

The sTeX3 Package Collection *

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

2022-09-07

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 2022-09-07)

Contents

I	Manual	1
1	What is sTeX?	2
2	Quickstart	3
2.1	Setup	3
2.1.1	Minimal Setup for the PDF-only Workflow	3
2.1.2	GIT-based Setup for the sTeX Development Version	3
2.1.3	sTeX Archives (Manual Setup)	4
2.1.4	The sTeX IDE	4
2.1.5	Manual Setup for Active Documents and Knowledge Management Services	4
2.2	A First sTeX Document	5
2.2.1	OMDoc/xhtml Conversion	8
2.2.2	MMT/OMDoc Conversion	9
3	Creating sTeX Content	10
3.1	How Knowledge is Organized in sTeX	10
3.2	sTeX Archives	11
3.2.1	The Local MathHub-Directory	11
3.2.2	The Structure of sTeX Archives	12
3.2.3	MANIFEST.MF-Files	12
3.2.4	Using Files in sTeX Archives Directly	13
3.3	Module, Symbol and Notation Declarations	14
3.3.1	The smodule-Environment	14
3.3.2	Declaring New Symbols and Notations	16
	Operator Notations	20
3.3.3	Argument Modes	20
	Mode-b Arguments	20
	Mode-a Arguments	21
	Mode-B Arguments	22
3.3.4	Type and Definiens Components	23
3.3.5	Precedences and Automated Bracketing	24
3.3.6	Variables	26
3.3.7	Variable Sequences	27
3.4	Module Inheritance and Structures	29
3.4.1	Multilinguality and Translations	29
3.4.2	Simple Inheritance and Namespaces	30
3.4.3	The mathstructure Environment	32
3.4.4	The copymodule Environment	35
3.4.5	The interpretmodule Environment	36
3.5	Primitive Symbols (The sTeX Metatheory)	37
4	Using sTeX Symbols	38
4.1	\symref and its variants	38
4.2	Marking Up Text and On-the-Fly Notations	39

5	<code>TeX</code> Statements	43
5.1	Definitions, Theorems, Examples, Paragraphs	43
5.2	Proofs	46
5.3	Highlighting and Presentation Customizations	51
6	Cross References	53
7	Additional Packages	55
7.1	Tikzinput: Treating TIKZ code as images	55
7.2	Modular Document Structuring	56
7.2.1	Introduction	56
7.2.2	Package Options	56
7.2.3	Document Fragments	56
7.2.4	Ending Documents Prematurely	58
7.2.5	Global Document Variables	58
7.3	Slides and Course Notes	58
7.3.1	Introduction	58
7.3.2	Package Options	59
7.3.3	Notes and Slides	59
7.3.4	Customizing Header and Footer Lines	60
7.3.5	Frame Images	61
7.3.6	Excursions	62
7.4	Representing Problems and Solutions	63
7.4.1	Introduction	63
7.4.2	Problems and Solutions	63
7.4.3	Markup for Added-Value Services	65
	Multiple Choice Blocks	65
	Filling-In Concrete Solutions	66
7.4.4	Including Problems	67
7.4.5	Testing and Spacing	68
7.5	Homeworks, Quizzes and Exams	68
7.5.1	Introduction	68
7.5.2	Package Options	68
7.5.3	Assignments	69
7.5.4	Including Assignments	69
7.5.5	Typesetting Exams	69
II	Documentation	71
8	<code>TeX</code>-Basics	72
8.1	Macros and Environments	72
8.1.1	HTML Annotations	72
8.1.2	Babel Languages	73
8.1.3	Auxiliary Methods	73

9	sTeX-MathHub	74
9.1	Macros and Environments	74
9.1.1	Files, Paths, URIs	74
9.1.2	MathHub Archives	75
9.1.3	Using Content in Archives	76
10	sTeX-References	77
10.1	Macros and Environments	77
10.1.1	Setting Reference Targets	77
10.1.2	Using References	78
11	sTeX-Modules	79
11.1	Macros and Environments	79
11.1.1	The <code>smodule</code> environment	81
12	sTeX-Module Inheritance	83
12.1	Macros and Environments	83
12.1.1	SMS Mode	83
12.1.2	Imports and Inheritance	84
13	sTeX-Symbols	86
13.1	Macros and Environments	86
14	sTeX-Terms	88
14.1	Macros and Environments	88
15	sTeX-Structural Features	90
15.1	Macros and Environments	90
15.1.1	Structures	90
16	sTeX-Statements	91
16.1	Macros and Environments	91
17	sTeX-Proofs: Structural Markup for Proofs	92
18	sTeX-Metatheory	93
18.1	Symbols	93
III	Extensions	94
19	Tikzinput: Treating TIKZ code as images	95
19.1	Macros and Environments	95
20	document-structure: Semantic Markup for Open Mathematical Documents in L^AT_EX	96
21	NotesSlides – Slides and Course Notes	97
22	problem.sty: An Infrastructure for formatting Problems	98

23	hwexam.sty/cls: An Infrastructure for formatting Assignments and Exams	99
IV	Implementation	100
24	STEX-Basics Implementation	101
24.1	The STEXDocument Class	101
24.2	Preliminaries	102
24.3	Messages and logging	102
24.4	HTML Annotations	103
24.5	Babel Languages	105
24.6	Persistence	107
24.7	Auxiliary Methods	107
25	STEX-MathHub Implementation	111
25.1	Generic Path Handling	111
25.2	PWD and kpsewhich	113
25.3	File Hooks and Tracking	114
25.4	MathHub Repositories	115
25.5	Using Content in Archives	120
26	STEX-References Implementation	125
26.1	Document URIs and URLs	125
26.2	Setting Reference Targets	127
26.3	Using References	130
27	STEX-Modules Implementation	136
27.1	The smodule environment	140
27.2	Invoking modules	146
28	STEX-Module Inheritance Implementation	148
28.1	SMS Mode	148
28.2	Inheritance	152
29	STEX-Symbols Implementation	158
29.1	Symbol Declarations	158
29.2	Notations	166
29.3	Variables	176
30	STEX-Terms Implementation	185
30.1	Symbol Invocations	185
30.2	Terms	193
30.3	Notation Components	198
30.4	Variables	200
30.5	Sequences	203
31	STEX-Structural Features Implementation	204
31.1	Imports with modification	205
31.2	The feature environment	213
31.3	Structure	213

32	sTeX-Statements Implementation	224
32.1	Definitions	224
32.2	Assertions	230
32.3	Examples	233
32.4	Logical Paragraphs	236
33	The Implementation	241
33.1	Proofs	241
34	sTeX-Others Implementation	250
35	sTeX-Metatheory Implementation	252
36	Tikzinput Implementation	255
37	document-structure.sty Implementation	258
37.1	Package Options	258
37.2	Document Structure	259
37.3	Front and Backmatter	263
37.4	Global Variables	265
38	NotesSlides – Implementation	266
38.1	Class and Package Options	266
38.2	Notes and Slides	268
38.3	Header and Footer Lines	272
38.4	Frame Images	274
38.5	Sectioning	275
38.6	Excursions	278
39	The Implementation	280
39.1	Package Options	280
39.2	Problems and Solutions	281
39.3	Markup for Added Value Services	288
39.4	Multiple Choice Blocks	288
39.5	Filling in Concrete Solutions	289
39.6	Including Problems	290
39.7	Reporting Metadata	291
39.8	Testing and Spacing	292
40	Implementation: The hwexam Package	294
40.1	Package Options	294
40.2	Assignments	295
40.3	Including Assignments	298
40.4	Typesetting Exams	299
40.5	Leftovers	301
41	References	302

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 `TeX` 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_{TE}X [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_{TE}X system already.

Chapter 2

Quickstart

2.1 Setup

There are two ways of using $\text{\texttt{sTeX}}$: as a

1. way of writing $\text{\texttt{L}^A\text{\texttt{T}}_E\text{\texttt{X}}$ 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 (and experimental) toolchain of knowledge management systems. Both workflows profit from an integrated development environment (IDE), which (also) automates setup as far as possible (see [subsection 2.1.4](#)).

2.1.1 Minimal Setup for the PDF-only Workflow

In the best of all worlds, there is no setup, as you already have a new version of $\text{\texttt{T}}_E\text{\texttt{X}}\text{\texttt{Live}}$ on your system as a $\text{\texttt{L}^A\text{\texttt{T}}_E\text{\texttt{X}}$ enthusiast. If not now is the time to install it; see [\[TL\]](#). You can usually update $\text{\texttt{T}}_E\text{\texttt{X}}\text{\texttt{Live}}$ via a package manager or the $\text{\texttt{T}}_E\text{\texttt{X}}\text{\texttt{Live}}$ manager **tlmgr**.

Alternatively, you can install $\text{\texttt{sTeX}}$ from CTAN, the Comprehensive $\text{\texttt{T}}_E\text{\texttt{X}}$ Archive Network; see [\[ST\]](#) for details.

2.1.2 GIT-based Setup for the $\text{\texttt{sTeX}}$ Development Version

If you want use the latest and greatest $\text{\texttt{sTeX}}$ packages that have not even been released to CTAN, then you can directly clone them from the $\text{\texttt{sTeX}}$ 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 $\text{\texttt{sTeX}}$ directory. Make sure to either clone the $\text{\texttt{sTeX}}$ 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 sTeX Archives (Manual Setup)

Writing semantically annotated sTeX becomes much easier, if we can use well-designed libraries of already annotated content. sTeX provides such libraries as sTeX 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 sTeX archive as an **archive path** `<apath>` and a name `<archive>`. We can clone the sTeX archive by the following command-line instructions:

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

Note that sTeX 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 sTeX too knows where to find its archives, we need to set a global system variable `MATHHUB`, that points to your local MathHub-directory (see [section 3.2](#)).

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

2.1.4 The sTeX IDE

We are currently working on an sTeX IDE as an sTeX plugin for VScode; see [\[S1a\]](#). It will feature a setup procedure that automates the setup described above (and below). For additional functionality see the (now obsolete) plugin for sTeX 1 [\[SLS; Stb\]](#).

2.1.5 Manual Setup for Active Documents and Knowledge Management Services

Foregoing on the sTeX 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 sTeX/MMT content archives.

- **sTeX Archives** If we only care about L^AT_EX 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) sTeX archives are cloned as well.

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

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

2.2 A First \LaTeX Document

Having set everything up, we can write a first \LaTeX 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     \[\defeq{\geometricSeries}{\definiens{
14       \infinitiesum{\svar{n}}{1}{
15         \realdivide[frac]{1}{
16           \realpower{2}{\svar{n}}
17         }
18       }}
19     \].\]
20   \end{sdefinition}
21   \begin{sassertion}[name=geometricSeriesConverges,type=theorem]
22     The \symname{geometricSeries} \symname{converges} towards $1$.
23   \end{sassertion}
24 \end{smodule}
25 \end{document}

```

Compiling this document with `pdflatex` should yield the output

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

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

Theorem 0.2. The **geometric series converges** towards 1.

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

Remark 2.2.1:

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 ??.

Let's investigate this document in detail to understand the respective parts of the \LaTeX 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[snglom/calculus]{series}
\importmodule[snglom/arithmetics]{realarith}
```

Next, we *import* two modules – `series` from the \TeX archive `snglom/calculus`, and `realarith` from the \TeX archive `snglom/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. `\infinitiesum`, `\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}` ...
`\defineend` The `sdefinition`-environment provides two additional macros, `\define` and `\defineend` which behave similarly to `\symname` and `\symref`, but explicitly mark the symbols as *being defined* in this environment, to allow for special highlighting.

```

\[\defeq{\geometricSeries}{\definien{
  \infinisum{\svar{n}}{1}{
    \realdivide[frac]{1}{
      \realpower{2}{\svar{n}}
    }
  }}
}\].\]

```

The next snippet – set in a math environment – uses several semantic macros imported from (or recursively via) `series` and `realarithmetics`, such as `\defeq`, `\infinisum`, 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.

2.2.1 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 `Ru\TeX` [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">
            <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?infinitesum"/>
  <OMV name="n"/>
```

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

```

<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 2.2.2:

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

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

2.2.2 Mmt/OMDoc Conversion

Another way to convert our document to *actual* MMT/OMDOC is to put it in an `TEX` archive (see ??) 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 3

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.

3.1 How Knowledge is Organized in sTeX

sTeX content is organized on multiple levels:

1. sTeX **archives** (see [section 3.2](#)) contain individual `.tex`-files.
2. These may contain sTeX **modules**, introduced via `\begin{smodule}{ModuleName}`.

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

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

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

3.2 $\text{\S}\text{\TeX}$ Archives

3.2.1 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.

3.2.2 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.

3.2.3 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`.

3.2.4 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 3.2.1:

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.

3.3 Module, Symbol and Notation Declarations

3.3.1 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)

```

3.3.2 Declaring New Symbols and Notations

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

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

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

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

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

Example 3

Input:

```

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

```

Output:

```

Given a foo, we can...

```

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

Example 4

Input:

```

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

```

Output:

```

this is a symbol taking two arguments.

```

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

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

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

Example 5

Input:

```

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

```

Output:

```

First: a; Second: b

```

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

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

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

Example 6

Input:

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

Output:

First: a ; Second: b



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



Note that it is required that

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

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

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

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

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

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

Example 7

Input:

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

Output:

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

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

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

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

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

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

Operator Notations

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

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

Example 8

Input:

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

Output:

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

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

3.3.3 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:

$\forall x,y,z.P$

3.3.4 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 $x \mapsto x+1$

3.3.5 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).g

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.

3.3.6 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;\neginfpref
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

3.3.7 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:

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

3.4 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.

3.4.1 Multilinguality and Translations

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

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

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

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

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

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

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

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

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

3.4.2 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, \TeX 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 \TeX 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`.

3.4.3 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  $\varMb := \mathsf{mathstruct}\{\varMb\{universe\}, \varMb\{op\}!, \varMb\{unit\}\}$ 
4 be a  $\mathsf{symname}\{monoid\}$  on  $\mathbb{Z}$  ...

```

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.

3.4.4 The copymodule Environment

TODO: explain

Given modules:

Example 29

Input:

```

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

```

Output:

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

Example 30

Input:

```

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

```

Output:

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

TODO: explain donotclone

3.4.5 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:

3.5 Primitive Symbols (The STEX Metatheory)

The `stex-metatheory` package contains 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 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 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 STEX collection due to the obiquity of the symbols involved. Note however, that the metatheory is for all practical purposes a “normal” STEX module, and the symbols contained “normal” STEX symbols.

Chapter 4

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?

4.1 `\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 `\symname`, 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 `\symname` (or if a symbol was introduced using `\symdecl*` without generating a semantic macro), we might prefer to use the *name* of a symbol directly for readability – e.g. we would want to write A `\symname{natural-number}` is... rather than A `\symname{Nat}` is.... \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 `\symname{Integers?addition}` or `\symname{RealNumbers?addition}` in the case where several `additions` are in scope.

4.2 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 7.2.1](#)) 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 5

STEX Statements

5.1 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 ?? 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 ??), `type=` for customization (see ??) 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 5.1.1 (An Example). $2+3$ is 5, $2\cdot 3$ is 6.

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

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

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

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

Example 39

Input:

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

```

1 \begin{mathstructure}{monoid}
2   \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 5.1.2 (Associativity). \circ is associative

Axiom 5.1.3 (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

5.2 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 5.2.1. $\sqrt{2}$ 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 5.2.2. $\sqrt{2}$ 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.

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

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

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

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

5.3 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*`.

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

All of these commands take one optional and two proper arguments, i.e.

`\stexpatch* [<type>] {<begin-code>} {<end-code>}`.

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

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

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

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

```

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

```

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

```

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

```

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

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

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

Chapter 6

Cross References

If we take features like `\inputref` and `\mhinput` (and the `sfragment`-environment, see [subsection 7.2.1](#)) seriously, and build large documents modularly from individually compiling documents for sections, chapters and so on, cross-referencing becomes an interesting problem.

Say, we have a document `main.tex`, which `\inputrefs` a section `section1.tex`, which references a definition with label `some_definition` in `section2.tex` (subsequently also inputted in `main.tex`). Then the numbering of the definition will depend on the *document context* in which the document fragment `section2.tex` occurs - in `section2.tex` itself (as a standalone document), it might be *Definition 1*, in `main.tex` it might be *Definition 3.1*, and in `section1.tex`, the definition *does not even occur*, so it needs to be referenced by some other text.

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

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

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

- The *label* of the reference target (e.g. `some_definition`),
- (optionally) the *file*/document containing the reference target (e.g. `section2`). This is not strictly necessary, but allows for additional disambiguation between possibly duplicate labels across files, and
- (optionally) the document context, in which we want to refer to the reference target (e.g. `main`).

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

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

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

If the referenced object does *not* occur in the current document however, `\sref` will refer to it by the object's name as it occurs in the file *<document-context>* in *<archive2>*.

For example, the reference to the `sfragment`-environment above will appear as “subsection 7.2.1 (Introduction) in the `gTEX3` manual” if you are reading this in the package documentation for `stex-references` directly, but as a linked “subsection 7.2.1” in the full documentation or manual. This is achieved using

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

For a further example, the following:

Part III

will say “Part III” (and link accordingly) in the full documentation, and “Part III (Extensions) in the full `gTEX3` documentation” everywhere else. This is achieved using

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

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

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

Chapter 7

Additional Packages

7.1 Tikzinput: Treating TIKZ code as images

image The behavior of the `tikzinput` package is determined by whether the `image` option is given. If it is not, then the `tikz` package is loaded, all other options are passed on to it and `\tikzinput{<file>}` inputs the TIKZ file `<file>.tex`; if not, only the `graphicx` package is loaded and `\tikzinput{<file>}` loads an image file `<file>.<ext>` generated from `<file>.tex`.

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

```
1 \documentclass{standalone}
2 \usepackage{tikz}
3 \usetikzpackage{...}
4 \begin{document}
5   \begin{tikzpicture}
6     ...
7   \end{tikzpicture}
8 \end{document}
```

The `standalone` class is a minimal \LaTeX 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 \LaTeX over it separately, e.g. for generating an image file from it.

\tikzinput This is exactly where the `tikzinput` package comes in: it supplies the `\tikzinput` macro, which – depending on the `image` option – either directly inputs the TIKZ picture (source) or tries to load an image file generated from it.

Concretely, if the `image` option is not set for the `tikzinput` package, then `\tikzinput[<opt>]{<file>}` disregards the optional argument `<opt>` and inputs `<file>.tex` via `\input` and resizes it to as specified in the `width` and `height` keys. If it is, `\tikzinput[<opt>]{<file>}` expands to `\includegraphics[<opt>]{<file>}`.

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

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

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

7.2 Modular Document Structuring

7.2.1 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.

7.2.2 Package Options

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

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

7.2.3 Document Fragments

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

```

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

```

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

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

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

```

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

```

Here we use two levels of `blindfragment`:

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

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

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

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

7.2.4 Ending Documents Prematurely

<hr/> <code>\prematurestop</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.
<code>\afterprematurestop</code> <hr/>	

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

Text fragments and modules can be made more re-usable by the use of global variables. For instance, the admin section of a course can be made course-independent (and therefore re-usable) by using variables (actually token registers) `courseAcronym` and `courseTitle` instead of the text itself. The variables can then be set in the `TeX` preamble of the course notes file.

7.2.5 Global Document Variables

To make document fragments more reusable, we sometimes want to make the content depend on the context. We use **document variables** for that.

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

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

7.3 Slides and Course Notes

7.3.1 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.

7.3.2 Package Options

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

<code>slides</code>	The options <code>slides</code> and <code>notes</code> switch between slides mode and notes mode (see subsection 7.3.3).
<code>notes</code>	
<code>sectocframes</code>	If the option <code>sectocframes</code> is given, then for the <code>sfragments</code> , special frames with the <code>sfragment</code> title (and number) are generated.
<code>frameimages</code>	If the option <code>frameimages</code> is set, then slide mode also shows the <code>\frameimage</code> -generated frames (see ??). If also the <code>fiboxed</code> option is given, the slides are surrounded by a box.
<code>fiboxed</code>	

7.3.3 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 `\notesfalse` is strongly discouraged however.



We need to give the title frame the `noframenumbering` option so that the frame numbering is kept in sync between the slides and the course notes.



The `beamer` class recommends not to use the `allowframebreaks` option on frames (even though it is very convenient). This holds even more in the `notesslides` case: At least in conjunction with `\newpage`, frame numbering behaves funnily (we have tried to fix this, but who knows).

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

`nparagraph` (*env.*) There are some environments that tend to occur at the top-level of `note` environments.

`nparagraph` (*env.*) We make convenience versions of these: e.g. the `nparagraph` environment is just an

`ndefinition` (*env.*) `sparagraph` inside a `note` environment (but looks nicer in the source, since it avoids one

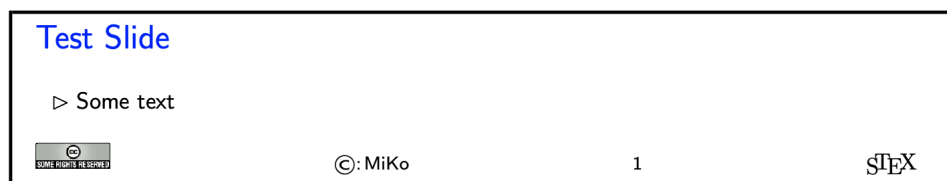
`nexample` (*env.*) level of source indenting). Similarly, we have the `nfragment`, `ndefinition`, `nexample`,

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

`nassertion` (*env.*)

7.3.4 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 \LaTeX 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.

7.3.5 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 \LaTeX 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: 

7.3.6 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.

7.4 Representing Problems and Solutions

7.4.1 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.

7.4.2 Problems and Solutions

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

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

`problem (env.)` The main environment provided by the `problempackage` is (surprise surprise) the `problem` environment. It is used to mark up problems and exercises. The environment takes an optional `KeyVal` argument with the keys `id` as an identifier that can be reference later, `pts` for the points to be gained from this exercise in homework or quiz situations, `min` for the estimated minutes needed to solve the problem, and finally `title` for an informative title of the problem.

Example 40

Input:

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

