1} }
3272     }
3273     \seq_map_inline:cn {l_stex_symdecl_#2_notations}{\begingroup
3274         \stex_debug:nn{Here}{Here:~##1}
3275         \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
3276         \edef \l_tmpa_tl {
3277             \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
3278             \exp_after:wN\exp_after:wN\exp_after:wN {
3279                 \exp_after:wN \l_tmpa_cs \l_tmpa_tl
3280             }
3281         }
3282
3283         \exp_after:wN \def \exp_after:wN \l_tmpa_tl
3284         \exp_after:wN #####\exp_after:wN 1 \exp_after:wN #####\exp_after:wN 2
3285         \exp_after:wN { \l_tmpa_tl }
3286
3287         \edef \l_tmpa_tl {
3288             \exp_after:wN \exp_not:n \exp_after:wN {
3289                 \l_tmpa_tl {##### 1}{##### 2}
3290             }
3291         }
3292
3293         \stex_debug:nn{Here}{Here:~\expandafter\detokenize\expandafter{\l_tmpa_tl}}
3294
3295     \stex_execute_in_module:x {
3296         \_stex_notation_restore_notation:nnnnn
3297         {#1}{##1}
3298         { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }
3299         { \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl} }
3300         {

```

```

3301         \cs_if_exist:cT{stex_op_notation_ #2\c_hash_str ##1 _cs}{
3302             \exp_args:NNo\exp_args:No\exp_not:n{\csname stex_op_notation_ #2\c_hash_str ##1
3303             }
3304         }
3305     }\endgroup
3306 }
3307 }
3308
3309 \NewDocumentCommand \copynotation {m m} {
3310     \stex_get_symbol:n { #1 }
3311     \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
3312     \stex_get_symbol:n { #2 }
3313     \exp_args:Noo
3314     \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
3315     \stex_smsmode_do:\ignorespacesandpars
3316 }
3317

```

(End definition for \setnotation. This function is documented on page 23.)

\symdef

```

3318 \keys_define:nn { stex / symdef } {
3319     name .str_set_x:N = \l_stex_symdecl_name_str ,
3320     args .str_set_x:N = \l_stex_symdecl_args_str ,
3321     type .tl_set:N = \l_stex_symdecl_type_tl ,
3322     def .tl_set:N = \l_stex_symdecl_definiens_tl ,
3323     reorder .str_set_x:N = \l_stex_symdecl_reorder_str ,
3324     op .tl_set:N = \l__stex_notation_op_tl ,
3325     % lang .str_set_x:N = \l__stex_notation_lang_str ,
3326     variant .str_set_x:N = \l__stex_notation_variant_str ,
3327     prec .str_set_x:N = \l__stex_notation_prec_str ,
3328     argnames .clist_set:N = \l_stex_symdecl_argnames_clist ,
3329     assoc .choices:nn =
3330         {bin,binl,binr,pre,conj,pwconj}
3331         {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
3332     unknown .code:n = \str_set:Nx
3333         \l__stex_notation_variant_str \l_keys_key_str
3334 }
3335
3336 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
3337     \str_clear:N \l_stex_symdecl_name_str
3338     \str_clear:N \l_stex_symdecl_args_str
3339     \str_clear:N \l_stex_symdecl_assoctype_str
3340     \str_clear:N \l_stex_symdecl_reorder_str
3341     \bool_set_false:N \l_stex_symdecl_local_bool
3342     \tl_clear:N \l_stex_symdecl_type_tl
3343     \tl_clear:N \l_stex_symdecl_definiens_tl
3344     \clist_clear:N \l_stex_symdecl_argnames_clist
3345     % \str_clear:N \l__stex_notation_lang_str
3346     \str_clear:N \l__stex_notation_variant_str
3347     \str_clear:N \l__stex_notation_prec_str
3348     \tl_clear:N \l__stex_notation_op_tl
3349
3350     \keys_set:nn { stex / symdef } { #1 }

```

```

3351 }
3352
3353 \NewDocumentCommand \symdef { m O{} } {
3354   \_stex_notation_symdef_args:n { #2 }
3355   \bool_set_true:N \l_stex_symdecl_make_macro_bool
3356   \stex_symdecl_do:n { #1 }
3357   \tl_set:Nn \l_stex_notation_after_do_tl {
3358     \_stex_notation_final:
3359     \stex_smsmode_do:\ignorespacesandpars
3360   }
3361   \str_set:Nx \l_stex_get_symbol_uri_str {
3362     \l_stex_current_module_str ? \l_stex_symdecl_name_str
3363   }
3364   \exp_args:Nx \stex_notation_do:nnnnn
3365     { \prop_item:cn {l_stex_symdecl\_l_stex_get_symbol_uri_str_prop } { args } }
3366     { \prop_item:cn {l_stex_symdecl\_l_stex_get_symbol_uri_str_prop } { arity } }
3367     { \l__stex_notation_variant_str }
3368     { \l__stex_notation_prec_str}
3369 }
3370 \stex_deactivate_macro:Nn \symdef {module~environments}
3371
3372 \keys_define:nn { stex / mmtdef } {
3373   name .str_set_x:N = \l_stex_symdecl_name_str ,
3374   args .str_set_x:N = \l_stex_symdecl_args_str ,
3375   reorder .str_set_x:N = \l_stex_symdecl_reorder_str ,
3376   op .tl_set:N = \l__stex_notation_op_tl ,
3377   % lang .str_set_x:N = \l__stex_notation_lang_str ,
3378   variant .str_set_x:N = \l__stex_notation_variant_str ,
3379   prec .str_set_x:N = \l__stex_notation_prec_str ,
3380   argnames .clist_set:N = \l_stex_symdecl_argnames_clist ,
3381   assoc .choices:nn =
3382     {bin,binl,binr,pre,conj,pwconj}
3383     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
3384   unknown .code:n = \str_set:Nx
3385     \l__stex_notation_variant_str \l_keys_key_str
3386 }
3387 \cs_new_protected:Nn \stex_mmtdef_args:n {
3388   \str_clear:N \l_stex_symdecl_name_str
3389   \str_clear:N \l_stex_symdecl_args_str
3390   \str_clear:N \l_stex_symdecl_assoctype_str
3391   \str_clear:N \l_stex_symdecl_reorder_str
3392   \bool_set_false:N \l_stex_symdecl_local_bool
3393   \clist_clear:N \l_stex_symdecl_argnames_clist
3394   % \str_clear:N \l__stex_notation_lang_str
3395   \str_clear:N \l__stex_notation_variant_str
3396   \str_clear:N \l__stex_notation_prec_str
3397   \tl_clear:N \l__stex_notation_op_tl
3398
3399   \keys_set:nn { stex / mmtdef } { #1 }
3400 }
3401
3402 \NewDocumentCommand \mmtdef {m O{} }{
3403   \_stex_mmtdef_args:n{ #2 }
3404   \bool_set_true:N \l_stex_symdecl_make_macro_bool

```

```

3405 \str_if_empty:NT \l_stex_symdecl_name_str {
3406   \str_set:Nx \l_stex_symdecl_name_str { #1 }
3407 }
3408 %\tl_set:Nx \l_stex_symdecl_definiens_tl {
3409 % \stex_annotate:nnn{ OMID }{
3410 %   \l_stex_module_mmtfor_str?\l_stex_symdecl_name_str
3411 % }{}
3412 %}
3413 \stex_symdecl_do:n { #1 }
3414 \stex_if_smsmode:F{
3415   \MMTrule{rules.stex.mmt.kwarc.info?SubstitutionRule}{
3416     \stex_annotate:nnn{ OMID }{
3417       \l_stex_current_module_str ? \l_stex_symdecl_name_str
3418     }{},
3419     \stex_annotate:nnn{ OMID }{
3420       \l_stex_module_mmtfor_str?\l_stex_symdecl_name_str
3421     }{}
3422   }
3423 }
3424 \tl_set:Nn \l_stex_notation_after_do_tl {
3425   \__stex_notation_final:
3426   \stex_smsmode_do:\ignorespacesandpars
3427 }
3428 \str_set:Nx \l_stex_get_symbol_uri_str {
3429   \l_stex_current_module_str ? \l_stex_symdecl_name_str
3430 }
3431 \exp_args:Nx \stex_notation_do:nnnnn
3432 { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str_prop } { args } }
3433 { \prop_item:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str_prop } { arity } }
3434 { \l__stex_notation_variant_str }
3435 { \l__stex_notation_prec_str}
3436 }

```

(End definition for `\symdef`. This function is documented on page 91.)

31.3 Variables

```

3437 <@@=stex_variables>
3438
3439 \keys_define:nn { stex / vardef } {
3440   name .str_set_x:N = \l__stex_variables_name_str ,
3441   args .str_set_x:N = \l__stex_variables_args_str ,
3442   type .tl_set:N = \l__stex_variables_type_tl ,
3443   def .tl_set:N = \l__stex_variables_def_tl ,
3444   op .tl_set:N = \l__stex_variables_op_tl ,
3445   prec .str_set_x:N = \l__stex_variables_prec_str ,
3446   reorder .str_set_x:N = \l__stex_variables_reorder_str ,
3447   argnames .clist_set:N = \l__stex_variables_argnames_clist ,
3448   assoc .choices:nn =
3449     {bin,binl,binr,pre,conj,pwconj}
3450     {\str_set:Nx \l__stex_variables ASSOCTYPE_str {\l_keys_choice_tl}},
3451   bind .choices:nn =
3452     {forall,exists}
3453     {\str_set:Nx \l__stex_variables BIND_str {\l_keys_choice_tl}}

```

```

3454 }
3455
3456 \cs_new_protected:Nn \__stex_variables_args:n {
3457   \str_clear:N \l__stex_variables_name_str
3458   \str_clear:N \l__stex_variables_args_str
3459   \str_clear:N \l__stex_variables_prec_str
3460   \str_clear:N \l__stex_variables_assoctype_str
3461   \str_clear:N \l__stex_variables_reorder_str
3462   \str_clear:N \l__stex_variables_bind_str
3463   \tl_clear:N \l__stex_variables_type_tl
3464   \tl_clear:N \l__stex_variables_def_tl
3465   \tl_clear:N \l__stex_variables_op_tl
3466   \clist_clear:N \l__stex_variables_argnames_clist
3467
3468   \keys_set:nn { stex / vardef } { #1 }
3469 }
3470
3471 \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{} } {
3472   \__stex_variables_args:n {#2}
3473   \str_if_empty:NT \l__stex_variables_name_str {
3474     \str_set:Nx \l__stex_variables_name_str { #1 }
3475   }
3476   \prop_clear:N \l_tmpa_prop
3477   \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
3478
3479   \int_zero:N \l_tmpb_int
3480   \bool_set_true:N \l_tmpa_bool
3481   \str_map_inline:Nn \l__stex_variables_args_str {
3482     \token_case_meaning:NnF ##1 {
3483       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
3484       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
3485       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
3486       {\tl_to_str:n a} {
3487         \bool_set_false:N \l_tmpa_bool
3488         \int_incr:N \l_tmpb_int
3489       }
3490       {\tl_to_str:n B} {
3491         \bool_set_false:N \l_tmpa_bool
3492         \int_incr:N \l_tmpb_int
3493       }
3494     }{
3495       \msg_error:nnxx{stex}{error/wrongargs}{
3496         variable~\l__stex_variables_name_str
3497       }{##1}
3498     }
3499   }
3500   \bool_if:NTF \l_tmpa_bool {
3501     % possibly numeric
3502     \str_if_empty:NTF \l__stex_variables_args_str {
3503       \prop_put:Nnn \l_tmpa_prop { args } {}
3504       \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
3505     }{
3506       \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
3507       \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }

```

```

3508     \str_clear:N \l_tmpa_str
3509     \int_step_inline:nn \l_tmpa_int {
3510         \str_put_right:Nn \l_tmpa_str i
3511     }
3512     \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
3513     \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
3514 }
3515 } {
3516     \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
3517     \prop_put:Nnx \l_tmpa_prop { arity }
3518     { \str_count:N \l__stex_variables_args_str }
3519 }
3520 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
3521 \tl_set:cx { #1 } { \stex_invoke_variable:n { \l__stex_variables_name_str } }
3522
3523 % argnames
3524
3525 \clist_clear:N \l_tmpa_clist
3526 \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } } {
3527     \clist_if_empty:NTF \l__stex_variables_argnames_clist {
3528         \clist_put_right:Nn \l_tmpa_clist { ##1 }
3529     } {
3530         \clist_pop:NN \l__stex_variables_argnames_clist \l_tmpa_tl
3531         \exp_args:NNx \clist_put_right:Nn \l_tmpa_clist { \c_dollar_str \l_tmpa_tl }
3532     }
3533 }
3534 \prop_put:Nnx \l_tmpa_prop { argnames } { \clist_use:Nn \l_tmpa_clist , }
3535
3536
3537 \prop_set_eq:cN { l_stex_symdecl_var://\l__stex_variables_name_str _prop } \l_tmpa_prop
3538
3539 \tl_if_empty:NF \l__stex_variables_op_tl {
3540     \cs_set:cpx {
3541         stex_var_op_notation_ \l__stex_variables_name_str _cs
3542     } { \exp_not:N \comp { \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
3543 }
3544
3545 \tl_set:Nn \l_stex_notation_after_do_tl {
3546     \exp_args:Nne \use:nn {
3547         \cs_generate_from_arg_count:cNnn { stex_var_notation_ \l__stex_variables_name_str _cs }
3548         \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
3549     } { {
3550         \exp_after:wN \exp_after:wN \exp_after:wN
3551         \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
3552         { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \STEXInter
3553     } }
3554
3555 \stex_if_do_html:T {
3556     \stex_annotate_invisible:nnn { vardecl } { \l__stex_variables_name_str } {
3557         \stex_annotate_invisible:nnn { precedence }
3558         { \l__stex_variables_prec_str } { }
3559         \tl_if_empty:NF \l__stex_variables_type_tl { \stex_annotate_invisible:nnn { type } { } { $ \
3560         \stex_annotate_invisible:nnn { args } { \l__stex_variables_args_str } { }
3561         \stex_annotate_invisible:nnn { macroname } { #1 } { }
3562         \tl_if_empty:NF \l__stex_variables_def_tl {

```

```

3562         \stex_annotate_invisible:nnn{definiens}{}
3563         {${\l__stex_variables_def_tl$}
3564     }
3565     \str_if_empty:NF \l__stex_variables_assoctype_str {
3566         \stex_annotate_invisible:nnn{assoctype}{\l__stex_variables_assoctype_str}{}
3567     }
3568     \str_if_empty:NF \l__stex_variables_reorder_str {
3569         \stex_annotate_invisible:nnn{reorderargs}{\l__stex_variables_reorder_str}{}
3570     }
3571     \int_zero:N \l_tmpa_int
3572     \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
3573     \tl_clear:N \l_tmpa_tl
3574     \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
3575         \int_incr:N \l_tmpa_int
3576         \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
3577         \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables
3578         \str_if_eq:VnTF \l_tmpb_str a {
3579             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3580                 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3581                 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3582             } }
3583         }{
3584             \str_if_eq:VnTF \l_tmpb_str B {
3585                 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3586                     \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3587                     \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3588                 } }
3589             }{
3590                 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3591                     \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
3592                 } }
3593             }
3594         }
3595     }
3596     \stex_annotate_invisible:nnn { notationcomp }{}{
3597         \str_set:Nx \STEXInternalCurrentSymbolStr {var://\l__stex_variables_name_str }
3598         $ \exp_args:Nno \use:nn { \use:c {
3599             stex_var_notation_\l__stex_variables_name_str _cs
3600         } } { \l_tmpa_tl } $
3601     }
3602     \tl_if_empty:NF \l__stex_variables_op_tl {
3603         \stex_annotate_invisible:nnn { notationopcomp }{}{
3604             ${\l__stex_variables_op_tl$}
3605         }
3606     }
3607 }
3608 \str_if_empty:NF \l__stex_variables_bind_str {
3609     \stex_annotate_invisible:nnn {bindtype}{\l__stex_variables_bind_str,\l__stex_variabl
3610 }
3611 }\ignorespacesandpars
3612 }
3613
3614 \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
3615 }

```



```

3616
3617 \cs_new:Nn \_stex_reset:N {
3618   \tl_if_exist:NTF #1 {
3619     \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
3620   }{
3621     \let \exp_not:N #1 \exp_not:N \undefined
3622   }
3623 }
3624
3625 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
3626   \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
3627   \exp_args:Nnx \use:nn {
3628     % TODO
3629     \stex_annotate_invisible:nnn {vardecl}{\clist_use:Nn\l__stex_variables_names,}{
3630       #2
3631     }
3632   }{
3633     \_stex_reset:N \varnot
3634     \_stex_reset:N \vartype
3635     \_stex_reset:N \vardefi
3636   }
3637 }
3638
3639 \NewDocumentCommand \vardef { s } {
3640   \IfBooleanTF#1 {
3641     \__stex_variables_do_complex:nn
3642   }{
3643     \__stex_variables_do_simple:nnn
3644   }
3645 }
3646
3647 \NewDocumentCommand \svar { 0{} m }{
3648   \tl_if_empty:nTF {#1}{
3649     \str_set:Nn \l_tmpa_str { #2 }
3650   }{
3651     \str_set:Nn \l_tmpa_str { #1 }
3652   }
3653   \_stex_term_omv:nn {
3654     var://\l_tmpa_str
3655   }{
3656     \exp_args:Nnx \use:nn {
3657       \def\comp{\_varcomp}
3658       \str_set:Nx \STEXInternalCurrentSymbolStr { var://\l_tmpa_str }
3659       \comp{ #2 }
3660     }{
3661       \_stex_reset:N \comp
3662       \_stex_reset:N \STEXInternalCurrentSymbolStr
3663     }
3664   }
3665 }
3666
3667
3668
3669 \keys_define:nn { stex / varseq } {

```

```

3670 name      .str_set_x:N = \l__stex_variables_name_str ,
3671 args      .int_set:N = \l__stex_variables_args_int ,
3672 type      .tl_set:N = \l__stex_variables_type_tl ,
3673 mid       .tl_set:N = \l__stex_variables_mid_tl ,
3674 bind      .choices:nn =
3675           {forall,exists}
3676           {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
3677 }
3678
3679 \cs_new_protected:Nn \__stex_variables_seq_args:n {
3680   \str_clear:N \l__stex_variables_name_str
3681   \int_set:Nn \l__stex_variables_args_int 1
3682   \tl_clear:N \l__stex_variables_type_tl
3683   \str_clear:N \l__stex_variables_bind_str
3684
3685   \keys_set:nn { stex / varseq } { #1 }
3686 }
3687
3688 \NewDocumentCommand \varseq {m O{} m m m}{
3689   \__stex_variables_seq_args:n { #2 }
3690   \str_if_empty:NT \l__stex_variables_name_str {
3691     \str_set:Nx \l__stex_variables_name_str { #1 }
3692   }
3693   \prop_clear:N \l_tmpa_prop
3694   \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
3695
3696   \seq_set_from_clist:Nn \l_tmpa_seq {#3}
3697   \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
3698     \msg_error:nnxx{stex}{error/seqlength}
3699     {\int_use:N \l__stex_variables_args_int}
3700     {\seq_count:N \l_tmpa_seq}
3701   }
3702   \seq_set_from_clist:Nn \l_tmpb_seq {#4}
3703   \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
3704     \msg_error:nnxx{stex}{error/seqlength}
3705     {\int_use:N \l__stex_variables_args_int}
3706     {\seq_count:N \l_tmpb_seq}
3707   }
3708   \prop_put:Nnn \l_tmpa_prop {starts} {#3}
3709   \prop_put:Nnn \l_tmpa_prop {ends} {#4}
3710
3711   \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str_cs}
3712   \cs_set:Npn {\int_use:N \l__stex_variables_args_int} { #5 }
3713
3714   % argnames
3715
3716   \clist_clear:N \l_tmpa_clist
3717   \int_step_inline:nn {\l__stex_variables_args_int} {
3718     \clist_put_right:Nn \l_tmpa_clist {##1}
3719   }
3720   \prop_put:Nnx \l_tmpa_prop {argnames} {\clist_use:Nn \l_tmpa_clist ,}
3721
3722
3723

```

```

3724 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str_cs}}
3725 \int_step_inline:nn \l__stex_variables_args_int {
3726   \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
3727 }
3728 \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
3729 \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
3730 \tl_if_empty:NF \l__stex_variables_mid_tl {
3731   \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
3732   \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
3733 }
3734 \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq_\l__stex_variables_name_str_cs}}
3735 \int_step_inline:nn \l__stex_variables_args_int {
3736   \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
3737 }
3738 \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
3739 \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
3740
3741
3742
3743 \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl
3744
3745 \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}
3746
3747 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str_cs}}
3748
3749 \int_step_inline:nn \l__stex_variables_args_int {
3750   \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
3751     \STEXInternalTermMathArgiii{i##1}{0}{\exp_not:n{####}##1}
3752   }}
3753 }
3754
3755 \tl_set:Nx \l_tmpa_tl {
3756   \STEXInternalTermMathOMaiiii { varseq://\l__stex_variables_name_str}{0}{
3757     \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
3758   }
3759 }
3760
3761 \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \STEXInternalSymbolAfterInvokationTL} }
3762
3763 \exp_args:Nno \use:nn {
3764   \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str_cs}
3765   \cs_set:Npn {\int_use:N \l__stex_variables_args_int}{\l_tmpa_tl}
3766
3767   \stex_debug:nn{sequences}{New~Sequence:~
3768     \expandafter\meaning\csname stex_varseq_\l__stex_variables_name_str_cs\endcsname\\~\\
3769     \prop_to_keyval:N \l_tmpa_prop
3770   }
3771   \prop_set_eq:cN {\l_stex_symdecl_varseq://\l__stex_variables_name_str_prop}\l_tmpa_prop
3772
3773   \stex_if_do_html:T{\stex_annotate_invisible:nnn{varseq}{\l__stex_variables_name_str}{
3774     \tl_if_empty:NF \l__stex_variables_type_tl {
3775       \stex_annotate:nnn {type}{\l__stex_variables_type_tl$}
3776     }
3777     \stex_annotate:nnn {args}{\int_use:N \l__stex_variables_args_int}{

```

```

3778 \str_if_empty:NF \l__stex_variables_bind_str {
3779   \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{}
3780 }
3781 \stex_annotate:nnn{startindex}{}{${#3$}
3782 \stex_annotate:nnn{endindex}{}{${#4$}
3783
3784 \tl_clear:N \l_tmpa_tl
3785 \int_step_inline:nn \l__stex_variables_args_int {
3786   \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3787     \stex_annotate:nnn{argmarker}{##1}{}
3788   } }
3789 }
3790 \stex_annotate_invisible:nnn { notationcomp }{}{
3791   \str_set:Nx \STEXInternalCurrentSymbolStr {varseq://\l__stex_variables_name_str }
3792   $ \exp_args:Nno \use:nn { \use:c {
3793     stex_varseq\l__stex_variables_name_str _cs
3794   } } { \l_tmpa_tl } $
3795 }
3796 \stex_annotate_invisible:nnn { notationopcomp }{}{
3797   $ \prop_item:Nn \l_tmpa_prop { notation } $
3798 }
3799
3800 }}
3801
3802 \ignorespacesandpars
3803 }
3804
3805
3806 \keys_define:nn { stex / mmtdecl } {
3807   name      .str_set_x:N = \l_stex_symdecl_name_str ,
3808   args      .str_set_x:N = \l_stex_symdecl_args_str ,
3809   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,
3810   reorder   .str_set_x:N = \l_stex_symdecl_reorder_str ,
3811   argnames  .clist_set:N = \l_stex_symdecl_argnames_clist ,
3812   assoc     .choices:nn =
3813     {bin,binl,binr,pre,conj,pwconj}
3814     {\str_set:Nx \l_stex_symdecl_assoc_type_str {\l_keys_choice_tl}}
3815 }
3816
3817 \cs_new_protected:Nn \_stex_mmtdecl_args:n {
3818   \str_clear:N \l_stex_symdecl_name_str
3819   \str_clear:N \l_stex_symdecl_args_str
3820   \str_clear:N \l_stex_symdecl_deprecate_str
3821   \str_clear:N \l_stex_symdecl_reorder_str
3822   \str_clear:N \l_stex_symdecl_assoc_type_str
3823   \bool_set_false:N \l_stex_symdecl_local_bool
3824   \clist_clear:N \l_stex_symdecl_argnames_clist
3825
3826   \keys_set:nn { stex / symdecl } { #1 }
3827 }
3828
3829 \NewDocumentCommand \mmtdecl { s m O{} } {
3830   \_stex_mmtdecl_args:n{#3}
3831   \IfBooleanTF #1 {

```

```

3832   \bool_set_false:N \l_stex_symdecl_make_macro_bool
3833 } {
3834   \bool_set_true:N \l_stex_symdecl_make_macro_bool
3835 }
3836 \str_if_empty:NT \l_stex_symdecl_name_str {
3837   \str_set:Nx \l_stex_symdecl_name_str { #1 }
3838 }
3839 %\tl_set:Nx \l_stex_symdecl_definiens_tl {
3840 %   \stex_annotate:nnn{ OMID }{
3841 %     \l_stex_module_mmtfor_str?\l_stex_symdecl_name_str
3842 %   }{}
3843 %}
3844 \stex_symdecl_do:n{#2}
3845 \MMTrule{rules.stex.mmt.kwarc.info?SubstitutionRule}{
3846   \stex_annotate:nnn{ OMID }{
3847     \l_stex_current_module_str ? \l_stex_symdecl_name_str
3848   }{},
3849   \stex_annotate:nnn{ OMID }{
3850     \l_stex_module_mmtfor_str?\l_stex_symdecl_name_str
3851   }{}
3852 }
3853 \stex_smsmode_do:
3854 }
3855
3856 \stex_deactivate_macro:Nn \mmtdecl {mmtinterface~environments}
3857 \stex_deactivate_macro:Nn \mmtdef {mmtinterface~environments}
3858
3859 </package>

```

Chapter 32

STEX -Terms Implementation

```
3860 <*package>
3861
3862 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
3863
3864 <@@=stex_terms>
3865
3866 Warnings and error messages
3867 \msg_new:nnn{stex}{error/nonotation}{
3868   Symbol~#1~invoked,~but~has~no~notation#2!
3869 }
3870 \msg_new:nnn{stex}{error/notationarg}{
3871   Error~in~parsing~notation~#1
3872 }
3873 \msg_new:nnn{stex}{error/noop}{
3874   Symbol~#1~has~no~operator~notation~for~notation~#2
3875 }
3876 \msg_new:nnn{stex}{error/notallowed}{
3877   Symbol~invocation~#1~not~allowed~in~notation~component~of~#2
3878 }
3879 \msg_new:nnn{stex}{error/doubleargument}{
3880   Argument~#1~of~symbol~#2~already~assigned
3881 }
3882 \msg_new:nnn{stex}{error/overarity}{
3883   Argument~#1~invalid~for~symbol~#2~with~arity~#3
3884 }
```

32.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
3884
3885
3886 \bool_new:N \l_stex_allow_semantic_bool
3887 \bool_set_true:N \l_stex_allow_semantic_bool
3888
```

```

3889 \cs_new_protected:Nn \stex_invoke_symbol:n {
3890   \ifvmode\indent\fi
3891   \bool_if:NTF \l_stex_allow_semantic_bool {
3892     \str_if_eq:eeF {
3893       \prop_item:cn {
3894         l_stex_symdecl_#1_prop
3895       }{ deprecate }
3896     }{}{
3897       \msg_warning:nxxx{stex}{warning/deprecated}{
3898         Symbol~#1
3899       }{
3900         \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
3901       }
3902     }
3903     \if_mode_math:
3904       \exp_after:wN \__stex_terms_invoke_math:n
3905     \else:
3906       \exp_after:wN \__stex_terms_invoke_text:n
3907     \fi: { #1 }
3908   }{
3909     \msg_error:nxxx{stex}{error/notallowed}{#1}{\STEXInternalCurrentSymbolStr}
3910   }
3911 }
3912
3913 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
3914   \peek_charcode_remove:NTF ! {
3915     \__stex_terms_invoke_op_custom:nn {#1}
3916   }{
3917     \__stex_terms_invoke_custom:nn {#1}
3918   }
3919 }
3920
3921 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
3922   \peek_charcode_remove:NTF ! {
3923     % operator
3924     \peek_charcode_remove:NTF * {
3925       % custom op
3926       \__stex_terms_invoke_op_custom:nn {#1}
3927     }{
3928       % op notation
3929       \peek_charcode:NTF [ {
3930         \__stex_terms_invoke_op_notation:nw {#1}
3931       }{
3932         \__stex_terms_invoke_op_notation:nw {#1}[]
3933       }
3934     }
3935   }{
3936     \peek_charcode_remove:NTF * {
3937       \__stex_terms_invoke_custom:nn {#1}
3938       % custom
3939     }{
3940       % normal
3941       \peek_charcode:NTF [ {
3942         \__stex_terms_invoke_notation:nw {#1}