```

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

```

Output:

Problem 7.4.1 (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.

7.4.3 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 7.4.2 (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 7.4.3 (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 7.4.4 (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 7.4.5 (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

7.4.4 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.

7.4.5 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`

7.5 Homeworks, Quizzes and Exams

7.5.1 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.

7.5.2 Package Options

`solutions` The `hwexam` package and class take the options `solutions`, `notes`, `hints`, `gnotes`, `pts`, `min`, and `boxed` that are just passed on to the `problems` package (cf. its documentation for a description of the intended behavior).
`notes`
`hints`
`gnotes`
`pts`
`min`

`multiple` Furthermore, the `hwexam` package takes the option `multiple` that allows to combine multiple assignment sheets into a compound document (the assignment sheets are treated as section, there is a table of contents, etc.).

`test` Finally, there is the option `test` that modifies the behavior to facilitate formatting tests. Only in `test` mode, the macros `\testspace`, `\testnewpage`, and `\testemptypage` have an effect: they generate space for the students to solve the given problems. Thus they can be left in the \LaTeX source.

7.5.3 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).

7.5.4 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.

7.5.5 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)

2022-09-07

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.	7.4.1	7.4.2	7.4.3	7.4.4	7.4.5	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 8

STEX-Basics

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

8.1 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>*.

8.1.1 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 $\langle content \rangle$ 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> .

8.1.2 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`.

8.1.3 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 $\langle cs \rangle$ throw an error, indicating that it is only allowed in the context of $\langle environments \rangle$.

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

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

Chapter 9

STEX-MathHub

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

9.1 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.
--------------------------------	---

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

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

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

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

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

9.1.2 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}`.

9.1.3 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-</code> mode, 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 10

STEX-References

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

10.1 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>
---	--

10.1.1 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> .
---	---

10.1.2 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 11

sTeX-Modules

This sub package contains code related to Modules

11.1 Macros and Environments

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

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

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

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

<hr/> <u><code>\c_stex_module_<URI>_constants</code></u> <hr/>	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`.

11.1.1 The `smodule` environment

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Chapter 12

STEX-Module Inheritance

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

12.1 Macros and Environments

12.1.1 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.
--------------------------------	--

12.1.2 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 13

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

Code related to symbol declarations and notations

13.1 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 $\text{\texttt{S}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$, but passed on to MMT for semantic services.
- **def**: An (ideally semantic) term, representing a *definiens*. Not used by $\text{\texttt{S}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$, 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 $\text{\texttt{S}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$ like an **i**-argument, but an application is turned into an $\text{\texttt{O}}\text{\texttt{M}}\text{\texttt{B}}\text{\texttt{i}}\text{\texttt{n}}\text{\texttt{d}}$ 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>.</p> <p>Ultimately stores the symbol $\langle URI \rangle$ in the property list <code>\l_stex_symdecl_<URI>_prop</code> with fields:</p> <ul style="list-style-type: none"> • <code>name</code> (string), • <code>module</code> (string), • <code>notations</code> (sequence of strings; initially empty), • <code>type</code> (token list), • <code>args</code> (string of is, as and bs), • <code>arity</code> (integer string), • <code>assocs</code> (integer string; number of associative arguments),
<hr/> <hr/> <code>\stex_all_symbols:n</code>	<p>Iterates over all currently available symbols. Requires two <code>\seq_map_break:</code> to break fully.</p>
<hr/> <hr/> <code>\stex_get_symbol:n</code>	<p>Computes the full URI of a symbol from a macro argument, e.g. the macro name, the macro itself, the full URI...</p>
<hr/> <hr/> <code>\notation</code>	<p><code>\notation[<args>]{<symbol>}{<notations⁺>}</code></p> <p>Introduces a new notation for $\langle symbol \rangle$, see <code>\stex_notation_do:nn</code></p>
<hr/> <hr/> <code>\stex_notation_do:nn</code>	<p><code>\stex_notation_do:nn{<URI>}{<notations⁺>}</code></p> <p>Implements the core functionality of <code>\notation</code>, and is called by <code>\notation</code> and <code>\symdef</code>.</p> <p>Ultimately stores the notation in the property list <code>\g_stex_notation_<URI>#<variant>#<lang>_prop</code> with fields:</p> <ul style="list-style-type: none"> • <code>symbol</code> (URI string), • <code>language</code> (string), • <code>variant</code> (string), • <code>opprec</code> (integer string), • <code>argprec</code> (sequence of integer strings)
<hr/> <hr/> <code>\symdef</code>	<p><code>\symdef[<args>]{<symbol>}{<notations⁺>}</code></p> <p>Combines <code>\symdecl</code> and <code>\notation</code> by introducing a new symbol and assigning a new notation for it.</p>

Chapter 14

STEX-Terms

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

14.1 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>\infprec</code> <code>\neginfprec</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 15

TeX-Structural Features

Code related to structural features

15.1 Macros and Environments

15.1.1 Structures

`mathstructure` (*env.*) TODO

Chapter 16

TeX-Statements

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

16.1 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 17

sTEX-Proofs: Structural Markup for Proofs

Chapter 18

sTeX-Metatheory

18.1 Symbols

Part III
Extensions

Chapter 19

Tikzinput: Treating TIKZ code as images

19.1 Macros and Environments

Chapter 20

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

Chapter 21

NotesSlides – Slides and Course Notes

Chapter 22

`problem.sty`: An Infrastructure for formatting Problems

Chapter 23

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

Part IV
Implementation

Chapter 24

\TeX -Basics Implementation

24.1 The \TeX Document Class

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

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

```

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

```

24.2 Preliminaries

```

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

```

\stex The sTeX logo:

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

(End definition for \stex and \sTeX. These functions are documented on page 72.)

24.3 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 72.)

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}

```

24.4 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 72.)

`\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 72.)

`\stex_annotate_html:nnn`

`\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 73.)

24.5 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 73.)

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

```


24.6 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 }

```

24.7 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 73.)

`\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 73.)

`\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 73.)

\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 \STEXtitle #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 \STEXtitle \stex_document_title:n
372   \tl_if_empty:NF \g_stex_document_title {
373     \stex_annotate_invisible:n{\noindent
374       \stex_annotate:nnn{doctitle}{-}{ \g_stex_document_title }
375     \par}
376   }
377   \let \stex_maketitle:\maketitle
378   \def \maketitle{
379     \tl_if_empty:NF \@title {
380       \exp_args:No \stex_document_title:n \@title
381     }
382     \stex_maketitle:
383   }

```

```

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

```

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

Chapter 25

STEX -MathHub Implementation

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

25.1 Generic Path Handling

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

`\stex_path_from_string:Nn`

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

```

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

```

(End definition for `\stex_path_from_string:Nn`. This function is documented on page 74.)

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

```

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

```

(End definition for `\stex_path_to_string:NN` and `\stex_path_to_string:N`. These functions are documented on page 74.)

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

```

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

```

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

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

```

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

```

```

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

```

(End definition for `\stex_path_canonicalize:N`. This function is documented on page 74.)

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

```

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

```

(End definition for `\stex_path_if_absolute:N`. This function is documented on page 74.)

25.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

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

```

(End definition for `\stex_kpsewhich:n`. This function is documented on page 74.)

We determine the PWD

`\c_stex_pwd_seq`
`\c_stex_pwd_str`

```

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

```

(End definition for `\c_stex_pwd_seq` and `\c_stex_pwd_str`. These variables are documented on page 74.)

25.3 File Hooks and Tracking

516 `<@@=stex_files>`

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

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

`\g_stex_files_stack` keeps track of file changes

```

517 \seq_gclear_new:N\g_stex_files_stack

```

(End definition for `\g_stex_files_stack`.)

`\c_stex_mainfile_seq`
`\c_stex_mainfile_str`

```

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

```

(End definition for `\c_stex_mainfile_seq` and `\c_stex_mainfile_str`. These variables are documented on page 74.)

`\g_stex_currentfile_seq`

```

521 \seq_gclear_new:N\g_stex_currentfile_seq

```

(End definition for `\g_stex_currentfile_seq`. This variable is documented on page 75.)

`\stex_filestack_push:n`

```

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

```



```

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

```

(End definition for `\stex_filestack_push:n`. This function is documented on page 75.)

`\stex_filestack_pop:`

```

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

```

(End definition for `\stex_filestack_pop:`. This function is documented on page 75.)

Hooks for the current file:

```

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

```

25.4 MathHub Repositories

```

555 <@=stex_mathhub>

```

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

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

```

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

```

```

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

```

(End definition for `\mathhub`, `\c_stex_mathhub_seq`, and `\c_stex_mathhub_str`. These variables are documented on page 75.)

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

```

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

```

```

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

```

(End definition for __stex_mathhub_do_manifest:n.)

\l_stex_mathhub_manifest_file_seq

```

621 \seq_new:N\l__stex_mathhub_manifest_file_seq

```

(End definition for \l_stex_mathhub_manifest_file_seq.)

__stex_mathhub_find_manifest:N

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

```

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

```

(End definition for __stex_mathhub_find_manifest:N.)

\c_stex_mathhub_manifest_ior

File variable used for MANIFEST-files

```

657 \ior_new:N \c__stex_mathhub_manifest_ior

```

(End definition for \c__stex_mathhub_manifest_ior.)

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

```

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

```

(End definition for _stex_mathhub_parse_manifest:n.)

\stex_set_current_repository:n

```

704 \cs_new_protected:Nn \stex_set_current_repository:n {

```

```

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

```

(End definition for `\stex_set_current_repository:n`. This function is documented on page 75.)

`\stex_require_repository:n`

```

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

```

(End definition for `\stex_require_repository:n`. This function is documented on page 75.)

`\l_stex_current_repository_prop` Current MathHub repository

```

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

```

(End definition for `\l_stex_current_repository_prop`. This variable is documented on page 75.)

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

```

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

```

```

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

```

(End definition for `\stex_in_repository:nn`. This function is documented on page 75.)

25.5 Using Content in Archives

`\mhpath`

```

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

```

(End definition for `\mhpath`. This function is documented on page 76.)

`\inputref`

`\mhinput`

```

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

```

```

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

```

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

`\addmhbibresource`

```

838 \cs_new_protected:Nn \__stex_mathhub_mhbibresource:nn {

```

```

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

```

(End definition for \addmhbibresource. This function is documented on page 76.)

\libinput

```

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

```

(End definition for \libinput. This function is documented on page 76.)

\libusepackage

```

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

```



```

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

```

(End definition for `\libusepackage`. This function is documented on page 76.)

`\mhgraphics`
`\cmhgraphics`

```

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

```

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

`\lstinputmhlisting`
`\clstinputmhlisting`

```

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

```

933

934 `\end{package}`

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

Chapter 26

STEX -References Implementation

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

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

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

26.1 Document URIs and URLs

`\l_stex_current_docns_str`

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

(End definition for `\l_stex_current_docns_str`. This variable is documented on page 77.)

`\stex_get_document_uri:`

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

```

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

```

(End definition for `\stex_get_document_uri`:. This function is documented on page 77.)

`\l_stex_current_docurl_str`

```

998 \str_new:N \l_stex_current_docurl_str

```

(End definition for `\l_stex_current_docurl_str`. This variable is documented on page 77.)

`\stex_get_document_url:`

```

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

```

```

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

```

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

26.2 Setting Reference Targets

```

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

```

Restoring references from .sref-files

\STEXInternalSrefRestoreTarget

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

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

\stex_ref_new_doc_target:n

```

1054 \seq_new:N \g_stex_ref_files_seq
1055
1056 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
1057   %\stex_get_document_uri:
1058   \str_clear:N \l__stex_refs_curr_label_str
1059   \str_set:Nx \l_tmpa_str { #1 }
1060   \str_if_empty:NT \l_tmpa_str {
1061     \int_gincr:N \l__stex_refs_unnamed_counter_int
1062     \str_set:Nx \l_tmpa_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
1063   }
1064   \str_set:Nx \l__stex_refs_curr_label_str {
1065     \l_stex_current_docns_str?\l_tmpa_str
1066   }
1067
1068   \exp_args:Noo \STEXInternalAuxAddDocRef\l_stex_current_docns_str\l_tmpa_str
1069
1070   %\seq_if_exist:cF{g__stex_refs_labels_\l_tmpa_str_seq}{
1071   %   \seq_new:c {g__stex_refs_labels_\l_tmpa_str_seq}
1072   %}
1073   %\seq_if_in:coF{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str {
1074   %   \seq_gput_right:co{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str
1075   %}
1076
1077
1078   \stex_if_smsmode:TF {
1079     %\stex_get_document_url:
1080     %\str_gset_eq:cN {sref_url_\l__stex_refs_curr_label_str_str}\l_stex_current_docurl_str
1081     %\str_gset_eq:cN {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_url_str
1082   }{
1083     \iow_now:Nx \c__stex_refs_refs_iow {
1084       \STEXInternalSrefRestoreTarget
1085       {\l_stex_current_docns_str}
1086       {\l_tmpa_str}
1087       {\@currentcounter}
1088       {\@currentlabel}
1089       {\tl_if_exist:NT\@currentlabelname{\exp_args:No\unexpanded\@currentlabelname}}
1090     }
1091     %\iow_now:Nx \c__stex_refs_refs_iow {
1092     %   {\l_stex_current_docns_str?\l_tmpa_str}~==~{\use:c{\@currentcounter autorefname}~\@currentlabelname}
1093     \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
1094     \immediate\write\@auxout{\STEXInternalAuxAddDocRef{\l_stex_current_docns_str}{\l_tmpa_str}}
1095     %\str_gset:cx {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_ref_str
1096   }
1097 }
1098 \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 77.)

The following is used to set the necessary macros in the .aux-file.

```

1099 \cs_new_protected:Npn \STEXInternalAuxAddDocRef #1 #2 {
1100   \exp_args:Nnx \seq_if_in:NnTF \g_stex_ref_files_seq {\detokenize{#1}} {
1101     \exp_args:Nnx \seq_if_in:cnF{g_stex_ref_ #1 _seq}{\detokenize{#2}}{
1102       \exp_args:Nnx \seq_gput_left:cn{g_stex_ref_ #1 _seq}{\detokenize{#2}}
1103     }
1104   }{
1105     \exp_args:Nnx \seq_gput_right:Nn \g_stex_ref_files_seq {\detokenize{#1}}
1106     %\seq_if_exist:cF{g_stex_ref_ #1 _seq}{
1107       \seq_new:c{g_stex_ref_ #1 _seq} % <- seq_new throws errors??
1108     %}
1109     \exp_args:Nnx \seq_gput_left:cn{g_stex_ref_ #1 _seq}{\detokenize{#2}}
1110   }
1111
1112   %\str_set:Nn \l_tmpa_str {#1?#2}
1113   %\str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
1114   %\seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
1115     % \seq_new:c {g__stex_refs_labels_#2_seq}
1116     %}
1117   %\seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
1118     % \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
1119     %}
1120 }

```

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

```

1121 \AtEndDocument{
1122   \def\STEXInternalAuxAddDocRef#1 #2 {}{}
1123 }

```

\stex_ref_new_sym_target:n

```

1124 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
1125
1126   % \stex_if_smsmode:TF {
1127   %   \str_if_exist:cF{sref_sym_#1_type}{
1128   %     \stex_get_document_url:
1129   %     \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str
1130   %     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
1131   %   }
1132   % }{
1133   %   \str_if_empty:NF \l__stex_refs_curr_label_str {
1134   %     \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
1135   %     \immediate\write\@auxout{
1136   %       \exp_not:N\expandafter\def\exp_not:N\csname \exp_not:N\detokenize{sref_sym_#1_label}
1137   %       \l__stex_refs_curr_label_str
1138   %     }
1139   %   }
1140   % }
1141 % }
1142 }

```

(End definition for \stex_ref_new_sym_target:n. This function is documented on page 77.)

26.3 Using References

`\sref` Optional arguments:

```

1143
1144 \keys_define:nn { stex / sref / 1 } {
1145   archive .str_set_x:N = \l__stex_refs_repo_str,
1146   file     .str_set_x:N = \l__stex_refs_file_str,
1147   % TODO get rid of this
1148   fallback .code:n = {},
1149   pre      .code:n = {},
1150   post     .code:n = {}
1151 }
1152 \cs_new_protected:Nn \__stex_refs_args_i:n {
1153   \str_clear:N \l__stex_refs_repo_str
1154   \str_clear:N \l__stex_refs_file_str
1155   \keys_set:nn { stex / sref / 1 } { #1 }
1156 }
1157 \keys_define:nn { stex / sref / 2 } {
1158   in       .str_set_x:N = \l__stex_refs_in_str,
1159   archive  .str_set_x:N = \l__stex_refs_repob_str,
1160   title    .tl_set:N = \l__stex_refs_title_tl
1161 }
1162 \cs_new_protected:Nn \__stex_refs_args_ii:n {
1163   \str_clear:N \l__stex_refs_in_str
1164   \tl_clear:N \l__stex_refs_title_tl
1165   \str_clear:N \l__stex_refs_repob_str
1166   \keys_set:nn { stex / sref / 2 } { #1 }
1167 }

```

The actual macro:

```

1168 \NewDocumentCommand \sref { 0{} m 0{} }{
1169   \__stex_refs_args_i:n{#1}
1170   \__stex_refs_args_ii:n{#3}
1171   \str_clear:N \l__stex_refs_uri_str
1172   \__stex_refs_find_uri:n{#2}
1173   \__stex_refs_do_sref:n{#2}
1174 }
1175 \NewDocumentCommand \extref { 0{} m m }{
1176   \__stex_refs_args_i:n{#1}
1177   \__stex_refs_args_ii:n{#3}
1178   \str_if_empty:NT \l__stex_refs_in_str {
1179     \msg_error:nn{stex}{error/extrefmissing}
1180   }
1181   \str_clear:N \l__stex_refs_uri_str
1182   \__stex_refs_find_uri:n{#2}
1183   \__stex_refs_do_sref_in:n{#2}
1184 }
1185
1186 \cs_new_protected:Nn \__stex_refs_find_uri:n {
1187   \stex_debug:nn{sref}{File: \l__stex_refs_file_str^^JRepo:\l__stex_refs_repo_str}
1188   \str_if_empty:NTF \l__stex_refs_file_str {
1189     \seq_if_exist:cT{g_stex_ref_\l_stex_current_docns_str_seq}{
1190       \seq_map_inline:cn{g_stex_ref_\l_stex_current_docns_str_seq}{
1191         \str_if_eq:nnT{#1}{##1}{

```



```

1192         \str_set_eq:NN \l__stex_refs_uri_str \l_stex_current_docns_str
1193         \seq_map_break:
1194     }
1195 }
1196 }
1197 \str_if_empty:NF \l__stex_refs_uri_str {
1198     \seq_map_inline:Nn \g_stex_ref_files_seq {
1199         \seq_map_inline:cn{g_stex_ref_###1_seq}{
1200             \str_if_eq:nnT{#1}{###1}{
1201                 \str_set:Nn \l__stex_refs_uri_str {##1}
1202                 \seq_map_break:n{\seq_map_break:}
1203             }
1204         }
1205     }
1206 }
1207 }{
1208     \str_if_empty:NTF \l__stex_refs_repo_str {
1209         \prop_if_exist:NTF \l_stex_current_repository_prop {
1210             \prop_get:NnN \l_stex_current_repository_prop { ns } \l__stex_refs_uri_str
1211             \str_set:Nx \l__stex_refs_uri_str {\l__stex_refs_uri_str / \l__stex_refs_file_str}
1212             \stex_path_from_string:Nn \l_tmpb_seq \l__stex_refs_uri_str
1213             \str_set:Nx \l__stex_refs_uri_str {\stex_path_to_string:N \l_tmpb_seq}
1214         }{
1215             \stex_path_from_string:Nn \l_tmpb_seq {
1216                 \stex_path_to_string:N \g_stex_currentfile_seq/ .. / \l__stex_refs_file_str
1217             }
1218             \str_set:Nx \l__stex_refs_uri_str {file:/\stex_path_to_string:N \l_tmpb_seq}
1219         }
1220     }{
1221         \stex_require_repository:n \l__stex_refs_repo_str
1222         \prop_get:cnN { c_stex_mathhub_\l__stex_refs_repo_str _manifest_prop } { ns } \l__stex_refs_uri_str
1223         \str_set:Nx \l__stex_refs_uri_str {\l__stex_refs_uri_str / \l__stex_refs_file_str}
1224         \stex_path_from_string:Nn \l_tmpb_seq \l__stex_refs_uri_str
1225         \str_set:Nx \l__stex_refs_uri_str {\stex_path_to_string:N \l_tmpb_seq}
1226     }
1227 }
1228 }
1229
1230 \cs_new_protected:Nn \__stex_refs_do_autoref:n{
1231     \cs_if_exist:cTF{autoref}{
1232         \exp_args:Nx\autoref{sref_#1}
1233     }{
1234         \exp_args:Nx\ref{sref_#1}
1235     }
1236 }
1237
1238 \cs_new_protected:Nn \__stex_refs_do_sref:n {
1239     \str_if_empty:NTF \l__stex_refs_uri_str {
1240         \str_if_empty:NTF \l__stex_refs_in_str {
1241             \__stex_refs_do_autoref:n{#1}
1242         }{
1243             \__stex_refs_do_sref_in:n{#1}
1244         }
1245     }{

```

```

1246 \exp_args:NNo \seq_if_in:NnTF \g_stex_ref_files_seq \l__stex_refs_uri_str {
1247 \exp_args:Nnx \seq_if_in:cnTF{\g_stex_ref_\l__stex_refs_uri_str _seq}{\detokenize{#1}}{
1248 \__stex_refs_do_autoref:n{\l__stex_refs_uri_str?#1}
1249 }{
1250 \str_if_empty:NTF \l__stex_refs_in_str {
1251 \__stex_refs_do_autoref:n{#1}
1252 }{
1253 \__stex_refs_do_sref_in:n{#1}
1254 }
1255 }
1256 }{
1257 \str_if_empty:NTF \l__stex_refs_in_str {
1258 \__stex_refs_do_autoref:n{#1}
1259 }{
1260 \__stex_refs_do_sref_in:n{#1}
1261 }
1262 }
1263 }
1264 }
1265
1266 \cs_new_protected:Nn \__stex_refs_restore_target:nnnnn {
1267 \str_if_empty:NTF \l__stex_refs_uri_str {
1268 \exp_args:No \str_if_eq:nnT \l__stex_refs_id_str {#2}{
1269 \tl_set:Nn \l__stex_refs_return_tl {
1270 \use:c{#3autorefname}~#4\tl_if_empty:nF{#5}{~(5)}~in~
1271 \tl_if_empty:nTF\l__stex_refs_title_tl{
1272 ???
1273 }\l__stex_refs_title_tl
1274 }
1275 }
1276 }{
1277 \stex_debug:nn{sref}{\l__stex_refs_uri_str}{~ == ~ #1 ~ ?}
1278 \exp_args:No \str_if_eq:nnT \l__stex_refs_uri_str {#1}{
1279 \stex_debug:nn{sref}{\l__stex_refs_id_str~ == ~ #2 ~ ?}
1280 \exp_args:No \str_if_eq:nnT \l__stex_refs_id_str {#2}{
1281 \stex_debug:nn{sref}{success!}
1282 \tl_set:Nn \l__stex_refs_return_tl {
1283 \use:c{#3autorefname}~#4\tl_if_empty:nF{#5}{~(5)}~in~
1284 \tl_if_empty:nTF\l__stex_refs_title_tl{
1285 ???
1286 }\l__stex_refs_title_tl
1287 }
1288 \endinput
1289 }
1290 }
1291 }
1292 }
1293
1294 \cs_new_protected:Nn \__stex_refs_do_sref_in:n {
1295 \stex_debug:nn{sref}{In: \l__stex_refs_in_str^^JRepo:\l__stex_refs_repo_str}
1296 \stex_debug:nn{sref}{URI: \l__stex_refs_uri_str?#1}
1297 %\msg_warning:nnn{stex}{warning/smsmissing}{<filename>}
1298 \begingroup\catcode13=9\relax\catcode10=9\relax
1299 \str_if_empty:NTF \l__stex_refs_repob_str {

```

```

1300 \prop_if_exist:NTF \l_stex_current_repository_prop {
1301   \str_set:Nx \l_tmpa_str {
1302     \c_stex_mathhub_str /
1303     \prop_item:Nn \l_stex_current_repository_prop { id }
1304     / source / \l__stex_refs_in_str .sref
1305   }
1306 }{
1307   \str_set:Nx \l_tmpa_str {
1308     \stex_path_to_string:N \g_stex_currentfile_seq/ .. / \l__stex_refs_in_str . sref
1309   }
1310 }
1311 }{
1312   \str_set:Nx \l_tmpa_str {
1313     \c_stex_mathhub_str / \l__stex_refs_repob_str
1314     / source / \l__stex_refs_in_str . sref
1315   }
1316 }
1317 \stex_path_from_string:Nn \l_tmpb_seq \l_tmpa_str
1318 \stex_path_to_string:NN \l_tmpb_seq \l_tmpa_str
1319 \stex_debug:nn{sref}{File: \l_tmpa_str}
1320 \exp_args:No \IfFileExists \l_tmpa_str {
1321   \tl_clear:N \l__stex_refs_return_tl
1322   \str_set:Nn \l__stex_refs_id_str {#1}
1323   \let\STEXInternalSrefRestoreTarget\__stex_refs_restore_target:nnnnn
1324   \use:c{@ @ input}{\l_tmpa_str}
1325   \exp_args:No \tl_if_empty:nTF \l__stex_refs_return_tl {
1326     \exp_args:Nnno \msg_warning:nnnn{stex}{warning/smslabelmissing}\l_tmpa_str{#1}
1327     \__stex_refs_do_autoref:n{
1328       \str_if_empty:NF\l__stex_refs_uri_str{\l__stex_refs_uri_str?}#1
1329     }
1330   }{
1331     \l__stex_refs_return_tl
1332   }
1333 }{
1334   \exp_args:Nnno \msg_warning:nnnn{stex}{warning/smsmissing}\l_tmpa_str
1335   \__stex_refs_do_autoref:n{
1336     \str_if_empty:NF\l__stex_refs_uri_str{\l__stex_refs_uri_str?}#1
1337   }
1338 }
1339 \endgroup
1340 }
1341
1342 % \__stex_refs_args:n { #1 }
1343 % \str_if_empty:NTF \l__stex_refs_indocument_str {
1344 %   \str_set:Nx \l_tmpa_str { #2 }
1345 %   \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
1346 %   \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
1347 %     \seq_if_exist:CTF{g__stex_refs_labels_\l_tmpa_str_seq}{
1348 %       \seq_get_left:cnF {g__stex_refs_labels_\l_tmpa_str_seq} \l_tmpa_str {
1349 %         \str_clear:N \l_tmpa_str
1350 %       }
1351 %     }{
1352 %       \str_clear:N \l_tmpa_str
1353 %     }

```

```

1354 %   }{
1355 %       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1356 %       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1357 %       \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
1358 %       \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str_seq}{
1359 %           \str_set_eq:NN \l_tmpc_str \l_tmpa_str
1360 %           \str_clear:N \l_tmpa_str
1361 %           \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str_seq} {
1362 %               \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
1363 %                   \str_range:nnn { ##1 }{-\l_tmpa_int}{ -1 }
1364 %               }{
1365 %                   \seq_map_break:n {
1366 %                       \str_set:Nn \l_tmpa_str { ##1 }
1367 %                   }
1368 %               }
1369 %           }
1370 %       }{
1371 %           \str_clear:N \l_tmpa_str
1372 %       }
1373 %   }
1374 %   \str_if_empty:NTF \l_tmpa_str {
1375 %       \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_li
1376 %   }{
1377 %       \str_if_eq:cNTF {sref_\l_tmpa_str_type} \c__stex_refs_ref_str {
1378 %           \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1379 %               \cs_if_exist:cTF{autoref}{
1380 %                   \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1381 %               }{
1382 %                   \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1383 %               }
1384 %           }{
1385 %               \ltx@ifpackageloaded{hyperref}{
1386 %                   \hyperref[sref_\l_tmpa_str]\l__stex_refs_linktext_tl
1387 %               }{
1388 %                   \l__stex_refs_linktext_tl
1389 %               }
1390 %           }
1391 %       }{
1392 %           \ltx@ifpackageloaded{hyperref}{
1393 %               \href{\use:c{sref_url_\l_tmpa_str_str}}{\tl_if_empty:NTF \l__stex_refs_linktext_
1394 %           }{
1395 %               \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_ref
1396 %           }
1397 %       }
1398 %   }
1399 %   }{
1400 %       % TODO
1401 %   }
1402 %}

```

(End definition for `\sref`. This function is documented on page 78.)

\srefsym

```

1403 \NewDocumentCommand \srefsym { 0{} m}{

```

```

1404 \stex_get_symbol:n { #2 }
1405 \__stex_refs_sym_aux:nn{#1}{\l_stex_get_symbol_uri_str}
1406 }
1407
1408 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1409
1410 % \str_if_exist:cTF {sref_sym_#2 _label_str }{
1411 % \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1412 % }{
1413 % \__stex_refs_args:n { #1 }
1414 % \str_if_empty:NTF \l__stex_refs_indocument_str {
1415 % \tl_if_exist:cTF{sref_sym_#2 _type}{
1416 % % doc uri in \l_tmpb_str
1417 % \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1418 % \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1419 % % reference
1420 % \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1421 % \cs_if_exist:cTF{autoref}{
1422 % \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1423 % }{
1424 % \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1425 % }
1426 % }{
1427 % \ltx@ifpackageloaded{hyperref}{
1428 % \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1429 % }{
1430 % \l__stex_refs_linktext_tl
1431 % }
1432 % }
1433 % }{
1434 % % URL
1435 % \ltx@ifpackageloaded{hyperref}{
1436 % \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl
1437 % }{
1438 % \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_fallback_tl
1439 % }
1440 % }
1441 % }{
1442 % \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_fallback_tl
1443 % }
1444 % }{
1445 % % TODO
1446 % }
1447 % }
1448 }

```

(End definition for `\srefsym`. This function is documented on page 78.)

`\srefsymuri`

```

1449 \cs_new_protected:Npn \srefsymuri #1 #2 { % TODO
1450 #2%\__stex_refs_sym_aux:nn[linktext={#2}]{#1}
1451 }

```

(End definition for `\srefsymuri`. This function is documented on page 78.)

```

1452 \</package>

```

Chapter 27

STEX -Modules Implementation

```
1453 <*package>
1454
1455 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1456
1457 <@@=stex_modules>
1458
1459     Warnings and error messages
1458 \msg_new:nnn{stex}{error/unknownmodule}{
1459     No~module~#1~found
1460 }
1461 \msg_new:nnn{stex}{error/syntax}{
1462     Syntax~error:~#1
1463 }
1464 \msg_new:nnn{stex}{error/siglanguage}{
1465     Module~#1~declares~signature~#2,~but~does~not~
1466     declare~its~language
1467 }
1468 \msg_new:nnn{stex}{warning/deprecated}{
1469     #1~is~deprecated;~please~use~#2~instead!
1470 }
1471
1472 \msg_new:nnn{stex}{error/conflictingmodules}{
1473     Conflicting~imports~for~module~#1
1474 }
1475
1476 \l_stex_current_module_str The current module:
1475 \str_new:N \l_stex_current_module_str
1476
1477 (End definition for \l_stex_current_module_str. This variable is documented on page 80.)
1478
1479 \l_stex_all_modules_seq Stores all available modules
1476 \seq_new:N \l_stex_all_modules_seq
1477
1478 (End definition for \l_stex_all_modules_seq. This variable is documented on page 80.)
```

```

\stex_if_in_module_p:
\stex_if_in_module:TF
1477 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1478   \str_if_empty:NTF \l_stex_current_module_str
1479   \prg_return_false: \prg_return_true:
1480 }

(End definition for \stex_if_in_module:TF. This function is documented on page 80.)

```

```

\stex_if_module_exists_p:n
\stex_if_module_exists:nTF
1481 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1482   \prop_if_exist:cTF { c_stex_module_#1_prop }
1483   \prg_return_true: \prg_return_false:
1484 }

(End definition for \stex_if_module_exists:nTF. This function is documented on page 80.)

```

```

\stex_add_to_current_module:n
\STEXexport
Only allowed within modules:
1485 \cs_new_protected:Nn \stex_execute_in_module:n { \stex_if_in_module:T {
1486   \stex_add_to_current_module:n { #1 }
1487   \stex_do_up_to_module:n { #1 }
1488 }}
1489 \cs_generate_variant:Nn \stex_execute_in_module:n {x}
1490
1491 \cs_new_protected:Nn \stex_add_to_current_module:n {
1492   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str_code} { #1 }
1493 }
1494 \cs_generate_variant:Nn \stex_add_to_current_module:n {x}
1495 \cs_new_protected:Npn \STEXexport {
1496   \ExplSyntaxOn
1497   \__stex_modules_export:n
1498 }
1499 \cs_new_protected:Nn \__stex_modules_export:n {
1500   \ignorespacesandpars#1\ExplSyntaxOff
1501   \stex_add_to_current_module:n { \ignorespacesandpars#1}
1502   \stex_smsmode_do:
1503 }
1504 \let \stex_module_export_helper:n \use:n
1505 \stex_deactivate_macro:Nn \STEXexport {module~environments}

(End definition for \stex_add_to_current_module:n and \STEXexport. These functions are documented
on page 80.)

```

```

\stex_add_constant_to_current_module:n
1506 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1507   \str_set:Nx \l_tmpa_str { #1 }
1508   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str_constants} { \l_tmpa_str }
1509 }

(End definition for \stex_add_constant_to_current_module:n. This function is documented on page
80.)

```

```

\stex_add_import_to_current_module:n
1510 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1511   \str_set:Nx \l_tmpa_str { #1 }
1512   \exp_args:Nno

```

```

1513 \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1514 \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1515 }
1516 }

```

(End definition for `\stex_add_import_to_current_module:n`. This function is documented on page 80.)

`\stex_collect_imports:n`

```

1517 \cs_new_protected:Nn \stex_collect_imports:n {
1518 \seq_clear:N \l_stex_collect_imports_seq
1519 \__stex_modules_collect_imports:n {#1}
1520 }
1521 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1522 \seq_map_inline:cn {c_stex_module_#1_imports} {
1523 \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1524 \__stex_modules_collect_imports:n { ##1 }
1525 }
1526 }
1527 \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1528 \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1529 }
1530 }

```

(End definition for `\stex_collect_imports:n`. This function is documented on page 80.)

`\stex_do_up_to_module:n`

```

1531 \int_new:N \l__stex_modules_group_depth_int
1532 \cs_new_protected:Nn \stex_do_up_to_module:n {
1533 \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1534 #1
1535 }{
1536 #1
1537 \expandafter \tl_gset:Nn
1538 \csname l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1539 \expandafter\expandafter\expandafter\endcsname
1540 \expandafter\expandafter\expandafter { \csname
1541 l__stex_modules_aftergroup_\l_stex_current_module_str _tl\endcsname #1 }
1542 \aftergroup\__stex_modules_aftergroup_do:
1543 }
1544 }
1545 \cs_generate_variant:Nn \stex_do_up_to_module:n {x}
1546 \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1547 \stex_debug:nn{aftergroup}{\cs_meaning:c{
1548 l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1549 }}}
1550 \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1551 \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1552 \tl_gclear:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1553 }{
1554 \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1555 \aftergroup\__stex_modules_aftergroup_do:
1556 }
1557 }
1558 \cs_new_protected:Nn \stex_reset_up_to_module:n {
1559 \expandafter\let\csname l__stex_modules_aftergroup_#1_tl\endcsname\undefined

```


1560 }

(End definition for `\stex_do_up_to_module:n`. This function is documented on page 80.)

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

1561

(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.

```

1562 \str_new:N \l_stex_module_ns_str
1563 \str_new:N \l_stex_module_subpath_str
1564 \cs_new_protected:Nn \__stex_modules_compute_namespace:nN {
1565   \seq_set_eq:NN \l_tmpa_seq #2
1566   % split off file extension
1567   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str % <- filename
1568   \exp_args:Nnno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1569   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str % <- filename without suffixes
1570   \seq_put_right:No \l_tmpa_seq \l_tmpb_str % <- file path including name without suffixes
1571
1572   \bool_set_true:N \l_tmpa_bool
1573   \bool_while_do:Nn \l_tmpa_bool {
1574     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1575     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1576       {source} { \bool_set_false:N \l_tmpa_bool }
1577     }{}{
1578       \seq_if_empty:NT \l_tmpa_seq {
1579         \bool_set_false:N \l_tmpa_bool
1580       }
1581     }
1582   }
1583
1584   \stex_path_to_string:NN \l_tmpa_seq \l_stex_module_subpath_str
1585   % \l_tmpa_seq <- sub-path relative to archive
1586   \str_if_empty:NTF \l_stex_module_subpath_str {
1587     \str_set:Nx \l_stex_module_ns_str {#1}
1588   }{
1589     \str_set:Nx \l_stex_module_ns_str {
1590       #1/\l_stex_module_subpath_str
1591     }
1592   }
1593 }
1594
1595 \cs_new_protected:Nn \stex_modules_current_namespace: {
1596   \str_clear:N \l_stex_module_subpath_str
1597   \prop_if_exist:NTF \l_stex_current_repository_prop {
1598     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1599     \__stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1600   }{
1601     % split off file extension
1602     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1603     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str

```

```

1604 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1605 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1606 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1607 \str_set:Nx \l_stex_module_ns_str {
1608   file:/\stex_path_to_string:N \l_tmpa_seq
1609 }
1610 }
1611 }

```

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

27.1 The smodule environment

smodule arguments:

```

1612 \keys_define:nn { stex / module } {
1613   title      .tl_set:N      = \smodulename ,
1614   type       .str_set_x:N    = \smodulename ,
1615   id         .str_set_x:N    = \smoduleid ,
1616   deprecate  .str_set_x:N    = \l_stex_module_deprecate_str ,
1617   ns         .str_set_x:N    = \l_stex_module_ns_str ,
1618   lang       .str_set_x:N    = \l_stex_module_lang_str ,
1619   sig        .str_set_x:N    = \l_stex_module_sig_str ,
1620   creators   .str_set_x:N    = \l_stex_module_creators_str ,
1621   contributors .str_set_x:N  = \l_stex_module_contributors_str ,
1622   meta       .str_set_x:N    = \l_stex_module_meta_str ,
1623   srccite    .str_set_x:N    = \l_stex_module_srccite_str
1624 }
1625
1626 \cs_new_protected:Nn \__stex_modules_args:n {
1627   \str_clear:N \smodulename
1628   \str_clear:N \smodulename
1629   \str_clear:N \smoduleid
1630   \str_clear:N \l_stex_module_ns_str
1631   \str_clear:N \l_stex_module_deprecate_str
1632   \str_clear:N \l_stex_module_lang_str
1633   \str_clear:N \l_stex_module_sig_str
1634   \str_clear:N \l_stex_module_creators_str
1635   \str_clear:N \l_stex_module_contributors_str
1636   \str_clear:N \l_stex_module_meta_str
1637   \str_clear:N \l_stex_module_srccite_str
1638   \keys_set:nn { stex / module } { #1 }
1639 }
1640
1641 % module parameters here? In the body?
1642

```

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

```

1643 \cs_new_protected:Nn \stex_module_setup:nn {
1644   \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1645   \str_set:Nx \l_stex_module_name_str { #2 }
1646   \__stex_modules_args:n { #1 }

```

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

Are we in a nested module?

```

1647 \stex_if_in_module:TF {
1648   % Nested module
1649   \prop_get:cnN {c_stex_module\_l_stex_current_module_str _prop}
1650   { ns } \l_stex_module_ns_str
1651   \str_set:Nx \l_stex_module_name_str {
1652     \prop_item:cn {c_stex_module\_l_stex_current_module_str _prop}
1653     { name } / \l_stex_module_name_str
1654   }
1655   \str_if_empty:NT \l_stex_module_lang_str {
1656     \str_set:Nx \l_stex_module_lang_str {
1657       \prop_item:cn {c_stex_module\_l_stex_current_module_str _prop}
1658       { lang }
1659     }
1660   }
1661 }{
1662   % not nested:
1663   \str_if_empty:NT \l_stex_module_ns_str {
1664     \stex_modules_current_namespace:
1665     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1666       / {\l_stex_module_ns_str}
1667     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1668     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1669       \str_set:Nx \l_stex_module_ns_str {
1670         \stex_path_to_string:N \l_tmpa_seq
1671       }
1672     }
1673   }
1674 }

```

Next, we determine the language of the module:

```

1675 \str_if_empty:NT \l_stex_module_lang_str {
1676   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1677   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1678   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1679   \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
1680     \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
1681       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
1682     }
1683   }
1684   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1685   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
1686     \seq_pop_right:NN \l_tmpa_seq \l_stex_module_lang_str
1687     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1688       inferred~from~file~name}
1689   }
1690 }
1691
1692 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
1693   \exp_args:NNo \stex_set_language:Nn \l_tmpa_str \l_stex_module_lang_str
1694 }}

```

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

```

1695 \str_if_empty:NTF \l_stex_module_sig_str {
1696   \exp_args:Nnx \prop_gset_from_keyval:cn {
1697     c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1698   } {
1699     name      = \l_stex_module_name_str ,
1700     ns       = \l_stex_module_ns_str ,
1701     file     = \exp_not:o { \g_stex_currentfile_seq } ,
1702     lang     = \l_stex_module_lang_str ,
1703     sig      = \l_stex_module_sig_str ,
1704     deprecate = \l_stex_module_deprecate_str ,
1705     meta     = \l_stex_module_meta_str
1706   }
1707   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1708   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1709   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _copymodules}
1710   \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1711   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1712 \str_if_empty:NT \l_stex_module_meta_str {
1713   \str_set:Nx \l_stex_module_meta_str {
1714     \c_stex_metatheory_ns_str ? Metatheory
1715   }
1716 }
1717 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1718   \bool_set_true:N \l_stex_in_meta_bool
1719   \exp_args:Nx \stex_add_to_current_module:n {
1720     \bool_set_true:N \l_stex_in_meta_bool
1721     \stex_activate_module:n {\l_stex_module_meta_str}
1722     \bool_set_false:N \l_stex_in_meta_bool
1723   }
1724   \stex_activate_module:n {\l_stex_module_meta_str}
1725   \bool_set_false:N \l_stex_in_meta_bool
1726 }
1727 }{
1728   \str_if_empty:NT \l_stex_module_lang_str {
1729     \msg_error:nnxx{stex}{error/siglanguage}{
1730       \l_stex_module_ns_str?\l_stex_module_name_str
1731     }\l_stex_module_sig_str}
1732   }
1733   \stex_debug:nn{modules}{Signature~\l_stex_module_sig_str~for~\l_stex_module_ns_str?\l_st
1734   \stex_if_module_exists:nTF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1735     \stex_debug:nn{modules}{(already exists)}
1736   }{
1737     \stex_debug:nn{modules}{(needs loading)}
1738     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1739     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1740     \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1741     \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1742     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1743     \str_set:Nx \l_tmpa_str {
1744       \stex_path_to_string:N \l_tmpa_seq /

```

```

1745     \l_tmpa_str . \l_stex_module_sig_str .tex
1746   }
1747   \IfFileExists \l_tmpa_str {
1748     \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1749       \str_clear:N \l_stex_current_module_str
1750       \seq_clear:N \l_stex_all_modules_seq
1751       \stex_debug:nn{modules}{Loading~signature}
1752     }
1753   }{
1754     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1755   }
1756 }
1757 \stex_if_smsmode:F {
1758   \stex_activate_module:n {
1759     \l_stex_module_ns_str ? \l_stex_module_name_str
1760   }
1761 }
1762 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1763 }
1764 \str_if_empty:NF \l_stex_module_deprecate_str {
1765   \msg_warning:nnxx{stex}{warning/deprecated}{
1766     Module~\l_stex_current_module_str
1767   }{
1768     \l_stex_module_deprecate_str
1769   }
1770 }
1771 \seq_put_right:Nx \l_stex_all_modules_seq {
1772   \l_stex_module_ns_str ? \l_stex_module_name_str
1773 }
1774 \tl_clear:c{l__stex_modules_aftergroup_\l_stex_module_ns_str ? \l_stex_module_name_str _tl}
1775 }

```

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

smodule (*env.*) The module environment.

```

\__stex_modules_begin_module: implements \begin{smodule}
1776 \cs_new_protected:Nn \__stex_modules_begin_module: {
1777   \stex_reactivate_macro:N \STEXexport
1778   \stex_reactivate_macro:N \importmodule
1779   \stex_reactivate_macro:N \symdecl
1780   \stex_reactivate_macro:N \notation
1781   \stex_reactivate_macro:N \symdef
1782 }
1783 \stex_debug:nn{modules}{
1784   New~module:\\
1785   Namespace:~\l_stex_module_ns_str\\
1786   Name:~\l_stex_module_name_str\\
1787   Language:~\l_stex_module_lang_str\\
1788   Signature:~\l_stex_module_sig_str\\
1789   Metatheory:~\l_stex_module_meta_str\\
1790   File:~\stex_path_to_string:N \g_stex_currentfile_seq
1791 }
1792

```

```

1793 \stex_if_do_html:T{
1794   \begin{stex_annotate_env} {theory} {
1795     \l_stex_module_ns_str ? \l_stex_module_name_str
1796   }
1797
1798   \stex_annotate_invisible:nnn{header}{} {
1799     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1800     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1801     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1802       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1803     }
1804     \str_if_empty:NF \smoduletype {
1805       \stex_annotate:nnn{type}{\smoduletype}{}
1806     }
1807   }
1808 }
1809 % TODO: Inherit metatheory for nested modules?
1810 }
1811 \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}`

```

1812 \cs_new_protected:Nn \_stex_modules_end_module: {
1813   \stex_debug:nn{modules}{Closing module~\prop_item:cn {c_stex_module\_l_stex_current_module}
1814   \stex_reset_up_to_module:n \l_stex_current_module_str
1815   \stex_if_smsmode:T {
1816     \stex_persist:x {
1817       \prop_set_from_keyval:cn{c_stex_module\_l_stex_current_module_str _prop}{
1818         \exp_after:wN \prop_to_keyval:N \csname c_stex_module\_l_stex_current_module_str _pr
1819       }
1820       \seq_set_from_clist:cn{c_stex_module\_l_stex_current_module_str _constants}{
1821         \seq_use:cn{c_stex_module\_l_stex_current_module_str _constants},
1822       }
1823       \seq_set_from_clist:cn{c_stex_module\_l_stex_current_module_str _imports}{
1824         \seq_use:cn{c_stex_module\_l_stex_current_module_str _imports},
1825       }
1826       \tl_set:cn {c_stex_module\_l_stex_current_module_str _code}
1827     }
1828     \exp_after:wN \let \exp_after:wN \l_tmpa_tl \csname c_stex_module\_l_stex_current_module
1829     \exp_after:wN \stex_persist:n \exp_after:wN { \exp_after:wN { \l_tmpa_tl } }
1830   }
1831 }

```

(End definition for `_stex_modules_end_module:.`)

The core environment

```

1832 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1833 \NewDocumentEnvironment { smodule } { 0 } { m } {
1834   \stex_module_setup:nn{#1}{#2}
1835   %\par
1836   \stex_if_smsmode:F{
1837     \tl_if_empty:NF \smoduletitle {
1838       \exp_args:No \stex_document_title:n \smoduletitle
1839     }

```

```

1840 \tl_clear:N \l_tmpa_tl
1841 \clist_map_inline:Nn \smodulotype {
1842   \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1843     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1844   }
1845 }
1846 \tl_if_empty:NTF \l_tmpa_tl {
1847   \__stex_modules_smodule_start:
1848 }{
1849   \l_tmpa_tl
1850 }
1851 }
1852 \__stex_modules_begin_module:
1853 \str_if_empty:NF \smoduleid {
1854   \stex_ref_new_doc_target:n \smoduleid
1855 }
1856 \stex_smsmode_do:
1857 } {
1858   \__stex_modules_end_module:
1859   \stex_if_smsmode:F {
1860     \end{stex_annotate_env}
1861     \clist_set:Nn \l_tmpa_clist \smodulotype
1862     \tl_clear:N \l_tmpa_tl
1863     \clist_map_inline:Nn \l_tmpa_clist {
1864       \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1865         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1866       }
1867     }
1868     \tl_if_empty:NTF \l_tmpa_tl {
1869       \__stex_modules_smodule_end:
1870     }{
1871       \l_tmpa_tl
1872     }
1873   }
1874 }

```

\stexpatchmodule

```

1875 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1876 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1877
1878 \newcommand\stexpatchmodule[3] [] {
1879   \str_set:Nx \l_tmpa_str{ #1 }
1880   \str_if_empty:NTF \l_tmpa_str {
1881     \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1882     \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1883   }{
1884     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1885     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1886   }
1887 }

```

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

27.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n 1888 \NewDocumentCommand \STEXModule { m } {
1889   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1890   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1891   \tl_set:Nn \l_tmpa_tl {
1892     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1893   }
1894   \seq_map_inline:Nn \l_stex_all_modules_seq {
1895     \str_set:Nn \l_tmpb_str { ##1 }
1896     \str_if_eq:eeT { \l_tmpa_str } {
1897       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1898     } {
1899       \seq_map_break:n {
1900         \tl_set:Nn \l_tmpa_tl {
1901           \stex_invoke_module:n { ##1 }
1902         }
1903       }
1904     }
1905   }
1906   \l_tmpa_tl
1907 }
1908
1909 \cs_new_protected:Nn \stex_invoke_module:n {
1910   \stex_debug:nn{modules}{Invoking~module~#1}
1911   \peek_charcode_remove:NTF ! {
1912     \__stex_modules_invoke_uri:nN { #1 }
1913   } {
1914     \peek_charcode_remove:NTF ? {
1915       \__stex_modules_invoke_symbol:nn { #1 }
1916     } {
1917       \msg_error:nnx{stex}{error/syntax}{
1918         ?~or~!~expected~after~
1919         \c_backslash_str STEXModule{#1}
1920       }
1921     }
1922   }
1923 }
1924
1925 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1926   \str_set:Nn #2 { #1 }
1927 }
1928
1929 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1930   \stex_invoke_symbol:n{#1?#2}
1931 }

```

(End definition for \STEXModule and \stex_invoke_module:n. These functions are documented on page 81.)