```

```

3943     }{
3944       \_stex_terms_invoke_notation:nw {#1}[]
3945     }
3946   }
3947 }
3948 }
3949
3950
3951 \cs_new_protected:Nn \_stex_terms_invoke_op_custom:nn {
3952   \exp_args:Nnx \use:nn {
3953     \def\comp{\_comp}
3954     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3955     \bool_set_false:N \l_stex_allow_semantic_bool
3956     \stex_mathml_intent:nn{#1}{
3957       \_stex_term_oms:nnn {#1}{#1 \c_hash_str CUSTOM-}{
3958         \comp{ #2 }
3959       }
3960     }
3961   }{
3962     \_stex_reset:N \comp
3963     \_stex_reset:N \STEXInternalCurrentSymbolStr
3964     \bool_set_true:N \l_stex_allow_semantic_bool
3965   }
3966 }
3967
3968 \keys_define:nn { stex / terms } {
3969   % lang      .tl_set_x:N = \l_stex_notation_lang_str ,
3970   variant .tl_set_x:N = \l_stex_notation_variant_str ,
3971   unknown .code:n      = \str_set:Nx
3972     \l_stex_notation_variant_str \l_keys_key_str
3973 }
3974
3975 \cs_new_protected:Nn \_stex_terms_args:n {
3976   % \str_clear:N \l_stex_notation_lang_str
3977   \str_clear:N \l_stex_notation_variant_str
3978
3979   \keys_set:nn { stex / terms } { #1 }
3980 }
3981
3982 \cs_new_protected:Nn \stex_find_notation:nn {
3983   \_stex_terms_args:n { #2 }
3984   \seq_if_empty:cTF {
3985     \l_stex_symdecl_ #1 _notations
3986   } {
3987     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
3988   } {
3989     \str_if_empty:NTF \l_stex_notation_variant_str {
3990       \seq_get_left:cN { \l_stex_symdecl_ #1 _notations } \l_stex_notation_variant_str
3991     } {
3992       \seq_if_in:cxTF { \l_stex_symdecl_ #1 _notations } {
3993         \l_stex_notation_variant_str
3994       } {
3995         % \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3996       } {

```



```

3997         \msg_error:nxxx{stex}{error/nonotation}{#1}{
3998         ~\l_stex_notation_variant_str
3999         }
4000     }
4001 }
4002 }
4003 }
4004
4005 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
4006     \exp_args:Nnx \use:nn {
4007         \def\comp{\_comp}
4008         \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
4009         \stex_find_notation:nn { #1 }{ #2 }
4010         \bool_set_false:N \l_stex_allow_semantic_bool
4011         \cs_if_exist:cTF {
4012             stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
4013         }{
4014             \_stex_term_oms:nnn { #1 }{
4015                 #1 \c_hash_str \l_stex_notation_variant_str
4016             }{
4017                 \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
4018             }
4019         }{
4020             \int_compare:nNnTF {\prop_item:cn {l_stex_symdecl_#1_prop}{arity}} = 0{
4021                 \cs_if_exist:cTF {
4022                     stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
4023                 }{
4024                     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4025                         \_stex_reset:N \comp
4026                         \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
4027                         \_stex_reset:N \STEXInternalCurrentSymbolStr
4028                         \bool_set_true:N \l_stex_allow_semantic_bool
4029                     }
4030                     \def\comp{\_comp}
4031                     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
4032                     \bool_set_false:N \l_stex_allow_semantic_bool
4033                     \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
4034                 }{
4035                     \msg_error:nxxx{stex}{error/nonotation}{#1}{
4036                     ~\l_stex_notation_variant_str
4037                     }
4038                 }
4039             }{
4040                 \msg_error:nxxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
4041             }
4042         }
4043     }{
4044         \_stex_reset:N \comp
4045         \_stex_reset:N \STEXInternalCurrentSymbolStr
4046         \bool_set_true:N \l_stex_allow_semantic_bool
4047     }
4048 }
4049
4050 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {

```

```

4051 \stex_find_notation:nn { #1 }{ #2 }
4052 \cs_if_exist:cTF {
4053   stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
4054 }{
4055   \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4056     \_stex_reset:N \comp
4057     \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
4058     \_stex_reset:N \STEXInternalCurrentSymbolStr
4059     \bool_set_true:N \l_stex_allow_semantic_bool
4060   }
4061   \def\comp{\_comp}
4062   \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
4063   \bool_set_false:N \l_stex_allow_semantic_bool
4064   \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
4065 }{
4066   \msg_error:nnxx{stex}{error/nonotation}{#1}{
4067     ~\l_stex_notation_variant_str
4068   }
4069 }
4070 }
4071
4072 \prop_new:N \l__stex_terms_custom_args_prop
4073 \clist_new:N \l_stex_argnames_seq
4074 \seq_new:N \l__stex_terms_tmp_seq
4075
4076 \cs_new_protected:Nn \__stex_terms_custom_comp:n{ \bool_set_false:N \l_stex_allow_semantic_bo
4077
4078 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
4079   \exp_args:Nnx \use:nn {
4080     \def\comp{\__stex_terms_custom_comp:n}
4081     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
4082     \prop_clear:N \l__stex_terms_custom_args_prop
4083     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
4084     \prop_get:cnN {
4085       l_stex_symdecl_#1 _prop
4086     }{ args } \l_tmpa_str
4087     \exp_args:NNx \seq_set_from_clist:Nn \l_stex_argnames_seq {
4088       \prop_item:cn {l_stex_symdecl_#1 _prop}{argnames}
4089     }
4090     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
4091     \tl_set:Nn \arg { \__stex_terms_arg: }
4092     \str_if_empty:NTF \l_tmpa_str {
4093       \stex_mathml_intent:nn{#1}{
4094         \_stex_term_oms:nnn {#1}{#1\c_hash_str CUSTOM-}{\ignorespaces#2}
4095       }
4096     }{
4097       \seq_clear:N \l__stex_terms_tmp_seq
4098       \exp_args:Nx\int_step_inline:nn{\prop_item:cn{l_stex_symdecl_#1 _prop}{arity}}{
4099         \tl_set:Nx \l__stex_terms_tmp_tl {\seq_item:Nn \l_stex_argnames_seq {##1}}
4100         \bool_lazy_or:nnT{
4101           \str_if_eq_p:nn{a}{\str_item:Nn \l_tmpa_str{##1}}
4102         }{
4103           \str_if_eq_p:nn{B}{\str_item:Nn \l_tmpa_str{##1}}
4104         }{

```

```

4105         \tl_put_right:Nn \l__stex_terms_tmp_tl +
4106     }
4107     \seq_put_right:No \l__stex_terms_tmp_seq \l__stex_terms_tmp_tl
4108 }
4109 \stex_mathml_intent:nn{
4110     #1[\prop_item:cn {l_stex_symdecl_#1 _prop}]{ args }(
4111         \seq_use:Nn \l__stex_terms_tmp_seq ,
4112     )
4113 }{
4114     \str_if_in:NnTF \l_tmpa_str b {
4115         \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
4116     }{
4117         \str_if_in:NnTF \l_tmpa_str B {
4118             \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
4119         }{
4120             \_stex_term_oma:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
4121         }
4122     }
4123 }
4124 }
4125 % TODO check that all arguments exist
4126 }{
4127     \_stex_reset:N \l_stex_argnames_seq
4128     \_stex_reset:N \STEXInternalCurrentSymbolStr
4129     \_stex_reset:N \arg
4130     \_stex_reset:N \comp
4131     \_stex_reset:N \l__stex_terms_custom_args_prop
4132     %\bool_set_true:N \l_stex_allow_semantic_bool
4133 }
4134 }
4135
4136 \NewDocumentCommand \__stex_terms_arg: { s O{} m}{
4137     \tl_if_empty:nTF {#2}{
4138         \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
4139         \bool_set_true:N \l_tmpa_bool
4140         \bool_do_while:Nn \l_tmpa_bool {
4141             \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int}
4142             \int_incr:N \l_tmpa_int
4143         }{
4144             \bool_set_false:N \l_tmpa_bool
4145         }
4146     }
4147     }{
4148         \int_set:Nn \l_tmpa_int { #2 }
4149     }
4150     \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
4151     \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
4152         \msg_error:nnxxx{stex}{error/overarity}
4153         {\int_use:N \l_tmpa_int}
4154         {\STEXInternalCurrentSymbolStr}
4155         {\str_count:N \l_tmpa_str}
4156     }
4157     \str_set:Nx \l_tmpa_str {\str_item:Nn \l_tmpa_str \l_tmpa_int}
4158     \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {

```

```

4159 \bool_lazy_any:nF {
4160   {\str_if_eq_p:Vn \l_tmpa_str {a}}
4161   {\str_if_eq_p:Vn \l_tmpa_str {B}}
4162 }{
4163   \msg_error:nnxx{stex}{error/doubleargument}
4164   {\int_use:N \l_tmpa_int}
4165   {\STEXInternalCurrentSymbolStr}
4166 }
4167 }
4168 \exp_args:NNx \prop_put:Nnn \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {\ign
4169 \bool_if:NTF \l_stex_allow_semantic_bool \use_i:nn {
4170   \bool_set_true:N \l_stex_allow_semantic_bool
4171   \use:nn
4172 }
4173 {
4174   \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq \l_tmpa_int}{
4175     \IfBooleanTF#1{
4176       \stex_annotate_invisible:n { %TODO
4177         \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{\ignorespaces#3}
4178       }
4179     }{ %TODO
4180       \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{\ignorespaces#3}
4181     }
4182   }}
4183   {\bool_set_false:N \l_stex_allow_semantic_bool}
4184 }
4185
4186
4187 \cs_new_protected:Nn \_stex_term_arg:nn {
4188   \bool_set_true:N \l_stex_allow_semantic_bool
4189   \stex_annotate:nnn{ arg }{ #1 }{ #2 }
4190   \bool_set_false:N \l_stex_allow_semantic_bool
4191 }
4192
4193 \cs_new_protected:Npn \STEXInternalTermMathArgiii #1#2#3 {
4194   \exp_args:Nnx \use:nn
4195     { \int_set:Nn \l__stex_terms_downprec { #2 }
4196       \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq \l_tmpa_int}{
4197         \_stex_term_arg:nn { #1 }{ #3 }
4198       }
4199     }
4200   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
4201 }

```

(End definition for `\stex_invoke_symbol:n`. This function is documented on page 92.)

`\STEXInternalTermMathAssocArgiiii`

```

4202 \cs_new_protected:Npn \STEXInternalTermMathAssocArgiiii #1#2#3#4#5 {
4203   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
4204   \tl_set:Nn \l_tmpb_tl {\STEXInternalTermMathArgiii{#5#1}{#2}}
4205   \tl_if_empty:nTF { #3 }{
4206     \STEXInternalTermMathArgiii{#5#1}{#2}{#3}
4207   }{
4208     \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 } }{

```

```

4209     \expandafter\if\expandafter\relax\noexpand#3
4210     \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_maybe_sequence:Nnn#3{#1}{#5}}
4211   \else
4212     \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_simple:nnn{#1}{#3}{#5}}
4213   \fi
4214   \l_tmpa_tl
4215 }{
4216   \__stex_terms_math_assoc_arg_simple:nnn{#1}{#3}{#5}
4217 }
4218 }
4219 }
4220
4221 \cs_new_protected:Nn \__stex_terms_math_assoc_arg_maybe_sequence:Nnn {
4222   \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
4223   \str_if_empty:NTF \l_tmpa_str {
4224     \exp_args:Nx \cs_if_eq:NNTF {
4225       \tl_head:N #1
4226     } \stex_invoke_sequence:n {
4227       \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
4228       \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
4229       \tl_set:Nx \l_tmpa_tl {\prop_item:cn {l_stex_symdecl_varseq://\l_tmpa_str _prop}{notat
4230       \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
4231       \tl_set:Nx \l_tmpa_tl {\exp_not:N \exp_not:n{
4232         \exp_not:n{\exp_args:Nnx \use:nn} {
4233           \exp_not:n {
4234             \def\comp{\_varcomp}
4235             \str_set:Nn \STEXInternalCurrentSymbolStr
4236             } {varseq://\l_tmpa_str}
4237             \exp_not:n{ ##1 }
4238           }{
4239             \exp_not:n {
4240               \_stex_reset:N \comp
4241               \_stex_reset:N \STEXInternalCurrentSymbolStr
4242             }
4243           }
4244         }}}
4245       \exp_args:Nno \use:n {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
4246       \seq_reverse:N \l_tmpa_seq
4247       \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
4248       \seq_map_inline:Nn \l_tmpa_seq {
4249         \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
4250           \exp_args:Nno
4251           \l_tmpa_cs { ##1 } \l_tmpa_tl
4252         }
4253       }
4254       \tl_set:Nx \l_tmpa_tl {
4255         \_stex_term_omv:nn {varseq://\l_tmpa_str}{
4256           \exp_args:No \exp_not:n \l_tmpa_tl
4257         }
4258       }
4259       \exp_args:No\l_tmpb_tl\l_tmpa_tl
4260     }{
4261       \__stex_terms_math_assoc_arg_simple:nnn{#2} { #1 }{#3}
4262     }

```

```

4263 } {
4264   \_stex_terms_math_assoc_arg_simple:nnn{#2} { #1 }{#3}
4265 }
4266
4267 }
4268
4269 \cs_new_protected:Nn \_stex_terms_math_assoc_arg_simple:nnn {
4270   \clist_set:Nn \l_tmpa_clist{ #2 }
4271   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
4272     \tl_set:Nn \l_tmpa_tl {
4273       \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq #1}{
4274         \_stex_term_arg:nn{A#3#1}{ #2 } }
4275     }
4276   }{
4277     \clist_reverse:N \l_tmpa_clist
4278     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
4279     \tl_set:Nx \l_tmpa_tl {
4280       \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq #1}{
4281         \_stex_term_arg:nn{A#3#1}{
4282           \exp_args:No \exp_not:n \l_tmpa_tl
4283         }
4284       }
4285     }
4286     \clist_map_inline:Nn \l_tmpa_clist {
4287       \exp_args:NNNo \exp_args:NNNo \tl_set:No \l_tmpa_tl {
4288         \exp_args:Nno
4289         \l_tmpa_cs {
4290           \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq #1}{
4291             \_stex_term_arg:nn{A#3#1}{##1}
4292           }
4293         } \l_tmpa_tl
4294       }
4295     }
4296     \exp_args:No\l_tmpb_tl\l_tmpa_tl
4297   }

```

(End definition for `\STEXInternalTermMathAssocArgiiii`. This function is documented on page 93.)

32.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
4298 \tl_const:Nx \infprec {\int_use:N \c_max_int}
4299 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
4300 \int_new:N \l__stex_terms_downprec
4301 \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
4302 \tl_set:Nn \l__stex_terms_left_bracket_str (
4303 \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

```

4304 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
4305   \bool_if:NTF \l__stex_terms_brackets_done_bool {
4306     \bool_set_false:N \l__stex_terms_brackets_done_bool
4307     #2
4308   } {
4309     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
4310       \bool_if:NTF \l__stex_inarray_bool { #2 }{
4311         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
4312         \dobrackets { #2 }
4313       }
4314     }{ #2 }
4315   }
4316 }
```

(End definition for `__stex_terms_maybe_brackets:nn`.)

`\dobrackets`

```

4317 \bool_new:N \l__stex_terms_brackets_done_bool
4318 %\RequirePackage{scalerel}
4319 \cs_new_protected:Npn \dobrackets #1 {
4320   %\ThisStyle{\if D\m@switch
4321   %   \exp_args:Nnx \use:nn
4322   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
4323   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
4324   %   \else
4325   %     \exp_args:Nnx \use:nn
4326   %     {
4327   %       \bool_set_true:N \l__stex_terms_brackets_done_bool
4328   %       \int_set:Nn \l__stex_terms_downprec \infpref
4329   %       \l__stex_terms_left_bracket_str
4330   %       #1
4331   %     }
4332   %     {
4333   %       \bool_set_false:N \l__stex_terms_brackets_done_bool
4334   %       \l__stex_terms_right_bracket_str
4335   %       \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
4336   %     }
4337   %\fi}
4338 }
```

(End definition for `\dobrackets`. This function is documented on page 93.)

`\withbrackets`

```

4339 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
4340   \exp_args:Nnx \use:nn
4341   {
4342     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
4343     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
4344     #3
4345   }
4346   {
```

```

4347 \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
4348 {\l__stex_terms_left_bracket_str}
4349 \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
4350 {\l__stex_terms_right_bracket_str}
4351 }
4352 }

```

(End definition for \withbrackets. This function is documented on page 93.)

\STEXinvisible

```

4353 \cs_new_protected:Npn \STEXinvisible #1 {
4354 \stex_annotate_invisible:n { #1 }
4355 }

```

(End definition for \STEXinvisible. This function is documented on page 93.)

OMDoc terms:

\STEXInternalTermMathOMSiiii

```

4356 \cs_new_protected:Nn \_stex_term_oms:nnn {
4357 \stex_annotate:nnn{ OMID }{ #2 }{
4358 #3
4359 }
4360 }
4361
4362 \cs_new_protected:Npn \STEXInternalTermMathOMSiiii #1#2#3#4 {
4363 \__stex_terms_maybe_brackets:nn { #3 }{
4364 \stex_mathml_intent:nn{#1} {
4365 \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
4366 }
4367 }
4368 }

```

(End definition for \STEXInternalTermMathOMSiiii. This function is documented on page 92.)

_stex_term_math_omv:nn

```

4369 \cs_new_protected:Nn \_stex_term_omv:nn {
4370 \stex_annotate:nnn{ OMV }{ #1 }{
4371 #2
4372 }
4373 }

```

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

\STEXInternalTermMathOMAiiai

```

4374 \cs_new_protected:Nn \_stex_term_oma:nnn {
4375 \stex_annotate:nnn{ OMA }{ #2 }{
4376 #3
4377 }
4378 }
4379
4380 \cs_new_protected:Npn \STEXInternalTermMathOMAiiai #1#2#3#4 {
4381 \exp_args:Nnx \use:nn {
4382 \seq_clear:N \l__stex_terms_tmp_seq
4383 \prop_if_exist:cT{l_stex_symdecl_#1 _prop}{
4384 \exp_args:NNx \seq_set_from_clist:Nn \l_stex_argnames_seq {

```



```

4385     \prop_item:cn {l_stex_symdecl_#1 _prop}{argnames}
4386   }
4387   \exp_args:Nx\int_step_inline:nn{\prop_item:cn{l_stex_symdecl_#1 _prop}{arity}}{
4388     \tl_set:Nx \l__stex_terms_tmp_tl {\seq_item:Nn \l_stex_argnames_seq {##1}}
4389     \bool_lazy_or:nnT{
4390       \str_if_eq_p:nn{a}{\str_item:Nn\l_tmpa_str{##1}}
4391     }{
4392       \str_if_eq_p:nn{B}{\str_item:Nn\l_tmpa_str{##1}}
4393     }{
4394       \tl_put_right:Nn \l__stex_terms_tmp_tl +
4395     }
4396     \seq_put_right:No \l__stex_terms_tmp_seq \l__stex_terms_tmp_tl
4397   }
4398 }
4399 \__stex_terms_maybe_brackets:nn { #3 }{
4400   \stex_mathml_intent:nn{
4401     #1[\prop_item:cn {l_stex_symdecl_#1 _prop}{ args }](
4402       \seq_use:Nn \l__stex_terms_tmp_seq ,
4403     )
4404   }{
4405     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
4406   }
4407 }
4408 }{
4409   \_stex_reset:N \l_stex_argnames_seq
4410 }
4411 }

```

(End definition for `\STEXInternalTermMathOMAi`. This function is documented on page 92.)

`\STEXInternalTermMathOMBiiii`

```

4412 \cs_new_protected:Nn \_stex_term_ombind:nnn {
4413   \stex_annotate:nnn{ OMBIND }{ #2 }{
4414     #3
4415   }
4416 }
4417
4418 \cs_new_protected:Npn \STEXInternalTermMathOMBiiii #1#2#3#4 {
4419   \exp_args:Nnx \use:nn {
4420     \seq_clear:N \l__stex_terms_tmp_seq
4421     \prop_if_exist:cT{l_stex_symdecl_#1 _prop}{
4422       \exp_args:NNx \seq_set_from_clist:Nn \l_stex_argnames_seq {
4423         \prop_item:cn {l_stex_symdecl_#1 _prop}{argnames}
4424       }
4425     }
4426     \exp_args:Nx\int_step_inline:nn{\prop_item:cn{l_stex_symdecl_#1 _prop}{arity}}{
4427       \tl_set:Nx \l__stex_terms_tmp_tl {\seq_item:Nn \l_stex_argnames_seq {##1}}
4428       \bool_lazy_or:nnT{
4429         \str_if_eq_p:nn{a}{\str_item:Nn\l_tmpa_str{##1}}
4430       }{
4431         \str_if_eq_p:nn{B}{\str_item:Nn\l_tmpa_str{##1}}
4432       }{
4433         \tl_put_right:Nn \l__stex_terms_tmp_tl +
4434       }
4435       \seq_put_right:No \l__stex_terms_tmp_seq \l__stex_terms_tmp_tl

```

```

4435     }
4436   }
4437   \__stex_terms_maybe_brackets:nn { #3 }{
4438     \stex_mathml_intent:nn{
4439       #1[\prop_item:cn {l_stex_symdecl_#1 _prop}{ args }](
4440         \seq_use:Nn \l__stex_terms_tmp_seq ,
4441       )
4442     }{
4443       \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
4444     }
4445   }
4446   }{
4447     \stex_reset:N \l_stex_argnames_seq
4448   }
4449 }

```

(End definition for `\STEXInternalTermMathOMBiiii`. This function is documented on page 92.)

`\symref`
`\symname`

```

4450 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
4451
4452 \keys_define:nn { stex / symname } {
4453   pre      .tl_set_x:N      = \l__stex_terms_pre_tl ,
4454   post     .tl_set_x:N      = \l__stex_terms_post_tl ,
4455   root     .tl_set_x:N      = \l__stex_terms_root_tl
4456 }
4457
4458 \cs_new_protected:Nn \stex_symname_args:n {
4459   \tl_clear:N \l__stex_terms_post_tl
4460   \tl_clear:N \l__stex_terms_pre_tl
4461   \tl_clear:N \l__stex_terms_root_str
4462   \keys_set:nn { stex / symname } { #1 }
4463 }
4464
4465 \NewDocumentCommand \symref { m m }{
4466   \let\compemph_uri_prev:\compemph@uri
4467   \let\compemph@uri\symrefemph@uri
4468   \STEXsymbol{#1}!\{ #2 }
4469   \let\compemph@uri\compemph_uri_prev:
4470 }
4471
4472 \NewDocumentCommand \synonym { 0{} m m }{
4473   \stex_symname_args:n { #1 }
4474   \let\compemph_uri_prev:\compemph@uri
4475   \let\compemph@uri\symrefemph@uri
4476   % TODO
4477   \STEXsymbol{#2}!\{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
4478   \let\compemph@uri\compemph_uri_prev:
4479 }
4480
4481 \NewDocumentCommand \symname { 0{} m }{
4482   \stex_symname_args:n { #1 }
4483   \stex_get_symbol:n { #2 }
4484   \str_set:Nx \l_tmpa_str {

```

```

4485 \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4486 }
4487 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4488
4489 \let\compemph_uri_prev:\compemph@uri
4490 \let\compemph@uri\symrefemph@uri
4491 \exp_args:Nnx \use:nn
4492 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmode*\fi{
4493   \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
4494 } }
4495 \let\compemph@uri\compemph_uri_prev:
4496 }
4497
4498 \NewDocumentCommand \Symname { 0{} m }{
4499   \stex_symname_args:n { #1 }
4500   \stex_get_symbol:n { #2 }
4501   \str_set:Nx \l_tmpa_str {
4502     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4503   }
4504   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4505   \let\compemph_uri_prev:\compemph@uri
4506   \let\compemph@uri\symrefemph@uri
4507   \exp_args:Nnx \use:nn
4508   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmode*\fi{
4509     \exp_after:wN \stex_capitalize:n \l_tmpa_str
4510     \l__stex_terms_post_tl
4511   } }
4512   \let\compemph@uri\compemph_uri_prev:
4513 }

```

(End definition for `\symref` and `\symname`. These functions are documented on page 92.)

32.3 Notation Components

```

4514 <@@=stex_notationcomps>

\comp
\compemph@uri
\compemph
\defemph
\defemph@uri
\symrefemph
\symrefemph@uri
\varemp
\varemp@uri

4515 \cs_new_protected:Npn \_comp #1 {
4516   \str_if_empty:NF \STEXInternalCurrentSymbolStr {
4517     \stex_html_backend:TF {
4518       \stex_annotate:nnn { comp }{ \STEXInternalCurrentSymbolStr }{ #1 }
4519     }{
4520       \exp_args:Nnx \compemph@uri { #1 } { \STEXInternalCurrentSymbolStr }
4521     }
4522   }
4523 }
4524
4525 \cs_new_protected:Npn \_varcomp #1 {
4526   \str_if_empty:NF \STEXInternalCurrentSymbolStr {
4527     \stex_html_backend:TF {
4528       \stex_annotate:nnn { varcomp }{ \STEXInternalCurrentSymbolStr }{ #1 }
4529     }{
4530       \exp_args:Nnx \varemp@uri { #1 } { \STEXInternalCurrentSymbolStr }
4531     }

```

```

4532 }
4533 }
4534
4535 \def\comp{\_comp}
4536
4537 \cs_new_protected:Npn \compemph@uri #1 #2 {
4538   \compemph{ #1 }
4539 }
4540
4541
4542 \cs_new_protected:Npn \compemph #1 {
4543   #1
4544 }
4545
4546 \cs_new_protected:Npn \defemph@uri #1 #2 {
4547   \defemph{#1}
4548 }
4549
4550 \cs_new_protected:Npn \defemph #1 {
4551   \textbf{#1}
4552 }
4553
4554 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
4555   \symrefemph{#1}
4556 }
4557
4558 \cs_new_protected:Npn \symrefemph #1 {
4559   \emph{#1}
4560 }
4561
4562 \cs_new_protected:Npn \varemp@uri #1 #2 {
4563   \varemp{#1}
4564 }
4565
4566 \cs_new_protected:Npn \varemp #1 {
4567   #1
4568 }

```

(End definition for `\comp` and others. These functions are documented on page 93.)