```

\stex_activate_module:n
1932 \bool_new:N \l_stex_in_meta_bool
1933 \bool_set_false:N \l_stex_in_meta_bool

```



```

1934 \cs_new_protected:Nn \stex_activate_module:n {
1935   \stex_debug:nn{modules}{Activating~module~#1}
1936   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1937     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1938     \use:c{ c_stex_module_#1_code }
1939   }
1940 }

```

(End definition for `\stex_activate_module:n`. This function is documented on page 82.)

`mmtinterface` (env.)

```

1941 \NewDocumentEnvironment { mmtinterface } { 0{} m m } {
1942   \begin{smodule}[#1]{#3}
1943     \str_set:Nx \l_stex_module_mmtfor_str {#2}
1944     \MMTinclude{#2}
1945     \stex_reactivate_macro:N \mmtdecl
1946     \stex_reactivate_macro:N \mmtdef
1947   }{
1948     \end{smodule}
1949   }
1950 \end{package}

```

Chapter 28

STEX -Module Inheritance Implementation

```
1951 <*package>
1952
1953 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1954
```

28.1 SMS Mode

```
1955 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1956 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1957 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1958 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1959
1960 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1961   \makeatletter
1962   \makeatother
1963   \ExplSyntaxOn
1964   \ExplSyntaxOff
1965   \rustexBREAK
1966 }
1967
1968 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1969   \symdef
1970   \importmodule
1971   \notation
1972   \symdecl
1973   \STEXexport
1974   \inlineass
1975   \inlinedef
1976   \inlineex
1977   \endinput
1978   \setnotation
```

```

1979 \copynotation
1980 \assign
1981 \renamedekl
1982 \donotcopy
1983 \instantiate
1984 \textsymdecl
1985 }
1986
1987 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1988   \tl_to_str:n {
1989     smodule,
1990     copymodule,
1991     interpretmodule,
1992     realization,
1993     sdefinition,
1994     sexample,
1995     sassertion,
1996     sparagraph,
1997     mathstructure
1998   }
1999 }

```

(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 83.)

`\stex_if_smsmode_p:`

```

\stex_if_smsmode:TF
2000 \bool_new:N \g__stex_smsmode_bool
2001 \bool_set_false:N \g__stex_smsmode_bool
2002 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
2003   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
2004 }

```

(End definition for `\stex_if_smsmode:TF`. This function is documented on page 83.)

`_stex_smsmode_in_smsmode:nn`

```

2005 \cs_new_protected:Nn \_stex_smsmode_in_smsmode:nn { \stex_suppress_html:n {
2006   \vbox_set:Nn \l_tmpa_box {
2007     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
2008     \bool_gset_true:N \g__stex_smsmode_bool
2009     #2
2010     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
2011   }
2012   \box_clear:N \l_tmpa_box
2013 } }

```

(End definition for `_stex_smsmode_in_smsmode:nn`.)

`\stex_file_in_smsmode:nn`

```

2014 \quark_new:N \q__stex_smsmode_break
2015
2016 \NewDocumentCommand \_stex_smsmode_importmodule: { 0{} m } {
2017   \seq_gput_right:Nn \l__stex_smsmode_importmodules_seq {{#1}{#2}}
2018   \stex_smsmode_do:
2019 }
2020

```

```

2021 \cs_new_protected:Nn \__stex_smsmode_module:nn {
2022   \__stex_modules_args:n{#1}
2023   \stex_if_in_module:F {
2024     \str_if_empty:NF \l_stex_module_sig_str {
2025       \stex_modules_current_namespace:
2026       \str_set:Nx \l_stex_module_name_str { #2 }
2027       \stex_if_module_exists:nF{\l_stex_module_ns_str?\l_stex_module_name_str}{
2028         \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
2029         \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
2030         \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
2031         \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
2032         \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
2033         \str_set:Nx \l_tmpa_str {
2034           \stex_path_to_string:N \l_tmpa_seq /
2035           \l_tmpa_str . \l_stex_module_sig_str .tex
2036         }
2037         \IfFileExists \l_tmpa_str {
2038           \exp_args:NNx \seq_gput_right:Nn \l__stex_smsmode_sigmodules_seq \l_tmpa_str
2039         }{
2040           \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
2041         }
2042       }
2043     }
2044   }
2045 }
2046
2047 \prg_new_conditional:Nnn \__stex_smsmode_check_import_pair:nn {T,F,TF} {
2048   %\stex_debug:nn{import-pair}{\detokenize{#1}~{#2}}
2049   \tl_if_empty:nTF{#1}{
2050     \prop_if_exist:NTF \l_stex_current_repository_prop
2051     {
2052       %\stex_debug:nn{import-pair}{in repository \prop_item:Nn \l_stex_current_repository_
2053       \prg_return_true:
2054     } {
2055       \seq_set_split:Nnn \l_tmpa_seq ? {#2}
2056       \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
2057       \tl_if_empty:NT \l_tmpa_tl {
2058         \seq_pop_left:NN \l_tmpa_seq \l_tmpa_tl
2059       }
2060       %\stex_debug:nn{import-pair}{\seq_use:Nn \l_tmpa_seq,~of~length~\seq_count:N \l_tmpa
2061       \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1
2062       \prg_return_true: \prg_return_false:
2063     }
2064   }\prg_return_true:
2065 }
2066
2067 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
2068   \stex_filestack_push:n{#1}
2069   \seq_gclear:N \l__stex_smsmode_importmodules_seq
2070   \seq_gclear:N \l__stex_smsmode_sigmodules_seq
2071   % ----- new -----
2072   \__stex_smsmode_in_smsmode:nn{#1}{
2073     \let\importmodule\__stex_smsmode_importmodule:
2074     \let\stex_module_setup:nn\__stex_smsmode_module:nn

```

```

2075 \let\__stex_modules_begin_module:\relax
2076 \let\__stex_modules_end_module:\relax
2077 \seq_clear:N \g_stex_smsmode_allowedenvs_seq
2078 \exp_args:NNx \seq_put_right:Nn \g_stex_smsmode_allowedenvs_seq {\tl_to_str:n{smodule}}
2079 \tl_clear:N \g_stex_smsmode_allowedmacros_tl
2080 \tl_clear:N \g_stex_smsmode_allowedmacros_escape_tl
2081 \tl_put_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\importmodule}
2082 \everyeof{\q__stex_smsmode_break\noexpand}
2083 \expandafter\expandafter\expandafter
2084 \stex_smsmode_do:
2085 \csname @ @ input\endcsname "#1"\relax
2086
2087 \seq_map_inline:Nn \l__stex_smsmode_sigmodules_seq {
2088   \stex_filestack_push:n{##1}
2089   \expandafter\expandafter\expandafter
2090   \stex_smsmode_do:
2091   \csname @ @ input\endcsname "##1"\relax
2092   \stex_filestack_pop:
2093 }
2094 }
2095 % ----- new -----
2096 \__stex_smsmode_in_smsmode:nn{#1} {
2097   #2
2098   % ----- new -----
2099   \begingroup
2100   \%stex_debug:nn{smsmode}{Here:~\seq_use:Nn\l__stex_smsmode_importmodules_seq, }
2101   \seq_map_inline:Nn \l__stex_smsmode_importmodules_seq {
2102     \__stex_smsmode_check_import_pair:nnT ##1 { \begingroup
2103       \stex_import_module_uri:nn ##1
2104       \stex_import_require_module:nnnn
2105       \l_stex_import_ns_str
2106       \l_stex_import_archive_str
2107       \l_stex_import_path_str
2108       \l_stex_import_name_str \endgroup
2109     }
2110   }
2111   \endgroup
2112   \%stex_debug:nn{smsmode}{Actually~loading~file~#1}
2113   % ----- new -----
2114   \everyeof{\q__stex_smsmode_break\noexpand}
2115   \expandafter\expandafter\expandafter
2116   \stex_smsmode_do:
2117   \csname @ @ input\endcsname "#1"\relax
2118 }
2119 \stex_filestack_pop:
2120 }

```

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

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

```

2121 \cs_new_protected:Npn \stex_smsmode_do: {
2122   \stex_if_smsmode:T {
2123     \__stex_smsmode_do:w

```

```

2124 }
2125 }
2126 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
2127   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
2128     \expandafter\if\expandafter\relax\noexpand#1
2129     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
2130   \else\expandafter\__stex_smsmode_do:w\fi
2131 }{
2132   \__stex_smsmode_do:w % #1
2133 }
2134 }
2135 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
2136   \cs_if_eq:NNTF #1 \q__stex_smsmode_break {
2137     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
2138       #1\__stex_smsmode_do:w
2139     }{
2140       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
2141         #1
2142       }{
2143         \cs_if_eq:NNTF \begin #1 {
2144           \__stex_smsmode_check_begin:n
2145         }{
2146           \cs_if_eq:NNTF \end #1 {
2147             \__stex_smsmode_check_end:n
2148           }{
2149             \__stex_smsmode_do:w
2150           }
2151         }
2152       }
2153     }
2154   }
2155 }
2156
2157 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
2158   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
2159     \begin{#1}
2160   }{
2161     \__stex_smsmode_do:w
2162   }
2163 }
2164 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
2165   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
2166     \end{#1}\__stex_smsmode_do:w
2167   }{
2168     \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
2169   }
2170 }

```

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

28.2 Inheritance

```

2171 <@@=stex_importmodule>

```

`\stex_import_module_uri:nn`

```
2172 \cs_new_protected:Nn \stex_import_module_uri:nn {
2173   \str_set:Nx \l_stex_import_archive_str { #1 }
2174   \str_set:Nn \l_stex_import_path_str { #2 }
2175
2176   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
2177   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
2178   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
2179
2180   \stex_modules_current_namespace:
2181   \bool_lazy_all:nTF {
2182     {\str_if_empty_p:N \l_stex_import_archive_str}
2183     {\str_if_empty_p:N \l_stex_import_path_str}
2184     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
2185   }{
2186     \str_set_eq:NN \l_stex_import_path_str \l_stex_module_subpath_str
2187     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
2188   }{
2189     \str_if_empty:NT \l_stex_import_archive_str {
2190       \prop_if_exist:NT \l_stex_current_repository_prop {
2191         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
2192       }
2193     }
2194     \str_if_empty:NTF \l_stex_import_archive_str {
2195       \str_if_empty:NF \l_stex_import_path_str {
2196         \stex_path_from_string:Nn \l_tmpb_seq {
2197           \l_stex_module_ns_str / .. / \l_stex_import_path_str
2198         }
2199         \str_set:Nx \l_stex_import_ns_str {\stex_path_to_string:N \l_tmpb_seq}
2200         \str_replace_once:Nnn \l_stex_import_ns_str {file://} {file:///}
2201       }
2202     }{
2203       \stex_require_repository:n \l_stex_import_archive_str
2204       \prop_get:cnN { c_stex_mathhub\_l_stex_import_archive_str_manifest_prop } { ns }
2205       \l_stex_import_ns_str
2206       \str_if_empty:NF \l_stex_import_path_str {
2207         \str_set:Nx \l_stex_import_ns_str {
2208           \l_stex_import_ns_str / \l_stex_import_path_str
2209         }
2210       }
2211     }
2212   }
2213 }
```

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

`\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`.

```
2214 \str_new:N \l_stex_import_name_str
2215 \str_new:N \l_stex_import_archive_str
2216 \str_new:N \l_stex_import_path_str
2217 \str_new:N \l_stex_import_ns_str
```

(End definition for `\l_stex_import_name_str` and others. These variables are documented on page 85.)

```

\stex_import_require_module:nnnn {{\ns}} {{\archive-ID}} {{\path}} {{\name}}
2218 \cs_new_protected:Nn \stex_import_require_module:nnnn {
2219   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
2220
2221     \stex_debug:nn{requiremodule}{Here:\~1:\~2:\~3:\~4}
2222
2223     \exp_args:NNxx \seq_set_split:Nnn \l_tmpa_seq {\tl_to_str:n{/}} {#4}
2224     \seq_get_left:NN \l_tmpa_seq \l_tmpc_str
2225
2226     %\stex_debug:nn{requiremodule}{Top~module:\l_tmpc_str}
2227
2228     % archive
2229     \str_set:Nx \l_tmpa_str { #2 }
2230     \str_if_empty:NTF \l_tmpa_str {
2231       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
2232       \seq_put_right:Nn \l_tmpa_seq {...}
2233     } {
2234       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
2235       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
2236       \seq_put_right:Nn \l_tmpa_seq { source }
2237     }
2238
2239     % path
2240     \str_set:Nx \l_tmpb_str { #3 }
2241     \str_if_empty:NTF \l_tmpb_str {
2242       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / \l_tmpc_str }
2243
2244       \ltx@ifpackageloaded{babel} {
2245         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
2246           { \language } \l_tmpb_str {
2247           \msg_error:nnx{stex}{error/unknownlanguage}{\language}
2248         }
2249       } {
2250         \str_clear:N \l_tmpb_str
2251       }
2252
2253       \stex_debug:nn{modules}{Checking~a1~\l_tmpa_str.\l_tmpb_str.tex}
2254       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2255         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2256       }{
2257         \stex_debug:nn{modules}{Checking~a2~\l_tmpa_str.tex}
2258         \IfFileExists{ \l_tmpa_str.tex }{
2259           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2260         }{
2261           % try english as default
2262           \stex_debug:nn{modules}{Checking~a3~\l_tmpa_str.en.tex}
2263           \IfFileExists{ \l_tmpa_str.en.tex }{
2264             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2265           }{
2266             \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2267           }
2268         }
2269       }
2270

```



```

2271 } {
2272   \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
2273   \seq_concat:NNN \l_tmpb_seq \l_tmpa_seq \l_tmpb_seq
2274
2275   \ltx@ifpackageloaded{babel} {
2276     \exp_args:NnX \prop_get:NnNF \c_stex_language_abbrevs_prop
2277       { \language } \l_tmpb_str {
2278       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
2279     }
2280   } {
2281     \str_clear:N \l_tmpb_str
2282   }
2283
2284   \stex_path_canonicalize:N \l_tmpb_seq
2285   \stex_path_to_string:NN \l_tmpb_seq \l_tmpa_str
2286
2287   \stex_debug:nn{modules}{Checking~b1~\l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex}
2288   \IfFileExists{ \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }{
2289     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }
2290   }{
2291     \stex_debug:nn{modules}{Checking~b2~\l_tmpa_str/\l_tmpc_str.tex}
2292     \IfFileExists{ \l_tmpa_str/\l_tmpc_str.tex }{
2293       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.tex }
2294     }{
2295       % try english as default
2296       \stex_debug:nn{modules}{Checking~b3~\l_tmpa_str/\l_tmpc_str.en.tex}
2297       \IfFileExists{ \l_tmpa_str/\l_tmpc_str.en.tex }{
2298         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.en.tex }
2299       }{
2300         \stex_debug:nn{modules}{Checking~b4~\l_tmpa_str.\l_tmpb_str.tex}
2301         \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2302           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2303         }{
2304           \stex_debug:nn{modules}{Checking~b4~\l_tmpa_str.tex}
2305           \IfFileExists{ \l_tmpa_str.tex }{
2306             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2307           }{
2308             % try english as default
2309             \stex_debug:nn{modules}{Checking~b5~\l_tmpa_str.en.tex}
2310             \IfFileExists{ \l_tmpa_str.en.tex }{
2311               \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2312             }{
2313               \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2314             }
2315           }
2316         }
2317       }
2318     }
2319   }
2320 }
2321
2322 \str_if_eq:eeF{\g__stex_importmodule_file_str}{\seq_use:Nn \g_stex_currentfile_seq /}{
2323   \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
2324     \seq_clear:N \l_stex_all_modules_seq

```

```

2325     \str_clear:N \l_stex_current_module_str
2326     \str_set:Nx \l_tmpb_str { #2 }
2327     \str_if_empty:NF \l_tmpb_str {
2328       \stex_set_current_repository:n { #2 }
2329     }
2330     \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
2331   }
2332
2333   \stex_if_module_exists:nF { #1 ? #4 } {
2334     \msg_error:nnx{stex}{error/unknownmodule}{
2335       #1?#4~(in~file~\g__stex_importmodule_file_str)
2336     }
2337   }
2338 }
2339
2340 }
2341 \stex_activate_module:n { #1 ? #4 }
2342 }

```

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

`\importmodule`

```

2343 \NewDocumentCommand \importmodule { 0{} m } {
2344   \stex_import_module_uri:nn { #1 } { #2 }
2345   \stex_debug:nn{modules}{Importing~module:~
2346     \l_stex_import_ns_str ? \l_stex_import_name_str
2347   }
2348   \stex_import_require_module:nnnn
2349   { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2350   { \l_stex_import_path_str } { \l_stex_import_name_str }
2351   \stex_if_smsmode:F {
2352     \stex_annotate_invisible:nnn
2353     {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
2354   }
2355   \exp_args:Nx \stex_add_to_current_module:n {
2356     \stex_import_require_module:nnnn
2357     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2358     { \l_stex_import_path_str } { \l_stex_import_name_str }
2359   }
2360   \exp_args:Nx \stex_add_import_to_current_module:n {
2361     \l_stex_import_ns_str ? \l_stex_import_name_str
2362   }
2363   \stex_smsmode_do:
2364   \ignorespacesandpars
2365 }
2366 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

(End definition for `\importmodule`. This function is documented on page 84.)

`\usemodule`

```

2367 \NewDocumentCommand \usemodule { 0{} m } {
2368   \stex_if_smsmode:F {
2369     \stex_import_module_uri:nn { #1 } { #2 }
2370     \stex_import_require_module:nnnn
2371     { \l_stex_import_ns_str } { \l_stex_import_archive_str }

```

```

2372 { \l_stex_import_path_str } { \l_stex_import_name_str }
2373 \stex_annotate_invisible:nnn
2374 {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
2375 }
2376 \stex_smsmode_do:
2377 \ignorespacesandpars
2378 }

```

(End definition for \usemodule. This function is documented on page 84.)

```

2379 \cs_new_protected:Nn \stex_csl_to_imports:Nn {
2380   \tl_if_empty:nF{#2}{
2381     \clist_set:Nn \l_tmpa_clist {#2}
2382     \clist_map_inline:Nn \l_tmpa_clist {
2383       \tl_if_head_eq_charcode:nNTF {##1} [{
2384         #1 ##1
2385       }{
2386         #1{##1}
2387       }
2388     }
2389   }
2390 }
2391 \cs_generate_variant:Nn \stex_csl_to_imports:Nn {No}
2392
2393
2394 \end{package}

```

Chapter 29

STEX -Symbols Implementation

```
2395 <*package>
2396
2397 %%%%%%%%%% symbols.dtx %%%%%%%%%%
2398
2399 \msg_new:nnn{stex}{error/wrongargs}{
2400   args~value~in~symbol~declaration~for~#1~
2401   needs~to~be~i,~a,~b~or~B,~but~#2~given
2402 }
2403 \msg_new:nnn{stex}{error/unknownsymbol}{
2404   No~symbol~#1~found!
2405 }
2406 \msg_new:nnn{stex}{error/seqlength}{
2407   Expected~#1~arguments;~got~#2!
2408 }
2409 \msg_new:nnn{stex}{error/unknownnotation}{
2410   Unknown~notation~#1~for~#2!
2411 }
```

29.1 Symbol Declarations

```
2412 <@@=stex_symdecl>
\stex_all_symbols:n Map over all available symbols
2413 \cs_new_protected:Nn \stex_all_symbols:n {
2414   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
2415   \seq_map_inline:Nn \l_stex_all_modules_seq {
2416     \seq_map_inline:cn{c_stex_module_##1_constants}{
2417       \__stex_symdecl_all_symbols_cs{##1?####1}
2418     }
2419   }
2420 }
```

(End definition for `\stex_all_symbols:n`. This function is documented on page [87](#).)

`\STEXsymbol`

```
2421 \NewDocumentCommand \STEXsymbol { m } {
2422   \stex_get_symbol:n { #1 }
2423   \exp_args:No
2424   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
2425 }
```

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

`symdecl` arguments:

```
2426 \keys_define:nn { stex / symdecl } {
2427   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2428   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2429   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
2430   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,
2431   align     .str_set:N    = \l_stex_symdecl_align_str , % TODO(?)
2432   gfc       .str_set:N    = \l_stex_symdecl_gfc_str , % TODO(?)
2433   def       .tl_set:N    = \l_stex_symdecl_definiens_tl ,
2434   reorder   .str_set_x:N = \l_stex_symdecl_reorder_str ,
2435   argnames  .clist_set:N = \l_stex_symdecl_argnames_clist ,
2436   assoc     .choices:nn  =
2437     {bin,binl,binr,pre,conj,pwconj}
2438     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}
2439 }
2440
2441 \bool_new:N \l_stex_symdecl_make_macro_bool
2442
2443 \cs_new_protected:Nn \__stex_symdecl_args:n {
2444   \str_clear:N \l_stex_symdecl_name_str
2445   \str_clear:N \l_stex_symdecl_args_str
2446   \str_clear:N \l_stex_symdecl_deprecate_str
2447   \str_clear:N \l_stex_symdecl_reorder_str
2448   \str_clear:N \l_stex_symdecl_assoctype_str
2449   \bool_set_false:N \l_stex_symdecl_local_bool
2450   \tl_clear:N \l_stex_symdecl_type_tl
2451   \tl_clear:N \l_stex_symdecl_definiens_tl
2452   \clist_clear:N \l_stex_symdecl_argnames_clist
2453
2454   \keys_set:nn { stex / symdecl } { #1 }
2455 }
```

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

```
2456
2457 \NewDocumentCommand \symdecl { s m O{} } {
2458   \__stex_symdecl_args:n { #3 }
2459   \IfBooleanTF #1 {
2460     \bool_set_false:N \l_stex_symdecl_make_macro_bool
2461   } {
2462     \bool_set_true:N \l_stex_symdecl_make_macro_bool
2463   }
2464   \stex_symdecl_do:n { #2 }
2465   \stex_smsmode_do:
2466 }
```

```

2467
2468 \cs_new_protected:Nn \stex_symdecl_do:nn {
2469   \__stex_symdecl_args:n{#1}
2470   \bool_set_false:N \l_stex_symdecl_make_macro_bool
2471   \stex_symdecl_do:n{#2}
2472 }
2473
2474 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

(End definition for \symdecl. This function is documented on page 86.)

\stex_symdecl_do:n

```

2475 \cs_new_protected:Nn \stex_symdecl_do:n {
2476   \stex_if_in_module:F {
2477     % TODO throw error? some default namespace?
2478   }
2479
2480   \str_if_empty:NT \l_stex_symdecl_name_str {
2481     \str_set:Nx \l_stex_symdecl_name_str { #1 }
2482   }
2483
2484   \prop_if_exist:cT { l_stex_symdecl_
2485     \l_stex_current_module_str ?
2486     \l_stex_symdecl_name_str
2487     _prop
2488   }{
2489     % TODO throw error (beware of circular dependencies)
2490   }
2491
2492   \prop_clear:N \l_tmpa_prop
2493   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
2494   \seq_clear:N \l_tmpa_seq
2495   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
2496   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
2497
2498   \str_if_empty:NT \l_stex_symdecl_deprecate_str {
2499     \str_if_empty:NF \l_stex_module_deprecate_str {
2500       \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
2501     }
2502   }
2503   \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
2504
2505   \exp_args:No \stex_add_constant_to_current_module:n {
2506     \l_stex_symdecl_name_str
2507   }
2508
2509   % arity/args
2510   \int_zero:N \l_tmpb_int
2511
2512   \bool_set_true:N \l_tmpa_bool
2513   \str_map_inline:Nn \l_stex_symdecl_args_str {
2514     \token_case_meaning:NnF ##1 {
2515       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2516       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }

```

```

2517     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2518     {\tl_to_str:n a} {
2519         \bool_set_false:N \l_tmpa_bool
2520         \int_incr:N \l_tmpb_int
2521     }
2522     {\tl_to_str:n B} {
2523         \bool_set_false:N \l_tmpa_bool
2524         \int_incr:N \l_tmpb_int
2525     }
2526 }{
2527     \msg_error:nnxx{stex}{error/wrongargs}{
2528         \l_stex_current_module_str ?
2529         \l_stex_symdecl_name_str
2530     }{##1}
2531 }
2532 }
2533
2534 \bool_if:NTF \l_tmpa_bool {
2535     % possibly numeric
2536     \str_if_empty:NTF \l_stex_symdecl_args_str {
2537         \prop_put:Nnn \l_tmpa_prop { args } {}
2538         \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2539     }{
2540         \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
2541         \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2542         \str_clear:N \l_tmpa_str
2543         \int_step_inline:nn \l_tmpa_int {
2544             \str_put_right:Nn \l_tmpa_str i
2545         }
2546         \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
2547     }
2548 } {
2549     \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
2550     \prop_put:Nnx \l_tmpa_prop { arity }
2551     { \str_count:N \l_stex_symdecl_args_str }
2552 }
2553 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2554
2555 \tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
2556     \prop_put:Nnx \l_tmpa_prop { defined }{ false }
2557 }{
2558     \prop_put:Nnx \l_tmpa_prop { defined }{ true }
2559 }
2560
2561 % argnames
2562
2563 \clist_clear:N \l_tmpa_clist
2564 \int_step_inline:nn {\prop_item:Nn \l_tmpa_prop {arity}} {
2565     \clist_if_empty:NTF \l_stex_symdecl_argnames_clist {
2566         \clist_put_right:Nn \l_tmpa_clist {##1}
2567     }{
2568         \clist_pop:NN \l_stex_symdecl_argnames_clist \l_tmpa_tl
2569         \exp_args:NNx \clist_put_right:Nn \l_tmpa_clist {\c_dollar_str\l_tmpa_tl}
2570     }

```

```

2571 }
2572 \prop_put:Nn \l_tmpa_prop {argnames} {\clist_use:Nn \l_tmpa_clist ,}
2573
2574 % semantic macro
2575
2576 \bool_if:NT \l_stex_symdecl_make_macro_bool {
2577   \exp_args:Nx \stex_do_up_to_module:n {
2578     \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2579       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2580     }}
2581   }
2582 }
2583
2584 \stex_debug:nn{symbols}{New~symbol:~
2585   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2586   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2587   Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
2588   Definiens:~\exp_not:o { \l_stex_symdecl_definiens_tl}
2589 }
2590
2591 % circular dependencies require this:
2592 \stex_if_do_html:T {
2593   \stex_annotate_invisible:nnn {symdecl} {
2594     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2595   } {
2596     \tl_if_empty:NF \l_stex_symdecl_type_tl {
2597       \stex_annotate_invisible:nnn{type}{\l_stex_symdecl_type_tl$}
2598     }
2599     \stex_annotate_invisible:nnn{args}{\prop_item:Nn \l_tmpa_prop { args }}{}
2600     \stex_annotate_invisible:nnn{macroname}{#1}{}
2601     \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2602       \stex_annotate_invisible:nnn{definiens}{}
2603       {\l_stex_symdecl_definiens_tl$}
2604     }
2605     \str_if_empty:NF \l_stex_symdecl_assoc_type_str {
2606       \stex_annotate_invisible:nnn{assoc_type}{\l_stex_symdecl_assoc_type_str}{}
2607     }
2608     \str_if_empty:NF \l_stex_symdecl_reorder_str {
2609       \stex_annotate_invisible:nnn{reorderargs}{\l_stex_symdecl_reorder_str}{}
2610     }
2611   }
2612 }
2613 \prop_if_exist:cF {
2614   \l_stex_symdecl_
2615   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2616   _prop
2617 } {
2618   \bool_if:NTF \l_stex_symdecl_local_bool \stex_do_up_to_module:x \stex_execute_in_module:
2619     \__stex_symdecl_restore_symbol:nnnnnnnn
2620       {\l_stex_symdecl_name_str}
2621       { \prop_item:Nn \l_tmpa_prop {args} }
2622       { \prop_item:Nn \l_tmpa_prop {arity} }
2623       { \prop_item:Nn \l_tmpa_prop {assoc} }
2624       { \prop_item:Nn \l_tmpa_prop {defined} }

```



```

2625         {\bool_if:NT \l_stex_symdecl_make_macro_bool {#1} }
2626         {\l_stex_current_module_str}
2627         { \prop_item:Nn \l_tmpa_prop {argnames} }
2628     }
2629 }
2630 }
2631 \cs_new_protected:Nn \__stex_symdecl_restore_symbol:nnnnnnnn {
2632   \prop_clear:N \l_tmpa_prop
2633   \prop_put:Nnn \l_tmpa_prop { module } { #7 }
2634   \prop_put:Nnn \l_tmpa_prop { name } { #1 }
2635   \prop_put:Nnn \l_tmpa_prop { args } { #2 }
2636   \prop_put:Nnn \l_tmpa_prop { arity } { #3 }
2637   \prop_put:Nnn \l_tmpa_prop { assocs } { #4 }
2638   \prop_put:Nnn \l_tmpa_prop { defined } { #5 }
2639   \prop_put:Nnn \l_tmpa_prop { argnames } { #8 }
2640   \tl_if_empty:nF{#6}{
2641     \tl_set:cx{#6}{\stex_invoke_symbol:n{\detokenize{#7 ? #1}}}
2642   }
2643   \prop_set_eq:cN{\l_stex_symdecl_ \detokenize{#7 ? #1} _prop}\l_tmpa_prop
2644   \seq_clear:c{\l_stex_symdecl_ \detokenize{#7 ? #1} _notations}
2645 }

```

(End definition for `\stex_symdecl_do:n`. This function is documented on page 87.)

`\textsymdecl`

```

2646
2647 \keys_define:nn { stex / textsymdecl } {
2648   name      .str_set_x:N = \l__stex_symdecl_name_str ,
2649   type      .tl_set:N    = \l__stex_symdecl_type_tl
2650 }
2651
2652 \cs_new_protected:Nn \stex_textsymdecl_args:n {
2653   \str_clear:N \l__stex_symdecl_name_str
2654   \tl_clear:N \l__stex_symdecl_type_tl
2655   \clist_clear:N \l_stex_symdecl_argnames_clist
2656   \keys_set:nn { stex / textsymdecl } { #1 }
2657 }
2658
2659 \NewDocumentCommand \textsymdecl {m O{} m} {
2660   \stex_textsymdecl_args:n { #2 }
2661   \str_if_empty:NTF \l__stex_symdecl_name_str {
2662     \__stex_symdecl_args:n{name=#1,#2}
2663   }{
2664     \__stex_symdecl_args:n{#2}
2665   }
2666   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2667   \stex_symdecl_do:n{#1-sym}
2668   \stex_execute_in_module:n{
2669     \cs_set_nopar:cpn{#1name}{
2670       \ifvmode\hbox_unpack:N\c_empty_box\fi
2671       \ifmmode\hbox{#3}\else#3\fi\hspace
2672     }
2673     \cs_set_nopar:cpn{#1}{
2674       \ifmmode\csname#1-sym\expandafter\endcsname\else

```

```

2675     \ifvmode\hbox_unpack:N\c_empty_box\fi
2676     \symref{#1-sym}{#3}\expandafter\xspace
2677     \fi
2678   }
2679 }
2680 \stex_execute_in_module:x{
2681   \__stex_notation_restore_notation:nnnnn
2682   {\l_stex_current_module_str?\tl_if_empty:N\l_stex_symdecl_name_str{#1}\l_stex_symdecl_name_str{#1}}{0}
2683   {\exp_not:n{\STEXInternalTermMathOMSiiii{\STEXInternalCurrentSymbolStr}{}}{\neginfprec}{\neginfprec}}{0}
2684   \comp{\hbox{#3}}\STEXInternalSymbolAfterInvokationTL
2685   }}}
2686   {}
2687 }
2688 }
2689 \stex_smsmode_do:
2690 }

```

(End definition for `\textsymdecl`. This function is documented on page 19.)

`\stex_get_symbol:n`

```

2691 \str_new:N \l_stex_get_symbol_uri_str
2692
2693 \cs_new_protected:Nn \stex_get_symbol:n {
2694   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2695     \tl_set:Nn \l_tmpa_tl { #1 }
2696     \__stex_symdecl_get_symbol_from_cs:
2697   }{
2698     % argument is a string
2699     % is it a command name?
2700     \cs_if_exist:cTF { #1 }{
2701       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2702       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2703       \str_if_empty:N\l_tmpa_str {
2704         \exp_args:Nx \cs_if_eq:NNTF {
2705           \tl_head:N \l_tmpa_tl
2706         } \stex_invoke_symbol:n {
2707           \__stex_symdecl_get_symbol_from_cs:
2708         }{
2709           \__stex_symdecl_get_symbol_from_string:n { #1 }
2710         }
2711       } {
2712         \__stex_symdecl_get_symbol_from_string:n { #1 }
2713       }
2714     }{
2715       % argument is not a command name
2716       \__stex_symdecl_get_symbol_from_string:n { #1 }
2717       % \l_stex_all_symbols_seq
2718     }
2719   }
2720   \str_if_eq:eeF {
2721     \prop_item:cn {
2722       \l_stex_symdecl\l_stex_get_symbol_uri_str _prop
2723     }{ deprecate }
2724   }{}{

```

```

2725     \msg_warning:nnxx{stex}{warning/deprecated}{
2726     Symbol~\l_stex_get_symbol_uri_str
2727     }{
2728     \prop_item:cn {l_stex_symdecl_l\l_stex_get_symbol_uri_str _prop}{ deprecate }
2729     }
2730 }
2731 }
2732
2733 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2734     \tl_set:Nn \l_tmpa_tl {
2735         \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2736     }
2737     \str_set:Nn \l_tmpa_str { #1 }
2738
2739     %\int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2740
2741     \str_if_in:NnTF \l_tmpa_str ? {
2742         \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
2743         \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
2744         \str_set:Nx \l_tmpb_str {\seq_use:Nn \l_tmpa_seq ?}
2745     }{
2746         \str_clear:N \l_tmpb_str
2747     }
2748     \str_if_empty:NnTF \l_tmpb_str {
2749         \seq_map_inline:Nn \l_stex_all_modules_seq {
2750             \seq_map_inline:cn{c_stex_module_##1_constants}{
2751                 \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2752                     \seq_map_break:n{\seq_map_break:n{
2753                         \tl_set:Nn \l_tmpa_tl {
2754                             \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2755                         }
2756                     }}
2757                 }
2758             }
2759         }
2760     }{
2761         \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpb_str }
2762         \seq_map_inline:Nn \l_stex_all_modules_seq {
2763             \str_if_eq:eeT{ \l_tmpb_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2764                 \seq_map_inline:cn{c_stex_module_##1_constants}{
2765                     \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2766                         \seq_map_break:n{\seq_map_break:n{
2767                             \tl_set:Nn \l_tmpa_tl {
2768                                 \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2769                             }
2770                         }}
2771                     }
2772                 }
2773             }
2774         }
2775     }
2776
2777     \l_tmpa_tl
2778 }

```

```

2779
2780 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
2781   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2782     { \tl_tail:N \l_tmpa_tl }
2783   \tl_if_single:NTF \l_tmpa_tl {
2784     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2785       \exp_after:wN \str_set:Nn \exp_after:wN
2786         \l_stex_get_symbol_uri_str \l_tmpa_tl
2787     }{
2788       % TODO
2789       % tail is not a single group
2790     }
2791   }{
2792     % TODO
2793     % tail is not a single group
2794   }
2795 }

```

(End definition for `\stex_get_symbol:n`. This function is documented on page 87.)

29.2 Notations

```

2796 <@@=stex_notation>
      notation arguments:
2797 \keys_define:nn { stex / notation } {
2798   % lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2799   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2800   prec     .str_set_x:N = \l__stex_notation_prec_str ,
2801   op       .tl_set:N = \l__stex_notation_op_tl ,
2802   primary .bool_set:N = \l__stex_notation_primary_bool ,
2803   primary .default:n = {true} ,
2804   hints    .str_set_x:N = \l__stex_notation_hints_str,
2805   unknown .code:n = \str_set:Nx
2806     \l__stex_notation_variant_str \l_keys_key_str
2807 }
2808
2809 \cs_new_protected:Nn \_stex_notation_args:n {
2810   % \str_clear:N \l__stex_notation_lang_str
2811   \str_clear:N \l__stex_notation_variant_str
2812   \str_clear:N \l__stex_notation_prec_str
2813   \str_clear:N \l__stex_notation_hints_str
2814   \tl_clear:N \l__stex_notation_op_tl
2815   \bool_set_false:N \l__stex_notation_primary_bool
2816
2817   \keys_set:nn { stex / notation } { #1 }
2818 }

```

\notation

```

2819 \NewDocumentCommand \notation { s m O{}} {
2820   \_stex_notation_args:n { #3 }
2821   \tl_clear:N \l_stex_symdecl_definiens_tl
2822   \stex_get_symbol:n { #2 }
2823   \tl_set:Nn \l_stex_notation_after_do_tl {

```

```

2824 \__stex_notation_final:
2825 \IfBooleanTF#1{
2826   \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2827 }{}
2828 \stex_smsmode_do:\ignorespacesandpars
2829 }
2830 \stex_notation_do:nnnnn
2831 { \prop_item:cn {\l_stex_symdecl\l_stex_get_symbol_uri_str _prop } { args } }
2832 { \prop_item:cn { \l_stex_symdecl\l_stex_get_symbol_uri_str _prop } { arity } }
2833 { \l__stex_notation_variant_str }
2834 { \l__stex_notation_prec_str }
2835 }
2836 \stex_deactivate_macro:Nn \notation {module-environments}

```

(End definition for \notation. This function is documented on page 87.)

\stex_notation_do:nnnnn

```

2837 \seq_new:N \l__stex_notation_precedences_seq
2838 \tl_new:N \l__stex_notation_opprec_tl
2839 \int_new:N \l__stex_notation_currarg_int
2840 \tl_new:N \STEXInternalSymbolAfterInvokationTL
2841
2842 \cs_new_protected:Nn \stex_notation_do:nnnnn {
2843   \let\STEXInternalCurrentSymbolStr\relax
2844   \seq_clear:N \l__stex_notation_precedences_seq
2845   \tl_clear:N \l__stex_notation_opprec_tl
2846   \str_set:Nx \l__stex_notation_args_str { #1 }
2847   \str_set:Nx \l__stex_notation_arity_str { #2 }
2848   \str_set:Nx \l__stex_notation_suffix_str { #3 }
2849   \str_set:Nx \l__stex_notation_prec_str { #4 }
2850
2851   % precedences
2852   \str_if_empty:NTF \l__stex_notation_prec_str {
2853     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2854       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2855     }{
2856       \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2857     }
2858   } {
2859     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2860       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2861       \int_step_inline:nn { \l__stex_notation_arity_str } {
2862         \exp_args:NNo
2863         \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2864       }
2865     }{
2866       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2867       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2868         \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2869         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2870           \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2871             \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2872           \seq_map_inline:Nn \l_tmpa_seq {
2873             \seq_put_right:Nn \l__stex_notation_precedences_seq { ##1 }

```

```

2874     }
2875   }
2876   }{
2877     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2878       \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2879     }{
2880       \tl_set:No \l__stex_notation_opprec_tl { 0 }
2881     }
2882   }
2883 }
2884 }
2885
2886 \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2887 \int_step_inline:nn { \l__stex_notation_arity_str } {
2888   \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
2889     \exp_args:NNo
2890     \seq_put_right:No \l__stex_notation_precedences_seq {
2891       \l__stex_notation_opprec_tl
2892     }
2893   }
2894 }
2895 \tl_clear:N \l_stex_notation_dummyargs_tl
2896
2897 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2898   \exp_args:NNe
2899   \cs_set:Npn \l_stex_notation_macrocode_cs {
2900     \STEXInternalTermMathOMSiiii { \STEXInternalCurrentSymbolStr }
2901     { \l__stex_notation_suffix_str }
2902     { \l__stex_notation_opprec_tl }
2903     { \exp_not:n { #5 } }
2904   }
2905   \l_stex_notation_after_do_tl
2906 }{
2907   \str_if_in:NnTF \l__stex_notation_args_str b {
2908     \exp_args:Nne \use:nn
2909     {
2910       \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2911       \cs_set:Npn \l__stex_notation_arity_str } { {
2912         \STEXInternalTermMathOMBiiii { \STEXInternalCurrentSymbolStr }
2913         { \l__stex_notation_suffix_str }
2914         { \l__stex_notation_opprec_tl }
2915         { \exp_not:n { #5 } }
2916       }
2917     }{
2918       \str_if_in:NnTF \l__stex_notation_args_str B {
2919         \exp_args:Nne \use:nn
2920         {
2921           \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2922           \cs_set:Npn \l__stex_notation_arity_str } { {
2923             \STEXInternalTermMathOMBiiii { \STEXInternalCurrentSymbolStr }
2924             { \l__stex_notation_suffix_str }
2925             { \l__stex_notation_opprec_tl }
2926             { \exp_not:n { #5 } }
2927           } }

```

```

2928   }{
2929     \exp_args:Nne \use:nn
2930     {
2931       \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2932       \cs_set:Npn \l__stex_notation_arity_str } { {
2933         \STEXInternalTermMathOMAiiai { \STEXInternalCurrentSymbolStr }
2934         { \l__stex_notation_suffix_str }
2935         { \l__stex_notation_opprec_tl }
2936         { \exp_not:n { #5 } }
2937       } }
2938     }
2939   }
2940
2941   \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2942   \int_zero:N \l__stex_notation_currarg_int
2943   \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2944   \__stex_notation_arguments:
2945 }
2946 }

```

(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

```

2947 \cs_new_protected:Nn \__stex_notation_arguments: {
2948   \int_incr:N \l__stex_notation_currarg_int
2949   \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2950     \l_stex_notation_after_do_tl
2951   }{
2952     \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2953     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2954     \str_if_eq:VnTF \l_tmpa_str a {
2955       \__stex_notation_argument_assoc:nn{a}
2956     }{
2957       \str_if_eq:VnTF \l_tmpa_str B {
2958         \__stex_notation_argument_assoc:nn{B}
2959       }{
2960         \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpb_str
2961         \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2962           { \STEXInternalTermMathArgiii
2963             { \l_tmpa_str\int_use:N \l__stex_notation_currarg_int }
2964             { \l_tmpb_str }
2965             { ###\int_use:N \l__stex_notation_currarg_int }
2966           }
2967         }
2968         \__stex_notation_arguments:
2969       }
2970     }
2971   }
2972 }

```

(End definition for `__stex_notation_arguments:.`)

`__stex_notation_argument_assoc:nn`

```

2973 \cs_new_protected:Nn \__stex_notation_argument_assoc:nn {

```

```

2974
2975 \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2976   {\l__stex_notation_arity_str}{
2977     #2
2978   }
2979 \int_zero:N \l_tmpa_int
2980 \tl_clear:N \l_tmpa_tl
2981 \str_map_inline:Nn \l__stex_notation_args_str {
2982   \int_incr:N \l_tmpa_int
2983   \tl_put_right:Nx \l_tmpa_tl {
2984     \str_if_eq:nnTF {##1}{a}{ } {} }{
2985     \str_if_eq:nnTF {##1}{B}{ } {} }{
2986     {\_stex_term_arg:nn{##1}\int_use:N \l_tmpa_int}{##### \int_use:N \l_tmpa_int}
2987   }
2988 }
2989 }
2990 }
2991 \exp_after:wN\exp_after:wN\exp_after:wN \def
2992 \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2993 \exp_after:wN\exp_after:wN\exp_after:wN ##
2994 \exp_after:wN\exp_after:wN\exp_after:wN 1
2995 \exp_after:wN\exp_after:wN\exp_after:wN ##
2996 \exp_after:wN\exp_after:wN\exp_after:wN 2
2997 \exp_after:wN\exp_after:wN\exp_after:wN {
2998   \exp_after:wN \exp_after:wN \exp_after:wN
2999   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
3000     \exp_after:wN \l_tmpa_cs \l_tmpa_tl
3001   }
3002 }
3003
3004 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
3005 \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
3006   \STEXInternalTermMathAssocArgiiii
3007   { \int_use:N \l__stex_notation_currarg_int }
3008   { \l_tmpa_str }
3009   { ####\int_use:N \l__stex_notation_currarg_int }
3010   { \l_tmpa_cs {####1} {####2} }
3011   {#1}
3012 } }
3013 \__stex_notation_arguments:
3014 }

```

(End definition for _stex_notation_argument_assoc:nn.)

_stex_notation_final: Called after processing all notation arguments