`\ellipses`

```

4569 \NewDocumentCommand \ellipses {} { \ldots }

```

(End definition for `\ellipses`. This function is documented on page 93.)

```

\parray
\prmatrix
\parrayline
\parraylineh
\parraycell
4570 \bool_new:N \l_stex_inparray_bool
4571 \bool_set_false:N \l_stex_inparray_bool
4572 \NewDocumentCommand \parray { m m } {
4573   \begin{group}
4574     \bool_set_true:N \l_stex_inparray_bool
4575     \begin{array}{#1}
4576       #2
4577     \end{array}
4578   \end{group}

```

```

4579 }
4580
4581 \NewDocumentCommand \prmatrix { m } {
4582   \begingroup
4583   \bool_set_true:N \l_stex_inarray_bool
4584   \begin{matrix}
4585     #1
4586   \end{matrix}
4587   \endgroup
4588 }
4589
4590 \def \maybepline {
4591   \bool_if:NT \l_stex_inarray_bool {\hline}
4592 }
4593
4594 \def \parrayline #1 #2 {
4595   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
4596 }
4597
4598 \def \pmrow #1 { \parrayline{}{ #1 } }
4599
4600 \def \parraylineh #1 #2 {
4601   #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
4602 }
4603
4604 \def \parraycell #1 {
4605   #1 \bool_if:NT \l_stex_inarray_bool {&}
4606 }

```

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

32.4 Variables

```

4607 <@@=stex_variables>

```

\stex_invoke_variable:n Invokes a variable

```

4608 \cs_new_protected:Nn \stex_invoke_variable:n {
4609   \if_mode_math:
4610     \exp_after:wN \__stex_variables_invoke_math:n
4611   \else:
4612     \exp_after:wN \__stex_variables_invoke_text:n
4613   \fi: {#1}
4614 }
4615
4616 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
4617   \peek_charcode_remove:NTF ! {
4618     \__stex_variables_invoke_op_custom:nn {#1}
4619   }{
4620     \__stex_variables_invoke_custom:nn {#1}
4621   }
4622 }
4623
4624
4625 \cs_new_protected:Nn \__stex_variables_invoke_math:n {

```

```

4626 \peek_charcode_remove:NTF ! {
4627   \peek_charcode_remove:NTF ! {
4628     \peek_charcode:NTF [ {
4629       % TODO throw error
4630     }{
4631       \__stex_variables_invoke_op_custom:nn
4632     }
4633   }{
4634     \__stex_variables_invoke_op:n { #1 }
4635   }
4636 }{
4637   \peek_charcode_remove:NTF * {
4638     \__stex_variables_invoke_custom:nn { #1 }
4639   }{
4640     \__stex_variables_invoke_math_ii:n { #1 }
4641   }
4642 }
4643 }
4644
4645 \cs_new_protected:Nn \__stex_variables_invoke_op_custom:nn {
4646   \exp_args:Nnx \use:nn {
4647     \def\comp{\_varcomp}
4648     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4649     \bool_set_false:N \l_stex_allow_semantic_bool
4650     \stex_term_omv:nn {var://#1}{
4651       \comp{ #2 }
4652     }
4653   }{
4654     \stex_reset:N \comp
4655     \stex_reset:N \STEXInternalCurrentSymbolStr
4656     \bool_set_true:N \l_stex_allow_semantic_bool
4657   }
4658 }
4659
4660 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
4661   \cs_if_exist:cTF {
4662     stex_var_op_notation_ #1 _cs
4663   }{
4664     \exp_args:Nnx \use:nn {
4665       \def\comp{\_varcomp}
4666       \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4667       \stex_term_omv:nn { var://#1 }{
4668         \use:c{stex_var_op_notation_ #1 _cs }
4669       }
4670     }{
4671       \stex_reset:N \comp
4672       \stex_reset:N \STEXInternalCurrentSymbolStr
4673     }
4674   }{
4675     \int_compare:nNnTF {\prop_item:cn {l_stex_symdecl_var://#1_prop}{arity}} = 0{
4676       \__stex_variables_invoke_math_ii:n {#1}
4677     }{
4678       \msg_error:nxxx{stex}{error/noop}{variable~#1}{}
4679     }

```

```

4680 }
4681 }
4682
4683 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n #1 {
4684   \cs_if_exist:cTF {
4685     stex_var_notation_#1_cs
4686   }{
4687     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4688       \_stex_reset:N \comp
4689       \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
4690       \_stex_reset:N \STEXInternalCurrentSymbolStr
4691       \bool_set_true:N \l_stex_allow_semantic_bool
4692     }
4693     \def\comp{\_varcomp}
4694     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4695     \bool_set_false:N \l_stex_allow_semantic_bool
4696     \use:c{stex_var_notation_#1_cs}
4697   }{
4698     \msg_error:nnxx{stex}{error/nonotation}{variable-#1}{s}
4699   }
4700 }
4701
4702 \cs_new_protected:Nn \__stex_variables_invoke_custom:nn {
4703   \exp_args:Nnx \use:nn {
4704     \def\comp{\_varcomp}
4705     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4706     \prop_clear:N \l__stex_terms_custom_args_prop
4707     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
4708     \prop_get:cnN {
4709       l_stex_symdecl_var://#1 _prop
4710     }{ args } \l_tmpa_str
4711     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
4712     \tl_set:Nn \arg { \_stex_terms_arg: }
4713     \str_if_empty:NTF \l_tmpa_str {
4714       \_stex_term_omv:nn {var://#1}{\ignorespaces#2}
4715     }{
4716       \str_if_in:NnTF \l_tmpa_str b {
4717         \_stex_term_ombind:nnn {var://#1}{\ignorespaces#2}
4718       }{
4719         \str_if_in:NnTF \l_tmpa_str B {
4720           \_stex_term_ombind:nnn {var://#1}{\ignorespaces#2}
4721         }{
4722           \_stex_term_oma:nnn {var://#1}{\ignorespaces#2}
4723         }
4724       }
4725     }
4726     % TODO check that all arguments exist
4727   }{
4728     \_stex_reset:N \STEXInternalCurrentSymbolStr
4729     \_stex_reset:N \arg
4730     \_stex_reset:N \comp
4731     \_stex_reset:N \l__stex_terms_custom_args_prop
4732     %\bool_set_true:N \l_stex_allow_semantic_bool
4733   }

```

```
4734 }
```

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

32.5 Sequences

```
4735 <@@=stex_sequences>
4736
4737 \cs_new_protected:Nn \stex_invoke_sequence:n {
4738   \peek_charcode_remove:NTF ! {
4739     \stex_term_omv:nn {varseq://#1}{
4740       \exp_args:Nnx \use:nn {
4741         \def\comp{\_varcomp}
4742         \str_set:Nn \STEXInternalCurrentSymbolStr {varseq://#1}
4743         \prop_item:cn{l_stex_symdecl_varseq://#1_prop}{notation}
4744       }{
4745         \_stex_reset:N \comp
4746         \_stex_reset:N \STEXInternalCurrentSymbolStr
4747       }
4748     }
4749   }{
4750     \bool_set_false:N \l_stex_allow_semantic_bool
4751     \def\comp{\_varcomp}
4752     \str_set:Nn \STEXInternalCurrentSymbolStr {varseq://#1}
4753     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4754       \_stex_reset:N \comp
4755       \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
4756       \_stex_reset:N \STEXInternalCurrentSymbolStr
4757       \bool_set_true:N \l_stex_allow_semantic_bool
4758     }
4759     \use:c { stex_varseq_#1_cs }
4760   }
4761 }
4762 </package>
```


Chapter 33

STEX -Structural Features Implementation

```
4763 <*package>
4764
4765 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4766
    Warnings and error messages
4767 \msg_new:nnn{stex}{error/copymodule/notallowed}{
4768   Symbol~#1~can~not~be~assigned~in~copymodule~#2
4769 }
4770 \msg_new:nnn{stex}{error/interpretmodule/noddefinens}{
4771   Symbol~#1~not~assigned~in~interpretmodule~#2
4772 }
4773
4774 \msg_new:nnn{stex}{error/unknownstructure}{
4775   No~structure~#1~found!
4776 }
4777
4778 \msg_new:nnn{stex}{error/unknownfield}{
4779   No~field~#1~in~instance~#2~found!\#3
4780 }
4781
4782 \msg_new:nnn{stex}{error/keyval}{
4783   Invalid~key=value~pair:#1
4784 }
4785 \msg_new:nnn{stex}{error/instantiate/missing}{
4786   Assignments~missing~in~instantiate:~#1
4787 }
4788 \msg_new:nnn{stex}{error/incompatible}{
4789   Incompatible~signature:~#1~(#2)~and~#3~(#4)
4790 }
4791
```

33.1 Imports with modification

```

4792 <@@=stex_copymodule>
4793 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
4794   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4795     \tl_set:Nn \l_tmpa_tl { #1 }
4796     \__stex_copymodule_get_symbol_from_cs:
4797   }{
4798     % argument is a string
4799     % is it a command name?
4800     \cs_if_exist:cTF { #1 }{
4801       \cs_set_eq:Nc \l_tmpa_tl { #1 }
4802       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4803       \str_if_empty:NTF \l_tmpa_str {
4804         \exp_args:Nx \cs_if_eq:NNTF {
4805           \tl_head:N \l_tmpa_tl
4806         } \stex_invoke_symbol:n {
4807           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
4808         }{
4809           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4810         }
4811       } {
4812         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4813       }
4814     }{
4815       % argument is not a command name
4816       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4817       % \l_stex_all_symbols_seq
4818     }
4819   }
4820 }
4821
4822 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
4823   \str_set:Nn \l_tmpa_str { #1 }
4824   \bool_set_false:N \l_tmpa_bool
4825   \bool_if:NF \l_tmpa_bool {
4826     \tl_set:Nn \l_tmpa_tl {
4827       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
4828     }
4829     \str_set:Nn \l_tmpa_str { #1 }
4830     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4831     \seq_map_inline:Nn #2 {
4832       \str_set:Nn \l_tmpb_str { ##1 }
4833       \str_if_eq:eeT { \l_tmpa_str } {
4834         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4835       } {
4836         \seq_map_break:n {
4837           \tl_set:Nn \l_tmpa_tl {
4838             \str_set:Nn \l_stex_get_symbol_uri_str {
4839               ##1
4840             }
4841           }
4842         }
4843       }

```

```

4844     }
4845     \l_tmpa_tl
4846   }
4847 }
4848
4849 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
4850   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4851     { \tl_tail:N \l_tmpa_tl }
4852   \tl_if_single:NTF \l_tmpa_tl {
4853     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
4854       \exp_after:wN \str_set:Nn \exp_after:wN
4855         \l_stex_get_symbol_uri_str \l_tmpa_tl
4856       \__stex_copymodule_get_symbol_check:n { #1 }
4857     }{
4858       % TODO
4859       % tail is not a single group
4860     }
4861   }{
4862     % TODO
4863     % tail is not a single group
4864   }
4865 }
4866
4867 \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
4868   \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
4869     \msg_error:nxxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
4870       :~\seq_use:Nn #1 {,~}
4871     }
4872   }
4873 }
4874
4875 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
4876   % import module
4877   \stex_import_module_uri:nn { #1 } { #2 }
4878   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
4879   \stex_import_require_module:nnnn
4880     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4881     { \l_stex_import_path_str } { \l_stex_import_name_str }
4882
4883   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
4884   \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
4885
4886   % fields
4887   \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
4888   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4889     \seq_map_inline:cn {c_stex_module_##1_constants}{
4890       \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
4891         ##1 ? ####1
4892       }
4893     }
4894   }
4895
4896   % setup prop
4897   \seq_clear:N \l_tmpa_seq

```

```

4898 \exp_args:Nn \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
4899   name      = \l_stex_current_copymodule_name_str ,
4900   module    = \l_stex_current_module_str ,
4901   from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
4902   includes  = \l_tmpa_seq %,
4903   % fields  = \l_tmpa_seq
4904 }
4905 \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
4906   as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
4907 \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_seq {,
4908 \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,
4909
4910 \stex_if_do_html:T {
4911   \begin{stex_annotate_env} {#4} {
4912     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4913   }
4914   \stex_annotate_invisible:nnn{domain}{\l_stex_import_ns_str ?\l_stex_import_name_str}{}
4915 }
4916 }
4917
4918 \cs_new_protected:Nn \stex_copymodule_end:n {
4919   % apply to every field
4920   \def \l_tmpa_cs ##1 ##2 {#1}
4921
4922   \tl_clear:N \__stex_copymodule_module_tl
4923   \tl_clear:N \__stex_copymodule_exec_tl
4924
4925   %\prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
4926   \seq_clear:N \__stex_copymodule_fields_seq
4927
4928   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4929     \seq_map_inline:cn {c_stex_module_##1_constants}{
4930
4931       \tl_clear:N \__stex_copymodule_curr_symbol_tl % <- wrap in current symbol html
4932       \l_tmpa_cs{##1}{####1}
4933
4934       \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
4935         \str_set_eq:Nc \__stex_copymodule_curr_name_str {l__stex_copymodule_copymodule_##1?####1_name_str}
4936         \stex_if_do_html:T {
4937           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4938             \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?####1_name_str}}
4939           }
4940         }
4941       }{
4942         \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str /
4943       }
4944
4945       \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
4946       \prop_put:Nnx \l_tmpa_prop { name } \__stex_copymodule_curr_name_str
4947       \prop_put:Nnx \l_tmpa_prop { module } \l_stex_current_module_str
4948
4949       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
4950         \stex_if_do_html:T {
4951           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {

```

```

4952     $\stex_annotate_invisible:nnn{definiens}{\exp_after:wN \exp_not:N\csname l__st
4953   }
4954 }
4955 \prop_put:Nnn \l_tmpa_prop { defined } { true }
4956 }
4957
4958 \stex_add_constant_to_current_module:n \__stex_copymodule_curr_name_str
4959 \tl_put_right:Nx \__stex_copymodule_module_tl {
4960   \seq_clear:c {l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_n
4961   \prop_set_from_keyval:cn {
4962     l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_name_str _prop
4963   }{
4964     \prop_to_keyval:N \l_tmpa_prop
4965   }
4966 }
4967
4968 \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
4969   \stex_if_do_html:T {
4970     \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4971       \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1
4972     }
4973   }
4974   \tl_put_right:Nx \__stex_copymodule_module_tl {
4975     \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
4976       \stex_invoke_symbol:n {
4977         \l_stex_current_module_str ? \__stex_copymodule_curr_name_str
4978       }
4979     }
4980   }
4981 }
4982
4983 \seq_put_right:Nx \__stex_copymodule_fields_seq {\l_stex_current_module_str ? \__stex_
4984
4985 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4986   \stex_copy_notations:nn {\l_stex_current_module_str ? \__stex_copymodule_curr_name_s
4987 }
4988
4989 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4990   \stex_if_do_html:TF{
4991     \stex_annotate_invisible:nnn{assignment} {##1?####1} { \exp_after:wN \exp_not:n \e
4992   }{
4993     \exp_after:wN \exp_not:n \exp_after:wN {\__stex_copymodule_curr_symbol_tl}
4994   }
4995 }
4996 }
4997 }
4998
4999
5000 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \__stex_copymodule_fields_seq
5001 \tl_put_left:Nx \__stex_copymodule_module_tl {
5002   \prop_set_from_keyval:cn {
5003     l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _pro
5004   }{
5005     \prop_to_keyval:N \l_stex_current_copymodule_prop

```

```

5006     }
5007   }
5008
5009   \seq_gput_right:cx{c_stex_module_\l_stex_current_module_str _copymodules}{
5010     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
5011   }
5012
5013   \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
5014   \stex_debug:nn{copymodule}{result:\meaning \__stex_copymodule_module_tl}
5015   \stex_debug:nn{copymodule}{output:\meaning \__stex_copymodule_exec_tl}
5016
5017   \__stex_copymodule_exec_tl
5018   \stex_if_do_html:T {
5019     \end{stex_annotate_env}
5020   }
5021 }
5022
5023 \NewDocumentEnvironment {copymodule} { 0{} m m}{
5024   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ copymodule }
5025   \stex_deactivate_macro:Nn \symdecl {module~environments}
5026   \stex_deactivate_macro:Nn \symdef {module~environments}
5027   \stex_deactivate_macro:Nn \notation {module~environments}
5028   \stex_reactivate_macro:N \assign
5029   \stex_reactivate_macro:N \renamedec1
5030   \stex_reactivate_macro:N \donotcopy
5031   \stex_smsmode_do:
5032 }{
5033   \stex_copymodule_end:n {}
5034 }
5035
5036 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
5037   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ interpretmodule }
5038   \stex_deactivate_macro:Nn \symdecl {module~environments}
5039   \stex_deactivate_macro:Nn \symdef {module~environments}
5040   \stex_deactivate_macro:Nn \notation {module~environments}
5041   \stex_reactivate_macro:N \assign
5042   \stex_reactivate_macro:N \renamedec1
5043   \stex_reactivate_macro:N \donotcopy
5044   \stex_smsmode_do:
5045 }{
5046   \stex_copymodule_end:n {
5047     \tl_if_exist:cF {
5048       l__stex_copymodule_copymodule_##1?##2_def_tl
5049     }{
5050       \str_if_eq:eeF {
5051         \prop_item:cn{
5052           l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
5053       }{ true }{
5054         \msg_error:nxxx{stex}{error/interpretmodule/nodéfiniens}{
5055           ##1?##2
5056         }{\l_stex_current_copymodule_name_str}
5057       }
5058     }
5059   }

```

```

5060 }
5061
5062 \iffalse \begin{stex_annotate_env} \fi
5063 \NewDocumentEnvironment {realization} { 0{} m}{
5064   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #2 }{ realize }
5065   \stex_deactivate_macro:Nn \symdecl {module~environments}
5066   \stex_deactivate_macro:Nn \symdef {module~environments}
5067   \stex_deactivate_macro:Nn \notation {module~environments}
5068   \stex_reactivate_macro:N \donotcopy
5069   \stex_reactivate_macro:N \assign
5070   \stex_smsmode_do:
5071 }{
5072   \stex_import_module_uri:nn { #1 } { #2 }
5073   \tl_clear:N \__stex_copymodule_exec_tl
5074   \tl_set:Nx \__stex_copymodule_module_tl {
5075     \stex_import_require_module:nnnn
5076     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
5077     { \l_stex_import_path_str } { \l_stex_import_name_str }
5078   }
5079   \exp_args:Nx \stex_add_import_to_current_module:n{
5080     \l_stex_import_ns_str ? \l_stex_import_name_str
5081   }
5082
5083   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
5084     \seq_map_inline:cn {c_stex_module_##1_constants}{
5085       \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str / #
5086       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
5087         \stex_if_do_html:T {
5088           \tl_put_right:Nx \__stex_copymodule_exec_tl {
5089             \stex_annotate_invisible:nnn{assignment} {##1?####1} {
5090               $\stex_annotate_invisible:nnn{definiens}{}{\exp_after:wN \exp_not:N\csname l__
5091             }
5092           }
5093         }
5094         \tl_put_right:Nx \__stex_copymodule_module_tl {
5095           \prop_put:cnn {l_stex_symdecl_##1?####1_prop}{ defined }{ true }
5096         }
5097       }
5098     }}
5099
5100   \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
5101
5102   \__stex_copymodule_exec_tl
5103   \stex_if_do_html:T {\end{stex_annotate_env}}
5104 }
5105
5106 \NewDocumentCommand \donotcopy { m }{
5107   \str_clear:N \l_stex_import_name_str
5108   \str_set:Nn \l_tmpa_str { #1 }
5109   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
5110   \seq_map_inline:Nn \l_stex_all_modules_seq {
5111     \str_set:Nn \l_tmpb_str { ##1 }
5112     \str_if_eq:eeT { \l_tmpa_str } {
5113       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }

```

```

5114 } {
5115   \seq_map_break:n {
5116     \stex_if_do_html:T {
5117       \stex_if_smsmode:F {
5118         \stex_annotate_invisible:nnn{donotcopy}{##1}{
5119           \stex_annotate:nnn{domain}{##1}{}}
5120       }
5121     }
5122   }
5123   \str_set_eq:NN \l_stex_import_name_str \l_tmpb_str
5124 }
5125 }
5126 \seq_map_inline:cn {c_stex_module_###1_copymodules}{
5127   \str_set:Nn \l_tmpb_str { ###1 }
5128   \str_if_eq:eeT { \l_tmpa_str } {
5129     \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
5130   } {
5131     \seq_map_break:n {\seq_map_break:n {
5132       \stex_if_do_html:T {
5133         \stex_if_smsmode:F {
5134           \stex_annotate_invisible:nnn{donotcopy}{####1}{
5135             \stex_annotate:nnn{domain}{
5136               \prop_item:cn {l_stex_copymodule_ ####1 _prop}{module}
5137             }{}}
5138         }
5139       }
5140     }
5141     \str_set:Nx \l_stex_import_name_str {
5142       \prop_item:cn {l_stex_copymodule_ ####1 _prop}{module}
5143     }
5144   }}
5145 }
5146 }
5147 }
5148 \str_if_empty:NTF \l_stex_import_name_str {
5149   % TODO throw error
5150 }{
5151   \stex_collect_imports:n {\l_stex_import_name_str }
5152   \seq_map_inline:Nn \l_stex_collect_imports_seq {
5153     \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
5154     \seq_map_inline:cn {c_stex_module_###1_constants}{
5155       \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? ####1 }
5156       \bool_lazy_any:nT {
5157         { \cs_if_exist_p:c {l__stex_copymodule_copymodule_###1?####1_name_str}}
5158         { \cs_if_exist_p:c {l__stex_copymodule_copymodule_###1?####1_macroname_str}}
5159         { \cs_if_exist_p:c {l__stex_copymodule_copymodule_###1?####1_def_tl}}
5160       }{
5161         % TODO throw error
5162       }
5163     }
5164   }
5165   \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
5166   \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_name_str }
5167   \prop_put:Nno \l_stex_current_copymodule_prop {includes} \l_tmpa_seq

```



```

5168 }
5169 \stex_smsmode_do:
5170 }
5171
5172 \NewDocumentCommand \assign { m m }{
5173   \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
5174   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
5175   \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _def_tl}{#2}
5176   \stex_smsmode_do:
5177 }
5178
5179 \keys_define:nn { stex / renamedec1 } {
5180   name          .str_set_x:N = \l_stex_renamedec1_name_str
5181 }
5182 \cs_new_protected:Nn \__stex_copymodule_renamedec1_args:n {
5183   \str_clear:N \l_stex_renamedec1_name_str
5184   \keys_set:nn { stex / renamedec1 } { #1 }
5185 }
5186
5187 \NewDocumentCommand \renamedec1 { O{} m m }{
5188   \__stex_copymodule_renamedec1_args:n { #1 }
5189   \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
5190   \stex_debug:nn{renamedec1}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
5191   \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
5192   \str_if_empty:NTF \l_stex_renamedec1_name_str {
5193     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
5194       \l_stex_get_symbol_uri_str
5195     } }
5196   } {
5197     \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex
5198     \stex_debug:nn{renamedec1}{@~\l_stex_current_module_str ? \l_stex_renamedec1_name_str}
5199     \prop_set_eq:cc {l_stex_symdecl_
5200       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
5201     _prop
5202     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
5203     \seq_set_eq:cc {l_stex_symdecl_
5204       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
5205     _notations
5206     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
5207     \prop_put:cnx {l_stex_symdecl_
5208       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
5209     _prop
5210     }{ name }{ \l_stex_renamedec1_name_str }
5211     \prop_put:cnx {l_stex_symdecl_
5212       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
5213     _prop
5214     }{ module }{ \l_stex_current_module_str }
5215     \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
5216       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
5217     }
5218     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
5219       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
5220     } }
5221   }

```

```

5222 \stex_smsmode_do:
5223 }
5224
5225 \stex_deactivate_macro:Nn \assign {copymodules}
5226 \stex_deactivate_macro:Nn \renamedekl {copymodules}
5227 \stex_deactivate_macro:Nn \donotcopy {copymodules}
5228
5229

```

33.2 The feature environment

`structural@feature (env.)`

```

5230 <@@=stex_features>
5231
5232 \NewDocumentEnvironment{structural_feature_module}{ m m m }{
5233   \stex_if_in_module:F {
5234     \msg_set:nnn{stex}{error/nomodule}{
5235       Structural~Feature~has~to~occur~in~a~module:\\
5236       Feature~#2~of~type~#1\\
5237       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
5238     }
5239     \msg_error:nn{stex}{error/nomodule}
5240   }
5241
5242   \str_set_eq:NN \l_stex_feature_parent_str \l_stex_current_module_str
5243
5244   \stex_module_setup:nn{meta=NONE}{#2 - #1}
5245
5246   \stex_if_do_html:T {
5247     \begin{stex_annotate_env}{ feature:#1 }{\l_stex_feature_parent_str ? #2 - #1}
5248     \stex_annotate_invisible:nnn{header}{}{ #3 }
5249   }
5250 }{
5251   \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
5252   \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
5253   \stex_debug:nn{features}{
5254     Feature: \l_stex_last_feature_str
5255   }
5256   \stex_if_do_html:T {
5257     \end{stex_annotate_env}
5258   }
5259 }

```

33.3 Structure

`structure (env.)`

```

5260 <@@=stex_structures>
5261 \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
5262   \prop_if_exist:cF {c_stex_module_ \l_stex_current_module_str _structures}{
5263     \prop_new:c {c_stex_module_ \l_stex_current_module_str _structures}
5264   }
5265   \prop_gput:cxn{c_stex_module_ \l_stex_current_module_str _structures}

```

```

5266     {#1}{#2}
5267 }
5268
5269 \keys_define:nn { stex / features / structure } {
5270   name          .str_set_x:N = \l__stex_structures_name_str ,
5271 }
5272
5273 \cs_new_protected:Nn \__stex_structures_structure_args:n {
5274   \str_clear:N \l__stex_structures_name_str
5275   \keys_set:nn { stex / features / structure } { #1 }
5276 }
5277 \NewDocumentEnvironment{mathstructure}{m O{}}{
5278   \begin{mathstructure_inner}{#1}[#2]
5279     \stex_smsmode_do:
5280     \ignorespacesandpars
5281   }\end{mathstructure_inner}}
5282 \NewDocumentEnvironment{mathstructure_inner}{m O{}}{
5283   \__stex_structures_structure_args:n { #2 }
5284   \str_if_empty:NT \l__stex_structures_name_str {
5285     \str_set:Nx \l__stex_structures_name_str { #1 }
5286   }
5287   \stex_suppress_html:n {
5288     \bool_set_true:N \l_stex_symdecl_make_macro_bool
5289     \exp_args:Nx \stex_symdecl_do:nn {
5290       name = \l__stex_structures_name_str ,
5291       def = {\STEXsymbol{module-type}}{
5292         \STEXInternalTermMathOMSiiii {
5293           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
5294             { ns } ?
5295           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
5296             { name } / \l__stex_structures_name_str - structure
5297         }{}{}{}
5298       }}
5299     }{ #1 }
5300   }
5301   \exp_args:Nnnx
5302   \begin{structural_feature_module}{ structure }
5303     { \l__stex_structures_name_str }{}
5304 }{
5305   \end{structural_feature_module}
5306   \_stex_reset_up_to_module:n \l_stex_last_feature_str
5307   \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
5308   \seq_clear:N \l_tmpa_seq
5309   \seq_map_inline:Nn \l_stex_collect_imports_seq {
5310     \seq_map_inline:cn{c_stex_module_##1_constants}{
5311       \seq_put_right:Nn \l_tmpa_seq { ##1 ? ####1 }
5312     }
5313   }
5314   \exp_args:Nnno
5315   \prop_gput:cn {c_stex_module_ \l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
5316   \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
5317   \stex_add_structure_to_current_module:nn
5318     \l__stex_structures_name_str
5319     \l_stex_last_feature_str

```

```

5320
5321 \stex_execute_in_module:x {
5322   \tl_set:cn { #1 }{
5323     \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures
5324   }
5325 }
5326 }
5327
5328 \cs_new:Nn \stex_invoke_structure:nn {
5329   \stex_invoke_symbol:n { #1?#2 }
5330 }
5331
5332 \cs_new_protected:Nn \stex_get_structure:n {
5333   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
5334     \tl_set:Nn \l_tmpa_tl { #1 }
5335     \__stex_structures_get_from_cs:
5336   }{
5337     \cs_if_exist:cTF { #1 }{
5338       \cs_set_eq:Nc \l_tmpa_cs { #1 }
5339       \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
5340       \str_if_empty:NNTF \l_tmpa_str {
5341         \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs} \stex_invoke_structure:nn {
5342           \__stex_structures_get_from_cs:
5343         }{
5344           \__stex_structures_get_from_string:n { #1 }
5345         }
5346       }{
5347         \__stex_structures_get_from_string:n { #1 }
5348       }
5349     }{
5350       \__stex_structures_get_from_string:n { #1 }
5351     }
5352   }
5353 }
5354
5355 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
5356   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
5357   { \tl_tail:N \l_tmpa_tl }
5358   \str_set:Nx \l_tmpa_str {
5359     \exp_after:wN \use_i:nn \l_tmpa_tl
5360   }
5361   \str_set:Nx \l_tmpb_str {
5362     \exp_after:wN \use_ii:nn \l_tmpa_tl
5363   }
5364   \str_set:Nx \l_stex_get_structure_str {
5365     \l_tmpa_str ? \l_tmpb_str
5366   }
5367   \str_set:Nx \l_stex_get_structure_module_str {
5368     \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
5369   }
5370 }
5371
5372 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
5373   \tl_set:Nn \l_tmpa_tl {

```

```

5374 \msg_error:nnn{stex}{error/unknownstructure}{#1}
5375 }
5376 \str_set:Nn \l_tmpa_str { #1 }
5377 \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
5378
5379 \seq_map_inline:Nn \l_stex_all_modules_seq {
5380   \prop_if_exist:cT {c_stex_module_##1_structures} {
5381     \prop_map_inline:cn {c_stex_module_##1_structures} {
5382       \exp_args:No \str_if_eq:nnT \l_tmpa_str {####1}{
5383         %\str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
5384         \prop_map_break:n{\seq_map_break:n{
5385           \tl_set:Nn \l_tmpa_tl {
5386             \str_set:Nn \l_stex_get_structure_str {##1?####1}
5387             \str_set:Nn \l_stex_get_structure_module_str {####2}
5388           }
5389         }}
5390       }
5391     }
5392   }
5393 }
5394 \l_tmpa_tl
5395 }

```

\instantiate

```

5396
5397 \NewDocumentEnvironment{usestructure}{m}{
5398   \stex_get_structure:n {#1}
5399   \exp_args:Nnx \stex_debug:nn{features}{using~structure:~\l_stex_get_structure_module_str}
5400   \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5401 }{}
5402
5403 \keys_define:nn { stex / instantiate } {
5404   name .str_set_x:N = \l__stex_structures_name_str
5405 }
5406 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
5407   \str_clear:N \l__stex_structures_name_str
5408   \keys_set:nn { stex / instantiate } { #1 }
5409 }
5410
5411 \NewDocumentEnvironment{extstructure}{m m O{}}{
5412   \begin{mathstructure_inner}{#1}[#3]
5413     \seq_set_split:Nnn\__stex_structures_extstructure_imports_seq,{#2}
5414     \seq_map_inline:Nn\__stex_structures_extstructure_imports_seq {
5415       \stex_get_structure:n {##1}
5416       \exp_args:Nnx \stex_debug:nn{features}{importing~structure:~\l_stex_get_structure_modu
5417       \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5418       \stex_if_smsmode:F {
5419         \stex_annotate_invisible:nnn
5420         {import} {\l_stex_get_structure_module_str} {}
5421       }
5422       \exp_args:Nx \stex_add_import_to_current_module:n {
5423         \l_stex_get_structure_module_str
5424       }
5425       \exp_args:Nx \stex_add_to_current_module:n {

```

```

5426         \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5427     }
5428 }
5429 \stex_smsmode_do:
5430 \ignorespacesandpars
5431 }{
5432 \end{mathstructure_inner}
5433 }
5434
5435 \NewDocumentEnvironment{extstructure*}{m m O{}}{
5436 % TODO
5437 \begin{extstructure}{#1}{#2}{#3}
5438 }{
5439 \end{extstructure}
5440 }
5441
5442 \NewDocumentCommand \instantiate {m O{ } m m O{}}{
5443 \begingroup
5444 \stex_get_structure:n {#3}
5445 \__stex_structures_instantiate_args:n { #2 }
5446 \str_if_empty:NT \l__stex_structures_name_str {
5447 \str_set:Nn \l__stex_structures_name_str { #1 }
5448 }
5449 \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5450 \seq_clear:N \l__stex_structures_fields_seq
5451 \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
5452 \seq_map_inline:Nn \l_stex_collect_imports_seq {
5453 \seq_map_inline:cn {c_stex_module_##1_constants}{
5454 \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? #####1 }
5455 }
5456 }
5457
5458 \tl_if_empty:nF{#5}{
5459 \seq_set_split:Nnn \l_tmpa_seq , {#5}
5460 \prop_clear:N \l_tmpa_prop
5461 \seq_map_inline:Nn \l_tmpa_seq {
5462 \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
5463 \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
5464 \msg_error:nnn{stex}{error/keyval}{##1}
5465 }
5466 \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
5467 \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
5468 \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
5469 \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}
5470 \exp_args:Nxx \str_if_eq:nnF
5471 {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5472 {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
5473 \msg_error:nnxxx{stex}{error/incompatible}
5474 {\l__stex_structures_dom_str}
5475 {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5476 {\l_stex_get_symbol_uri_str}
5477 {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
5478 }
5479 \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str

```

```

5480     }
5481 }
5482
5483 \seq_map_inline:Nn \l__stex_structures_fields_seq {
5484   \str_set:Nx \l_tmpa_str {field:\l__stex_structures_name_str . \prop_item:cn {l_stex_sy
5485   \stex_debug:nn{instantiate}{Field~\l_tmpa_str :~##1}
5486
5487   \stex_add_constant_to_current_module:n {\l_tmpa_str}
5488   \stex_execute_in_module:x {
5489     \prop_set_from_keyval:cn { l_stex_symdecl_ \l_stex_current_module_str?\l_tmpa_str _p
5490       name   = \l_tmpa_str ,
5491       args   = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
5492       arity  = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
5493       assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs} ,
5494       argnames = {\prop_item:cn {l_stex_symdecl_##1_prop}{argnames}}
5495     }
5496     \seq_clear:c {l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _notations}
5497   }
5498
5499   \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
5500     \stex_find_notation:nn{##1}{ }
5501     \stex_execute_in_module:x {
5502       \seq_put_right:cn {l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _notation
5503     }
5504
5505     \stex_copy_control_sequence_ii:ccN
5506     {stex_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_notation_
5507     {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5508     \l_tmpa_tl
5509     \exp_args:No \stex_execute_in_module:n \l_tmpa_tl
5510
5511
5512     \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
5513       \tl_set_eq:Nc \l_tmpa_cs {stex_op_notation_##1\c_hash_str \l_stex_notation_variant
5514       \stex_execute_in_module:x {
5515         \tl_set:cn
5516         {stex_op_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_not
5517         { \exp_args:No \exp_not:n \l_tmpa_cs}
5518       }
5519     }
5520
5521   }
5522
5523   \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\l_stex_cur
5524 }
5525
5526 \stex_execute_in_module:x {
5527   \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
5528   domain = \l_stex_get_structure_module_str ,
5529   \prop_to_keyval:N \l_tmpa_prop
5530 }
5531 \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
5532 }
5533 \stex_debug:nn{instantiate}{

```

```

5534 Instance~\l_stex_current_module_str?\l__stex_structures_name_str \\  

5535 \prop_to_keyval:N \l_tmpa_prop  

5536 }  

5537 \exp_args:Nxx \stex_symdecl_do:nn {  

5538   type={\STEXsymbol{module-type}}{  

5539     \STEXInternalTermMathOMSiiii {  

5540       \l_stex_get_structure_module_str  

5541     }{}{0}{}  

5542   }}  

5543 }{\l__stex_structures_name_str}  

5544 % {  

5545   \str_set:Nx \l_stex_get_symbol_uri_str {\l_stex_current_module_str?\l__stex_structures  

5546   \tl_set:Nn \l_stex_notation_after_do_tl {\__stex_notation_final:}  

5547   \stex_notation_do:nnnnn{}{0}{}{\comp{#4}}  

5548 % }  

5549 %\exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}  

5550 \endgroup  

5551 \stex_smsmode_do:\ignorespacesandpars  

5552 }  

5553  

5554 \cs_new_protected:Nn \stex_symbol_or_var:n {  

5555   \cs_if_exist:cTF{#1}{  

5556     \cs_set_eq:Nc \l_tmpa_tl { #1 }  

5557     \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }  

5558     \str_if_empty:NTF \l_tmpa_str {  

5559       \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }  

5560       \stex_invoke_variable:n {  

5561         \bool_set_true:N \l_stex_symbol_or_var_bool  

5562         \bool_set_false:N \l_stex_instance_or_symbol_bool  

5563         \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}  

5564         \tl_set:Nx \l_tmpa_tl {\exp_after:wN \use:n \l_tmpa_tl}  

5565         \str_set:Nx \l_stex_get_symbol_uri_str {  

5566           \exp_after:wN \use:n \l_tmpa_tl  

5567         }  

5568       }{ % TODO \stex_invoke_varinstance:n  

5569         \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl } \stex_invoke_varinstance:n {  

5570           \bool_set_true:N \l_stex_symbol_or_var_bool  

5571           \bool_set_true:N \l_stex_instance_or_symbol_bool  

5572           \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}  

5573           \tl_set:Nx \l_tmpa_tl {\exp_after:wN \use:n \l_tmpa_tl}  

5574           \str_set:Nx \l_stex_get_symbol_uri_str {  

5575             \exp_after:wN \use:n \l_tmpa_tl  

5576           }  

5577         }{  

5578           \bool_set_false:N \l_stex_symbol_or_var_bool  

5579           \stex_get_symbol:n{#1}  

5580         }  

5581       }  

5582     }{  

5583       \__stex_structures_symbolorvar_from_string:n{ #1 }  

5584     }  

5585   }{  

5586     \__stex_structures_symbolorvar_from_string:n{ #1 }  

5587   }  


```



```

5588 }
5589
5590 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
5591   \prop_if_exist:cTF {l_stex_symdecl_var://#1 _prop}{
5592     \bool_set_true:N \l_stex_symbol_or_var_bool
5593     \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
5594   }{
5595     \bool_set_false:N \l_stex_symbol_or_var_bool
5596     \stex_get_symbol:n{#1}
5597   }
5598 }
5599
5600 \keys_define:nn { stex / varinstantiate } {
5601   name .str_set_x:N = \l__stex_structures_name_str,
5602   bind .choices:nn =
5603     {forall,exists}
5604     {\str_set:Nx \l__stex_structures_bind_str {\l_keys_choice_tl}}
5605 }
5606
5607 \cs_new_protected:Nn \__stex_structures_varinstantiate_args:n {
5608   \str_clear:N \l__stex_structures_name_str
5609   \str_clear:N \l__stex_structures_bind_str
5610   \keys_set:nn { stex / varinstantiate } { #1 }
5611 }
5612
5613 \NewDocumentCommand \varinstantiate {m O{}} m m O{}{}{
5614   \begingroup
5615     \stex_get_structure:n {#3}
5616     \__stex_structures_varinstantiate_args:n { #2 }
5617     \str_if_empty:NT \l__stex_structures_name_str {
5618       \str_set:Nn \l__stex_structures_name_str { #1 }
5619     }
5620     \stex_if_do_html:TF{
5621       \stex_annotate:nnn{varinstance}{\l__stex_structures_name_str}
5622     }{\use:n}
5623     {
5624       \stex_if_do_html:T{
5625         \stex_annotate_invisible:nnn{domain}{\l_stex_get_structure_module_str}{}
5626       }
5627       \seq_clear:N \l__stex_structures_fields_seq
5628       \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
5629       \seq_map_inline:Nn \l_stex_collect_imports_seq {
5630         \seq_map_inline:cn {c_stex_module_##1_constants}{
5631           \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
5632         }
5633       }
5634       \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5635       \prop_clear:N \l_tmpa_prop
5636       \tl_if_empty:nF {#5} {
5637         \seq_set_split:Nnn \l_tmpa_seq , {#5}
5638         \seq_map_inline:Nn \l_tmpa_seq {
5639           \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
5640           \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
5641             \msg_error:nnn{stex}{error/keyval}{##1}

```

```

5642     }
5643     \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_structures_dom_str
5644     \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
5645     \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_uri_str
5646     \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
5647     \stex_if_do_html:T{
5648         \stex_annotate:nnn{assign}{\l__stex_structures_dom_str,
5649         \bool_if:NTF\l_stex_symbol_or_var_bool{var://}{\l_stex_get_symbol_uri_str}{}}
5650     }
5651     \bool_if:NTF \l_stex_symbol_or_var_bool {
5652         \exp_args:Nxx \str_if_eq:nnF
5653             {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5654             {\prop_item:cn{l_stex_symdecl_var://\l_stex_get_symbol_uri_str _prop}{args}}}{
5655             \msg_error:nnxxxx{stex}{error/incompatible}
5656             {\l__stex_structures_dom_str}
5657             {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5658             {\l_stex_get_symbol_uri_str}
5659             {\prop_item:cn{l_stex_symdecl_var://\l_stex_get_symbol_uri_str _prop}{args}}
5660         }
5661         \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n {
5662     }}{
5663         \exp_args:Nxx \str_if_eq:nnF
5664             {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5665             {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}}{
5666             \msg_error:nnxxxx{stex}{error/incompatible}
5667             {\l__stex_structures_dom_str}
5668             {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5669             {\l_stex_get_symbol_uri_str}
5670             {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
5671         }
5672         \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n {
5673     }}
5674     }
5675 }
5676 \tl_gclear:N \g__stex_structures_aftergroup_tl
5677 \seq_map_inline:Nn \l__stex_structures_fields_seq {
5678     \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdecl_
5679     \stex_debug:nn{varinstantiate}{Field~\l_tmpa_str :~##1}
5680     \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
5681         \stex_find_notation:nn{##1}{}
5682         \cs_gset_eq:cc{g__stex_structures_tmpa_\l_tmpa_str _cs}
5683             {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5684         \stex_debug:nn{varinstantiate}{Notation:~\cs_meaning:c{g__stex_structures_tmpa_\l
5685         \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
5686             \cs_gset_eq:cc {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
5687             {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5688             \stex_debug:nn{varinstantiate}{Operator~Notation:~\cs_meaning:c{g__stex_struct
5689     }}
5690 }
5691 }
5692 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
5693     \prop_set_from_keyval:cn { l_stex_symdecl_ var://\l_tmpa_str _prop}{
5694         name      = \l_tmpa_str ,
5695         args      = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,

```

```

5696         arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
5697         assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs} ,
5698         argnames = {\prop_item:cn {l_stex_symdecl_##1_prop}{argnames}} ,
5699     }
5700     \cs_set_eq:cc {stex_var_notation_\l_tmpa_str_cs}
5701     {g__stex_structures_tmpa_\l_tmpa_str_cs}
5702     \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str_cs}
5703     {g__stex_structures_tmpa_op_\l_tmpa_str_cs}
5704 }
5705 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\stex_inv
5706 }
5707 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
5708     \prop_set_from_keyval:cn {l_stex_varinstance_\l__stex_structures_name_str_prop }{
5709         domain = \l_stex_get_structure_module_str ,
5710         \prop_to_keyval:N \l_tmpa_prop
5711     }
5712     \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
5713     \tl_set:cn {l_stex_varinstance_\l__stex_structures_name_str_op_tl}{
5714         \exp_args:Nnx \exp_not:N \use:nn {
5715             \str_set:Nn \exp_not:N \STEXInternalCurrentSymbolStr {var://\l__stex_structures_
5716                 \_stex_term_omv:nn {var://\l__stex_structures_name_str}{
5717                     \exp_not:n{
5718                         \_varcomp{#4}
5719                     }
5720                 }
5721             }{
5722                 \exp_not:n{\_stex_reset:N \STEXInternalCurrentSymbolStr}
5723             }
5724         }
5725     }
5726 }
5727 \stex_debug:nn{varinstantiate}{\expandafter\detokenize\expandafter{\g__stex_structures_a
5728     \aftergroup\g__stex_structures_aftergroup_tl
5729 }
5730 \endgroup
5731 \stex_smsmode_do:\ignorespacesandpars
5732 }
5733 \cs_new_protected:Nn \stex_invoke_instance:n {
5734     \peek_charcode_remove:NTF ! {
5735         \stex_invoke_symbol:n{#1}
5736     }{
5737         \_stex_invoke_instance:nn {#1}
5738     }
5739 }
5740
5741
5742 \cs_new_protected:Nn \stex_invoke_varinstance:n {
5743     \peek_charcode_remove:NTF ! {
5744         \exp_args:Nnx \use:nn {
5745             \def\comp{\_varcomp}
5746             \use:c{l_stex_varinstance_#1_op_tl}
5747         }{
5748             \_stex_reset:N \comp
5749         }

```

```

5750 }{
5751   \_stex_invoke_varinstance:nn {#1}
5752 }
5753 }
5754
5755 \cs_new_protected:Nn \_stex_invoke_instance:nn {
5756   \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
5757     \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
5758   }{
5759     \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
5760     \msg_error:nnxxx{stex}{error/unknownfield}{#2}{#1}{
5761       \prop_to_keyval:N \l_tmpa_prop
5762     }
5763   }
5764 }
5765
5766 \cs_new_protected:Nn \_stex_invoke_varinstance:nn {
5767   \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
5768     \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
5769     \l_tmpa_tl
5770   }{
5771     \msg_error:nnnnn{stex}{error/unknownfield}{#2}{#1}{
5772   }
5773 }

```

(End definition for `\instantiate`. This function is documented on page 38.)

`\stex_invoke_structure:nnn`

```

5774 % #1: URI of the instance
5775 % #2: URI of the instantiated module
5776 \cs_new_protected:Nn \stex_invoke_structure:nnn {
5777   \tl_if_empty:nTF{ #3 }{
5778     \prop_set_eq:Nc \l__stex_structures_structure_prop {
5779       c_stex_feature_ #2 _prop
5780     }
5781     \tl_clear:N \l_tmpa_tl
5782     \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
5783     \seq_map_inline:Nn \l_tmpa_seq {
5784       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
5785       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
5786       \cs_if_exist:cT {
5787         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
5788       }{
5789         \tl_if_empty:NF \l_tmpa_tl {
5790           \tl_put_right:Nn \l_tmpa_tl {,}
5791         }
5792         \tl_put_right:Nx \l_tmpa_tl {
5793           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
5794         }
5795       }
5796     }
5797     \exp_args:No \mathstrut \l_tmpa_tl
5798   }{
5799     \stex_invoke_symbol:n{#1/#3}

```

```

5800 }
5801 }

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

5802 </package>

```

Chapter 34

STEX -Statements Implementation

```
5803 <*package>
5804
5805 %%%%%%%%%%% features.dtx %%%%%%%%%%%
5806
5807 <@@=stex_statements>
    Warnings and error messages
5808
\titleemph
5809 \def\titleemph#1{\textbf{#1}}
    (End definition for \titleemph. This function is documented on page ??.)
```

34.1 Definitions

definiendum

```
5810 \keys_define:nn {stex / definiendum }{
5811   pre      .tl_set:N      = \l__stex_statements_definiendum_pre_tl,
5812   post     .tl_set:N      = \l__stex_statements_definiendum_post_tl,
5813   root     .str_set_x:N    = \l__stex_statements_definiendum_root_str,
5814   gfa      .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
5815 }
5816 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
5817   \str_clear:N \l__stex_statements_definiendum_root_str
5818   \tl_clear:N \l__stex_statements_definiendum_post_tl
5819   \str_clear:N \l__stex_statements_definiendum_gfa_str
5820   \keys_set:nn { stex / definiendum }{ #1 }
5821 }
5822 \NewDocumentCommand \definiendum { O{} m m } {
5823   \__stex_statements_definiendum_args:n { #1 }
5824   \stex_get_symbol:n { #2 }
5825   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5826   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
5827     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

```

5828     \tl_set:Nn \l_tmpa_tl { #3 }
5829   } {
5830     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
5831     \tl_set:Nn \l_tmpa_tl {
5832       \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
5833     }
5834   }
5835 } {
5836   \tl_set:Nn \l_tmpa_tl { #3 }
5837 }
5838
5839 % TODO root
5840 \stex_html_backend:TF {
5841   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
5842 } {
5843   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
5844 }
5845 }
5846 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

(End definition for definiendum. This function is documented on page 48.)

definame

```

5847
5848 \NewDocumentCommand \definame { 0{ } m } {
5849   \__stex_statements_definiendum_args:n { #1 }
5850   % TODO: root
5851   \stex_get_symbol:n { #2 }
5852   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5853   \str_set:Nx \l_tmpa_str {
5854     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5855   }
5856   \str_replace_all:Nnn \l_tmpa_str {-} {~}
5857   \stex_html_backend:TF {
5858     \stex_if_do_html:T {
5859       \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5860         \l_tmpa_str\l__stex_statements_definiendum_post_tl
5861       }
5862     }
5863   } {
5864     \exp_args:Nnx \defemph@uri {
5865       \l_tmpa_str\l__stex_statements_definiendum_post_tl
5866     } { \l_stex_get_symbol_uri_str }
5867   }
5868 }
5869 \stex_deactivate_macro:Nn \definame {definition~environments}
5870
5871 \NewDocumentCommand \Definame { 0{ } m } {
5872   \__stex_statements_definiendum_args:n { #1 }
5873   \stex_get_symbol:n { #2 }
5874   \str_set:Nx \l_tmpa_str {
5875     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5876   }
5877   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}

```

```

5878 \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5879 \stex_html_backend:TF {
5880   \stex_if_do_html:T {
5881     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5882       \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5883     }
5884   }
5885 } {
5886   \exp_args:Nnx \defemph@uri {
5887     \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5888   } { \l_stex_get_symbol_uri_str }
5889 }
5890 }
5891 \stex_deactivate_macro:Nn \Definame {definition-environments}
5892
5893 \NewDocumentCommand \premise { m }{
5894   \noindent\stex_annotate:nnn{ premise }{}{\ignorespaces #1 }
5895 }
5896 \NewDocumentCommand \conclusion { m }{
5897   \noindent\stex_annotate:nnn{ conclusion }{}{\ignorespaces #1 }
5898 }
5899 \NewDocumentCommand \definiens { 0{} m }{
5900   \str_clear:N \l_stex_get_symbol_uri_str
5901   \tl_if_empty:nF {#1} {
5902     \stex_get_symbol:n { #1 }
5903   }
5904   \str_if_empty:NT \l_stex_get_symbol_uri_str {
5905     \int_compare:nNnTF {\clist_count:N \l__stex_statements_sdefinition_for_clist} = 1 {
5906       \str_set:Nx \l_stex_get_symbol_uri_str {\clist_item:Nn \l__stex_statements_sdefinition
5907     }{
5908       % TODO throw error
5909     }
5910   }
5911   \str_if_eq:eeT {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{module}}
5912   {\l_stex_current_module_str}{
5913     \str_if_eq:eeF {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defin
5914   }{true}{
5915     \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5916     \exp_args:Nx \stex_add_to_current_module:n {
5917       \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5918     }
5919   }
5920 }
5921 \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
5922 }
5923
5924 \NewDocumentCommand \varbindforall {m}{
5925   \stex_symbol_or_var:n {#1}
5926   \bool_if:NTF\l_stex_symbol_or_var_bool{
5927     \stex_if_do_html:T {
5928       \stex_annotate_invisible:nnn {bindtype}{forall,\l_stex_get_symbol_uri_str}{}
5929     }
5930   }{
5931     % todo throw error

```



```

5932 }
5933 }
5934
5935 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
5936 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
5937 \stex_deactivate_macro:Nn \definiens {definition~environments}
5938 \stex_deactivate_macro:Nn \varbindforall {definition~or~assertion~environments}
5939

```

(End definition for definame. This function is documented on page 48.)

sdefinition (env.)

```

5940
5941 \keys_define:nn {stex / sdefinition }{
5942   type      .str_set_x:N = \sdefinitiontype,
5943   id        .str_set_x:N = \sdefinitionid,
5944   name      .str_set_x:N = \sdefinitionname,
5945   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5946   title     .tl_set:N     = \sdefinitiontitle
5947 }
5948 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
5949   \str_clear:N \sdefinitiontype
5950   \str_clear:N \sdefinitionid
5951   \str_clear:N \sdefinitionname
5952   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5953   \tl_clear:N \sdefinitiontitle
5954   \keys_set:nn { stex / sdefinition }{ #1 }
5955 }
5956
5957 \NewDocumentEnvironment{sdefinition}{0{}}{
5958   \__stex_statements_sdefinition_args:n{ #1 }
5959   \stex_reactivate_macro:N \definiendum
5960   \stex_reactivate_macro:N \definame
5961   \stex_reactivate_macro:N \Definame
5962   \stex_reactivate_macro:N \premise
5963   \stex_reactivate_macro:N \definiens
5964   \stex_reactivate_macro:N \varbindforall
5965   \stex_if_smsmode:F{
5966     \seq_clear:N \l_tmpb_seq
5967     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5968       \tl_if_empty:nF{ ##1 }{
5969         \stex_get_symbol:n { ##1 }
5970         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5971           \l_stex_get_symbol_uri_str
5972         }
5973       }
5974     }
5975     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5976     \exp_args:Nnnx
5977     \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpb_seq {,}}
5978     \str_if_empty:NF \sdefinitiontype {
5979       \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{ }
5980     }
5981     \str_if_empty:NF \sdefinitionname {

```

```

5982     \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5983   }
5984   \clist_set:No \l_tmpa_clist \sdefinitiontype
5985   \tl_clear:N \l_tmpa_tl
5986   \clist_map_inline:Nn \l_tmpa_clist {
5987     \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
5988       \tl_set:Nn \l_tmpa_tl {
5989         \stex_patch_counters:
5990         \use:c{__stex_statements_sdefinition_##1_start:}
5991         \stex_unpatch_counters:
5992       }
5993     }
5994   }
5995   \tl_if_empty:NTF \l_tmpa_tl {
5996     \__stex_statements_sdefinition_start:
5997   }{
5998     \l_tmpa_tl
5999   }
6000 }
6001 \stex_ref_new_doc_target:n \sdefinitionid
6002 \stex_smsmode_do:
6003 }{
6004   \stex_suppress_html:n {
6005     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
6006   }
6007   \stex_if_smsmode:F {
6008     \clist_set:No \l_tmpa_clist \sdefinitiontype
6009     \tl_clear:N \l_tmpa_tl
6010     \clist_map_inline:Nn \l_tmpa_clist {
6011       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
6012         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
6013       }
6014     }
6015     \tl_if_empty:NTF \l_tmpa_tl {
6016       \__stex_statements_sdefinition_end:
6017     }{
6018       \l_tmpa_tl
6019     }
6020     \end{stex_annotate_env}
6021   }
6022 }

```

\stexpatchdefinition

```

6023 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
6024   \stex_par:\noindent\titllemph{Definition\tl_if_empty:NF \sdefinitiontitle {
6025     ~(\sdefinitiontitle)
6026   }~}
6027 }
6028 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\stex_par:\medskip}
6029
6030 \newcommand\stexpatchdefinition[3] [] {
6031   \str_set:Nx \l_tmpa_str{ #1 }
6032   \str_if_empty:NTF \l_tmpa_str {
6033     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }

```

```

6034     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
6035   }{
6036     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2
6037     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
6038   }
6039 }

```

(End definition for `\stexpatchdefinition`. This function is documented on page 55.)

`\inlinedef` inline:

```

6040 \keys_define:nn {stex / inlinedef }{
6041   type      .str_set_x:N = \sdefinitiontype,
6042   id        .str_set_x:N = \sdefinitionid,
6043   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
6044   name      .str_set_x:N = \sdefinitionname
6045 }
6046 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
6047   \str_clear:N \sdefinitiontype
6048   \str_clear:N \sdefinitionid
6049   \str_clear:N \sdefinitionname
6050   \clist_clear:N \l__stex_statements_sdefinition_for_clist
6051   \keys_set:nn { stex / inlinedef }{ #1 }
6052 }
6053 \NewDocumentCommand \inlinedef { 0{} m } {
6054   \beginngroup
6055   \__stex_statements_inlinedef_args:n{ #1 }
6056   \stex_reactivate_macro:N \definiendum
6057   \stex_reactivate_macro:N \definame
6058   \stex_reactivate_macro:N \Definame
6059   \stex_reactivate_macro:N \premise
6060   \stex_reactivate_macro:N \definiens
6061   \stex_reactivate_macro:N \varbindforall
6062   \stex_ref_new_doc_target:n \sdefinitionid
6063   \stex_if_smsmode:TF{\stex_suppress_html:n {
6064     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
6065   }}{
6066     \seq_clear:N \l_tmpb_seq
6067     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
6068       \tl_if_empty:nF{ ##1 }{
6069         \stex_get_symbol:n { ##1 }
6070         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6071           \l_stex_get_symbol_uri_str
6072         }
6073       }
6074     }
6075     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
6076     \ifvmode\noindent\fi
6077     \exp_args:Nnx
6078     \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpb_seq {,}}{
6079       \str_if_empty:NF \sdefinitiontype {
6080         \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{
6081         }
6082       }
6083       \str_if_empty:NF \sdefinitionname {

```

```

6084         \stex_suppress_html:n{\stex_symdecl_do:nn}{\sdefinitionname}}
6085         \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}}
6086     }
6087 }
6088 }
6089 \endgroup
6090 \stex_smsmode_do:
6091 }

```

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

34.2 Assertions

`sassertion (env.)`

```

6092
6093 \keys_define:nn {stex / sassertion }{
6094     type      .str_set_x:N = \sassertiontype,
6095     id        .str_set_x:N = \sassertionid,
6096     title     .tl_set:N    = \sassertiontitle ,
6097     for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
6098     name      .str_set_x:N = \sassertionname
6099 }
6100 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
6101     \str_clear:N \sassertiontype
6102     \str_clear:N \sassertionid
6103     \str_clear:N \sassertionname
6104     \clist_clear:N \l__stex_statements_sassertion_for_clist
6105     \tl_clear:N \sassertiontitle
6106     \keys_set:nn { stex / sassertion }{ #1 }
6107 }
6108
6109 %\tl_new:N \g__stex_statements_aftergroup_tl
6110
6111 \NewDocumentEnvironment{sassertion}{0{}}{
6112     \__stex_statements_sassertion_args:n{ #1 }
6113     \stex_reactivate_macro:N \premise
6114     \stex_reactivate_macro:N \conclusion
6115     \stex_reactivate_macro:N \varbindforall
6116     \stex_if_smsmode:F {
6117         \seq_clear:N \l_tmpb_seq
6118         \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
6119             \tl_if_empty:nF{ ##1 }{
6120                 \stex_get_symbol:n { ##1 }
6121                 \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6122                     \l_stex_get_symbol_uri_str
6123                 }
6124             }
6125         }
6126         \exp_args:Nnnx
6127         \begin{sassertion}{\seq_use:Nn \l_tmpb_seq {,}}
6128         \str_if_empty:NF \sassertiontype {
6129             \stex_annotate_invisible:nnn{type}{\sassertiontype}{}}
6130     }

```

```

6131 \str_if_empty:NF \sassertionname {
6132   \stex_annotate_invisible:nnn{statementname}{\sassertionname}{\}
6133 }
6134 \clist_set:No \l_tmpa_clist \sassertiontype
6135 \tl_clear:N \l_tmpa_tl
6136 \clist_map_inline:Nn \l_tmpa_clist {
6137   \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
6138     \tl_set:Nn \l_tmpa_tl {
6139       \stex_patch_counters:
6140       \use:c{__stex_statements_sassertion_##1_start:}
6141       \stex_unpatch_counters:
6142     }
6143   }
6144 }
6145 \tl_if_empty:NTF \l_tmpa_tl {
6146   \__stex_statements_sassertion_start:
6147 }{
6148   \l_tmpa_tl
6149 }
6150 }
6151 \str_if_empty:NTF \sassertionid {
6152   \str_if_empty:NF \sassertionname {
6153     \stex_ref_new_doc_target:n {}
6154   }
6155 } {
6156   \stex_ref_new_doc_target:n \sassertionid
6157 }
6158 \stex_smsmode_do:
6159 }{
6160   \str_if_empty:NF \sassertionname {
6161     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
6162     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
6163   }
6164   \stex_if_smsmode:F {
6165     \clist_set:No \l_tmpa_clist \sassertiontype
6166     \tl_clear:N \l_tmpa_tl
6167     \clist_map_inline:Nn \l_tmpa_clist {
6168       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
6169         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
6170       }
6171     }
6172     \tl_if_empty:NTF \l_tmpa_tl {
6173       \__stex_statements_sassertion_end:
6174     }{
6175       \l_tmpa_tl
6176     }
6177     \end{stex_annotate_env}
6178   }
6179 }

```

\stexpatchassertion

```

6180
6181 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
6182   \stex_par:\noindent\titilemph{Assertion~\tl_if_empty:NF \sassertiontitle {

```

```

6183     (\sassertiontitle)
6184   }~}
6185 }
6186 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\stex_par:\medskip}
6187
6188 \newcommand\stexpatchassertion[3] [] {
6189   \str_set:Nx \l_tmpa_str{ #1 }
6190   \str_if_empty:NTF \l_tmpa_str {
6191     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
6192     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
6193   }{
6194     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
6195     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
6196   }
6197 }

```

(End definition for `\stexpatchassertion`. This function is documented on page 55.)