```

3015 \cs_new_protected:Nn \__stex_notation_restore_notation:nnnnn {
3016   \cs_generate_from_arg_count:cNnn{stex_notation_\detokenize{#1} \c_hash_str \detokenize{#2}}
3017   \cs_set_nopar:Npn {#3}{#4}
3018   \tl_if_empty:nF {#5}{
3019     \tl_set:cn{stex_op_notation_\detokenize{#1} \c_hash_str \detokenize{#2}_cs}{\comp{ #5 }
3020   }
3021   \seq_if_exist:cT { l_stex_symdecl_\detokenize{#1} _notations }{
3022     \seq_put_right:cx { l_stex_symdecl_\detokenize{#1} _notations } { \detokenize{#2} }
3023   }

```



```

3024 }
3025
3026 \cs_new_protected:Nn \__stex_notation_final: {
3027
3028   \stex_execute_in_module:x {
3029     \__stex_notation_restore_notation:nnnnn
3030     {\l_stex_get_symbol_uri_str}
3031     {\l__stex_notation_suffix_str}
3032     {\l__stex_notation_arity_str}
3033     {
3034       \exp_after:wN \exp_after:wN \exp_after:wN
3035       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
3036       { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \STEXInt
3037     }
3038     {\exp_args:No \exp_not:n \l__stex_notation_op_tl }
3039   }
3040
3041   \stex_debug:nn{symbols}{
3042     Notation~\l__stex_notation_suffix_str
3043     ~for~\l_stex_get_symbol_uri_str^^J
3044     Operator~precedence:~\l__stex_notation_opprec_tl^^J
3045     Argument~precedences:~
3046     \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
3047     Notation: \cs_meaning:c {
3048       stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
3049       \l__stex_notation_suffix_str
3050       _cs
3051     }
3052   }
3053   % HTML annotations
3054   \stex_if_do_html:T {
3055     \stex_annotate_invisible:nnn { notation }
3056     { \l_stex_get_symbol_uri_str } {
3057       \stex_annotate_invisible:nnn { notationfragment }
3058       { \l__stex_notation_suffix_str }{}
3059       \stex_annotate_invisible:nnn { precedence }
3060       { \l__stex_notation_prec_str }{}
3061
3062       \int_zero:N \l_tmpa_int
3063       \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
3064       \tl_clear:N \l_tmpa_tl
3065       \int_step_inline:nn { \l__stex_notation_arity_str }{
3066         \int_incr:N \l_tmpa_int
3067         \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
3068         \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_rema
3069         \str_if_eq:VnTF \l_tmpb_str a {
3070           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3071             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3072             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3073           } }
3074         }{
3075           \str_if_eq:VnTF \l_tmpb_str B {
3076             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3077               \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,

```

```

3078         \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{\}
3079     } }
3080   }{
3081     \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3082       \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{\}
3083     } }
3084   }
3085 }
3086 }
3087 \stex_annotate_invisible:nnn { notationcomp }{ }{
3088   \str_set:Nx \STEXInternalCurrentSymbolStr {\l_stex_get_symbol_uri_str }
3089   $ \exp_args:Nno \use:nn { \use:c {
3090     stex_notation_ \STEXInternalCurrentSymbolStr
3091     \c_hash_str \l__stex_notation_suffix_str_cs
3092   } } { \l_tmpa_tl } $
3093 }
3094 \tl_if_empty:NF \l__stex_notation_op_tl {
3095   \stex_annotate_invisible:nnn { notationopcomp }{ }{
3096     $\l__stex_notation_op_tl$
3097   }
3098 }
3099 }
3100 }
3101 }

```

(End definition for `_stex_notation_final:`.)

`\setnotation`

```

3102 \keys_define:nn { stex / setnotation } {
3103   % lang      .tl_set_x:N = \l__stex_notation_lang_str ,
3104   variant .tl_set_x:N = \l__stex_notation_variant_str ,
3105   unknown .code:n      = \str_set:Nx
3106     \l__stex_notation_variant_str \l_keys_key_str
3107 }
3108
3109 \cs_new_protected:Nn \_stex_setnotation_args:n {
3110   % \str_clear:N \l__stex_notation_lang_str
3111   \str_clear:N \l__stex_notation_variant_str
3112   \keys_set:nn { stex / setnotation } { #1 }
3113 }
3114
3115 \cs_new_protected:Nn \_stex_notation_setnotation:nn {
3116   \seq_if_exist:cnTF{l_stex_symdecl_#1_notations}{
3117     \seq_remove_all:cn { l_stex_symdecl_#1_notations }{ #2 }
3118     \seq_put_left:cn { l_stex_symdecl_#1_notations }{ #2 }
3119   }
3120 }
3121
3122 \cs_new_protected:Nn \stex_setnotation:n {
3123   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1_notations }
3124     { \l__stex_notation_variant_str }{
3125     \stex_execute_in_module:x{ \_stex_notation_setnotation:nn {#1}{\l__stex_notation_vari
3126     \stex_debug:nn {notations}{
3127       Setting~default~notation~

```

```

3128         {\l__stex_notation_variant_str }~for~
3129         #1 \\\
3130         \expandafter\meaning\csname
3131         l_stex_symdecl_#1 _notations\endcsname
3132     }
3133 }{
3134     \msg_error:nnxx{stex}{unknownnotation}{\l__stex_notation_variant_str}{#1}
3135 }
3136 }
3137
3138 \NewDocumentCommand \setnotation {m m} {
3139     \stex_get_symbol:n { #1 }
3140     \_stex_setnotation_args:n { #2 }
3141     \stex_setnotation:n{\l_stex_get_symbol_uri_str}
3142     \stex_smsmode_do:\ignorespacesandpars
3143 }
3144
3145 \cs_new_protected:Nn \stex_copy_notations:nn {
3146     \stex_debug:nn {notations}{
3147         Copying~notations~from~#2~to~#1\\
3148         \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
3149     }
3150     \tl_clear:N \l_tmpa_tl
3151     \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
3152         \tl_put_right:Nn \l_tmpa_tl { {##### #1} }
3153     }
3154     \seq_map_inline:cn {l_stex_symdecl_#2_notations}{\begingroup
3155         \stex_debug:nn{Here}{Here:~##1}
3156         \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
3157         \edef \l_tmpa_tl {
3158             \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
3159             \exp_after:wN\exp_after:wN\exp_after:wN {
3160                 \exp_after:wN \l_tmpa_cs \l_tmpa_tl
3161             }
3162         }
3163
3164         \exp_after:wN \def \exp_after:wN \l_tmpa_tl
3165         \exp_after:wN #####\exp_after:wN 1 \exp_after:wN #####\exp_after:wN 2
3166         \exp_after:wN { \l_tmpa_tl }
3167
3168         \edef \l_tmpa_tl {
3169             \exp_after:wN \exp_not:n \exp_after:wN {
3170                 \l_tmpa_tl {##### 1}{##### 2}
3171             }
3172         }
3173
3174         \stex_debug:nn{Here}{Here:~\expandafter\detokenize\expandafter{\l_tmpa_tl}}
3175
3176     \stex_execute_in_module:x {
3177         \_stex_notation_restore_notation:nnnnn
3178         {#1}{##1}
3179         { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }
3180         { \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl} }
3181         {

```

```

3182         \cs_if_exist:cT{stex_op_notation_ #2\c_hash_str ##1 _cs}{
3183             \exp_args:NNo\exp_args:No\exp_not:n{\csname stex_op_notation_ #2\c_hash_str ##1
3184             }
3185         }
3186     }\endgroup
3187 }
3188 }
3189
3190 \NewDocumentCommand \copynotation {m m} {
3191     \stex_get_symbol:n { #1 }
3192     \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
3193     \stex_get_symbol:n { #2 }
3194     \exp_args:Noo
3195     \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
3196     \stex_smsmode_do:\ignorespacesandpars
3197 }
3198

```

(End definition for \setnotation. This function is documented on page 19.)

\symdef

```

3199 \keys_define:nn { stex / symdef } {
3200     name .str_set_x:N = \l_stex_symdecl_name_str ,
3201     args .str_set_x:N = \l_stex_symdecl_args_str ,
3202     type .tl_set:N = \l_stex_symdecl_type_tl ,
3203     def .tl_set:N = \l_stex_symdecl_definiens_tl ,
3204     reorder .str_set_x:N = \l_stex_symdecl_reorder_str ,
3205     op .tl_set:N = \l__stex_notation_op_tl ,
3206     % lang .str_set_x:N = \l__stex_notation_lang_str ,
3207     variant .str_set_x:N = \l__stex_notation_variant_str ,
3208     prec .str_set_x:N = \l__stex_notation_prec_str ,
3209     argnames .clist_set:N = \l_stex_symdecl_argnames_clist ,
3210     assoc .choices:nn =
3211         {bin,binl,binr,pre,conj,pwconj}
3212         {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
3213     unknown .code:n = \str_set:Nx
3214         \l__stex_notation_variant_str \l_keys_key_str
3215 }
3216
3217 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
3218     \str_clear:N \l_stex_symdecl_name_str
3219     \str_clear:N \l_stex_symdecl_args_str
3220     \str_clear:N \l_stex_symdecl_assoctype_str
3221     \str_clear:N \l_stex_symdecl_reorder_str
3222     \bool_set_false:N \l_stex_symdecl_local_bool
3223     \tl_clear:N \l_stex_symdecl_type_tl
3224     \tl_clear:N \l_stex_symdecl_definiens_tl
3225     \clist_clear:N \l_stex_symdecl_argnames_clist
3226     % \str_clear:N \l__stex_notation_lang_str
3227     \str_clear:N \l__stex_notation_variant_str
3228     \str_clear:N \l__stex_notation_prec_str
3229     \tl_clear:N \l__stex_notation_op_tl
3230
3231     \keys_set:nn { stex / symdef } { #1 }

```

```

3232 }
3233
3234 \NewDocumentCommand \symdef { m O{} } {
3235   \_stex_notation_symdef_args:n { #2 }
3236   \bool_set_true:N \l_stex_symdecl_make_macro_bool
3237   \stex_symdecl_do:n { #1 }
3238   \tl_set:Nn \l_stex_notation_after_do_tl {
3239     \_stex_notation_final:
3240     \stex_smsmode_do:\ignorespacesandpars
3241   }
3242   \str_set:Nx \l_stex_get_symbol_uri_str {
3243     \l_stex_current_module_str ? \l_stex_symdecl_name_str
3244   }
3245   \exp_args:Nx \stex_notation_do:nnnnn
3246     { \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str_prop } { args } }
3247     { \prop_item:cn { l_stex_symdecl\l_stex_get_symbol_uri_str_prop } { arity } }
3248     { \l__stex_notation_variant_str }
3249     { \l__stex_notation_prec_str}
3250 }
3251 \stex_deactivate_macro:Nn \symdef {module~environments}
3252
3253 \keys_define:nn { stex / mmtdef } {
3254   name .str_set_x:N = \l_stex_symdecl_name_str ,
3255   args .str_set_x:N = \l_stex_symdecl_args_str ,
3256   reorder .str_set_x:N = \l_stex_symdecl_reorder_str ,
3257   op .tl_set:N = \l__stex_notation_op_tl ,
3258   % lang .str_set_x:N = \l__stex_notation_lang_str ,
3259   variant .str_set_x:N = \l__stex_notation_variant_str ,
3260   prec .str_set_x:N = \l__stex_notation_prec_str ,
3261   argnames .clist_set:N = \l_stex_symdecl_argnames_clist ,
3262   assoc .choices:nn =
3263     {bin,binl,binr,pre,conj,pwconj}
3264     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
3265   unknown .code:n = \str_set:Nx
3266     \l__stex_notation_variant_str \l_keys_key_str
3267 }
3268 \cs_new_protected:Nn \_stex_mmtdef_args:n {
3269   \str_clear:N \l_stex_symdecl_name_str
3270   \str_clear:N \l_stex_symdecl_args_str
3271   \str_clear:N \l_stex_symdecl_assoctype_str
3272   \str_clear:N \l_stex_symdecl_reorder_str
3273   \clist_clear:N \l_stex_symdecl_argnames_clist
3274   % \str_clear:N \l__stex_notation_lang_str
3275   \str_clear:N \l__stex_notation_variant_str
3276   \str_clear:N \l__stex_notation_prec_str
3277   \tl_clear:N \l__stex_notation_op_tl
3278
3279   \keys_set:nn { stex / mmtdef } { #1 }
3280 }
3281
3282 \NewDocumentCommand \mmtdef {m O{} } {}
3283   \_stex_mmtdef_args:n{ #2 }
3284   \bool_set_true:N \l_stex_symdecl_make_macro_bool
3285   \str_if_empty:NT \l_stex_symdecl_name_str {

```

```

3286   \str_set:Nx \l_stex_symdecl_name_str { #1 }
3287 }
3288 %\tl_set:Nx \l_stex_symdecl_definiens_tl {
3289 %   \stex_annotate:nnn{ OMID }{
3290 %     \l_stex_module_mmtfor_str?\l_stex_symdecl_name_str
3291 %   }{}
3292 %}
3293 \stex_symdecl_do:n { #1 }
3294 \MMTrule{rules.stex.mmt.kwarc.info?SubstitutionRule}{
3295   \stex_annotate:nnn{ OMID }{
3296     \l_stex_current_module_str ? \l_stex_symdecl_name_str
3297   }{},
3298   \stex_annotate:nnn{ OMID }{
3299     \l_stex_module_mmtfor_str?\l_stex_symdecl_name_str
3300   }{}
3301 }
3302 \tl_set:Nn \l_stex_notation_after_do_tl {
3303   \__stex_notation_final:
3304   \stex_smsmode_do:\ignorespacesandpars
3305 }
3306 \str_set:Nx \l_stex_get_symbol_uri_str {
3307   \l_stex_current_module_str ? \l_stex_symdecl_name_str
3308 }
3309 \exp_args:Nx \stex_notation_do:nnnnn
3310 { \prop_item:cn {l_stex_symdecl\_l_stex_get_symbol_uri_str_prop } { args } }
3311 { \prop_item:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str_prop } { arity } }
3312 { \l_stex_notation_variant_str }
3313 { \l__stex_notation_prec_str}
3314 }

```

(End definition for `\symdef`. This function is documented on page 87.)

29.3 Variables

```

3315 <@@=stex_variables>
3316
3317 \keys_define:nn { stex / vardef } {
3318   name      .str_set_x:N = \l__stex_variables_name_str ,
3319   args      .str_set_x:N = \l__stex_variables_args_str ,
3320   type      .tl_set:N    = \l__stex_variables_type_tl ,
3321   def       .tl_set:N    = \l__stex_variables_def_tl ,
3322   op        .tl_set:N    = \l__stex_variables_op_tl ,
3323   prec      .str_set_x:N = \l__stex_variables_prec_str ,
3324   reorder   .str_set_x:N = \l__stex_variables_reorder_str ,
3325   argnames   .clist_set:N = \l__stex_variables_argnames_clist ,
3326   assoc     .choices:nn  =
3327     {bin,binl,binr,pre,conj,pwconj}
3328     {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
3329   bind      .choices:nn  =
3330     {forall,exists}
3331     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
3332 }
3333
3334 \cs_new_protected:Nn \__stex_variables_args:n {

```

```

3335 \str_clear:N \l__stex_variables_name_str
3336 \str_clear:N \l__stex_variables_args_str
3337 \str_clear:N \l__stex_variables_prec_str
3338 \str_clear:N \l__stex_variables_assoctype_str
3339 \str_clear:N \l__stex_variables_reorder_str
3340 \str_clear:N \l__stex_variables_bind_str
3341 \tl_clear:N \l__stex_variables_type_tl
3342 \tl_clear:N \l__stex_variables_def_tl
3343 \tl_clear:N \l__stex_variables_op_tl
3344 \clist_clear:N \l__stex_variables_argnames_clist
3345
3346 \keys_set:nn { stex / vardef } { #1 }
3347 }
3348
3349 \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{}} {
3350   \__stex_variables_args:n {#2}
3351   \str_if_empty:NT \l__stex_variables_name_str {
3352     \str_set:Nx \l__stex_variables_name_str { #1 }
3353   }
3354   \prop_clear:N \l_tmpa_prop
3355   \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
3356
3357   \int_zero:N \l_tmpb_int
3358   \bool_set_true:N \l_tmpa_bool
3359   \str_map_inline:Nn \l__stex_variables_args_str {
3360     \token_case_meaning:NnF ##1 {
3361       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
3362       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
3363       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
3364       {\tl_to_str:n a} {
3365         \bool_set_false:N \l_tmpa_bool
3366         \int_incr:N \l_tmpb_int
3367       }
3368       {\tl_to_str:n B} {
3369         \bool_set_false:N \l_tmpa_bool
3370         \int_incr:N \l_tmpb_int
3371       }
3372     }{
3373       \msg_error:nnxx{stex}{error/wrongargs}{
3374         variable~\l__stex_variables_name_str
3375       }{##1}
3376     }
3377   }
3378   \bool_if:NTF \l_tmpa_bool {
3379     % possibly numeric
3380     \str_if_empty:NTF \l__stex_variables_args_str {
3381       \prop_put:Nnn \l_tmpa_prop { args } {}
3382       \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
3383     }{
3384       \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
3385       \prop_put:Nnn \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
3386       \str_clear:N \l_tmpa_str
3387       \int_step_inline:nn \l_tmpa_int {
3388         \str_put_right:Nn \l_tmpa_str i

```

```

3389     }
3390     \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
3391     \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
3392   }
3393 } {
3394   \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
3395   \prop_put:Nnx \l_tmpa_prop { arity }
3396   { \str_count:N \l__stex_variables_args_str }
3397 }
3398 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
3399 \tl_set:cx { #1 } { \stex_invoke_variable:n { \l__stex_variables_name_str } }
3400
3401 % argnames
3402
3403 \clist_clear:N \l_tmpa_clist
3404 \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } } {
3405   \clist_if_empty:NTF \l__stex_variables_argnames_clist {
3406     \clist_put_right:Nn \l_tmpa_clist { ##1 }
3407   } {
3408     \clist_pop:NN \l__stex_variables_argnames_clist \l_tmpa_tl
3409     \exp_args:NNx \clist_put_right:Nn \l_tmpa_clist { \c_dollar_str \l_tmpa_tl }
3410   }
3411 }
3412 \prop_put:Nnx \l_tmpa_prop { argnames } { \clist_use:Nn \l_tmpa_clist , }
3413
3414
3415 \prop_set_eq:cN { l_stex_symdecl_var://\l__stex_variables_name_str _prop } \l_tmpa_prop
3416
3417 \tl_if_empty:NF \l__stex_variables_op_tl {
3418   \cs_set:cpx {
3419     stex_var_op_notation_ \l__stex_variables_name_str _cs
3420   } { \exp_not:N \comp { \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
3421 }
3422
3423 \tl_set:Nn \l_stex_notation_after_do_tl {
3424   \exp_args:Nne \use:nn {
3425     \cs_generate_from_arg_count:cNnn { stex_var_notation_ \l__stex_variables_name_str _cs }
3426     \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
3427   } {{
3428     \exp_after:wN \exp_after:wN \exp_after:wN
3429     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
3430     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \STEXInter
3431   }}
3432 \stex_if_do_html:T {
3433   \stex_annotate_invisible:nnn { vardecl } { \l__stex_variables_name_str } {
3434     \stex_annotate_invisible:nnn { precedence }
3435     { \l__stex_variables_prec_str } {}
3436     \tl_if_empty:NF \l__stex_variables_type_tl { \stex_annotate_invisible:nnn { type } { } { $ \l
3437     \stex_annotate_invisible:nnn { args } { \l__stex_variables_args_str } {}
3438     \stex_annotate_invisible:nnn { macroname } { #1 } {}
3439     \tl_if_empty:NF \l__stex_variables_def_tl {
3440       \stex_annotate_invisible:nnn { definiens } {}
3441       { $ \l__stex_variables_def_tl $ }
3442     }

```



```

3443 \str_if_empty:NF \l__stex_variables_assoctype_str {
3444 \stex_annotate_invisible:nnn{assoctype}{\l__stex_variables_assoctype_str}{}}
3445 }
3446 \str_if_empty:NF \l__stex_variables_reorder_str {
3447 \stex_annotate_invisible:nnn{reorderargs}{\l__stex_variables_reorder_str}{}}
3448 }
3449 \int_zero:N \l_tmpa_int
3450 \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
3451 \tl_clear:N \l_tmpa_tl
3452 \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
3453 \int_incr:N \l_tmpa_int
3454 \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
3455 \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables
3456 \str_if_eq:VnTF \l_tmpb_str a {
3457 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3458 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3459 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3460 } }
3461 }{
3462 \str_if_eq:VnTF \l_tmpb_str B {
3463 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3464 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3465 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3466 } }
3467 }{
3468 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3469 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
3470 } }
3471 }
3472 }
3473 }
3474 \stex_annotate_invisible:nnn { notationcomp }{}{
3475 \str_set:Nx \STEXInternalCurrentSymbolStr {var://\l__stex_variables_name_str }
3476 $ \exp_args:Nno \use:nn { \use:c {
3477 stex_var_notation_\l__stex_variables_name_str _cs
3478 } } { \l_tmpa_tl } $
3479 }
3480 \tl_if_empty:NF \l__stex_variables_op_tl {
3481 \stex_annotate_invisible:nnn { notationopcomp }{}{
3482 $\l__stex_variables_op_tl$
3483 }
3484 }
3485 }
3486 \str_if_empty:NF \l__stex_variables_bind_str {
3487 \stex_annotate_invisible:nnn {bindtype}{\l__stex_variables_bind_str,\l__stex_variab
3488 }
3489 }\ignorespacesandpars
3490 }
3491
3492 \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
3493 }
3494
3495 \cs_new:Nn \stex_reset:N {
3496 \tl_if_exist:NTF #1 {

```

```

3497 \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
3498 }{
3499 \let \exp_not:N #1 \exp_not:N \undefined
3500 }
3501 }
3502
3503 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
3504 \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
3505 \exp_args:Nnx \use:nn {
3506 % TODO
3507 \stex_annotate_invisible:nnn {vardecl}{\clist_use:Nn\l__stex_variables_names,}{
3508 #2
3509 }
3510 }{
3511 \stex_reset:N \varnot
3512 \stex_reset:N \vartype
3513 \stex_reset:N \vardefi
3514 }
3515 }
3516
3517 \NewDocumentCommand \vardef { s } {
3518 \IfBooleanTF#1 {
3519 \__stex_variables_do_complex:nn
3520 }{
3521 \__stex_variables_do_simple:nnn
3522 }
3523 }
3524
3525 \NewDocumentCommand \svar { 0{} m }{
3526 \tl_if_empty:nTF {#1}{
3527 \str_set:Nn \l_tmpa_str { #2 }
3528 }{
3529 \str_set:Nn \l_tmpa_str { #1 }
3530 }
3531 \stex_term_omv:nn {
3532 var://\l_tmpa_str
3533 }{
3534 \exp_args:Nnx \use:nn {
3535 \def\comp{\_varcomp}
3536 \str_set:Nx \STEXInternalCurrentSymbolStr { var://\l_tmpa_str }
3537 \comp{ #2 }
3538 }{
3539 \stex_reset:N \comp
3540 \stex_reset:N \STEXInternalCurrentSymbolStr
3541 }
3542 }
3543 }
3544
3545
3546
3547 \keys_define:nn { stex / varseq } {
3548 name .str_set_x:N = \l__stex_variables_name_str ,
3549 args .int_set:N = \l__stex_variables_args_int ,
3550 type .tl_set:N = \l__stex_variables_type_tl ,

```