`\inlineass` inline:

```

6198 \keys_define:nn {stex / inlineass }{
6199   type      .str_set_x:N = \sassertiontype,
6200   id        .str_set_x:N = \sassertionid,
6201   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
6202   name      .str_set_x:N = \sassertionname
6203 }
6204 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
6205   \str_clear:N \sassertiontype
6206   \str_clear:N \sassertionid
6207   \str_clear:N \sassertionname
6208   \clist_clear:N \l__stex_statements_sassertion_for_clist
6209   \keys_set:nn { stex / inlineass }{ #1 }
6210 }
6211 \NewDocumentCommand \inlineass { 0{} m } {
6212   \begingroup
6213   \stex_reactivate_macro:N \premise
6214   \stex_reactivate_macro:N \conclusion
6215   \stex_reactivate_macro:N \varbindforall
6216   \__stex_statements_inlineass_args:n{ #1 }
6217   \str_if_empty:NTF \sassertionid {
6218     \str_if_empty:NF \sassertionname {
6219       \stex_ref_new_doc_target:n {}
6220     }
6221   } {
6222     \stex_ref_new_doc_target:n \sassertionid
6223   }
6224
6225   \stex_if_smsmode:TF{
6226     \str_if_empty:NF \sassertionname {
6227       \stex_suppress_html:n{\stex_symdecl_do:nn}{\sassertionname}}
6228     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
6229   }
6230 }{
6231   \seq_clear:N \l_tmpb_seq
6232   \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {

```

```

6233     \tl_if_empty:nF{ ##1 }{
6234         \stex_get_symbol:n { ##1 }
6235         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6236             \l_stex_get_symbol_uri_str
6237         }
6238     }
6239 }
6240 \ifvmode\noindent\fi
6241 \exp_args:Nnx
6242 \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpb_seq {,}}{
6243     \str_if_empty:NF \sassertiontype {
6244         \stex_annotate_invisible:nnn{typestrings}{\sassertiontype}{ }
6245     }
6246     #2
6247     \str_if_empty:NF \sassertionname {
6248         \stex_suppress_html:n{\stex_symdecl_do:nn{ }{\sassertionname}}
6249         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
6250         \stex_annotate_invisible:nnn{statementname}{\sassertionname}{ }
6251     }
6252 }
6253 }
6254 \endgroup
6255 \stex_smsmode_do:
6256 }

```

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

34.3 Examples

`sexample (env.)`

```

6257
6258 \keys_define:nn {stex / sexample }{
6259     type      .str_set_x:N = \exampletype,
6260     id        .str_set_x:N = \sexampleid,
6261     title     .tl_set:N    = \sexampletitle,
6262     name      .str_set_x:N = \sexamplename ,
6263     for       .clist_set:N = \l__stex_statements_sexample_for_clist,
6264 }
6265 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
6266     \str_clear:N \sexampletype
6267     \str_clear:N \sexampleid
6268     \str_clear:N \sexamplename
6269     \tl_clear:N \sexampletitle
6270     \clist_clear:N \l__stex_statements_sexample_for_clist
6271     \keys_set:nn { stex / sexample }{ #1 }
6272 }
6273
6274 \NewDocumentEnvironment{sexample}{0{}}{
6275     \__stex_statements_sexample_args:n{ #1 }
6276     \stex_reactivate_macro:N \premise
6277     \stex_reactivate_macro:N \conclusion
6278     \stex_if_smsmode:F {
6279         \seq_clear:N \l_tmpb_seq

```

```

6280 \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
6281   \tl_if_empty:NF{ ##1 }{
6282     \stex_get_symbol:n { ##1 }
6283     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6284       \l_stex_get_symbol_uri_str
6285     }
6286   }
6287 }
6288 \exp_args:Nnnx
6289 \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpb_seq {,}}
6290 \str_if_empty:NF \sexamplotype {
6291   \stex_annotate_invisible:nnn{typestrings}{\sexampletype}{\}
6292 }
6293 \str_if_empty:NF \sexamplename {
6294   \stex_annotate_invisible:nnn{statementname}{\sexamplename}{\}
6295 }
6296 \clist_set:No \l_tmpa_clist \sexamplotype
6297 \tl_clear:N \l_tmpa_tl
6298 \clist_map_inline:Nn \l_tmpa_clist {
6299   \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
6300     \tl_set:Nn \l_tmpa_tl {
6301       \stex_patch_counters:
6302       \use:c{__stex_statements_sexample_##1_start:}
6303       \stex_unpatch_counters:
6304     }
6305   }
6306 }
6307 \tl_if_empty:NTF \l_tmpa_tl {
6308   \__stex_statements_sexample_start:
6309 }{
6310   \l_tmpa_tl
6311 }
6312 }
6313 \str_if_empty:NF \sexampleid {
6314   \stex_ref_new_doc_target:n \sexampleid
6315 }
6316 \stex_smsmode_do:
6317 ){
6318   \str_if_empty:NF \sexamplename {
6319     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
6320   }
6321   \stex_if_smsmode:F {
6322     \clist_set:No \l_tmpa_clist \sexamplotype
6323     \tl_clear:N \l_tmpa_tl
6324     \clist_map_inline:Nn \l_tmpa_clist {
6325       \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
6326         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
6327       }
6328     }
6329     \tl_if_empty:NTF \l_tmpa_tl {
6330       \__stex_statements_sexample_end:
6331     }{
6332       \l_tmpa_tl
6333     }

```



```

6334 \end{stex_annotate_env}
6335 }
6336 }

```

`\stexpatchexample`

```

6337
6338 \cs_new_protected:Nn \__stex_statements_sexample_start: {
6339 \stex_par:\noindent\titllemph{Example~\tl_if_empty:NF \sexampletile {
6340 (\sexampletile)
6341 }~}
6342 }
6343 \cs_new_protected:Nn \__stex_statements_sexample_end: {\stex_par:\medskip}
6344
6345 \newcommand\stexpatchexample[3] [] {
6346 \str_set:Nx \l_tmpa_str{ #1 }
6347 \str_if_empty:NTF \l_tmpa_str {
6348 \tl_set:Nn \__stex_statements_sexample_start: { #2 }
6349 \tl_set:Nn \__stex_statements_sexample_end: { #3 }
6350 }{
6351 \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
6352 \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
6353 }
6354 }

```

(End definition for `\stexpatchexample`. This function is documented on page 55.)

`\inlineex` inline:

```

6355 \keys_define:nn {stex / inlineex }{
6356 type .str_set_x:N = \sexamplotype,
6357 id .str_set_x:N = \sexampleid,
6358 for .clist_set:N = \l__stex_statements_sexample_for_clist ,
6359 name .str_set_x:N = \sexamplename
6360 }
6361 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
6362 \str_clear:N \sexamplotype
6363 \str_clear:N \sexampleid
6364 \str_clear:N \sexamplename
6365 \clist_clear:N \l__stex_statements_sexample_for_clist
6366 \keys_set:nn { stex / inlineex }{ #1 }
6367 }
6368 \NewDocumentCommand \inlineex { 0{} m } {
6369 \begingroup
6370 \stex_reactivate_macro:N \premise
6371 \stex_reactivate_macro:N \conclusion
6372 \__stex_statements_inlineex_args:n{ #1 }
6373 \str_if_empty:NF \sexampleid {
6374 \stex_ref_new_doc_target:n \sexampleid
6375 }
6376 \stex_if_smsmode:TF{
6377 \str_if_empty:NF \sexamplename {
6378 \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sexamplename}}
6379 }
6380 }{
6381 \seq_clear:N \l_tmpb_seq

```

```

6382 \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
6383   \tl_if_empty:nF{ ##1 }{
6384     \stex_get_symbol:n { ##1 }
6385     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6386       \l_stex_get_symbol_uri_str
6387     }
6388   }
6389 }
6390 \ifvmode\noindent\fi
6391 \exp_args:Nnx
6392 \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpb_seq {,}}{
6393   \str_if_empty:NF \sexamplotype {
6394     \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{}
6395   }
6396   #2
6397   \str_if_empty:NF \sexamplename {
6398     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
6399     \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}
6400   }
6401 }
6402 }
6403 \endgroup
6404 \stex_smsmode_do:
6405 }

```

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

34.4 Logical Paragraphs

sparagraph (*env.*)

```

6406 \keys_define:nn { stex / sparagraph } {
6407   id      .str_set:N      = \sparagraphid ,
6408   title   .tl_set:N       = \l_stex_sparagraph_title_tl ,
6409   type    .str_set:N      = \sparagraphtype ,
6410   for     .clist_set:N    = \l__stex_statements_sparagraph_for_clist ,
6411   from    .tl_set:N       = \sparagraphfrom ,
6412   to      .tl_set:N       = \sparagraphto ,
6413   start   .tl_set:N       = \l_stex_sparagraph_start_tl ,
6414   name    .str_set:N      = \sparagraphname ,
6415   imports .tl_set:N       = \l__stex_statements_sparagraph_imports_tl
6416 }
6417
6418 \cs_new_protected:Nn \stex_sparagraph_args:n {
6419   \tl_clear:N \l_stex_sparagraph_title_tl
6420   \tl_clear:N \sparagraphfrom
6421   \tl_clear:N \sparagraphto
6422   \tl_clear:N \l_stex_sparagraph_start_tl
6423   \tl_clear:N \l__stex_statements_sparagraph_imports_tl
6424   \str_clear:N \sparagraphid
6425   \str_clear:N \sparagraphtype
6426   \clist_clear:N \l__stex_statements_sparagraph_for_clist
6427   \str_clear:N \sparagraphname
6428   \keys_set:nn { stex / sparagraph }{ #1 }

```

```

6429 }
6430 \newif\if@in@omtext\@in@omtextfalse
6431
6432 \NewDocumentEnvironment {sparagraph} { 0{} } {
6433   \stex_sparagraph_args:n { #1 }
6434   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
6435     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
6436   }{
6437     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
6438   }
6439   \@in@omtexttrue
6440   \stex_if_smsmode:F {
6441     \seq_clear:N \l_tmpb_seq
6442     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
6443       \tl_if_empty:NF{ ##1 }{
6444         \stex_get_symbol:n { ##1 }
6445         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6446           \l_stex_get_symbol_uri_str
6447         }
6448       }
6449     }
6450     \exp_args:Nnnx
6451     \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}
6452     \str_if_empty:NF \sparagraphtype {
6453       \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{ }
6454     }
6455     \str_if_empty:NF \sparagraphfrom {
6456       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{ }
6457     }
6458     \str_if_empty:NF \sparagraphto {
6459       \stex_annotate_invisible:nnn{to}{\sparagraphto}{ }
6460     }
6461     \str_if_empty:NF \sparagraphname {
6462       \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{ }
6463     }
6464     \clist_set:No \l_tmpa_clist \sparagraphtype
6465     \tl_clear:N \l_tmpa_tl
6466     \clist_map_inline:Nn \sparagraphtype {
6467       \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
6468         \tl_set:Nn \l_tmpa_tl {
6469           \stex_patch_counters:
6470           \use:c{__stex_statements_sparagraph_##1_start:}
6471           \stex_unpatch_counters:
6472         }
6473       }
6474     }
6475     \stex_csl_to_imports:No \usemodule \l__stex_statements_sparagraph_imports_tl
6476     \tl_if_empty:NTF \l_tmpa_tl {
6477       \__stex_statements_sparagraph_start:
6478     }{
6479       \l_tmpa_tl
6480     }
6481   }
6482   \clist_set:No \l_tmpa_clist \sparagraphtype

```

```

6483 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}
6484 {
6485   \stex_reactivate_macro:N \definiendum
6486   \stex_reactivate_macro:N \definame
6487   \stex_reactivate_macro:N \Definame
6488   \stex_reactivate_macro:N \premise
6489   \stex_reactivate_macro:N \definiens
6490 }
6491 \str_if_empty:NTF \sparagraphid {
6492   \str_if_empty:NTF \sparagraphname {
6493     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
6494       \stex_ref_new_doc_target:n {}
6495     }
6496   } {
6497     \stex_ref_new_doc_target:n {}
6498   }
6499 } {
6500   \stex_ref_new_doc_target:n \sparagraphid
6501 }
6502 \exp_args:NNx
6503 \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
6504   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
6505     \tl_if_empty:nF{ ##1 }{
6506       \stex_get_symbol:n { ##1 }
6507       \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
6508     }
6509   }
6510 }
6511 \stex_smsmode_do:
6512 \ignorespacesandpars
6513 }{
6514   \str_if_empty:NF \sparagraphname {
6515     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
6516     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
6517   }
6518   \stex_if_smsmode:F {
6519     \clist_set:No \l_tmpa_clist \sparagraphtype
6520     \tl_clear:N \l_tmpa_tl
6521     \clist_map_inline:Nn \l_tmpa_clist {
6522       \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
6523         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
6524       }
6525     }
6526     \tl_if_empty:NTF \l_tmpa_tl {
6527       \__stex_statements_sparagraph_end:
6528     }{
6529       \l_tmpa_tl
6530     }
6531     \end{stex_annotate_env}
6532   }
6533 }

```

\stexpatchparagraph

6534

```

6535 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
6536   \stex_par:\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
6537     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
6538       \titleemph{\l_stex_sparagraph_title_tl}:~
6539     }
6540   }{
6541     \titleemph{\l_stex_sparagraph_start_tl}~
6542   }
6543 }
6544 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\stex_par:\medskip}
6545
6546 \newcommand\stexpatchparagraph[3] [] {
6547   \str_set:Nx \l_tmpa_str{ #1 }
6548   \str_if_empty:NTF \l_tmpa_str {
6549     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
6550     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
6551   }{
6552     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
6553     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
6554   }
6555 }
6556
6557 \keys_define:nn { stex / inlinepara } {
6558   id      .str_set:N = \sparagraphid ,
6559   type    .str_set:N = \sparagraphtype ,
6560   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
6561   from    .tl_set:N   = \sparagraphfrom ,
6562   to      .tl_set:N   = \sparagraphto ,
6563   name    .str_set:N   = \sparagraphname
6564 }
6565 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
6566   \tl_clear:N \sparagraphfrom
6567   \tl_clear:N \sparagraphto
6568   \str_clear:N \sparagraphid
6569   \str_clear:N \sparagraphtype
6570   \clist_clear:N \l__stex_statements_sparagraph_for_clist
6571   \str_clear:N \sparagraphname
6572   \keys_set:nn { stex / inlinepara }{ #1 }
6573 }
6574 \NewDocumentCommand \inlinepara { 0{} m } {
6575   \begingroup
6576   \__stex_statements_inlinepara_args:n{ #1 }
6577   \clist_set:No \l_tmpa_clist \sparagraphtype
6578   \str_if_empty:NTF \sparagraphid {
6579     \str_if_empty:NTF \sparagraphname {
6580       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
6581         \stex_ref_new_doc_target:n {}
6582       }
6583     } {
6584       \stex_ref_new_doc_target:n {}
6585     }
6586   } {
6587     \stex_ref_new_doc_target:n \sparagraphid
6588   }

```

```

6589 \stex_if_smsmode:TF{
6590   \str_if_empty:NF \sparagraphname {
6591     \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
6592     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
6593   }
6594 }{
6595   \seq_clear:N \l_tmpb_seq
6596   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
6597     \tl_if_empty:NF{ ##1 }{
6598       \stex_get_symbol:n { ##1 }
6599       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6600         \l_stex_get_symbol_uri_str
6601       }
6602     }
6603   }
6604   \ifvmode\noindent\fi
6605   \exp_args:Nnx
6606   \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}{
6607     \str_if_empty:NF \sparagraphtype {
6608       \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
6609     }
6610     \str_if_empty:NF \sparagraphfrom {
6611       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
6612     }
6613     \str_if_empty:NF \sparagraphto {
6614       \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
6615     }
6616     \str_if_empty:NF \sparagraphname {
6617       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
6618       \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
6619       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
6620     }
6621     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
6622       \clist_map_inline:Nn \l_tmpb_seq {
6623         \stex_ref_new_sym_target:n {##1}
6624       }
6625     }
6626     #2
6627   }
6628 }
6629 \endgroup
6630 \stex_smsmode_do:
6631 }
6632

```

(End definition for `\stexpatchparagraph`. This function is documented on page 55.)

```

6633 </package>

```

Chapter 35

The Implementation

```
6634 <*package>
6635 <@@=stex_sproof>
6636
6637 %%%%%%%%%%    sproof.dtx    %%%%%%%%%%
6638
```

35.1 Proofs

We first define some keys for the proof environment.

```
6639 \keys_define:nn { stex / spf } {
6640   id          .str_set_x:N = \spfid,
6641   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,
6642   from        .tl_set:N    = \l__stex_sproof_spf_from_tl ,
6643   proofend    .tl_set:N    = \l__stex_sproof_spf_proofend_tl,
6644   type        .str_set_x:N = \spftype,
6645   title       .tl_set:N    = \spftitle,
6646   continues   .tl_set:N    = \l__stex_sproof_spf_continues_tl,
6647   functions   .tl_set:N    = \l__stex_sproof_spf_functions_tl,
6648   term        .tl_set:N    = \l__stex_sproof_spf_term_tl,
6649   method      .tl_set:N    = \l__stex_sproof_spf_method_tl,
6650   hide        .bool_set:N  = \l__stex_sproof_spf_hide_bool
6651 }
6652 \cs_new_protected:Nn \l__stex_sproof_spf_args:n {
6653   \str_clear:N \spfid
6654   \tl_clear:N \l__stex_sproof_spf_for_tl
6655   \tl_clear:N \l__stex_sproof_spf_from_tl
6656   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
6657   \str_clear:N \spftype
6658   \tl_clear:N \spftitle
6659   \tl_clear:N \l__stex_sproof_spf_continues_tl
6660   \tl_clear:N \l__stex_sproof_spf_term_tl
6661   \tl_clear:N \l__stex_sproof_spf_functions_tl
6662   \tl_clear:N \l__stex_sproof_spf_method_tl
6663   \bool_set_false:N \l__stex_sproof_spf_hide_bool
6664   \keys_set:nn { stex / spf }{ #1 }
6665 }
6666 \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`

```
6667 \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.

```
6668 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
6669 \cs_new_protected:Npn \sproofnumber {
6670   \int_set:Nn \l_tmpa_int {1}
6671   \bool_while_do:nn {
6672     \int_compare_p:nNn {
6673       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6674     } > 0
6675   }{
6676     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
6677     \int_incr:N \l_tmpa_int
6678   }
6679 }
6680 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
6681   \int_set:Nn \l_tmpa_int {1}
6682   \bool_while_do:nn {
6683     \int_compare_p:nNn {
6684       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6685     } > 0
6686   }{
6687     \int_incr:N \l_tmpa_int
6688   }
6689   \int_compare:nNnF \l_tmpa_int = 1 {
6690     \int_decr:N \l_tmpa_int
6691   }
6692   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
6693     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1
6694   }
6695 }
6696
6697 \cs_new_protected:Npn \__stex_sproof_add_counter: {
6698   \int_set:Nn \l_tmpa_int {1}
6699   \bool_while_do:nn {
6700     \int_compare_p:nNn {
6701       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6702     } > 0
6703   }{
6704     \int_incr:N \l_tmpa_int
6705   }
6706   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 1 }
6707 }
6708
```



```

6709 \cs_new_protected:Npn \__stex_sproof_remove_counter: {
6710   \int_set:Nn \l_tmpa_int {1}
6711   \bool_while_do:nn {
6712     \int_compare_p:nNn {
6713       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6714     } > 0
6715   }{
6716     \int_incr:N \l_tmpa_int
6717   }
6718   \int_decr:N \l_tmpa_int
6719   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 0 }
6720 }

```

\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

```

6721 \def\sproof@box{
6722   \ltx@ifpackageloaded{amssymb}{\square$}{
6723     \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
6724   }
6725 }
6726 \def\sproofend{
6727   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
6728     \hfil\hfill\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
6729   }
6730 }

```

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

spf@*kw

```

6731 \def\spf@proofsketch@kw{Proof~Sketch}
6732 \def\spf@proof@kw{Proof}
6733 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

6734 \AddToHook{begindocument}{
6735   \ltx@ifpackageloaded{babel}{
6736     \makeatletter
6737     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6738     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6739       \input{sproof-ngerman.ldf}
6740     }
6741     \clist_if_in:NnT \l_tmpa_clist {finnish}{
6742       \input{sproof-finnish.ldf}
6743     }
6744     \clist_if_in:NnT \l_tmpa_clist {french}{
6745       \input{sproof-french.ldf}
6746     }
6747     \clist_if_in:NnT \l_tmpa_clist {russian}{
6748       \input{sproof-russian.ldf}
6749     }
6750     \makeatother
6751   }{}
6752 }

```

spfsketch

```

6753 \newcommand\spfsketch[2] [] {
6754   \begin{group}
6755   \let \premise \stex_proof_premise:
6756   \stex_sproof_spf_args:n{#1}
6757   \stex_if_smsmode:TF {
6758     \str_if_empty:NF \spfid {
6759       \stex_ref_new_doc_target:n \spfid
6760     }
6761   }{
6762     \seq_clear:N \l_tmpa_seq
6763     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6764       \tl_if_empty:nF{ ##1 }{
6765         \stex_get_symbol:n { ##1 }
6766         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6767           \l_stex_get_symbol_uri_str
6768         }
6769       }
6770     }
6771     \exp_args:Nnx
6772     \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {},,}{
6773       \str_if_empty:NF \spftype {
6774         \stex_annotate_invisible:nnn{type}{\spftype}{
6775         }
6776       }
6777       \clist_set:Nn \l_tmpa_clist \spftype
6778       \tl_set:Nn \l_tmpa_tl {
6779         \titleemph{
6780           \tl_if_empty:NTF \spftitle {
6781             \spf@proofsketch@kw
6782           }{
6783             \spftitle
6784           }
6785         }
6786       }
6787       \clist_map_inline:Nn \l_tmpa_clist {
6788         \exp_args:Nn \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6789           \tl_clear:N \l_tmpa_tl
6790         }
6791       }
6792       \str_if_empty:NF \spfid {
6793         \stex_ref_new_doc_target:n \spfid
6794       }
6795       \l_tmpa_tl #2 \sproofend
6796     }
6797   }
6798   \endgroup
6799 }
6800

```

(End definition for *spfsketch*. This function is documented on page 54.)

```

\stex_sproof_maybe_comment:
\stex_sproof_maybe_comment_end:
\stex_sproof_start_comment:
6801 \bool_set_false:N \l__stex_sproof_in_spfblock_bool

```

```

6802
6803 \cs_new_protected:Nn \__stex_sproof_maybe_comment: {
6804   \bool_if:NF \l__stex_sproof_in_spfblock_bool {
6805     \par \setbox \l_tmpa_box \vbox \bgroup \everypar{\__stex_sproof_start_comment:}
6806   }
6807 }
6808 \cs_new_protected:Nn \__stex_sproof_maybe_comment_end: {
6809   \bool_if:NF \l__stex_sproof_in_spfblock_bool { \egroup }
6810 }
6811 \cs_new_protected:Nn \__stex_sproof_start_comment: {
6812   \csname @ @ par\endcsname\egroup\item[]\bgroup\stexcommentfont
6813 }
6814

```

(End definition for __stex_sproof_maybe_comment:, __stex_sproof_maybe_comment_end:, and __stex_sproof_start_comment:.)

\stexcommentfont

```

6815 \cs_new_protected:Npn \stexcommentfont {
6816   \small\itshape
6817 }

```

(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.

```

6818 \cs_new_protected:Nn \__stex_sproof_start_env:nnn {
6819   \seq_clear:N \l_tmpa_seq
6820   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6821     \tl_if_empty:NF{ ##1 }{
6822       \stex_get_symbol:n { ##1 }
6823       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6824         \l_stex_get_symbol_uri_str
6825       }
6826     }
6827   }
6828   \exp_args:Nnnx
6829   \begin{stex_annotate_env}{#1}{\seq_use:Nn \l_tmpa_seq {,}}
6830   \str_if_empty:NF \spftype {
6831     \stex_annotate_invisible:nnn{type}{\spftype}{}
6832   }
6833   #3 {\~\stex_annotate:nnn{spftitle}{}{#2}}
6834   \str_if_empty:NF \spfid {
6835     \stex_ref_new_doc_target:n \spfid
6836   }
6837   \begin{stex_annotate_env}{spfbody}{\bool_if:NTF \l__stex_sproof_spf_hide_bool {false}{true}
6838   \bool_if:NT \l__stex_sproof_spf_hide_bool{
6839     \stex_html_backend:F{\setbox\l_tmpa_box\vbox\bgroup}
6840   }
6841   \begin{list}{}{
6842     \setlength\topsep{0pt}
6843     \setlength\parsep{0pt}
6844     \setlength\rightmargin{0pt}

```

```

6845 } \_stex_sproof_maybe_comment:
6846 }
6847 }
6848 \cs_new_protected:Nn \_stex_sproof_end_env:n {
6849   \stex_if_smsmode:F{
6850     \_stex_sproof_maybe_comment_end:
6851     \end{list}
6852     \bool_if:NT \l__stex_sproof_spf_hide_bool{
6853       \stex_html_backend:F{\egroup}
6854     }
6855     \clist_set:No \l_tmpa_clist \spftype
6856     #1
6857     \end{stex_annotate_env}
6858     \end{stex_annotate_env}
6859   }
6860 }
6861 \NewDocumentEnvironment{sproof}{s O{} m}{
6862   \intarray_gzero:N \l__stex_sproof_counter_intarray
6863   \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
6864   \stex_reactivate_macro:N \yield
6865   \stex_reactivate_macro:N \eqstep
6866   \stex_reactivate_macro:N \assumption
6867   \stex_reactivate_macro:N \conclude
6868   \stex_reactivate_macro:N \spfstep
6869   \_stex_sproof_spf_args:n{#2}
6870   \stex_if_smsmode:TF {
6871     \str_if_empty:NF \spfid {
6872       \stex_ref_new_doc_target:n \spfid
6873     }
6874   }{
6875     \_stex_sproof_start_env:nnn{sproof}{#3}{
6876       \clist_set:No \l_tmpa_clist \spftype
6877       \tl_clear:N \l_tmpa_tl
6878       \clist_map_inline:Nn \l_tmpa_clist {
6879         \tl_if_exist:cT {\_stex_sproof_sproof_##1_start:}{
6880           \tl_set:Nn \l_tmpa_tl {\use:c{\_stex_sproof_sproof_##1_start:}}
6881         }
6882         \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6883           \tl_set:Nn \l_tmpa_tl {\use:n{}}
6884         }
6885       }
6886       \tl_if_empty:NTF \l_tmpa_tl {
6887         \_stex_sproof_sproof_start:
6888       }{
6889         \l_tmpa_tl
6890       }
6891     }
6892   }
6893   \stex_smsmode_do:
6894 }{\_stex_sproof_end_env:n{
6895   \tl_clear:N \l_tmpa_tl
6896   \clist_map_inline:Nn \l_tmpa_clist {
6897     \tl_if_exist:cT {\_stex_sproof_sproof_##1_end:}{
6898       \tl_set:Nn \l_tmpa_tl {\use:c{\_stex_sproof_sproof_##1_end:}}

```

```

6899     }
6900   }
6901   \tl_if_empty:NTF \l_tmpa_tl {
6902     \__stex_sproof_sproof_end:
6903   }{
6904     \l_tmpa_tl
6905   }
6906 }}
6907 \NewDocumentEnvironment{subproof}{s O{} m}{
6908   \__stex_sproof_spf_args:n{#2}
6909   \stex_if_smsmode:TF {
6910     \str_if_empty:NF \spfid {
6911       \stex_ref_new_doc_target:n \spfid
6912     }
6913   }{
6914     \__stex_sproof_start_env:nnn{subproof}{\item[\sproofnumber]\ignorespacesandpars #3}{}
6915   }
6916   \__stex_sproof_add_counter:
6917   \stex_smsmode_do:
6918 }{\__stex_sproof_remove_counter:\__stex_sproof_end_env:n{
6919   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6920     \__stex_sproof_inc_counter:
6921   }
6922   \aftergroup\__stex_sproof_maybe_comment:
6923 }
6924 \AddToHook{env/subproof/before}{\__stex_sproof_maybe_comment_end:}
6925
6926 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
6927   \par\noindent\titleemph{
6928     \tl_if_empty:NTF \spftype {
6929       \spf@proof@kw
6930     }{
6931       \spftype
6932     }
6933   }:
6934 }
6935 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
6936
6937 \newcommand\stexpatchproof[3] [] {
6938   \str_set:Nx \l_tmpa_str{ #1 }
6939   \str_if_empty:NTF \l_tmpa_str {
6940     \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
6941     \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
6942   }{
6943     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
6944     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
6945   }
6946 }

\pstep
\conclude
\assumption
\have
\eqstep

```

```

6947
6948 \keys_define:nn { stex / spfsteps } {
6949   id          .str_set_x:N = \spfstepid,
6950   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,

```

```

6951 type      .str_set_x:N = \spftype,
6952 title     .tl_set:N = \spftitle,
6953 method    .tl_set:N = \l__stex_sproof_spf_method_tl,
6954 term      .tl_set:N = \l__stex_sproof_spf_term_tl
6955 }
6956 \cs_new_protected:Nn \__stex_sproof_spfstep_args:n {
6957 \str_clear:N \spfstepid
6958 \clist_clear:N \l__stex_sproof_spf_for_clist
6959 \str_clear:N \spftype
6960 \tl_clear:N \l__stex_sproof_spf_method_tl
6961 \tl_clear:N \l__stex_sproof_spf_term_tl
6962 %\bool_set_false:N \l__stex_sproof_inc_counter_bool
6963 \keys_set:nn { stex / spfsteps }{ #1 }
6964 }
6965
6966 \cs_new_protected:Nn \__stex_sproof_make_step_macro:Nnnnn {
6967 \NewDocumentCommand #1 {s O{} +m} {
6968 \__stex_sproof_maybe_comment_end:
6969
6970 \__stex_sproof_spfstep_args:n{##2}
6971 \stex_annotate:nnn{spfstep}{#2}{
6972 \tl_if_empty:NF \l__stex_sproof_spf_term_tl {
6973 \stex_annotate_invisible:nnn{spfyield}{}{\l__stex_sproof_spf_term_tl$}
6974 }
6975 \bool_if:NTF \l__stex_sproof_in_spfblock_bool {
6976 #4
6977 }{
6978 \item[\IfBooleanTF ##1 {}{#3}]
6979 }
6980 \ignorespacesandpars ##3
6981 }
6982 \bool_if:NF \l__stex_sproof_in_spfblock_bool { \IfBooleanTF ##1 {}{ #5 } }
6983 \__stex_sproof_maybe_comment:
6984 }
6985 \stex_deactivate_macro:Nn #1 {sproof~environments}
6986 }
6987
6988 \__stex_sproof_make_step_macro:Nnnnn \assumption {assumption} \sproofnumber {} \__stex_sproof
6989 \__stex_sproof_make_step_macro:Nnnnn \conclude {conclusion} {\$ \Rightarrow$} {} {}
6990 \__stex_sproof_make_step_macro:Nnnnn \spfstep {} \sproofnumber {} \__stex_sproof_inc_counter
6991
6992 \NewDocumentCommand \eqstep {s m}{
6993 \__stex_sproof_maybe_comment_end:
6994 \bool_if:NTF \l__stex_sproof_in_spfblock_bool {
6995 $=$
6996 }{
6997 \item[$=$]
6998 }
6999 $\stex_annotate:nnn{spfstep}{eq}{ #2 }$
7000 \__stex_sproof_maybe_comment:
7001 }
7002 \stex_deactivate_macro:Nn \eqstep {sproof~environments}
7003
7004 \NewDocumentCommand \yield {+m}{

```

```

7005 \stex_annotate:nnn{spfyield}{\}{ #1 }
7006 }
7007 \stex_deactivate_macro:Nn \yield {sproof~environments}
7008
7009 \NewDocumentEnvironment{spfblock}{\}{\{
7010 \item[]
7011 \bool_set_true:N \l__stex_sproof_in_spfblock_bool
7012 }\{
7013 \aftergroup\__stex_sproof_maybe_comment:
7014 }
7015 \AddToHook{env/spfblock/before}{\__stex_sproof_maybe_comment_end:}
7016

```

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

`\spfidea`

```

7017 \NewDocumentCommand\spfidea{0\} +m\{
7018 \__stex_sproof_spf_args:n{#1}
7019 \titleemph{
7020 \tl_if_empty:NTF \spftype {Proof-Idea}{
7021 \spftype
7022 }~#2
7023 }~#2
7024 \sproofend
7025 }

```

(End definition for `\spfidea`. This function is documented on page 54.)

```

7026 \newcommand\spfjust[1]{
7027 #1
7028 }
7029 \</package>

```

Some auxiliary code, and clean up to be executed at the end of the package.

Chapter 36

STEX -Others Implementation

```
7030 <*package>
7031
7032 %%%%%%%%%% others.dtx %%%%%%%%%%
7033
7034 <@@=stex_others>
    Warnings and error messages
7035 % None

\MSC Math subject classifier

7036 \NewDocumentCommand \MSC {m} {
7037 % TODO
7038 }

(End definition for \MSC. This function is documented on page ??.)
    Patching tikzinput, if loaded

7039 \@ifpackageloaded{tikzinput}{
7040 \RequirePackage{stex-tikzinput}
7041 }{}
7042
7043 \bool_if:NT \c_stex_persist_mode_bool {
7044 \let__stex_notation_restore_notation_old:nnnnn
7045 \__stex_notation_restore_notation:nnnnn
7046 \def__stex_notation_restore_notation_new:nnnnn#1#2#3#4#5{
7047 \__stex_notation_restore_notation_old:nnnnn{#1}{#2}{#3}{#4}{#5}
7048 \ExplSyntaxOn
7049 }
7050 \def__stex_notation_restore_notation:nnnnn{
7051 \ExplSyntaxOff
7052 \catcode'\sim10
7053 \__stex_notation_restore_notation_new:nnnnn
7054 }
7055 \input{\jobname.sms}
7056 \let__stex_notation_restore_notation:nnnnn
7057 \__stex_notation_restore_notation_old:nnnnn
7058 \prop_if_exist:NT\c_stex_mathhub_main_manifest_prop{
```



```

7059     \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
7060     \l_tmpa_str
7061     \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str_manifest_prop }
7062     \c_stex_mathhub_main_manifest_prop
7063     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
7064   }
7065 }
7066
7067 \stex_get_document_uri:
7068 </package>

```

Chapter 37

STEX -Metatheory Implementation

```
7069 <*package>
7070 <@@=stex_modules>
7071
7072 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
7073
7074 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX/meta}
7075 \begingroup
7076 \stex_module_setup:nn{
7077   ns=\c_stex_metatheory_ns_str,
7078   meta=NONE
7079 }{Metatheory}
7080 \stex_reactivate_macro:N \symdecl
7081 \stex_reactivate_macro:N \notation
7082 \stex_reactivate_macro:N \symdef
7083 \ExplSyntaxOff
7084 \csname stex_suppress_html:n\endcsname{
7085   % is-a (a:A, a \in A, a is an A, etc.)
7086   \symdecl{isa}[args=ai]
7087   \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
7088   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
7089   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
7090
7091   % bind (\forall, \Pi, \lambda etc.)
7092   \symdecl{bind}[args=Bi,assoc=pre]
7093   \notation{bind}[depfun,prec=nobrackets,op={(\cdot)\;\to\;\cdot}]{\comp( #1 \comp{}\;\to\;)}
7094   \notation{bind}[forall]{\comp\forall #1.\;#2}{##1 \comp, ##2}
7095   \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
7096
7097   % implicit bind
7098   \symdecl{implicitbind}[args=Bi,assoc=pre]
7099   \notation{implicitbind}[braces,prec=nobrackets,op={(\cdot\_I\;\cdot)}]{\comp\{ #1 \comp{
7100   \notation{implicitbind}[depfun,prec=nobrackets]{\comp( #1 \comp{}\;\to\_I\; } #2}{##1 \comp,
7101   \notation{implicitbind}[Pi]{\comp\prod^I_{#1}#2}{##1\comp,##2}
7102
7103   % dummy variable
```

```

7104 \symdecl{dummyvar}
7105 \notation{dummyvar}[underscore]{\comp\_}
7106 \notation{dummyvar}[dot]{\comp\cdot}
7107 \notation{dummyvar}[dash]{\comp{\rm --}}
7108
7109 %fromto (function space, Hom-set, implication etc.)
7110 \symdecl{fromto}[args=ai]
7111 \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
7112 \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
7113
7114 % mapto (lambda etc.)
7115 \symdecl{mapto}[args=Bi]
7116 \notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
7117 \notation{mapto}[lambda]{\comp\lambda #1 \comp.; #2}{#1 \comp, #2}
7118 \notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.; #2}{#1 \comp, #2}
7119
7120 % function/operator application
7121 \symdecl{apply}[args=ia]
7122 \notation{apply}[prec=0;0x\infpres,parens,op=\cdot(\cdot)]{#1 \comp( #2 \comp)}{##1 \comp,
7123 \notation{apply}[prec=0;0x\infpres,lambda]{#1 \; #2 }{##1 \; ; ##2}
7124
7125 % collection of propositions/booleans/truth values
7126 \symdecl{prop}[name=proposition]
7127 \notation{prop}[prop]{\comp{\rm prop}}
7128 \notation{prop}[BOOL]{\comp{\rm BOOL}}
7129
7130 \symdecl{judgmentholds}[args=1]
7131 \notation{judgmentholds}[vdash,op=\vdash]{\comp\vdash\; #1}
7132
7133 % sequences
7134 \symdecl{seqtype}[args=1]
7135 \notation{seqtype}[kleene]{#1^{\comp\ast}}
7136
7137 \symdecl{seqexpr}[args=a]
7138 \notation{seqexpr}[angle,prec=nobrackets]{\comp\langle #1\comp\rangle}{##1\comp,##2}
7139
7140 \symdef{seqmap}[args=abi,setlike]{\comp\{#3 \comp| #2\comp\in \dobrackets{#1} \comp\}}{##1
7141 \symdef{seqprepend}[args=ia]{#1 \comp{::} #2}{##1 \comp, ##2}
7142 \symdef{seqappend}[args=ai]{#1 \comp{::} #2}{##1 \comp, ##2}
7143 \symdef{seqfoldleft}[args=iabbi]{ \comp{foldl}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\comp}
7144 \symdef{seqfoldright}[args=iabbi,op=foldr]{ \comp{foldr}\dobrackets{#1,#2}\dobrackets{#3\comp
7145 \symdef{seqhead}[args=a]{\comp{head}\dobrackets{#1}}{##1 \comp, ##2}
7146 \symdef{seqtail}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
7147 \symdef{seqlast}[args=a]{\comp{last}\dobrackets{#1}}{##1 \comp, ##2}
7148 \symdef{seqinit}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
7149
7150 \symdef{sequence-index}[args=2,li,prec=nobrackets]{\{#1\}_{#2}}
7151 \notation{sequence-index}[ui,prec=nobrackets]{\{#1\}^{\#2}}
7152
7153 \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{,\ellipses}}{##1\comp,##2}
7154 \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{,\ellipses,}#2}{##1\comp,##2}
7155 \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{,\ellipses,}#2\comp{,\ellipses,}#
7156
7157 % nat literals

```

```

7158 \symdef{natliteral}{\comp{\mathtt{Ord}}}
7159
7160 % letin (''let'', local definitions, variable substitution)
7161 \symdecl{letin}[args=bii]
7162 \notation{letin}{let}{\comp{\rm let}}\;#1\comp{=}\#2\;\comp{\rm in}}\;#3}
7163 \notation{letin}{subst}{#3 \comp[ #1 \comp/ #2 \comp]}
7164 \notation{letin}{frac}{#3 \comp[ \frac{#2}{#1} \comp]}
7165
7166 % structures
7167 \symdecl*{module-type}[args=1]
7168 \notation{module-type}{\comp{\mathtt{MOD}}}\;#1}
7169 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
7170 \notation{mathstruct}{angle,prec=nobrackets}{\comp\angle #1 \comp\rangle}{##1 \comp, ##2}
7171
7172 % objects
7173 \symdecl{object}
7174 \notation{object}{\comp{\mathtt{OBJECT}}}
7175
7176 }
7177
7178 % The following are abbreviations in the sTeX corpus that are left over from earlier
7179 % developments. They will eventually be phased out.
7180
7181 \ExplSyntaxOn
7182 \stex_add_to_current_module:n{
7183   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
7184   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
7185   \def\livar{\csname sequence-index\endcsname[li]}
7186   \def\uivar{\csname sequence-index\endcsname[ui]}
7187   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
7188   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
7189 }
7190 \__stex_modules_end_module:
7191 \endgroup
7192
7193
7194 \str_set:Nn \l_stex_metatheory_str {http://mathhub.info/sTeX/meta?Metatheory}
7195
7196 \NewDocumentCommand \setmetatheory {0{} m}{
7197   \stex_import_module_uri:nn { #1 } { #2 }
7198   \stex_import_require_module:nnnn
7199   { \l_stex_import_ns_str } { \l_stex_import_archive_str }
7200   { \l_stex_import_path_str } { \l_stex_import_name_str }
7201   \str_set:Nx \l_stex_metatheory_str { \l_stex_import_ns_str ? \l_stex_import_name_str }
7202   \stex_smsmode_do:
7203 }
7204
7205 </package>

```

Chapter 38

Tikzinput Implementation

```
7206 <@@=tikzinput>
7207 <*package>
7208
7209 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
7210
7211 \ProvidesExplPackage{tikzinput}{2022/09/14}{3.2.0}{tikzinput package}
7212 \RequirePackage{l3keys2e}
7213
7214 \keys_define:nn { tikzinput } {
7215   image .bool_set:N = \c_tikzinput_image_bool,
7216   image .default:n = false ,
7217   unknown .code:n = {}
7218 }
7219
7220 \ProcessKeysOptions { tikzinput }
7221
7222 \bool_if:NTF \c_tikzinput_image_bool {
7223   \RequirePackage{graphicx}
7224
7225   \providecommand\usetikzlibrary[]{}
7226   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
7227 }{
7228   \RequirePackage{tikz}
7229   \RequirePackage{standalone}
7230
7231   \newcommand \tikzinput [2] [] {
7232     \setkeys{Gin}{#1}
7233     \ifx \Gin@ewidth \Gin@exclamation
7234       \ifx \Gin@eheight \Gin@exclamation
7235         \input { #2 }
7236       \else
7237         \resizebox{!}{ \Gin@eheight }{
7238           \input { #2 }
7239         }
7240       \fi
7241     \else
7242       \ifx \Gin@eheight \Gin@exclamation
7243         \resizebox{ \Gin@ewidth }{!}{
```

```

7244         \input { #2 }
7245     }
7246     \else
7247         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
7248             \input { #2 }
7249         }
7250     \fi
7251 \fi
7252 }
7253 }
7254
7255 \newcommand \ctikzinput [2] [] {
7256     \begin{center}
7257         \tikzinput [#1] {#2}
7258     \end{center}
7259 }
7260
7261 \@ifpackageloaded{stex}{
7262     \RequirePackage{stex-tikzinput}
7263 }{}
7264
7265 </package>
7266 <*stex>
7267 \ProvidesExplPackage{stex-tikzinput}{2022/09/14}{3.2.0}{stex-tikzinput}
7268 \RequirePackage{stex}
7269 \RequirePackage{tikzinput}
7270
7271 \newcommand\mhtikzinput[2] [] {%
7272     \def\Gin@mhrepos{ }\setkeys{Gin}{#1}%
7273     \stex_in_repository:nn\Gin@mhrepos{
7274         \tikzinput[#1]{\mhp@hpath{##1}{#2}}
7275     }
7276 }
7277 \newcommand\cmhtikzinput[2] [] {\begin{center}\mhtikzinput[#1]{#2}\end{center}}
7278
7279 \cs_new_protected:Nn \__tikzinput_usetikzlibrary:nn {
7280     \pgfkeys@spdef\pgf@temp{#1}
7281     \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
7282     \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgf@temp
7283     \expandafter\edef\csname tikz@library@#1@atcode\endcsname{\the\catcode'\@}
7284     \expandafter\edef\csname tikz@library@#1@barcode\endcsname{\the\catcode'\|}
7285     \expandafter\edef\csname tikz@library@#1@dollarcode\endcsname{\the\catcode'\$}
7286     \catcode'\@=11
7287     \catcode'\|=12
7288     \catcode'\$=3
7289     \pgfutil@InputIfFileExists{#2}{-}{-}
7290     \catcode'\@=\csname tikz@library@#1@atcode\endcsname
7291     \catcode'\|=\csname tikz@library@#1@barcode\endcsname
7292     \catcode'\$=\csname tikz@library@#1@dollarcode\endcsname
7293 }
7294
7295
7296 \newcommand\libusetikzlibrary[1]{

```

```

7297 \prop_if_exist:NF \l_stex_current_repository_prop {
7298   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
7299 }
7300 \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
7301   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
7302 }
7303 \seq_clear:N \l__tikzinput_libinput_files_seq
7304 \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
7305 \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
7306
7307 \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
7308   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / tikzlibrary
7309   \IfFileExists{ \l_tmpa_str }{
7310     \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
7311   }{}
7312   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
7313   \seq_put_right:No \l_tmpa_seq \l_tmpa_str
7314 }
7315
7316 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / tikzlibrary #1 .code.t
7317 \IfFileExists{ \l_tmpa_str }{
7318   \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
7319 }{}
7320
7321 \seq_if_empty:NTF \l__tikzinput_libinput_files_seq {
7322   \msg_error:nxxx{stex}{error/nofile}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .code.t
7323 }{
7324   \int_compare:nNnTF {\seq_count:N \l__tikzinput_libinput_files_seq} = 1 {
7325     \seq_map_inline:Nn \l__tikzinput_libinput_files_seq {
7326       \__tikzinput_usetikzlibrary:nn{#1}{ ##1 }
7327     }
7328   }{
7329     \msg_error:nxxx{stex}{error/twofiles}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .co
7330   }
7331 }
7332 }
7333 </stex>

```

Chapter 39

document-structure.sty Implementation

```
7334 \*package>
7335 \@@=document_structure>
7336 \ProvidesExplPackage{document-structure}{2022/09/14}{3.2.0}{Modular Document Structure}
7337 \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).

```
7338
7339 \keys_define:nn{ document-structure }{
7340   class      .str_set_x:N = \c_document_structure_class_str,
7341   topsect    .str_set_x:N = \c_document_structure_topsect_str,
7342   unknown    .code:n      = {
7343     \PassOptionsToClass{\CurrentOption}{stex}
7344     \PassOptionsToClass{\CurrentOption}{tikzinput}
7345   }
7346   % showignores .bool_set:N = \c_document_structure_showignores_bool,
7347 }
7348 \ProcessKeysOptions{ document-structure }
7349 \str_if_empty:NT \c_document_structure_class_str {
7350   \str_set:Nn \c_document_structure_class_str {article}
7351 }
7352 \str_if_empty:NT \c_document_structure_topsect_str {
7353   \str_set:Nn \c_document_structure_topsect_str {section}
7354 }
```

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

```
7355 \RequirePackage{xspace}
7356 \RequirePackage{comment}
7357 \RequirePackage{stex}
7358 \AddToHook{begindocument}{
```



```

7359 \ltx@ifpackageloaded{babel}{
7360     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7361     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
7362         \makeatletter\input{document-structure-ngerman.ldf}\makeatother
7363     }
7364 }{}
7365 }

```

`\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.

```

7366 \int_new:N \l_document_structure_section_level_int
7367 \str_case:NnF \c_document_structure_topsect_str {
7368     {part}}{
7369     \int_set:Nn \l_document_structure_section_level_int {0}
7370 }
7371 {chapter}}{
7372     \int_set:Nn \l_document_structure_section_level_int {1}
7373 }
7374 }{
7375     \str_case:NnF \c_document_structure_class_str {
7376         {book}}{
7377             \int_set:Nn \l_document_structure_section_level_int {0}
7378         }
7379         {report}}{
7380             \int_set:Nn \l_document_structure_section_level_int {0}
7381         }
7382     }{
7383         \int_set:Nn \l_document_structure_section_level_int {2}
7384     }
7385 }

```

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

```

7386 \def\current@section@level{document}%
7387 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
7388 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%

```

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

`\skipfragment`

```

7389 \cs_new_protected:Npn \skipfragment {

```

⁹EdNOTE: MK: we may have to experiment with the more powerful uppercasing macro from `mfirstuc.sty` once we internationalize.

```

7390 \ifcase\l_document_structure_section_level_int
7391 \or\stepcounter{part}
7392 \or\stepcounter{chapter}
7393 \or\stepcounter{section}
7394 \or\stepcounter{subsection}
7395 \or\stepcounter{subsubsection}
7396 \or\stepcounter{paragraph}
7397 \or\stepcounter{subparagraph}
7398 \fi
7399 }

```

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

`blindfragment (env.)`

```

7400 \newcommand\at@begin@blindsfragment[1]{
7401 \newenvironment{blindfragment}
7402 {
7403 \int_incr:N\l_document_structure_section_level_int
7404 \at@begin@blindsfragment\l_document_structure_section_level_int
7405 }{}

```

`\sfragment@nonum` convenience macro: `\sfragment@nonum{<level>}{<title>}` makes an unnumbered sectioning with title `<title>` at level `<level>`.

```

7406 \newcommand\sfragment@nonum[2]{
7407 \ifx\hyper@anchor\@undefined\else\phantomsection\fi
7408 \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
7409 }

```

(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 `rdfmata` package has been loaded. In the end we call `\sref@label@id` to enable crossreferencing.

```

7410 \newcommand\sfragment@num[2]{
7411 \tl_if_empty:NTF \l__document_structure_sfragment_short_tl {
7412 \@nameuse{#1}{#2}
7413 }{
7414 \cs_if_exist:NTF\rdfmata@sectioning{
7415 \@nameuse{rdfmata@#1@old}[\l__document_structure_sfragment_short_tl]{#2}
7416 }{
7417 \@nameuse{#1}[\l__document_structure_sfragment_short_tl]{#2}
7418 }
7419 }
7420 %\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\sfragment@id
7421 }

```

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

`sfragment (env.)`

```

7422 \keys_define:nn { document-structure / sfragment }{
7423 id .str_set_x:N = \l__document_structure_sfragment_id_str,
7424 date .str_set_x:N = \l__document_structure_sfragment_date_str,

```

```

7425 creators      .clist_set:N = \l__document_structure_sfragment_creators_clist,
7426 contributors  .clist_set:N = \l__document_structure_sfragment_contributors_clist,
7427 srccite        .tl_set:N     = \l__document_structure_sfragment_srccite_tl,
7428 type          .tl_set:N     = \l__document_structure_sfragment_type_tl,
7429 short         .tl_set:N     = \l__document_structure_sfragment_short_tl,
7430 intro         .tl_set:N     = \l__document_structure_sfragment_intro_tl,
7431 imports       .tl_set:N     = \l__document_structure_sfragment_imports_tl,
7432 loadmodules   .bool_set:N   = \l__document_structure_sfragment_loadmodules_bool
7433 }
7434 \cs_new_protected:Nn \l__document_structure_sfragment_args:n {
7435   \str_clear:N \l__document_structure_sfragment_id_str
7436   \str_clear:N \l__document_structure_sfragment_date_str
7437   \clist_clear:N \l__document_structure_sfragment_creators_clist
7438   \clist_clear:N \l__document_structure_sfragment_contributors_clist
7439   \tl_clear:N \l__document_structure_sfragment_srccite_tl
7440   \tl_clear:N \l__document_structure_sfragment_type_tl
7441   \tl_clear:N \l__document_structure_sfragment_short_tl
7442   \tl_clear:N \l__document_structure_sfragment_imports_tl
7443   \tl_clear:N \l__document_structure_sfragment_intro_tl
7444   \bool_set_false:N \l__document_structure_sfragment_loadmodules_bool
7445   \keys_set:nn { document-structure / sfragment } { #1 }
7446 }

```

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.

```

7447 \newif\if@mainmatter\@mainmattertrue
7448 \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.

```

7449 \keys_define:nn { document-structure / sectioning }{
7450   name      .str_set_x:N = \l__document_structure_sect_name_str  ,
7451   ref       .str_set_x:N = \l__document_structure_sect_ref_str   ,
7452   clear     .bool_set:N  = \l__document_structure_sect_clear_bool ,
7453   clear     .default:n   = {true}                                ,
7454   num       .bool_set:N  = \l__document_structure_sect_num_bool  ,
7455   num       .default:n   = {true}
7456 }
7457 \cs_new_protected:Nn \l__document_structure_sect_args:n {
7458   \str_clear:N \l__document_structure_sect_name_str
7459   \str_clear:N \l__document_structure_sect_ref_str
7460   \bool_set_false:N \l__document_structure_sect_clear_bool
7461   \bool_set_false:N \l__document_structure_sect_num_bool
7462   \keys_set:nn { document-structure / sectioning } { #1 }
7463 }
7464 \newcommand\omdoc@sectioning[3][]{
7465   \l__document_structure_sect_args:n {#1 }
7466   \let\omdoc@sect@name\l__document_structure_sect_name_str
7467   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
7468   \if@mainmatter% numbering not overridden by frontmatter, etc.
7469     \bool_if:NTF \l__document_structure_sect_num_bool {
7470       \sfragment@num{#2}{#3}
7471     }{

```

```

7472     \sfragment@nonum{#2}{#3}
7473   }
7474   \def\current@section@level{\omdoc@sect@name}
7475   \else
7476     \sfragment@nonum{#2}{#3}
7477   \fi
7478 }% 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.

```

7479 \newcommand\sfragment@redefine@addtocontents[1]{%
7480 %\edef\__document_structureimport{#1}%
7481 %\@for\@I:=\__document_structureimport\do{%
7482 %\edef\@path{\csname module@\@I @path\endcsname}%
7483 %\@ifundefined{tf@toc}\relax%
7484 %   {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
7485 %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
7486 %\def\addcontentsline##1##2##3{%
7487 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
7488 %\else% hyperref.sty not loaded
7489 %\def\addcontentsline##1##2##3{%
7490 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
7491 %\fi
7492 }% 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.

```

7493 \newenvironment{sfragment}[2][ ]% keys, title
7494 {
7495   \__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.

```

7496   \stex_csl_to_imports:No \usemodule \l__document_structure_sfragment_imports_tl
7497
7498   \bool_if:NT \l__document_structure_sfragment_loadmodules_bool {
7499     \sfragment@redefine@addtocontents{
7500       %\@ifundefined{module@id}\used@modules%
7501       %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}
7502     }
7503   }

```

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

```

7504
7505   \stex_document_title:n { #2 }
7506
7507   \stex_patch_counters:
7508   \int_incr:N\l__document_structure_section_level_int
7509   \ifcase\l__document_structure_section_level_int
7510     \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
7511     \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
7512     \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}

```

```

7513 \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
7514 \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
7515 \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#1}
7516 \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#1}
7517 \fi
7518 \at@begin@sfragment[#1]\l__document_structure_section_level_int{#2}
7519 \str_if_empty:NF \l__document_structure_sfragment_id_str {
7520   \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
7521 }
7522 \stex_unpatch_counters:
7523 }% for customization
7524 {}

```

and finally, we localize the sections

```

7525 \newcommand\omdoc@part@kw{Part}
7526 \newcommand\omdoc@chapter@kw{Chapter}
7527 \newcommand\omdoc@section@kw{Section}
7528 \newcommand\omdoc@subsection@kw{Subsection}
7529 \newcommand\omdoc@subsubsection@kw{Subsubsection}
7530 \newcommand\omdoc@paragraph@kw{paragraph}
7531 \newcommand\omdoc@subparagraph@kw{subparagraph}

```

39.3 Front and Backmatter

Index markup is provided by the `omtext` package [[Kohlhase:smmf:git](#)], so in the `document-structure` package we only need to supply the corresponding `\printindex` command, if it is not already defined

`\printindex`

```

7532 \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).

```

7533 \cs_if_exist:NTF\frontmatter{
7534   \let\__document_structure_orig_frontmatter\frontmatter
7535   \let\frontmatter\relax
7536 }{
7537   \tl_set:Nn\__document_structure_orig_frontmatter{
7538     \clearpage
7539     \@mainmatterfalse
7540     \pagenumbering{roman}
7541   }
7542 }
7543 \cs_if_exist:NTF\backmatter{
7544   \let\__document_structure_orig_backmatter\backmatter
7545   \let\backmatter\relax
7546 }{
7547   \tl_set:Nn\__document_structure_orig_backmatter{
7548     \clearpage
7549     \@mainmatterfalse

```

```

7550     \pagenumbering{roman}
7551   }
7552 }

```

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.

```

7553 \newenvironment{frontmatter}{
7554   \_document\_structure\_orig\_frontmatter
7555 }{
7556   \cs\_if\_exist:NTF\mainmatter{
7557     \mainmatter
7558   }{
7559     \clearpage
7560     \@mainmattertrue
7561     \pagenumbering{arabic}
7562   }
7563 }

```

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

```

7564 \newenvironment{backmatter}{
7565   \_document\_structure\_orig\_backmatter
7566 }{
7567   \cs\_if\_exist:NTF\mainmatter{
7568     \mainmatter
7569   }{
7570     \clearpage
7571     \@mainmattertrue
7572     \pagenumbering{arabic}
7573   }
7574 }

```

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

```

7575 \@mainmattertrue\pagenumbering{arabic}

```

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

```

7576 \def \c\_document\_structure\_document\_str{document}
7577 \newcommand\afterprematurestop{}
7578 \def\prematurestop@endsfragment{
7579   \unless\ifx\@currenvir\c\_document\_structure\_document\_str
7580     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
7581       \expandafter\prematurestop@endsfragment
7582     }
7583 }
7584 \providecommand\prematurestop{
7585   \message{Stopping~sTeX~processing~prematurely}
7586   \prematurestop@endsfragment
7587   \afterprematurestop
7588   \end{document}
7589 }

```

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

39.4 Global Variables

\setSGvar set a global variable

```
7590 \RequirePackage{etoolbox}
7591 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}
```

(End definition for \setSGvar. This function is documented on page 62.)