```

3551 mid .tl_set:N = \l__stex_variables_mid_tl ,
3552 bind .choices:nn =
3553 {forall,exists}
3554 {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
3555 }
3556
3557 \cs_new_protected:Nn \__stex_variables_seq_args:n {
3558 \str_clear:N \l__stex_variables_name_str
3559 \int_set:Nn \l__stex_variables_args_int 1
3560 \tl_clear:N \l__stex_variables_type_tl
3561 \str_clear:N \l__stex_variables_bind_str
3562
3563 \keys_set:nn { stex / varseq } { #1 }
3564 }
3565
3566 \NewDocumentCommand \varseq {m O{} m m m}{
3567 \__stex_variables_seq_args:n { #2 }
3568 \str_if_empty:NT \l__stex_variables_name_str {
3569 \str_set:Nx \l__stex_variables_name_str { #1 }
3570 }
3571 \prop_clear:N \l_tmpa_prop
3572 \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
3573
3574 \seq_set_from_clist:Nn \l_tmpa_seq {#3}
3575 \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
3576 \msg_error:nnxx{stex}{error/seqlength}
3577 {\int_use:N \l__stex_variables_args_int}
3578 {\seq_count:N \l_tmpa_seq}
3579 }
3580 \seq_set_from_clist:Nn \l_tmpb_seq {#4}
3581 \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
3582 \msg_error:nnxx{stex}{error/seqlength}
3583 {\int_use:N \l__stex_variables_args_int}
3584 {\seq_count:N \l_tmpb_seq}
3585 }
3586 \prop_put:Nnn \l_tmpa_prop {starts} {#3}
3587 \prop_put:Nnn \l_tmpa_prop {ends} {#4}
3588
3589 \cs_generate_from_arg_count:cNnn {stex_varseq\l__stex_variables_name_str _cs}
3590 \cs_set:Npn {\int_use:N \l__stex_variables_args_int} { #5 }
3591
3592 % argnames
3593
3594 \clist_clear:N \l_tmpa_clist
3595 \int_step_inline:nn {\l__stex_variables_args_int} {
3596 \clist_put_right:Nn \l_tmpa_clist {##1}
3597 }
3598 \prop_put:Nnx \l_tmpa_prop {argnames} {\clist_use:Nn \l_tmpa_clist ,}
3599
3600
3601
3602
3603 \exp_args:NNo \tl_set:N \l_tmpa_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}
3604 \int_step_inline:nn \l__stex_variables_args_int {

```

```

3605 \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
3606 }
3607 \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
3608 \tl_put_right:Nn \l_tmpa_tl {\,\ellipses,}
3609 \tl_if_empty:NF \l__stex_variables_mid_tl {
3610 \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
3611 \tl_put_right:Nn \l_tmpa_tl {\,\ellipses,}
3612 }
3613 \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3614 \int_step_inline:nn \l__stex_variables_args_int {
3615 \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
3616 }
3617 \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
3618 \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
3619
3620
3621 \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl
3622
3623 \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}
3624
3625 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3626
3627 \int_step_inline:nn \l__stex_variables_args_int {
3628 \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
3629 \STEXInternalTermMathArgiii{i##1}{0}{\exp_not:n{####}##1}
3630 }}
3631 }
3632
3633 \tl_set:Nx \l_tmpa_tl {
3634 \STEXInternalTermMathOMaiiii { varseq://\l__stex_variables_name_str}{0}{
3635 \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
3636 }
3637 }
3638
3639 \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \STEXInternalSymbolAfterInvokationTL} }
3640
3641 \exp_args:Nno \use:nn {
3642 \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
3643 \cs_set:Npn {\int_use:N \l__stex_variables_args_int}{\l_tmpa_tl}
3644
3645 \stex_debug:nn{sequences}{New~Sequence:~
3646 \expandafter\meaning\csname stex_varseq_\l__stex_variables_name_str _cs\endcsname\\~\\
3647 \prop_to_keyval:N \l_tmpa_prop
3648 }
3649 \prop_set_eq:cN {\l_stex_symdecl_varseq://\l__stex_variables_name_str _prop}\l_tmpa_prop
3650
3651 \stex_if_do_html:T{\stex_annotate_invisible:nnn{varseq}{\l__stex_variables_name_str}{
3652 \tl_if_empty:NF \l__stex_variables_type_tl {
3653 \stex_annotate:nnn {type}{\{$\l__stex_variables_type_tl$}
3654 }
3655 \stex_annotate:nnn {args}{\int_use:N \l__stex_variables_args_int}{}
3656 \str_if_empty:NF \l__stex_variables_bind_str {
3657 \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{}
3658 }

```

```

3659 \stex_annotate:nnn{startindex}{#3$}
3660 \stex_annotate:nnn{endindex}{#4$}
3661
3662 \tl_clear:N \l_tmpa_tl
3663 \int_step_inline:nn \l__stex_variables_args_int {
3664   \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3665     \stex_annotate:nnn{argmarker}{##1}{}
3666   } }
3667 }
3668 \stex_annotate_invisible:nnn { notationcomp }{}{
3669   \str_set:Nx \STEXInternalCurrentSymbolStr {varseq://\l__stex_variables_name_str }
3670   $ \exp_args:Nno \use:nn { \use:c {
3671     stex_varseq_\l__stex_variables_name_str _cs
3672   } } { \l_tmpa_tl } $
3673 }
3674 \stex_annotate_invisible:nnn { notationopcomp }{}{
3675   $ \prop_item:Nn \l_tmpa_prop { notation } $
3676 }
3677
3678 }}
3679
3680 \ignorespacesandpars
3681 }
3682
3683
3684 \keys_define:nn { stex / mmtdecl } {
3685   name      .str_set_x:N = \l_stex_symdecl_name_str ,
3686   args      .str_set_x:N = \l_stex_symdecl_args_str ,
3687   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,
3688   reorder   .str_set_x:N = \l_stex_symdecl_reorder_str ,
3689   argnames  .clist_set:N = \l_stex_symdecl_argnames_clist ,
3690   assoc     .choices:nn =
3691     {bin,binl,binr,pre,conj,pwconj}
3692     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}
3693 }
3694
3695 \cs_new_protected:Nn \_stex_mmtdecl_args:n {
3696   \str_clear:N \l_stex_symdecl_name_str
3697   \str_clear:N \l_stex_symdecl_args_str
3698   \str_clear:N \l_stex_symdecl_deprecate_str
3699   \str_clear:N \l_stex_symdecl_reorder_str
3700   \str_clear:N \l_stex_symdecl_assoctype_str
3701   \bool_set_false:N \l_stex_symdecl_local_bool
3702   \clist_clear:N \l_stex_symdecl_argnames_clist
3703
3704   \keys_set:nn { stex / symdecl } { #1 }
3705 }
3706
3707 \NewDocumentCommand \mmtdecl { s m O{} } {
3708   \_stex_mmtdecl_args:n{#3}
3709   \IfBooleanTF #1 {
3710     \bool_set_false:N \l_stex_symdecl_make_macro_bool
3711   } {
3712     \bool_set_true:N \l_stex_symdecl_make_macro_bool

```

```

3713 }
3714 \str_if_empty:NT \l_stex_symdecl_name_str {
3715   \str_set:Nx \l_stex_symdecl_name_str { #1 }
3716 }
3717 %\tl_set:Nx \l_stex_symdecl_definiens_tl {
3718 % \stex_annotate:nnn{ OMID }{
3719 %   \l_stex_module_mmtfor_str?\l_stex_symdecl_name_str
3720 % }{}
3721 %}
3722 \stex_symdecl_do:n{#2}
3723 \MMTrule{rules.stex.mmt.kwarc.info?SubstitutionRule}{
3724   \stex_annotate:nnn{ OMID }{
3725     \l_stex_current_module_str ? \l_stex_symdecl_name_str
3726   }{},
3727   \stex_annotate:nnn{ OMID }{
3728     \l_stex_module_mmtfor_str?\l_stex_symdecl_name_str
3729   }{}
3730 }
3731 \stex_smsmode_do:
3732 }
3733
3734 \stex_deactivate_macro:Nn \mmtdecl {mmtinterface~environments}
3735 \stex_deactivate_macro:Nn \mmtdef {mmtinterface~environments}
3736
3737 </package>

```

Chapter 30

STEX -Terms Implementation

```
3738 <*package>
3739
3740 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
3741
3742 <@@=stex_terms>
3743
3744 Warnings and error messages
3745 \msg_new:nnn{stex}{error/nonotation}{
3746   Symbol~#1~invoked,~but~has~no~notation#2!
3747 }
3748 \msg_new:nnn{stex}{error/notationarg}{
3749   Error~in~parsing~notation~#1
3750 }
3751 \msg_new:nnn{stex}{error/noop}{
3752   Symbol~#1~has~no~operator~notation~for~notation~#2
3753 }
3754 \msg_new:nnn{stex}{error/notallowed}{
3755   Symbol~invocation~#1~not~allowed~in~notation~component~of~#2
3756 }
3757 \msg_new:nnn{stex}{error/doubleargument}{
3758   Argument~#1~of~symbol~#2~already~assigned
3759 }
3760 \msg_new:nnn{stex}{error/overarity}{
3761   Argument~#1~invalid~for~symbol~#2~with~arity~#3
3762 }
```

30.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
3762
3763
3764 \bool_new:N \l_stex_allow_semantic_bool
3765 \bool_set_true:N \l_stex_allow_semantic_bool
3766
```

```

3767 \cs_new_protected:Nn \stex_invoke_symbol:n {
3768   \ifvmode\indent\fi
3769   \bool_if:NTF \l_stex_allow_semantic_bool {
3770     \str_if_eq:eeF {
3771       \prop_item:cn {
3772         l_stex_symdecl_#1_prop
3773       }{ deprecate }
3774     }{}{
3775       \msg_warning:nxxx{stex}{warning/deprecated}{
3776         Symbol~#1
3777       }{
3778         \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
3779       }
3780     }
3781     \if_mode_math:
3782     \exp_after:wN \__stex_terms_invoke_math:n
3783     \else:
3784     \exp_after:wN \__stex_terms_invoke_text:n
3785     \fi: { #1 }
3786   }{
3787     \msg_error:nxxx{stex}{error/notallowed}{#1}{\STEXInternalCurrentSymbolStr}
3788   }
3789 }
3790
3791 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
3792   \peek_charcode_remove:NTF ! {
3793     \__stex_terms_invoke_op_custom:nn {#1}
3794   }{
3795     \__stex_terms_invoke_custom:nn {#1}
3796   }
3797 }
3798
3799 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
3800   \peek_charcode_remove:NTF ! {
3801     % operator
3802     \peek_charcode_remove:NTF * {
3803       % custom op
3804       \__stex_terms_invoke_op_custom:nn {#1}
3805     }{
3806       % op notation
3807       \peek_charcode:NTF [ {
3808         \__stex_terms_invoke_op_notation:nw {#1}
3809       }{
3810         \__stex_terms_invoke_op_notation:nw {#1}[]
3811       }
3812     }
3813   }{
3814     \peek_charcode_remove:NTF * {
3815       \__stex_terms_invoke_custom:nn {#1}
3816       % custom
3817     }{
3818       % normal
3819       \peek_charcode:NTF [ {
3820         \__stex_terms_invoke_notation:nw {#1}

```



```

3821     }{
3822       \_stex_terms_invoke_notation:nw {#1}[]
3823     }
3824   }
3825 }
3826 }
3827
3828
3829 \cs_new_protected:Nn \_stex_terms_invoke_op_custom:nn {
3830   \exp_args:Nnx \use:nn {
3831     \def\comp{\_comp}
3832     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3833     \bool_set_false:N \l_stex_allow_semantic_bool
3834     \stex_mathml_intent:nn{#1}{
3835       \_stex_term_oms:nnn {#1}{#1 \c_hash_str CUSTOM-}{
3836         \comp{ #2 }
3837       }
3838     }
3839   }{
3840     \_stex_reset:N \comp
3841     \_stex_reset:N \STEXInternalCurrentSymbolStr
3842     \bool_set_true:N \l_stex_allow_semantic_bool
3843   }
3844 }
3845
3846 \keys_define:nn { stex / terms } {
3847   % lang      .tl_set_x:N = \l_stex_notation_lang_str ,
3848   variant .tl_set_x:N = \l_stex_notation_variant_str ,
3849   unknown .code:n      = \str_set:Nx
3850     \l_stex_notation_variant_str \l_keys_key_str
3851 }
3852
3853 \cs_new_protected:Nn \_stex_terms_args:n {
3854   % \str_clear:N \l_stex_notation_lang_str
3855   \str_clear:N \l_stex_notation_variant_str
3856
3857   \keys_set:nn { stex / terms } { #1 }
3858 }
3859
3860 \cs_new_protected:Nn \stex_find_notation:nn {
3861   \_stex_terms_args:n { #2 }
3862   \seq_if_empty:cTF {
3863     \l_stex_symdecl_ #1 _notations
3864   } {
3865     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
3866   } {
3867     \str_if_empty:NTF \l_stex_notation_variant_str {
3868       \seq_get_left:cN { \l_stex_symdecl_ #1 _notations } \l_stex_notation_variant_str
3869     } {
3870       \seq_if_in:cxTF { \l_stex_symdecl_ #1 _notations } {
3871         \l_stex_notation_variant_str
3872       } {
3873         % \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3874       } {

```

```

3875         \msg_error:nxxx{stex}{error/nonotation}{#1}{
3876         ~\l_stex_notation_variant_str
3877     }
3878 }
3879 }
3880 }
3881 }
3882
3883 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3884     \exp_args:Nnx \use:nn {
3885         \def\comp{\_comp}
3886         \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3887         \stex_find_notation:nn { #1 }{ #2 }
3888         \bool_set_false:N \l_stex_allow_semantic_bool
3889         \cs_if_exist:cTF {
3890             stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3891         }{
3892             \_stex_term_oms:nnn { #1 }{
3893                 #1 \c_hash_str \l_stex_notation_variant_str
3894             }{
3895                 \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3896             }
3897         }{
3898             \int_compare:nNnTF {\prop_item:cn {\l_stex_symdecl_#1_prop}{arity}} = 0{
3899                 \cs_if_exist:cTF {
3900                     stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3901                 }{
3902                     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
3903                         \_stex_reset:N \comp
3904                         \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
3905                         \_stex_reset:N \STEXInternalCurrentSymbolStr
3906                         \bool_set_true:N \l_stex_allow_semantic_bool
3907                     }
3908                     \def\comp{\_comp}
3909                     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3910                     \bool_set_false:N \l_stex_allow_semantic_bool
3911                     \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3912                 }{
3913                     \msg_error:nxxx{stex}{error/nonotation}{#1}{
3914                     ~\l_stex_notation_variant_str
3915                     }
3916                 }
3917             }{
3918                 \msg_error:nxxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3919             }
3920         }
3921     }{
3922         \_stex_reset:N \comp
3923         \_stex_reset:N \STEXInternalCurrentSymbolStr
3924         \bool_set_true:N \l_stex_allow_semantic_bool
3925     }
3926 }
3927
3928 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {

```

```

3929 \stex_find_notation:nn { #1 }{ #2 }
3930 \cs_if_exist:cTF {
3931   stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3932 }{
3933   \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
3934     \_stex_reset:N \comp
3935     \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
3936     \_stex_reset:N \STEXInternalCurrentSymbolStr
3937     \bool_set_true:N \l_stex_allow_semantic_bool
3938   }
3939   \def\comp{\_comp}
3940   \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3941   \bool_set_false:N \l_stex_allow_semantic_bool
3942   \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3943 }{
3944   \msg_error:nnxx{stex}{error/nonotation}{#1}{
3945     ~\l_stex_notation_variant_str
3946   }
3947 }
3948 }
3949
3950 \prop_new:N \l__stex_terms_custom_args_prop
3951 \clist_new:N \l_stex_argnames_seq
3952 \seq_new:N \l__stex_terms_tmp_seq
3953
3954 \cs_new_protected:Nn \__stex_terms_custom_comp:n{ \bool_set_false:N \l_stex_allow_semantic_bo
3955
3956 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3957   \exp_args:Nnx \use:nn {
3958     \def\comp{\__stex_terms_custom_comp:n}
3959     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3960     \prop_clear:N \l__stex_terms_custom_args_prop
3961     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3962     \prop_get:cnN {
3963       l_stex_symdecl_#1 _prop
3964     }{ args } \l_tmpa_str
3965     \exp_args:NNx \seq_set_from_clist:Nn \l_stex_argnames_seq {
3966       \prop_item:cn {l_stex_symdecl_#1 _prop}{argnames}
3967     }
3968     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3969     \tl_set:Nn \arg { \__stex_terms_arg: }
3970     \str_if_empty:NTF \l_tmpa_str {
3971       \stex_mathml_intent:nn{#1}{
3972         \_stex_term_oms:nnn {#1}{#1\c_hash_str CUSTOM-}{\ignorespaces#2}
3973       }
3974     }{
3975       \seq_clear:N \l__stex_terms_tmp_seq
3976       \exp_args:Nx\int_step_inline:nn{\prop_item:cn{l_stex_symdecl_#1 _prop}{arity}}{
3977         \tl_set:Nx \l__stex_terms_tmp_tl {\seq_item:Nn \l_stex_argnames_seq {##1}}
3978         \bool_lazy_or:nnT{
3979           \str_if_eq_p:nn{a}{\str_item:Nn \l_tmpa_str{##1}}
3980         }{
3981           \str_if_eq_p:nn{B}{\str_item:Nn \l_tmpa_str{##1}}
3982         }{

```

```

3983         \tl_put_right:Nn \l__stex_terms_tmp_tl +
3984     }
3985     \seq_put_right:No \l__stex_terms_tmp_seq \l__stex_terms_tmp_tl
3986 }
3987 \stex_mathml_intent:nn{
3988     #1[\prop_item:cn {l_stex_symdecl_#1 _prop}]{ args }(
3989     \seq_use:Nn \l__stex_terms_tmp_seq ,
3990 )
3991 }{
3992     \str_if_in:NnTF \l_tmpa_str b {
3993         \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
3994     }{
3995         \str_if_in:NnTF \l_tmpa_str B {
3996             \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
3997         }{
3998             \_stex_term_oma:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
3999         }
4000     }
4001 }
4002 }
4003 % TODO check that all arguments exist
4004 }{
4005     \_stex_reset:N \l_stex_argnames_seq
4006     \_stex_reset:N \STEXInternalCurrentSymbolStr
4007     \_stex_reset:N \arg
4008     \_stex_reset:N \comp
4009     \_stex_reset:N \l__stex_terms_custom_args_prop
4010     %\bool_set_true:N \l_stex_allow_semantic_bool
4011 }
4012 }
4013
4014 \NewDocumentCommand \__stex_terms_arg: { s O{} m}{
4015     \tl_if_empty:nTF {#2}{
4016         \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
4017         \bool_set_true:N \l_tmpa_bool
4018         \bool_do_while:Nn \l_tmpa_bool {
4019             \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int}
4020             \int_incr:N \l_tmpa_int
4021         }{
4022             \bool_set_false:N \l_tmpa_bool
4023         }
4024     }
4025     }{
4026         \int_set:Nn \l_tmpa_int { #2 }
4027     }
4028     \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
4029     \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
4030         \msg_error:nnxxx{stex}{error/overarity}
4031         {\int_use:N \l_tmpa_int}
4032         {\STEXInternalCurrentSymbolStr}
4033         {\str_count:N \l_tmpa_str}
4034     }
4035     \str_set:Nx \l_tmpa_str {\str_item:Nn \l_tmpa_str \l_tmpa_int}
4036     \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {

```

```

4037 \bool_lazy_any:nF {
4038   {\str_if_eq_p:Vn \l_tmpa_str {a}}
4039   {\str_if_eq_p:Vn \l_tmpa_str {B}}
4040 }{
4041   \msg_error:nnxx{stex}{error/doubleargument}
4042   {\int_use:N \l_tmpa_int}
4043   {\STEXInternalCurrentSymbolStr}
4044 }
4045 }
4046 \exp_args:NNx \prop_put:Nnn \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {\ign
4047 \bool_if:NTF \l_stex_allow_semantic_bool \use_i:nn {
4048   \bool_set_true:N \l_stex_allow_semantic_bool
4049   \use:nn
4050 }
4051 {
4052 \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq \l_tmpa_int}{
4053   \IfBooleanTF#1{
4054     \stex_annotate_invisible:n { %TODO
4055       \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{\ignorespaces#3}
4056     }
4057   }{ %TODO
4058     \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{\ignorespaces#3}
4059   }
4060 }}
4061 {\bool_set_false:N \l_stex_allow_semantic_bool}
4062 }
4063
4064
4065 \cs_new_protected:Nn \_stex_term_arg:nn {
4066   \bool_set_true:N \l_stex_allow_semantic_bool
4067   \stex_annotate:nnn{ arg }{ #1 }{ #2 }
4068   \bool_set_false:N \l_stex_allow_semantic_bool
4069 }
4070
4071 \cs_new_protected:Npn \STEXInternalTermMathArgiii #1#2#3 {
4072   \exp_args:Nnx \use:nn
4073   { \int_set:Nn \l__stex_terms_downprec { #2 }
4074     \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq \l_tmpa_int}{
4075       \_stex_term_arg:nn { #1 }{ #3 }
4076     }
4077   }
4078   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
4079 }

```

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

`\STEXInternalTermMathAssocArgiiii`

```

4080 \cs_new_protected:Npn \STEXInternalTermMathAssocArgiiii #1#2#3#4#5 {
4081   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
4082   \tl_set:Nn \l_tmpb_tl {\STEXInternalTermMathArgiii{#5#1}{#2}}
4083   \tl_if_empty:nTF { #3 }{
4084     \STEXInternalTermMathArgiii{#5#1}{#2}{}
4085   }{
4086     \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 }}{

```

```

4087 \expandafter\if\expandafter\relax\noexpand#3
4088 \tl_set:Nn \l_tmpa_tl {\_stex_terms_math_assoc_arg_maybe_sequence:Nnn#3{#1}{#5}}
4089 \else
4090 \tl_set:Nn \l_tmpa_tl {\_stex_terms_math_assoc_arg_simple:nnn{#1}{#3}{#5}}
4091 \fi
4092 \l_tmpa_tl
4093 }{
4094 \_stex_terms_math_assoc_arg_simple:nnn{#1}{#3}{#5}
4095 }
4096 }
4097 }
4098
4099 \cs_new_protected:Nn \_stex_terms_math_assoc_arg_maybe_sequence:Nnn {
4100 \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
4101 \str_if_empty:NTF \l_tmpa_str {
4102 \exp_args:Nx \cs_if_eq:NNTF {
4103 \tl_head:N #1
4104 } \stex_invoke_sequence:n {
4105 \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
4106 \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
4107 \tl_set:Nx \l_tmpa_tl {\prop_item:cn {l_stex_symdecl_varseq://\l_tmpa_str _prop}{notat
4108 \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
4109 \tl_set:Nx \l_tmpa_tl {\exp_not:N \exp_not:n{
4110 \exp_not:n{\exp_args:Nnx \use:nn} {
4111 \exp_not:n {
4112 \def\comp{\_varcomp}
4113 \str_set:Nn \STEXInternalCurrentSymbolStr
4114 } {varseq://\l_tmpa_str}
4115 \exp_not:n{ ##1 }
4116 }{
4117 \exp_not:n {
4118 \_stex_reset:N \comp
4119 \_stex_reset:N \STEXInternalCurrentSymbolStr
4120 }
4121 }
4122 }}}
4123 \exp_args:Nno \use:n {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
4124 \seq_reverse:N \l_tmpa_seq
4125 \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
4126 \seq_map_inline:Nn \l_tmpa_seq {
4127 \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
4128 \exp_args:Nno
4129 \l_tmpa_cs { ##1 } \l_tmpa_tl
4130 }
4131 }
4132 \tl_set:Nx \l_tmpa_tl {
4133 \_stex_term_omv:nn {varseq://\l_tmpa_str}{
4134 \exp_args:No \exp_not:n \l_tmpa_tl
4135 }
4136 }
4137 \exp_args:No\l_tmpb_tl\l_tmpa_tl
4138 }{
4139 \_stex_terms_math_assoc_arg_simple:nnn{#2} { #1 }{#3}
4140 }

```

```

4141 } {
4142   \_stex_terms_math_assoc_arg_simple:nnn{#2} { #1 }{#3}
4143 }
4144
4145 }
4146
4147 \cs_new_protected:Nn \_stex_terms_math_assoc_arg_simple:nnn {
4148   \clist_set:Nn \l_tmpa_clist{ #2 }
4149   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
4150     \tl_set:Nn \l_tmpa_tl {
4151       \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq #1}{
4152         \_stex_term_arg:nn{A#3#1}{ #2 } }
4153     }
4154   }{
4155     \clist_reverse:N \l_tmpa_clist
4156     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
4157     \tl_set:Nx \l_tmpa_tl {
4158       \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq #1}{
4159         \_stex_term_arg:nn{A#3#1}{
4160           \exp_args:No \exp_not:n \l_tmpa_tl
4161         }
4162       }
4163     }
4164     \clist_map_inline:Nn \l_tmpa_clist {
4165       \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
4166         \l_tmpa_cs {
4167           \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq #1}{
4168             \_stex_term_arg:nn{A#3#1}{##1}
4169           }
4170         } \l_tmpa_tl
4171       }
4172     }
4173   }
4174   \exp_args:No\l_tmpb_tl\l_tmpa_tl
4175 }

```

(End definition for `\STEXInternalTermMathAssocArgiiii`. This function is documented on page 89.)