\useSGvar use a global variable

```
7592 \newrobustcmd\useSGvar[1]{%
7593   \@ifundefined{sTeX@Gvar@#1}
7594   {\PackageError{document-structure}
7595    {The sTeX Global variable #1 is undefined}
7596    {set it with \protect\setSGvar}}
7597   \@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.

```
7598 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
7599   \@ifundefined{sTeX@Gvar@#1}
7600   {\PackageError{document-structure}
7601    {The sTeX Global variable #1 is undefined}
7602    {set it with \protect\setSGvar}}
7603   {\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.

```
7604 \*cls)
7605 \@@=notesslides)
7606 \ProvidesExplClass{notesslides}{2022/09/14}{3.2.0}{notesslides Class}
7607 \RequirePackage{13keys2e}
7608
7609 \keys_define:nn{notesslides / cls}{
7610   class .str_set_x:N = \c__notesslides_class_str,
7611   notes .bool_set:N = \c__notesslides_notes_bool ,
7612   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
7613   docopt .str_set_x:N = \c__notesslides_docopt_str,
7614   unknown .code:n = {
7615     \PassOptionsToPackage{\CurrentOption}{document-structure}
7616     \PassOptionsToClass{\CurrentOption}{beamer}
7617     \PassOptionsToPackage{\CurrentOption}{notesslides}
7618     \PassOptionsToPackage{\CurrentOption}{stex}
7619   }
7620 }
7621 \ProcessKeysOptions{ notesslides / cls }
7622
7623 \str_if_empty:NF \c__notesslides_class_str {
7624   \PassOptionsToPackage{class=\c__notesslides_class_str}{document-structure}
7625 }
7626
7627 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{book}{
7628   \PassOptionsToPackage{defaultttopsect=part}{notesslides}
7629 }
7630 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{report}{
7631   \PassOptionsToPackage{defaultttopsect=part}{notesslides}
7632 }
7633
7634 \RequirePackage{stex}
```



```

7635 \stex_html_backend:T {
7636   \bool_set_true:N\c__notesslides_notes_bool
7637 }
7638
7639 \bool_if:NTF \c__notesslides_notes_bool {
7640   \PassOptionsToPackage{notes=true}{notesslides}
7641   \message{notesslides.cls:~Formatting~course~materials~in~notes~mode}
7642 }{
7643   \PassOptionsToPackage{notes=false}{notesslides}
7644   \message{notesslides.cls:~Formatting~course~materials~in~slides~mode}
7645 }
7646 </cls>

```

now we do the same for the notesslides package.

```

7647 <*package>
7648 \ProvidesExplPackage{notesslides}{2022/09/14}{3.2.0}{notesslides Package}
7649 \RequirePackage{l3keys2e}
7650
7651 \keys_define:nn{notesslides / pkg}{
7652   topsect          .str_set_x:N = \c__notesslides_topsect_str,
7653   defaulttopsect   .str_set_x:N = \c__notesslides_defaulttopsec_str,
7654   notes            .bool_set:N = \c__notesslides_notes_bool ,
7655   slides           .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
7656   sectocframes     .bool_set:N = \c__notesslides_sectocframes_bool ,
7657   frameimages      .bool_set:N = \c__notesslides_frameimages_bool ,
7658   fiboxed          .bool_set:N = \c__notesslides_fiboxed_bool ,
7659   nopproblems      .bool_set:N = \c__notesslides_nopproblems_bool,
7660   unknown          .code:n      = {
7661     \PassOptionsToClass{\CurrentOption}{stex}
7662     \PassOptionsToClass{\CurrentOption}{tikzinput}
7663   }
7664 }
7665 \ProcessKeysOptions{ notesslides / pkg }
7666
7667 \RequirePackage{stex}
7668 \stex_html_backend:T {
7669   \bool_set_true:N\c__notesslides_notes_bool
7670 }
7671
7672 \newif\ifnotes
7673 \bool_if:NTF \c__notesslides_notes_bool {
7674   \notesttrue
7675 }{
7676   \notesfalse
7677 }
7678

```

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

```

7679 \str_if_empty:NTF \c__notesslides_topsect_str {
7680   \str_set_eq:NN \__notesslides_topsect \c__notesslides_defaulttopsec_str
7681 }{
7682   \str_set_eq:NN \__notesslides_topsect \c__notesslides_topsect_str
7683 }
7684 \PassOptionsToPackage{topsect=\__notesslides_topsect}{document-structure}

```

```
7685 </package>
```

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

```
7686 <*cls>
7687 \bool_if:NTF \c__notesslides_notes_bool {
7688   \str_if_empty:NT \c__notesslides_class_str {
7689     \str_set:Nn \c__notesslides_class_str {article}
7690   }
7691   \exp_after:wN\LoadClass\exp_after:wN[\c__notesslides_dcopt_str]
7692     {\c__notesslides_class_str}
7693 }{
7694   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
7695   \newcounter{Item}
7696   \newcounter{paragraph}
7697   \newcounter{subparagraph}
7698   \newcounter{Hfootnote}
7699 }
7700 \RequirePackage{document-structure}
```

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

```
7701 \RequirePackage{notesslides}
7702 </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.

```
7703 <*package>
7704 \bool_if:NT \c__notesslides_notes_bool {
7705   \RequirePackage{a4wide}
7706   \RequirePackage{marginnote}
7707   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
7708   \RequirePackage{mdframed}
7709   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
7710   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
7711 }
7712 \RequirePackage{stex-tikzinput}
7713 \RequirePackage{comment}
7714 \RequirePackage{url}
7715 \RequirePackage{graphicx}
7716 \RequirePackage{pgf}
7717 \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.

```
7718 \bool_if:NT \c__notesslides_notes_bool {
7719   \renewcommand\usetheme[2][\usepackage{#1}{beamertheme#2}]
7720 }
```

```

7721 \NewDocumentCommand \libusetheme {0{} m} {
7722   \libusepackage[#1]{beamertheme#2}
7723 }
7724

```

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

```

7725 \newcounter{slide}
7726 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
7727 \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.

```

7728 \bool_if:NTF \c__notesslides_notes_bool {
7729   \renewenvironment{note}{\ignorespaces}{}
7730 }{
7731   \excludecomment{note}
7732 }

```

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.

```

7733 \bool_if:NT \c__notesslides_notes_bool {
7734   \newlength{\slideframewidth}
7735   \setlength{\slideframewidth}{1.5pt}

```

frame (*env.*) We first define the keys.

```

7736 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
7737   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
7738     \bool_set_true:N #1
7739   }{
7740     \bool_set_false:N #1
7741   }
7742 }
7743 \keys_define:nn{notesslides / frame}{
7744   label .str_set_x:N = \l__notesslides_frame_label_str,
7745   allowframebreaks .code:n = {
7746     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
7747   },
7748   allowdisplaybreaks .code:n = {
7749     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
7750   },
7751   fragile .code:n = {
7752     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
7753   },
7754   shrink .code:n = {
7755     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
7756   },
7757   squeeze .code:n = {
7758     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
7759   },
7760   t .code:n = {

```

```

7761     \_notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
7762   },
7763   unknown    .code:n      = {}
7764 }
7765 \cs_new_protected:Nn \_notesslides_frame_args:n {
7766   \str_clear:N \l__notesslides_frame_label_str
7767   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
7768   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
7769   \bool_set_true:N \l__notesslides_frame_fragile_bool
7770   \bool_set_true:N \l__notesslides_frame_shrink_bool
7771   \bool_set_true:N \l__notesslides_frame_squeeze_bool
7772   \bool_set_true:N \l__notesslides_frame_t_bool
7773   \keys_set:nn { notesslides / frame }{ #1 }
7774 }

```

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

```

7775 \renewenvironment{frame}[1][]{
7776   \_notesslides_frame_args:n{#1}
7777   \sffamily
7778   \stepcounter{slide}
7779   \def\@currentlabel{\theslide}
7780   \str_if_empty:NF \l__notesslides_frame_label_str {
7781     \label{\l__notesslides_frame_label_str}
7782   }

```

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

```

7783 \def\itemize@level{outer}
7784 \def\itemize@outer{outer}
7785 \def\itemize@inner{inner}
7786 \renewcommand\newpage{\addtocounter{framenum}{1}}
7787 %\newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
7788 \renewenvironment{itemize}{
7789   \ifx\itemize@level\itemize@outer
7790     \def\itemize@label{$\rhd$}
7791   \fi
7792   \ifx\itemize@level\itemize@inner
7793     \def\itemize@label{$\scriptstyle\rhd$}
7794   \fi
7795   \begin{list}
7796     {\itemize@label}
7797     {\setlength{\labelsep}{.3em}
7798      \setlength{\labelwidth}{.5em}
7799      \setlength{\leftmargin}{1.5em}
7800     }
7801   \edef\itemize@level{\itemize@inner}
7802 }{
7803   \end{list}
7804 }

```

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

```

7805 \stex_html_backend:TF {
7806   \begin{stex_annotate_env}{frame}{}\vbox\bgroup
7807     \mdf@patchamsthm
7808   }{
7809     \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwid

```

```

7810     }
7811   }{
7812     \stex_html_backend:TF {
7813       \miko@slidelabel\egroup\end{stex_annotate_env}
7814     }\medskip\miko@slidelabel\end{mdframed}}
7815   }

```

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

`\frametitle`

```

7816   \renewcommand{\frametitle}[1]{
7817     \stex_document_title:n { #1 }
7818     {\Large\bf\sf\color{blue}{#1}}\medskip
7819   }
7820 }

```

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

EdN:10

`\pause` 10

```

7821 \bool_if:NT \c__notesslides_notes_bool {
7822   \newcommand\pause{}
7823 }

```

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

`nparagraph (env.)`

```

7824 \bool_if:NTF \c__notesslides_notes_bool {
7825   \newenvironment{nparagraph}[1] [] {\begin{sparagraph}[#1]}\end{sparagraph}}
7826 }{
7827   \excludecomment{nparagraph}
7828 }

```

`nfragment (env.)`

```

7829 \bool_if:NTF \c__notesslides_notes_bool {
7830   \newenvironment{nfragment}[2] [] {\begin{sfragment}[#1][#2]}\end{sfragment}}
7831 }{
7832   \excludecomment{nfragment}
7833 }

```

`ndefinition (env.)`

```

7834 \bool_if:NTF \c__notesslides_notes_bool {
7835   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}\end{sdefinition}}
7836 }{
7837   \excludecomment{ndefinition}
7838 }

```

`nassertion (env.)`

```

7839 \bool_if:NTF \c__notesslides_notes_bool {
7840   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}\end{sassertion}}
7841 }{
7842   \excludecomment{nassertion}
7843 }

```

¹⁰EDNOTE: MK: fake it in notes mode for now

`nsproof (env.)`

```
7844 \bool_if:NTF \c__notesslides_notes_bool {
7845   \newenvironment{nproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
7846 }{
7847   \excludecomment{nproof}
7848 }
```

`nexample (env.)`

```
7849 \bool_if:NTF \c__notesslides_notes_bool {
7850   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}{\end{sexample}}
7851 }{
7852   \excludecomment{nexample}
7853 }
```

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

```
7854 \def\inputref@preskip{\smallskip}
7855 \def\inputref@postskip{\medskip}
```

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

`\inputref*`

```
7856 \let\orig@inputref\inputref
7857 \def\inputref{@ifstar\ninputref\orig@inputref}
7858 \newcommand\ninputref[2] []{
7859   \bool_if:NT \c__notesslides_notes_bool {
7860     \orig@inputref[#1]{#2}
7861   }
7862 }
```

(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>}`.

```
7863 \newlength{\slidelogoheight}
7864
7865 \RequirePackage{graphicx}
7866
7867 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7868 \providecommand\mhgraphics[2] []{
7869   \def\Gin@mhrepos{}\setkeys{Gin}{#1}
7870   \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}
7871 }
7872
7873 \bool_if:NTF \c__notesslides_notes_bool {
7874   \setlength{\slidelogoheight}{.4cm}
7875 }{
7876   \setlength{\slidelogoheight}{.25cm}
7877 }
```

```

7878 \ifcsname slidelogo\endcsname\else
7879 \newsavebox{\slidelogo}
7880 \sbox{\slidelogo}{\sTeX}
7881 \fi
7882 \newrobustcmd{\setslidelogo}[2][]{
7883 \tl_if_empty:nTF{#1}{
7884 \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#2}}
7885 }{
7886 \sbox{\slidelogo}{\mhgraphics[height=\slidelogoheight,mhrepos=#1]{#2}}
7887 }
7888 }

```

(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.

```

7889 \bool_if:NT \c__notesslides_notes_bool {
7890 \def\author{\@dblarg\@ns@author}
7891 \long\def\@ns@author[#1]#2{%
7892 \def\c__notesslides_shortauthor{#1}%
7893 \def\@author{#2}
7894 }
7895 }

```

(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.

```

7896 \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.

```

7897 \def\copyrightnotice{%
7898 \footnotesize\copyright : \hspace{.3ex}%
7899 \ifcsname source\endcsname\source\else%
7900 \ifcsname c__notesslides_shortauthor\endcsname\c__notesslides_shortauthor\else%
7901 \PackageWarning{notesslides}{Author/Source-undefined-in-copyright-notice}%
7902 ?source/author?\fi%
7903 \fi}
7904 \newsavebox{\cclogo}
7905 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{stex-cc_somerights}}
7906 \newif\ifcchref\cchreffalse
7907 \AtBeginDocument{
7908 \@ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
7909 }
7910 \def\licensing{
7911 \ifcchref
7912 \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
7913 \else
7914 {\usebox{\cclogo}}

```

```

7915 \fi
7916 }
7917 \newrobustcmd{\setlicensing}[2][]{
7918   \def\@url{#1}
7919   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
7920   \ifx\@url\@empty
7921     \def\licensing{\usebox{\cclogo}}
7922   \else
7923     \def\licensing{
7924       \ifcchref
7925         \href{#1}{\usebox{\cclogo}}
7926       \else
7927         {\usebox{\cclogo}}
7928     \fi
7929   }
7930 \fi
7931 }

```

(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
7932 \newrobustcmd\miko@slidelabel{
7933   \vbox to \slidelogoheight{
7934     \vss\hbox to \slidewidth
7935       {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
7936   }
7937 }

```

(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.

```

7938 \def\Gin@mhrepos{}
7939 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7940 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
7941 \newrobustcmd\frameimage[2][]{
7942   \stepcounter{slide}
7943   \bool_if:NT \c__notesslides_frameimages_bool {
7944     \def\Gin@ewidth{}\setkeys{Gin}{#1}
7945     \bool_if:NF \c__notesslides_notes_bool { \vfill }
7946     \begin{center}
7947       \bool_if:NTF \c__notesslides_fiboxed_bool {
7948         \fbox{
7949           \ifx\Gin@ewidth\@empty
7950             \ifx\Gin@mhrepos\@empty
7951               \mhgraphics[width=\slidewidth,#1]{#2}
7952             \else
7953               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7954             \fi
7955           \else% Gin@ewidth empty

```

¹¹EdNOTE: see that we can use the themes for the slides some day. This is all fake.


```

7956         \ifx\Gin@mhrepos\@empty
7957         \mhgraphics[#1]{#2}
7958     \else
7959         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7960     \fi
7961 \fi% Gin@ewidth empty
7962 }
7963 }{
7964     \ifx\Gin@ewidth\@empty
7965     \ifx\Gin@mhrepos\@empty
7966         \mhgraphics[width=\slidewidth,#1]{#2}
7967     \else
7968         \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7969     \fi
7970     \ifx\Gin@mhrepos\@empty
7971         \mhgraphics[#1]{#2}
7972     \else
7973         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7974     \fi
7975     \fi% Gin@ewidth empty
7976 }
7977 \end{center}
7978 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
7979 \bool_if:NF \c__notesslides_notes_bool { \vfill }
7980 }
7981 } % 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`.

```

7982 \stex_html_backend:F {
7983     \bool_if:NT \c__notesslides_sectocframes_bool {
7984         \str_if_eq:VnTF \__notesslidestopsect{part}{
7985             \newcounter{chapter}\counterwithin*{section}{chapter}
7986         }{
7987             \str_if_eq:VnT\__notesslidestopsect{chapter}{
7988                 \newcounter{chapter}\counterwithin*{section}{chapter}
7989             }
7990         }
7991     }
7992 }

```

`\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
7993 \def\part@prefix{}
7994 \@ifpackageloaded{document-structure}{
7995     \str_case:VnF \__notesslidestopsect {

```

```

7996 {part}{
7997   \int_set:Nn \l_document_structure_section_level_int {0}
7998   \def\thesection{\arabic{chapter}.\arabic{section}}
7999   \def\part@prefix{\arabic{chapter}.}
8000 }
8001 {chapter}{
8002   \int_set:Nn \l_document_structure_section_level_int {1}
8003   \def\thesection{\arabic{chapter}.\arabic{section}}
8004   \def\part@prefix{\arabic{chapter}.}
8005 }
8006 }{
8007   \int_set:Nn \l_document_structure_section_level_int {2}
8008   \def\part@prefix{}
8009 }
8010 }
8011
8012 \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.)`

```

8013 \renewenvironment{sfragment}[2][]{
8014   \__document_structure_sfragment_args:n { #1 }
8015   \int_incr:N \l_document_structure_section_level_int
8016   \bool_if:NT \c__notesslides_sectocframes_bool {
8017     \stepcounter{slide}
8018     \begin{frame}[noframenumbering]
8019     \vfill\Large\centering
8020     \red{
8021       \ifcase\l_document_structure_section_level_int\or
8022         \stepcounter{part}
8023         \def\__notesslideslabel{\omdoc@part@kw}~\Roman{part}}
8024         \addcontentsline{toc}{part}{\protect\numberline{\thepart}#2}
8025         \pdfbookmark[0]{\thepart\ #2}{part.\thepart}
8026         \def\currentsectionlevel{\omdoc@part@kw}
8027       \or
8028         \stepcounter{chapter}
8029         \def\__notesslideslabel{\omdoc@chapter@kw}~\arabic{chapter}}
8030         \addcontentsline{toc}{chapter}{\protect\numberline{\thechapter}#2}
8031         \pdfbookmark[1]{\thechapter\ #2}{chapter.\cs_if_exist:cT{\thepart}\thepart.\thechapter}
8032         \def\currentsectionlevel{\omdoc@chapter@kw}
8033       \or
8034         \stepcounter{section}
8035         \def\__notesslideslabel{\part@prefix\arabic{section}}
8036         \addcontentsline{toc}{section}{\protect\numberline{\thesection}#2}
8037         \pdfbookmark[2]{\cs_if_exist:cT{\thechapter}{\thechapter}.\thesection\ #2}
8038         {section.\cs_if_exist:cT{\thepart}{\thepart}.\cs_if_exist:cT{\thechapter}{\thechapter}.\thepart}
8039         \def\currentsectionlevel{\omdoc@section@kw}
8040       \or
8041         \stepcounter{subsection}
8042         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
8043         \addcontentsline{toc}{subsection}{\protect\numberline{\thesubsection}#2}

```

```

8044         \pdfbookmark[3]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection
8045         {subsection.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection}
8046         \def\currentsectionlevel{\omdoc@subsection@kw}
8047     \or
8048         \stepcounter{subsubsection}
8049         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
8050         \addcontentsline{toc}{subsubsection}{\protect\numberline{\thesubsubsection}#2}
8051         \pdfbookmark[4]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsubsection}
8052         {subsubsection.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsubsection}
8053         \def\currentsectionlevel{\omdoc@subsubsection@kw}
8054     \or
8055         \stepcounter{paragraph}
8056         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{paragraph}}
8057         \addcontentsline{toc}{paragraph}{\protect\numberline{\theparagraph}#2}
8058         \pdfbookmark[5]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsubsection}
8059         {paragraph.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsubsection}
8060         \def\currentsectionlevel{\omdoc@paragraph@kw}
8061     \else
8062         \def\__notesslideslabel{}
8063         \def\currentsectionlevel{\omdoc@paragraph@kw}
8064     \fi% end ifcase
8065     \__notesslideslabel\quad #2%
8066 }%
8067 \vfill%
8068 \end{frame}%
8069 }
8070 \str_if_empty:NF \l__document_structure_sfragment_id_str {
8071     \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
8072 }
8073 }{}
8074 }

```

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

```

8075 \def\inserttheorembodyfont{\normalfont}
8076 %\bool_if:NF \c__notesslides_notes_bool {
8077 % \defbeamertemplate{theorem begin}{miko}
8078 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
8079 % \ifx\inserttheoremaddition@empty\else\ (\inserttheoremaddition)\fi%
8080 % \inserttheorempunctuation\inserttheorembodyfont\xspace}
8081 % \defbeamertemplate{theorem end}{miko}{\fi}

```

and we set it as the default one.

```

8082 % \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.

```

8083 % \expandafter\def\csname Parent2\endcsname{}
8084 %}
8085
8086 \AddToHook{begindocument}{ % this does not work for some reason
8087     \setbeamertemplate{theorems}[ams style]
8088 }
8089 \bool_if:NT \c__notesslides_notes_bool {
8090     \renewenvironment{columns}[1][\fi]{}{}

```

```

8091     \par\noindent%
8092     \begin{minipage}%
8093     \slidewidth\centering\leavevmode%
8094   }{%
8095     \end{minipage}\par\noindent%
8096   }%
8097   \newsavebox\columnbox%
8098   \renewenvironment<>{column}[2][]{%
8099     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
8100   }{%
8101     \end{minipage}\end{lrbox}\usebox\columnbox%
8102   }%
8103 }

8104 \bool_if:NTF \c__notesslides_noproblems_bool {
8105   \newenvironment{problems}{}{}
8106 }{
8107   \excludacomment{problems}
8108 }

```

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.

```

8109 \gdef\printexcursions{}
8110 \newcommand\excursionref[2]{% label, text
8111   \bool_if:NT \c__notesslides_notes_bool {
8112     \begin{sparagraph}[title=Excursion]
8113       #2 \sref[fallback=the appendix]{#1}.
8114     \end{sparagraph}
8115   }
8116 }
8117 \newcommand\activate@excursion[2][{}{
8118   \gappto\printexcursions{\inputref{#1}{#2}}
8119 }
8120 \newcommand\excursion[4][{}{ repos, label, path, text
8121   \bool_if:NT \c__notesslides_notes_bool {
8122     \activate@excursion{#1}{#3}\excursionref{#2}{#4}
8123   }
8124 }

```

(End definition for `\excursion`. This function is documented on page 66.)

\excursiongroup

```

8125 \keys_define:nn{notesslides / excursiongroup }{
8126   id          .str_set_x:N = \l__notesslides_excursion_id_str,
8127   intro       .tl_set:N    = \l__notesslides_excursion_intro_tl,
8128   mhrepos     .str_set_x:N = \l__notesslides_excursion_mhrepos_str
8129 }
8130 \cs_new_protected:Nn \__notesslides_excursion_args:n {
8131   \tl_clear:N \l__notesslides_excursion_intro_tl
8132   \str_clear:N \l__notesslides_excursion_id_str

```

```

8133 \str_clear:N \l__notesslides_excursion_mhrepos_str
8134 \keys_set:nn {notesslides / excursionsgroup }{ #1 }
8135 }
8136 \newcommand\excursionsgroup[1][]{
8137   \__notesslides_excursion_args:n{ #1 }
8138   \ifdefempty\printexcursions{}% only if there are excursions
8139   {\begin{note}
8140     \begin{sfragment}[#1]{Excursions}%
8141     \ifdefempty\l__notesslides_excursion_intro_tl}{
8142       \inputref[\l__notesslides_excursion_mhrepos_str]{
8143         \l__notesslides_excursion_intro_tl
8144       }
8145     }
8146     \printexcursions%
8147     \end{sfragment}
8148   \end{note}}
8149 }
8150 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
8151 \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.

```
8152 <*package>
8153 <@@=problems>
8154 \ProvidesExplPackage{problem}{2022/09/14}{3.2.0}{Semantic Markup for Problems}
8155 \RequirePackage{13keys2e}
8156 \RequirePackage{amssymb}% for \Box
8157
8158 \keys_define:nn { problem / pkg }{
8159   notes      .default:n    = { true },
8160   notes      .bool_set:N   = \c__problems_notes_bool,
8161   gnotes     .default:n    = { true },
8162   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
8163   hints      .default:n    = { true },
8164   hints      .bool_set:N   = \c__problems_hints_bool,
8165   solutions  .default:n    = { true },
8166   solutions  .bool_set:N   = \c__problems_solutions_bool,
8167   pts        .default:n    = { true },
8168   pts        .bool_set:N   = \c__problems_pts_bool,
8169   min        .default:n    = { true },
8170   min        .bool_set:N   = \c__problems_min_bool,
8171   boxed      .default:n    = { true },
8172   boxed      .bool_set:N   = \c__problems_boxed_bool,
8173   test       .default:n    = { true },
8174   test       .bool_set:N   = \c__problems_test_bool,
8175   unknown    .code:n       = {
8176     \PassOptionsToPackage{\CurrentOption}{stex}
8177   }
8178 }
8179 \newif\ifsolutions
8180
8181 \ProcessKeysOptions{ problem / pkg }
8182 \bool_if:NTF \c__problems_solutions_bool {
8183   \solutionstrue
```

```

8184 }{
8185   \solutionsfalse
8186 }
8187 \RequirePackage{stex}

```

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

```

8188 \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.

```

8189 \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.

```

8190 \def\prob@problem@kw{Problem}
8191 \def\prob@solution@kw{Solution}
8192 \def\prob@hint@kw{Hint}
8193 \def\prob@note@kw{Note}
8194 \def\prob@grade@kw{Grading}
8195 \def\prob@pt@kw{pt}
8196 \def\prob@min@kw{min}
8197 \def\prob@correct@kw{Correct}
8198 \def\prob@wrong@kw{Wrong}

```

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

For the other languages, we set up triggers

```

8199 \AddToHook{begindocument}{
8200   \ltx@ifpackageloaded{babel}{
8201     \makeatletter
8202     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
8203     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
8204       \input{problem-ngerman.ldf}
8205     }
8206     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
8207       \input{problem-finnish.ldf}
8208     }
8209     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
8210       \input{problem-french.ldf}
8211     }
8212     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
8213       \input{problem-russian.ldf}
8214     }
8215     \makeatother
8216   }{ }
8217 }

```

41.2 Problems and Solutions

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

```

8218 \keys_define:nn{ problem / problem }{
8219   id .str_set_x:N = \l__problems_prob_id_str,

```

```

8220 pts      .tl_set:N      = \l__problems_prob_pts_tl,
8221 min      .tl_set:N      = \l__problems_prob_min_tl,
8222 title     .tl_set:N      = \l__problems_prob_title_tl,
8223 type      .tl_set:N      = \l__problems_prob_type_tl,
8224 imports   .tl_set:N      = \l__problems_prob_imports_tl,
8225 name      .str_set_x:N    = \l__problems_prob_name_str,
8226 refnum     .int_set:N     = \l__problems_prob_refnum_int
8227 }
8228 \cs_new_protected:Nn \__problems_prob_args:n {
8229   \str_clear:N \l__problems_prob_id_str
8230   \str_clear:N \l__problems_prob_name_str
8231   \tl_clear:N \l__problems_prob_pts_tl
8232   \tl_clear:N \l__problems_prob_min_tl
8233   \tl_clear:N \l__problems_prob_title_tl
8234   \tl_clear:N \l__problems_prob_type_tl
8235   \tl_clear:N \l__problems_prob_imports_tl
8236   \int_zero_new:N \l__problems_prob_refnum_int
8237   \keys_set:nn { problem / problem }{ #1 }
8238   \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
8239     \let\l__problems_prob_refnum_int\undefined
8240   }
8241 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

8242 \newcounter{problem}[section]
8243 \newcommand\numberproblemsin[1]{\@addtoreset{problem}{#1}}
8244 \def\theplainsproblem{\arabic{problem}}
8245 \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.

```

8246 \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

```

8247 \newcommand\prob@number{
8248   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
8249     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
8250   }{
8251     \int_if_exist:NTF \l__problems_prob_refnum_int {
8252       \prob@label{\int_use:N \l__problems_prob_refnum_int }
8253     }{
8254       \prob@label\theplainsproblem
8255     }
8256   }
8257 }
8258 \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.

```

8259 \newcommand\prob@title[3]{%
8260   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
8261     #2 \l__problems_inclprob_title_tl #3
8262   }{
8263     \tl_if_empty:NTF \l__problems_prob_title_tl {
8264       #1
8265     }{
8266       #2 \l__problems_prob_title_tl #3
8267     }
8268   }
8269 }
```

(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.

```

8270 \def\prob@heading{
8271   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
8272   %\sref@label@id{\prob@problem@kw~\prob@number}{~}
8273 }
```

(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.)`

```

8274 \newenvironment{sproblem}[1][]{
8275   \__problems_prob_args:n{#1}%\sref@target%
8276   \@in@omtexttrue% we are in a statement (for inline definitions)
8277   \refstepcounter{sproblem}\record@problem
8278   \def\current@section@level{\prob@problem@kw}
8279
8280   \str_if_empty:NT \l__problems_prob_name_str {
8281     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
8282     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
8283     \seq_get_left:NN \l_tmpa_seq \l__problems_prob_name_str
8284   }
8285
8286   \stex_if_do_html:T{
8287     \tl_if_empty:NF \l__problems_prob_title_tl {
8288       \exp_args:No \stex_document_title:n \l__problems_prob_title_tl
8289     }
8290   }
8291
8292   \exp_args:Nno\stex_module_setup:nn{type=problem}\l__problems_prob_name_str
8293
8294   \stex_reactivate_macro:N \STEXexport
8295   \stex_reactivate_macro:N \importmodule
```

```

8296 \stex_reactivate_macro:N \symdecl
8297 \stex_reactivate_macro:N \notation
8298 \stex_reactivate_macro:N \symdef
8299
8300 \stex_if_do_html:T{
8301   \begin{stex_annotate_env} {problem} {
8302     \l_stex_module_ns_str ? \l_stex_module_name_str
8303   }
8304
8305   \stex_annotate_invisible:nnn{header}{} {
8306     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
8307     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
8308     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
8309       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
8310     }
8311   }
8312 }
8313
8314 \stex_csl_to_imports:No \importmodule \l__problems_prob_imports_tl
8315
8316
8317 \tl_if_exist:NTF \l__problems_inclprob_type_tl {
8318   \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
8319 }{
8320   \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
8321 }
8322 \str_if_exist:NTF \l__problems_inclprob_id_str {
8323   \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
8324 }{
8325   \str_set_eq:NN \sproblemid \l__problems_prob_id_str
8326 }
8327
8328
8329 \stex_if_smsmode:F {
8330   \clist_set:No \l_tmpa_clist \sproblemtype
8331   \tl_clear:N \l_tmpa_tl
8332   \clist_map_inline:Nn \l_tmpa_clist {
8333     \tl_if_exist:cT {__problems_sproblem_##1_start:}{
8334       \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_start:}}
8335     }
8336   }
8337   \tl_if_empty:NTF \l_tmpa_tl {
8338     \__problems_sproblem_start:
8339   }{
8340     \l_tmpa_tl
8341   }
8342 }
8343 \stex_ref_new_doc_target:n \sproblemid
8344 \stex_if_smsmode:TF \stex_smsmode_do: \ignorespacesandpars
8345 }{
8346   \__stex_modules_end_module:
8347   \stex_if_smsmode:F{
8348     \clist_set:No \l_tmpa_clist \sproblemtype
8349     \tl_clear:N \l_tmpa_tl

```

```

8350 \clist_map_inline:Nn \l_tmpa_clist {
8351   \tl_if_exist:cT {__problems_sproblem_##1_end:}{
8352     \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_end:}}
8353   }
8354 }
8355 \tl_if_empty:NTF \l_tmpa_tl {
8356   \__problems_sproblem_end:
8357 }{
8358   \l_tmpa_tl
8359 }
8360 }
8361 \stex_if_do_html:T{
8362   \end{stex_annotate_env}
8363 }
8364
8365 \smallskip
8366 }
8367
8368 \seq_put_right:Nx\g_stex_smsmode_allowedenvs_seq{\tl_to_str:n{sproblem}}
8369
8370
8371
8372 \cs_new_protected:Nn \__problems_sproblem_start: {
8373   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
8374 }
8375 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
8376
8377 \newcommand\stexpatchproblem[3][] {
8378   \str_set:Nx \l_tmpa_str{ #1 }
8379   \str_if_empty:NTF \l_tmpa_str {
8380     \tl_set:Nn \__problems_sproblem_start: { #2 }
8381     \tl_set:Nn \__problems_sproblem_end: { #3 }
8382   }{
8383     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
8384     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
8385   }
8386 }
8387
8388
8389 \bool_if:NT \c__problems_boxed_bool {
8390   \surroundwithmdframed{problem}
8391 }

```

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

```

8392 \def\record@problem{
8393   \protected@write\@auxout{}
8394   {
8395     \string\@problem{\prob@number}
8396     {
8397       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
8398         \l__problems_inclprob_pts_tl
8399       }{
8400         \l__problems_prob_pts_tl
8401       }

```

```

8402 }%
8403 {
8404   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
8405     \l__problems_inclprob_min_tl
8406   }{
8407     \l__problems_prob_min_tl
8408   }
8409 }
8410 }
8411 }

```

(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).

```

8412 \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.

```

8413 \keys_define:nn { problem / solution }{
8414   id          .str_set_x:N = \l__problems_solution_id_str ,
8415   for         .str_set_x:N = \l__problems_solution_for_str ,
8416   type       .str_set_x:N = \l__problems_solution_type_str ,
8417   title      .tl_set:N     = \l__problems_solution_title_tl
8418 }
8419 \cs_new_protected:Nn \__problems_solution_args:n {
8420   \str_clear:N \l__problems_solution_id_str
8421   \str_clear:N \l__problems_solution_type_str
8422   \str_clear:N \l__problems_solution_for_str
8423   \tl_clear:N \l__problems_solution_title_tl
8424   \keys_set:nn { problem / solution }{ #1 }
8425 }

```

`\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.

```

8426 \box_new:N \l__problems_solution_box
8427 \newenvironment{solution}[1][{}]{
8428   \__problems_solution_args:n{#1}
8429   \stex_html_backend:TF{
8430     \stex_if_do_html:T{
8431       \begin{stex_annotate_env}{solution}{}
8432       \str_if_empty:NF \l__problems_solution_type_str {
8433         \par\noindent
8434         \stex_annotate_invisible:nnn{typestrings}{\sexampletype}{}
8435       }
8436       \noindent\textbf{Solution}\tl_if_empty:NF\l__problems_solution_title_tl{~(\l__problem
8437     }
8438   }{
8439     \setbox\l__problems_solution_box\vbox\bgroup
8440     \par\smallskip\hrule\smallskip
8441     \noindent\textbf{Solution}\tl_if_empty:NF\l__problems_solution_title_tl{~(\l__problems
8442   }

```

```

8443 }{
8444   \stex_html_backend:TF{
8445     \stex_if_do_html:T{
8446       \end{stex_annotate_env}
8447     }
8448   }{
8449     \smallskip\hrule
8450     \egroup
8451     \bool_if:NT \c__problems_solutions_bool {
8452       \strut\par\noindent
8453       \box\l__problems_solution_box
8454     }
8455   }
8456 }
8457
8458 \newcommand\startsolutions{
8459   \bool_set_true:N \c__problems_solutions_bool
8460   \solutionstrue
8461   % \specialcomment{solution}{\@startsolution}{
8462   %   \bool_if:NF \c__problems_boxed_bool {
8463   %     \hrule\medskip
8464   %   }
8465   %   \end{small}}%
8466   % }
8467   % \bool_if:NT \c__problems_boxed_bool {
8468   %   \surroundwithhmdframed{solution}
8469   % }
8470 }

```

(End definition for \startsolutions. This function is documented on page 68.)

\stopsolutions

```

8471 \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.)

```

8472 \bool_if:NTF \c__problems_notes_bool {
8473   \newenvironment{exnote}[1][]{
8474     \par\smallskip\hrule\smallskip
8475     \noindent\textbf{\prob@note@kw :~ }\small
8476   }{
8477     \smallskip\hrule
8478   }
8479   }{
8480     \excludecomment{exnote}
8481   }

```

hint (env.)

```

8482 \bool_if:NTF \c__problems_notes_bool {
8483   \newenvironment{hint}[1][]{
8484     \par\smallskip\hrule\smallskip
8485     \noindent\textbf{\prob@hint@kw :~ }\small
8486   }{

```

```

8487     \smallskip\hrule
8488   }
8489   \newenvironment{exhint}[1][{}{
8490     \par\smallskip\hrule\smallskip
8491     \noindent\textbf{\prob@hint@kw :~ }\small
8492   }{
8493     \smallskip\hrule
8494   }
8495   }{
8496     \excludecomment{hint}
8497     \excludecomment{exhint}
8498   }

```

gnote (*env.*)

```

8499   \bool_if:NTF \c__problems_notes_bool {
8500     \newenvironment{gnote}[1][{}{
8501       \par\smallskip\hrule\smallskip
8502       \noindent\textbf{\prob@gnote@kw :~ }\small
8503     }{
8504       \smallskip\hrule
8505     }
8506     }{
8507       \excludecomment{gnote}
8508     }

```

41.3 Markup for Added Value Services

41.4 Multiple Choice Blocks

EdN:12

mcb (*env.*)¹²

```

8509   \newenvironment{mcb}{
8510     \begin{enumerate}
8511   }{
8512     \end{enumerate}
8513   }

```

we define the keys for the mcb macro

```

8514   \cs_new_protected:Nn \__problems_do_yes_param:Nn {
8515     \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
8516       \bool_set_true:N #1
8517     }{
8518       \bool_set_false:N #1
8519     }
8520   }
8521   \keys_define:nn { problem / mcb }{
8522     id      .str_set_x:N = \l__problems_mcc_id_str ,
8523     feedback .tl_set:N   = \l__problems_mcc_feedback_tl ,
8524     T       .default:n   = { false } ,
8525     T       .bool_set:N  = \l__problems_mcc_t_bool ,
8526     F       .default:n   = { false } ,

```

¹²EdNOTE: MK: maybe import something better here from a dedicated MC package

```

8527 F      .bool_set:N    = \l__problems_mcc_f_bool ,
8528 Ttext   .tl_set:N     = \l__problems_mcc_Ttext_tl ,
8529 Ftext   .tl_set:N     = \l__problems_mcc_Ftext_tl
8530 }
8531 \cs_new_protected:Nn \l__problems_mcc_args:n {
8532   \str_clear:N \l__problems_mcc_id_str
8533   \tl_clear:N \l__problems_mcc_feedback_tl
8534   \bool_set_false:N \l__problems_mcc_t_bool
8535   \bool_set_false:N \l__problems_mcc_f_bool
8536   \tl_clear:N \l__problems_mcc_Ttext_tl
8537   \tl_clear:N \l__problems_mcc_Ftext_tl
8538   \str_clear:N \l__problems_mcc_id_str
8539   \keys_set:nn { problem / mcc }{ #1 }
8540 }

```

\mcc

```

8541 \def\mccTrueText{\textbf{\prob@correct@kw!~}}
8542 \def\mccFalseText{\textbf{\prob@wrong@kw!~}}
8543 \newcommand\mcc[2][] {
8544   \l__problems_mcc_args:n{ #1 }
8545   \item[{$\Box$}] #2
8546   \bool_if:NT \c__problems_solutions_bool {
8547     \
8548     \bool_if:NT \l__problems_mcc_t_bool {
8549       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccTrueText\l__problems_mcc_Ttext_tl
8550     }
8551     \bool_if:NT \l__problems_mcc_f_bool {
8552       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccFalseText\l__problems_mcc_Ftext_tl
8553     }
8554     \tl_if_empty:NF \l__problems_mcc_feedback_tl {
8555       \emph{\l__problems_mcc_feedback_tl}
8556     }
8557   }
8558 } %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.

```

8559 \newcommand\fillinsol[2][] {%
8560   \def\@test{#1}
8561   \quad%
8562   \ifsolutions\textcolor{red}{\#1!}\else%
8563   \fbox{\ifx\@test\@empty\phantom{\huge{21}}\else\hspace{#1}\fi}%
8564   \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.

```

8565
8566 \keys_define:nn{ problem / inclproblem }{
8567   id      .str_set_x:N = \l__problems_inclprob_id_str,
8568   pts     .tl_set:N    = \l__problems_inclprob_pts_tl,
8569   min     .tl_set:N    = \l__problems_inclprob_min_tl,
8570   title   .tl_set:N    = \l__problems_inclprob_title_tl,
8571   refnum  .int_set:N    = \l__problems_inclprob_refnum_int,
8572   type    .tl_set:N    = \l__problems_inclprob_type_tl,
8573   mhrepos .str_set_x:N = \l__problems_inclprob_mhrepos_str
8574 }
8575 \cs_new_protected:Nn \l__problems_inclprob_args:n {
8576   \str_clear:N \l__problems_prob_id_str
8577   \tl_clear:N \l__problems_inclprob_pts_tl
8578   \tl_clear:N \l__problems_inclprob_min_tl
8579   \tl_clear:N \l__problems_inclprob_title_tl
8580   \tl_clear:N \l__problems_inclprob_type_tl
8581   \int_zero_new:N \l__problems_inclprob_refnum_int
8582   \str_clear:N \l__problems_inclprob_mhrepos_str
8583   \keys_set:nn { problem / inclproblem }{ #1 }
8584   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
8585     \let\l__problems_inclprob_pts_tl\undefined
8586   }
8587   \tl_if_empty:NT \l__problems_inclprob_min_tl {
8588     \let\l__problems_inclprob_min_tl\undefined
8589   }
8590   \tl_if_empty:NT \l__problems_inclprob_title_tl {
8591     \let\l__problems_inclprob_title_tl\undefined
8592   }
8593   \tl_if_empty:NT \l__problems_inclprob_type_tl {
8594     \let\l__problems_inclprob_type_tl\undefined
8595   }
8596   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
8597     \let\l__problems_inclprob_refnum_int\undefined
8598   }
8599 }
8600
8601 \cs_new_protected:Nn \l__problems_inclprob_clear: {
8602   \let\l__problems_inclprob_id_str\undefined
8603   \let\l__problems_inclprob_pts_tl\undefined
8604   \let\l__problems_inclprob_min_tl\undefined
8605   \let\l__problems_inclprob_title_tl\undefined
8606   \let\l__problems_inclprob_type_tl\undefined
8607   \let\l__problems_inclprob_refnum_int\undefined
8608   \let\l__problems_inclprob_mhrepos_str\undefined
8609 }
8610 \l__problems_inclprob_clear:
8611
8612 \newcommand\includeproblem[2][ ]{
8613   \l__problems_inclprob_args:n{ #1 }

```



```

8614 \exp_args:No \stex_in_repository:nn\l__problems_inclprob_mhrepos_str{
8615   \stex_html_backend:TF {
8616     \str_clear:N \l_tmpa_str
8617     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
8618       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
8619     }
8620     \stex_annotate_invisible:nnn{includeproblem}{
8621       \l_tmpa_str / #2
8622     }{}
8623   }{
8624     \begingroup
8625     \inputreftrue
8626     \tl_if_empty:nTF{ ##1 }{
8627       \input{#2}
8628     }{
8629       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
8630     }
8631     \endgroup
8632   }
8633 }
8634 \__problems_inclprob_clear:
8635 }

```

(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.

```

8636 \AddToHook{enddocument}{
8637   \bool_if:NT \c__problems_pts_bool {
8638     \message{Total:~\arabic{pts}~points}
8639   }
8640   \bool_if:NT \c__problems_min_bool {
8641     \message{Total:~\arabic{min}~minutes}
8642   }
8643 }

```

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

```

8644 \def\pts#1{
8645   \bool_if:NT \c__problems_pts_bool {
8646     \marginpar{#1~\prob@pt@kw}
8647   }
8648 }
8649 \def\min#1{
8650   \bool_if:NT \c__problems_min_bool {
8651     \marginpar{#1~\prob@min@kw}
8652   }
8653 }

```

`\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.

```

8654 \newcounter{pts}
8655 \def\show@pts{
8656   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
8657     \bool_if:NT \c__problems_pts_bool {
8658       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
8659       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
8660     }
8661   }{
8662     \tl_if_exist:NT \l__problems_prob_pts_tl {
8663       \bool_if:NT \c__problems_pts_bool {
8664         \tl_if_empty:NT\l__problems_prob_pts_tl{
8665           \tl_set:Nn \l__problems_prob_pts_tl {0}
8666         }
8667         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
8668         \addtocounter{pts}{\l__problems_prob_pts_tl}
8669       }
8670     }
8671   }
8672 }

```

(End definition for \show@pts. This function is documented on page ??.)
and now the same for the minutes

\show@min

```

8673 \newcounter{min}
8674 \def\show@min{
8675   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
8676     \bool_if:NT \c__problems_min_bool {
8677       \marginpar{\l__problems_inclprob_min_tl\ min}
8678       \addtocounter{min}{\l__problems_inclprob_min_tl}
8679     }
8680   }{
8681     \tl_if_exist:NT \l__problems_prob_min_tl {
8682       \bool_if:NT \c__problems_min_bool {
8683         \tl_if_empty:NT\l__problems_prob_min_tl{
8684           \tl_set:Nn \l__problems_prob_min_tl {0}
8685         }
8686         \marginpar{\l__problems_prob_min_tl\ min}
8687         \addtocounter{min}{\l__problems_prob_min_tl}
8688       }
8689     }
8690   }
8691 }
8692 \</package>

```

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

41.8 Testing and Spacing

\testspace

```

8693 \newcommand\testspace[1]{\bool_if:NT \c__problems_boxed_bool {\vspace*{#1}}}

```

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

`\testnewpage`

```
8694 \newcommand\testnewpage{\bool_if:NT \c__problems_boxed_bool {\newpage}}
```

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

`\testemptypage`

```
8695 \newcommand\testemptypage[1] [] {%
```

```
8696 \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`

```
8697 \newcommand\testsmallspace{\testspace{1cm}}
```

```
8698 \newcommand\testmedspace{\testspace{2cm}}
```

```
8699 \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.

```
8700 {*package}
8701 \ProvidesExplPackage{hwexam}{2022/09/14}{3.2.0}{homework assignments and exams}
8702 \RequirePackage{13keys2e}
8703
8704 \newif\iftest\testfalse
8705 \DeclareOption{test}{\testtrue\PassOptionsToPackage{\CurrentOption}{problem}}
8706 \newif\ifmultiple\multiplefalse
8707 \DeclareOption{multiple}{\multipletrue}
8708 \DeclareOption{lang}{\PassOptionsToPackage{\CurrentOption}{problem}}
8709 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
8710 \ProcessOptions
```

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

```
8711 \RequirePackage{keyval}[1997/11/10]
8712 \RequirePackage{problem}
```

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

```
8713 \newcommand\hwexam@assignment@kw{Assignment}
8714 \newcommand\hwexam@given@kw{Given}
8715 \newcommand\hwexam@due@kw{Due}
8716 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~blank~for~extra~space}
8717 \newcommand\hwexam@minutes@kw{minutes}
8718 \newcommand\correction@probs@kw{prob.}
8719 \newcommand\correction@pts@kw{total}
8720 \newcommand\correction@reached@kw{reached}
8721 \newcommand\correction@sum@kw{Sum}
8722 \newcommand\correction@grade@kw{grade}
8723 \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

```

8724 \AddToHook{begindocument}{
8725 \ltx@ifpackageloaded{babel}{
8726 \makeatletter
8727 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
8728 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
8729 \input{hwexam-ngerman.ldf}
8730 }
8731 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
8732 \input{hwexam-finnish.ldf}
8733 }
8734 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
8735 \input{hwexam-french.ldf}
8736 }
8737 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
8738 \input{hwexam-russian.ldf}
8739 }
8740 \makeatother
8741 }{}
8742 }
8743

```

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.

```

8744 \newcounter{assignment}
8745 %\numberproblemsin{assignment}

```

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

```

8746 \keys_define:nn { hwexam / assignment } {
8747 id .str_set:N = \l_@@_assign_id_str,
8748 number .int_set:N = \l_@@_assign_number_int,
8749 title .tl_set:N = \l_@@_assign_title_tl,
8750 type .tl_set:N = \l_@@_assign_type_tl,
8751 given .tl_set:N = \l_@@_assign_given_tl,
8752 due .tl_set:N = \l_@@_assign_due_tl,
8753 loadmodules .code:n = {
8754 \bool_set_true:N \l_@@_assign_loadmodules_bool
8755 }
8756 }
8757 \cs_new_protected:Nn \_@@_assignment_args:n {
8758 \str_clear:N \l_@@_assign_id_str
8759 \int_set:Nn \l_@@_assign_number_int {-1}
8760 \tl_clear:N \l_@@_assign_title_tl
8761 \tl_clear:N \l_@@_assign_type_tl
8762 \tl_clear:N \l_@@_assign_given_tl
8763 \tl_clear:N \l_@@_assign_due_tl
8764 \bool_set_false:N \l_@@_assign_loadmodules_bool
8765 \keys_set:nn { hwexam / assignment }{ #1 }
8766 }

```

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.

```

8767 \newcommand\given@due[2]{
8768 \bool_lazy_all:nF {
8769 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
8770 { \tl_if_empty_p:V \l_@@_assign_given_tl }
8771 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
8772 { \tl_if_empty_p:V \l_@@_assign_due_tl }
8773 }{ #1 }
8774
8775 \tl_if_empty:NTF \l_@@_inclasssign_given_tl {
8776 \tl_if_empty:NF \l_@@_assign_given_tl {
8777 \hwexam@given@kw\xspace\l_@@_assign_given_tl
8778 }
8779 }{
8780 \hwexam@given@kw\xspace\l_@@_inclasssign_given_tl
8781 }
8782
8783 \bool_lazy_or:nnF {
8784 \bool_lazy_and_p:nn {
8785 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
8786 }{
8787 \tl_if_empty_p:V \l_@@_assign_due_tl
8788 }
8789 }{
8790 \bool_lazy_and_p:nn {
8791 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
8792 }{
8793 \tl_if_empty_p:V \l_@@_assign_due_tl
8794 }
8795 }{ ,~ }
8796
8797 \tl_if_empty:NTF \l_@@_inclasssign_due_tl {
8798 \tl_if_empty:NF \l_@@_assign_due_tl {
8799 \hwexam@due@kw\xspace \l_@@_assign_due_tl
8800 }
8801 }{
8802 \hwexam@due@kw\xspace \l_@@_inclasssign_due_tl
8803 }
8804
8805 \bool_lazy_all:nF {
8806 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
8807 { \tl_if_empty_p:V \l_@@_assign_given_tl }
8808 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
8809 { \tl_if_empty_p:V \l_@@_assign_due_tl }
8810 }{ #2 }
8811 }

```

`\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.

```

8812 \newcommand\assignment@title[3]{
8813 \tl_if_empty:NTF \l_@@_inclasssign_title_tl {
8814 \tl_if_empty:NTF \l_@@_assign_title_tl {
8815 #1
8816 }{
8817 #2\l_@@_assign_title_tl#3
8818 }
8819 }{
8820 #2\l_@@_inclasssign_title_tl#3
8821 }
8822 }

```

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

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

```

8823 \newcommand\assignment@number{
8824 \int_compare:nNnTF \l_@@_inclasssign_number_int = {-1} {
8825 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8826 \arabic{assignment}
8827 } {
8828 \int_use:N \l_@@_assign_number_int
8829 }
8830 }{
8831 \int_use:N \l_@@_inclasssign_number_int
8832 }
8833 }

```

(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.

```

8834 \newenvironment{assignment}[1][]{
8835 \_@@_assignment_args:n { #1 }
8836 %\sref@target
8837 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8838 \global\stepcounter{assignment}
8839 }{
8840 \global\setcounter{assignment}{\int_use:N\l_@@_assign_number_int}
8841 }
8842 \setcounter{sproblem}{0}
8843 \renewcommand\prob@label[1]{\assignment@number.##1}
8844 \def\current@section@level{\document@hwexamtype}
8845 %\sref@label{id{\document@hwexamtype \thesection}
8846 \begin{@assignment}
8847 }{
8848 \end{@assignment}
8849 }

```

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

```

8850 \def\ass@title{
8851 {\protect\document@hwexamtype}\arabic{assignment}
8852 \assignment@title{}\;\;{}{}\; -- \given@due{}\;}
8853 }
8854 \ifmultiple
8855 \newenvironment{@assignment}{
8856 \bool_if:NTF \l_@@_assign_loadmodules_bool {
8857 \begin{sfragment}[loadmodules]{\ass@title}
8858 }{
8859 \begin{sfragment}{\ass@title}
8860 }
8861 }{
8862 \end{sfragment}
8863 }

```

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

```

8864 \else
8865 \newenvironment{@assignment}{
8866 \begin{center}\bf
8867 \Large@title\strut\
8868 \document@hwexamtype\arabic{assignment}\assignment@title{}\;\;{}{}\;
8869 \large\given@due{--\;\;{}{}\;--}
8870 \end{center}
8871 }{}
8872 \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.

```

8873 \keys_define:nn { hwexam / inclassignment } {
8874 %id .str_set_x:N = \l_@@_assign_id_str,
8875 number .int_set:N = \l_@@_inclassign_number_int,
8876 title .tl_set:N = \l_@@_inclassign_title_tl,
8877 type .tl_set:N = \l_@@_inclassign_type_tl,
8878 given .tl_set:N = \l_@@_inclassign_given_tl,
8879 due .tl_set:N = \l_@@_inclassign_due_tl,
8880 mhrepos .str_set_x:N = \l_@@_inclassign_mhrepos_str
8881 }
8882 \cs_new_protected:Nn \_@@_inclassignment_args:n {
8883 \int_set:Nn \l_@@_inclassign_number_int {-1}
8884 \tl_clear:N \l_@@_inclassign_title_tl
8885 \tl_clear:N \l_@@_inclassign_type_tl
8886 \tl_clear:N \l_@@_inclassign_given_tl
8887 \tl_clear:N \l_@@_inclassign_due_tl
8888 \str_clear:N \l_@@_inclassign_mhrepos_str
8889 \keys_set:nn { hwexam / inclassignment }{ #1 }
8890 }
8891 \_@@_inclassignment_args:n {}
8892
8893 \newcommand\inputassignment[2][{}]{

```



```

8894 \_@@_inclassassignment_args:n { #1 }
8895 \str_if_empty:NTF \l_@@_inclasssign_mhrepos_str {
8896   \input{#2}
8897 }{
8898   \stex_in_repository:nn{\l_@@_inclasssign_mhrepos_str}{
8899     \input{\mhpath{\l_@@_inclasssign_mhrepos_str}{#2}}
8900   }
8901 }
8902 \_@@_inclassassignment_args:n {}
8903 }
8904 \newcommand\includeassignment[2][]{
8905   \newpage
8906   \inputassignment[#1]{#2}
8907 }

```

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

42.4 Typesetting Exams

\quizheading

```

8908 \ExplSyntaxOff
8909 \newcommand\quizheading[1]{%
8910   \def\@tas{#1}%
8911   \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
8912   \ifx\@tas\@empty\else%
8913     \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
8914   \fi%
8915 }
8916 \ExplSyntaxOn

```

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

\testheading

```

8917
8918 \def\hwexamheader{\input{hwexam-default.header}}
8919
8920 \def\hwexamminutes{
8921   \tl_if_empty:NTF \testheading@duration {
8922     {\testheading@min}~\hwexam@minutes@kw
8923   }{
8924     \testheading@duration
8925   }
8926 }
8927
8928 \keys_define:nn { hwexam / testheading } {
8929   min .tl_set:N = \testheading@min,
8930   duration .tl_set:N = \testheading@duration,
8931   reqpts .tl_set:N = \testheading@reqpts,
8932   tools .tl_set:N = \testheading@tools
8933 }
8934 \cs_new_protected:Nn \_@@_testheading_args:n {
8935   \tl_clear:N \testheading@min
8936   \tl_clear:N \testheading@duration

```

```

8937 \tl_clear:N \testheading@reqpts
8938 \tl_clear:N \testheading@tools
8939 \keys_set:nn { hwexam / testheading }{ #1 }
8940 }
8941 \newenvironment{testheading}[1][ ]{
8942 \_@@_testheading_args:n{ #1 }
8943 \newcount\check@time\check@time=\testheading@min
8944 \advance\check@time by -\theassignment@totalmin
8945 \newif\if@bonuspoints
8946 \tl_if_empty:NTF \testheading@reqpts {
8947 \@bonuspointsfalse
8948 }{
8949 \newcount\bonus@pts
8950 \bonus@pts=\theassignment@totalpts
8951 \advance\bonus@pts by -\testheading@reqpts
8952 \edef\bonus@pts{\the\bonus@pts}
8953 \@bonuspointstrue
8954 }
8955 \edef\check@time{\the\check@time}
8956
8957 \makeatletter\hwexamheader\makeatother
8958 }{
8959 \newpage
8960 }

```

(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.

```

8961 <@@=problems>
8962 \renewcommand\@problem[3]{
8963 \stepcounter{assignment@probs}
8964 \def\__problemspts{#2}
8965 \ifx\__problemspts\empty\else
8966 \addtocounter{assignment@totalpts}{#2}
8967 \fi
8968 \def\__problemsmin{#3}\ifx\__problemsmin\empty\else\addtocounter{assignment@totalmin}{#3}\fi
8969 \xdef\correction@probs{\correction@probs & #1}%
8970 \xdef\correction@pts{\correction@pts & #2}
8971 \xdef\correction@reached{\correction@reached &}
8972 }
8973 <@@=hwexam>

```

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

\correction@table This macro generates the correction table

```

8974 \newcounter{assignment@probs}
8975 \newcounter{assignment@totalpts}
8976 \newcounter{assignment@totalmin}
8977 \def\correction@probs{\correction@probs@kw}
8978 \def\correction@pts{\correction@pts@kw}
8979 \def\correction@reached{\correction@reached@kw}
8980 \stepcounter{assignment@probs}
8981 \newcommand\correction@table{

```

```

8982 \resizebox{\textwidth}{!}{%
8983 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
8984 &\multicolumn{\theassignment@probs}{c|}{%|
8985 {\footnotesize\correction@forgrading@kw} &\\ \hline
8986 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
8987 \correction@pts & \theassignment@totalpts & \\ \hline
8988 \correction@reached & & \\ [.7cm] \hline
8989 \end{tabular}}
8990 \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).