30.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
4176 \tl_const:Nx \infprec {\int_use:N \c_max_int}
4177 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
4178 \int_new:N \l__stex_terms_downprec
4179 \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 89.)

Bracketing:

```

\l__stex_terms_left_bracket_str
\l__stex_terms_right_bracket_str
4180 \tl_set:Nn \l__stex_terms_left_bracket_str (
4181 \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

```

4182 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
4183   \bool_if:NTF \l__stex_terms_brackets_done_bool {
4184     \bool_set_false:N \l__stex_terms_brackets_done_bool
4185     #2
4186   } {
4187     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
4188       \bool_if:NTF \l__stex_inarray_bool { #2 }{
4189         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
4190         \dobrackets { #2 }
4191       }
4192     }{ #2 }
4193   }
4194 }
```

(End definition for `__stex_terms_maybe_brackets:nn`.)

`\dobrackets`

```

4195 \bool_new:N \l__stex_terms_brackets_done_bool
4196 %\RequirePackage{scalerel}
4197 \cs_new_protected:Npn \dobrackets #1 {
4198   %\ThisStyle{\if D\m@switch
4199   %   \exp_args:Nnx \use:nn
4200   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
4201   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
4202   %   \else
4203   %   \exp_args:Nnx \use:nn
4204   %   {
4205   %     \bool_set_true:N \l__stex_terms_brackets_done_bool
4206   %     \int_set:Nn \l__stex_terms_downprec \infpref
4207   %     \l__stex_terms_left_bracket_str
4208   %     #1
4209   %   }
4210   %   {
4211   %     \bool_set_false:N \l__stex_terms_brackets_done_bool
4212   %     \l__stex_terms_right_bracket_str
4213   %     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
4214   %   }
4215   %\fi}
4216 }
```

(End definition for `\dobrackets`. This function is documented on page 89.)

`\withbrackets`

```

4217 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
4218   \exp_args:Nnx \use:nn
4219   {
4220     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
4221     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
4222     #3
4223   }
4224 }
```



```

4225 \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
4226 {\l__stex_terms_left_bracket_str}
4227 \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
4228 {\l__stex_terms_right_bracket_str}
4229 }
4230 }

```

(End definition for `\withbrackets`. This function is documented on page 89.)

`\STEXinvisible`

```

4231 \cs_new_protected:Npn \STEXinvisible #1 {
4232 \stex_annotate_invisible:n { #1 }
4233 }

```

(End definition for `\STEXinvisible`. This function is documented on page 89.)
OMDoc terms:

`\STEXInternalTermMathOMSiiii`

```

4234 \cs_new_protected:Nn \_stex_term_oms:nnn {
4235 \stex_annotate:nnn{ OMID }{ #2 }{
4236 #3
4237 }
4238 }
4239
4240 \cs_new_protected:Npn \STEXInternalTermMathOMSiiii #1#2#3#4 {
4241 \__stex_terms_maybe_brackets:nn { #3 }{
4242 \stex_mathml_intent:nn{#1} {
4243 \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
4244 }
4245 }
4246 }

```

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

`_stex_term_math_omv:nn`

```

4247 \cs_new_protected:Nn \_stex_term_omv:nn {
4248 \stex_annotate:nnn{ OMV }{ #1 }{
4249 #2
4250 }
4251 }

```

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

`\STEXInternalTermMathOMAiiai`

```

4252 \cs_new_protected:Nn \_stex_term_oma:nnn {
4253 \stex_annotate:nnn{ OMA }{ #2 }{
4254 #3
4255 }
4256 }
4257
4258 \cs_new_protected:Npn \STEXInternalTermMathOMAiiai #1#2#3#4 {
4259 \exp_args:Nnx \use:nn {
4260 \seq_clear:N \l__stex_terms_tmp_seq
4261 \prop_if_exist:cT{l_stex_symdecl_#1 _prop}{
4262 \exp_args:NNx \seq_set_from_clist:Nn \l_stex_argnames_seq {

```

```

4263     \prop_item:cn {l_stex_symdecl_#1 _prop}{argnames}
4264   }
4265   \exp_args:Nx\int_step_inline:nn{\prop_item:cn{l_stex_symdecl_#1 _prop}{arity}}{
4266     \tl_set:Nx \l__stex_terms_tmp_tl {\seq_item:Nn \l_stex_argnames_seq {##1}}
4267     \bool_lazy_or:nnT{
4268       \str_if_eq_p:nn{a}{\str_item:Nn\l_tmpa_str{##1}}
4269     }{
4270       \str_if_eq_p:nn{B}{\str_item:Nn\l_tmpa_str{##1}}
4271     }{
4272       \tl_put_right:Nn \l__stex_terms_tmp_tl +
4273     }
4274     \seq_put_right:No \l__stex_terms_tmp_seq \l__stex_terms_tmp_tl
4275   }
4276 }
4277 \__stex_terms_maybe_brackets:nn { #3 }{
4278   \stex_mathml_intent:nn{
4279     #1[\prop_item:cn {l_stex_symdecl_#1 _prop}{ args }](
4280       \seq_use:Nn \l__stex_terms_tmp_seq ,
4281     )
4282   }{
4283     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
4284   }
4285 }
4286 }{
4287   \_stex_reset:N \l_stex_argnames_seq
4288 }
4289 }

```

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

`\STEXInternalTermMathOMB`

```

4290 \cs_new_protected:Nn \_stex_term_ombind:nnn {
4291   \stex_annotate:nnn{ OMBIND }{ #2 }{
4292     #3
4293   }
4294 }
4295
4296 \cs_new_protected:Npn \STEXInternalTermMathOMB #1#2#3#4 {
4297   \exp_args:Nnx \use:nn {
4298     \seq_clear:N \l__stex_terms_tmp_seq
4299     \prop_if_exist:cT{l_stex_symdecl_#1 _prop}{
4300       \exp_args:NNx \seq_set_from_clist:Nn \l_stex_argnames_seq {
4301         \prop_item:cn {l_stex_symdecl_#1 _prop}{argnames}
4302       }
4303       \exp_args:Nx\int_step_inline:nn{\prop_item:cn{l_stex_symdecl_#1 _prop}{arity}}{
4304         \tl_set:Nx \l__stex_terms_tmp_tl {\seq_item:Nn \l_stex_argnames_seq {##1}}
4305         \bool_lazy_or:nnT{
4306           \str_if_eq_p:nn{a}{\str_item:Nn\l_tmpa_str{##1}}
4307         }{
4308           \str_if_eq_p:nn{B}{\str_item:Nn\l_tmpa_str{##1}}
4309         }{
4310           \tl_put_right:Nn \l__stex_terms_tmp_tl +
4311         }
4312         \seq_put_right:No \l__stex_terms_tmp_seq \l__stex_terms_tmp_tl

```

```

4313     }
4314   }
4315   \__stex_terms_maybe_brackets:nn { #3 }{
4316     \stex_mathml_intent:nn{
4317       #1[\prop_item:cn {l_stex_symdecl_#1 _prop}{ args }](
4318         \seq_use:Nn \l__stex_terms_tmp_seq ,
4319       )
4320     }{
4321       \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
4322     }
4323   }
4324   }{
4325     \stex_reset:N \l_stex_argnames_seq
4326   }
4327 }

```

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

`\symref`
`\symname`

```

4328 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
4329
4330 \keys_define:nn { stex / symname } {
4331   pre      .tl_set_x:N      = \l__stex_terms_pre_tl ,
4332   post     .tl_set_x:N      = \l__stex_terms_post_tl ,
4333   root     .tl_set_x:N      = \l__stex_terms_root_tl
4334 }
4335
4336 \cs_new_protected:Nn \stex_symname_args:n {
4337   \tl_clear:N \l__stex_terms_post_tl
4338   \tl_clear:N \l__stex_terms_pre_tl
4339   \tl_clear:N \l__stex_terms_root_str
4340   \keys_set:nn { stex / symname } { #1 }
4341 }
4342
4343 \NewDocumentCommand \symref { m m }{
4344   \let\compemph_uri_prev:\compemph@uri
4345   \let\compemph@uri\symrefemph@uri
4346   \STEXsymbol{#1}!\{ #2 }
4347   \let\compemph@uri\compemph_uri_prev:
4348 }
4349
4350 \NewDocumentCommand \synonym { 0{} m m }{
4351   \stex_symname_args:n { #1 }
4352   \let\compemph_uri_prev:\compemph@uri
4353   \let\compemph@uri\symrefemph@uri
4354   % TODO
4355   \STEXsymbol{#2}!\{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
4356   \let\compemph@uri\compemph_uri_prev:
4357 }
4358
4359 \NewDocumentCommand \symname { 0{} m }{
4360   \stex_symname_args:n { #1 }
4361   \stex_get_symbol:n { #2 }
4362   \str_set:Nx \l_tmpa_str {

```

```

4363 \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4364 }
4365 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4366
4367 \let\compemph_uri_prev:\compemph@uri
4368 \let\compemph@uri\symrefemph@uri
4369 \exp_args:Nnx \use:nn
4370 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmode*\fi{
4371 \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
4372 } }
4373 \let\compemph@uri\compemph_uri_prev:
4374 }
4375
4376 \NewDocumentCommand \Symname { 0{} m }{
4377 \stex_symname_args:n { #1 }
4378 \stex_get_symbol:n { #2 }
4379 \str_set:Nx \l_tmpa_str {
4380 \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4381 }
4382 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4383 \let\compemph_uri_prev:\compemph@uri
4384 \let\compemph@uri\symrefemph@uri
4385 \exp_args:Nnx \use:nn
4386 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmode*\fi{
4387 \exp_after:wN \stex_capitalize:n \l_tmpa_str
4388 \l__stex_terms_post_tl
4389 } }
4390 \let\compemph@uri\compemph_uri_prev:
4391 }

```

(End definition for `\symref` and `\symname`. These functions are documented on page 88.)

30.3 Notation Components

```

4392 <@@=stex_notationcomps>
\comp
\compemph@uri
\compemph
\defemph
\defemph@uri
\symrefemph
\symrefemph@uri
\varemp
\varemp@uri
4393 \cs_new_protected:Npn \_comp #1 {
4394 \str_if_empty:NF \STEXInternalCurrentSymbolStr {
4395 \stex_html_backend:TF {
4396 \stex_annotate:nnn { comp }{ \STEXInternalCurrentSymbolStr }{ #1 }
4397 }{
4398 \exp_args:Nnx \compemph@uri { #1 } { \STEXInternalCurrentSymbolStr }
4399 }
4400 }
4401 }
4402
4403 \cs_new_protected:Npn \_varcomp #1 {
4404 \str_if_empty:NF \STEXInternalCurrentSymbolStr {
4405 \stex_html_backend:TF {
4406 \stex_annotate:nnn { varcomp }{ \STEXInternalCurrentSymbolStr }{ #1 }
4407 }{
4408 \exp_args:Nnx \varemp@uri { #1 } { \STEXInternalCurrentSymbolStr }
4409 }

```

```

4410 }
4411 }
4412
4413 \def\comp{\_comp}
4414
4415 \cs_new_protected:Npn \compemph@uri #1 #2 {
4416   \compemph{ #1 }
4417 }
4418
4419
4420 \cs_new_protected:Npn \compemph #1 {
4421   #1
4422 }
4423
4424 \cs_new_protected:Npn \defemph@uri #1 #2 {
4425   \defemph{#1}
4426 }
4427
4428 \cs_new_protected:Npn \defemph #1 {
4429   \textbf{#1}
4430 }
4431
4432 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
4433   \symrefemph{#1}
4434 }
4435
4436 \cs_new_protected:Npn \symrefemph #1 {
4437   \emph{#1}
4438 }
4439
4440 \cs_new_protected:Npn \varemp@uri #1 #2 {
4441   \varemp{#1}
4442 }
4443
4444 \cs_new_protected:Npn \varemp #1 {
4445   #1
4446 }

```

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

`\ellipses`

```

4447 \NewDocumentCommand \ellipses {} { \ldots }

```

(End definition for `\ellipses`. This function is documented on page 89.)

```

\parray
\prmatrix
\parrayline
\parraylineh
\parraycell
4448 \bool_new:N \l_stex_inarray_bool
4449 \bool_set_false:N \l_stex_inarray_bool
4450 \NewDocumentCommand \parray { m m } {
4451   \begingroup
4452   \bool_set_true:N \l_stex_inarray_bool
4453   \begin{array}{#1}
4454     #2
4455   \end{array}
4456 \endgroup

```

```

4457 }
4458
4459 \NewDocumentCommand \prmatrix { m } {
4460   \begingroup
4461   \bool_set_true:N \l_stex_inarray_bool
4462   \begin{matrix}
4463     #1
4464   \end{matrix}
4465   \endgroup
4466 }
4467
4468 \def \maybepline {
4469   \bool_if:NT \l_stex_inarray_bool {\hline}
4470 }
4471
4472 \def \parrayline #1 #2 {
4473   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
4474 }
4475
4476 \def \pmrow #1 { \parrayline{}{ #1 } }
4477
4478 \def \parraylineh #1 #2 {
4479   #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
4480 }
4481
4482 \def \parraycell #1 {
4483   #1 \bool_if:NT \l_stex_inarray_bool {&}
4484 }

```

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

30.4 Variables

```

4485 \@@=stex_variables

```

\stex_invoke_variable:n Invokes a variable

```

4486 \cs_new_protected:Nn \stex_invoke_variable:n {
4487   \if_mode_math:
4488     \exp_after:wN \__stex_variables_invoke_math:n
4489   \else:
4490     \exp_after:wN \__stex_variables_invoke_text:n
4491   \fi: {#1}
4492 }
4493
4494 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
4495   \peek_charcode_remove:NTF ! {
4496     \__stex_variables_invoke_op_custom:nn {#1}
4497   }{
4498     \__stex_variables_invoke_custom:nn {#1}
4499   }
4500 }
4501
4502
4503 \cs_new_protected:Nn \__stex_variables_invoke_math:n {

```

```

4504 \peek_charcode_remove:NTF ! {
4505   \peek_charcode_remove:NTF ! {
4506     \peek_charcode:NTF [ {
4507       % TODO throw error
4508     }{
4509       \__stex_variables_invoke_op_custom:nn
4510     }
4511   }{
4512     \__stex_variables_invoke_op:n { #1 }
4513   }
4514 }{
4515   \peek_charcode_remove:NTF * {
4516     \__stex_variables_invoke_custom:nn { #1 }
4517   }{
4518     \__stex_variables_invoke_math_ii:n { #1 }
4519   }
4520 }
4521 }
4522
4523 \cs_new_protected:Nn \__stex_variables_invoke_op_custom:nn {
4524   \exp_args:Nnx \use:nn {
4525     \def\comp{\_varcomp}
4526     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4527     \bool_set_false:N \l_stex_allow_semantic_bool
4528     \_stex_term_omv:nn {var://#1}{
4529       \comp{ #2 }
4530     }
4531   }{
4532     \_stex_reset:N \comp
4533     \_stex_reset:N \STEXInternalCurrentSymbolStr
4534     \bool_set_true:N \l_stex_allow_semantic_bool
4535   }
4536 }
4537
4538 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
4539   \cs_if_exist:cTF {
4540     stex_var_op_notation_ #1 _cs
4541   }{
4542     \exp_args:Nnx \use:nn {
4543       \def\comp{\_varcomp}
4544       \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4545       \_stex_term_omv:nn { var://#1 }{
4546         \use:c{stex_var_op_notation_ #1 _cs }
4547       }
4548     }{
4549       \_stex_reset:N \comp
4550       \_stex_reset:N \STEXInternalCurrentSymbolStr
4551     }
4552   }{
4553     \int_compare:nNnTF {\prop_item:cn {l_stex_symdecl_var://#1_prop}{arity}} = 0{
4554       \__stex_variables_invoke_math_ii:n {#1}
4555     }{
4556       \msg_error:nxxx{stex}{error/noop}{variable~#1}{}
4557     }

```

```

4558 }
4559 }
4560
4561 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n #1 {
4562   \cs_if_exist:cTF {
4563     stex_var_notation_#1_cs
4564   }{
4565     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4566       \_stex_reset:N \comp
4567       \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
4568       \_stex_reset:N \STEXInternalCurrentSymbolStr
4569       \bool_set_true:N \l_stex_allow_semantic_bool
4570     }
4571     \def\comp{\_varcomp}
4572     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4573     \bool_set_false:N \l_stex_allow_semantic_bool
4574     \use:c{stex_var_notation_#1_cs}
4575   }{
4576     \msg_error:nnxx{stex}{error/nonotation}{variable-#1}{s}
4577   }
4578 }
4579
4580 \cs_new_protected:Nn \__stex_variables_invoke_custom:nn {
4581   \exp_args:Nnx \use:nn {
4582     \def\comp{\_varcomp}
4583     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4584     \prop_clear:N \l__stex_terms_custom_args_prop
4585     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
4586     \prop_get:cnN {
4587       l_stex_symdecl_var://#1 _prop
4588     }{ args } \l_tmpa_str
4589     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
4590     \tl_set:Nn \arg { \_stex_terms_arg: }
4591     \str_if_empty:NTF \l_tmpa_str {
4592       \_stex_term_omv:nn {var://#1}{\ignorespaces#2}
4593     }{
4594       \str_if_in:NnTF \l_tmpa_str b {
4595         \_stex_term_ombind:nnn {var://#1}{\ignorespaces#2}
4596       }{
4597         \str_if_in:NnTF \l_tmpa_str B {
4598           \_stex_term_ombind:nnn {var://#1}{\ignorespaces#2}
4599         }{
4600           \_stex_term_oma:nnn {var://#1}{\ignorespaces#2}
4601         }
4602       }
4603     }
4604     % TODO check that all arguments exist
4605   }{
4606     \_stex_reset:N \STEXInternalCurrentSymbolStr
4607     \_stex_reset:N \arg
4608     \_stex_reset:N \comp
4609     \_stex_reset:N \l__stex_terms_custom_args_prop
4610     %\bool_set_true:N \l_stex_allow_semantic_bool
4611   }

```



```
4612 }
```

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

30.5 Sequences

```
4613 <@@=stex_sequences>
4614
4615 \cs_new_protected:Nn \stex_invoke_sequence:n {
4616   \peek_charcode_remove:NTF ! {
4617     \stex_term_omv:nn {varseq://#1}{
4618       \exp_args:Nnx \use:nn {
4619         \def\comp{\_varcomp}
4620         \str_set:Nn \STEXInternalCurrentSymbolStr {varseq://#1}
4621         \prop_item:cn{l_stex_symdecl_varseq://#1_prop}{notation}
4622       }{
4623         \_stex_reset:N \comp
4624         \_stex_reset:N \STEXInternalCurrentSymbolStr
4625       }
4626     }
4627   }{
4628     \bool_set_false:N \l_stex_allow_semantic_bool
4629     \def\comp{\_varcomp}
4630     \str_set:Nn \STEXInternalCurrentSymbolStr {varseq://#1}
4631     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4632       \_stex_reset:N \comp
4633       \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
4634       \_stex_reset:N \STEXInternalCurrentSymbolStr
4635       \bool_set_true:N \l_stex_allow_semantic_bool
4636     }
4637     \use:c { stex_varseq_#1_cs }
4638   }
4639 }
4640 </package>
```

Chapter 31

STEX -Structural Features Implementation

```
4641 ⟨*package⟩
4642
4643 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4644
4645 Warnings and error messages
4646 \msg_new:nnn{stex}{error/copymodule/notallowed}{
4647   Symbol~#1~can~not~be~assigned~in~copymodule~#2
4648 }
4649 \msg_new:nnn{stex}{error/interpretmodule/nodfiniens}{
4650   Symbol~#1~not~assigned~in~interpretmodule~#2
4651 }
4652 \msg_new:nnn{stex}{error/unknownstructure}{
4653   No~structure~#1~found!
4654 }
4655
4656 \msg_new:nnn{stex}{error/unknownfield}{
4657   No~field~#1~in~instance~#2~found!\#3
4658 }
4659
4660 \msg_new:nnn{stex}{error/keyval}{
4661   Invalid~key=value~pair:#1
4662 }
4663 \msg_new:nnn{stex}{error/instantiate/missing}{
4664   Assignments~missing~in~instantiate:~#1
4665 }
4666 \msg_new:nnn{stex}{error/incompatible}{
4667   Incompatible~signature:~#1~(#2)~and~#3~(#4)
4668 }
4669
```

31.1 Imports with modification

```

4670 <@@=stex_copymodule>
4671 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
4672   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4673     \tl_set:Nn \l_tmpa_tl { #1 }
4674     \__stex_copymodule_get_symbol_from_cs:
4675   }{
4676     % argument is a string
4677     % is it a command name?
4678     \cs_if_exist:cTF { #1 }{
4679       \cs_set_eq:Nc \l_tmpa_tl { #1 }
4680       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4681       \str_if_empty:NNTF \l_tmpa_str {
4682         \exp_args:Nx \cs_if_eq:NNTF {
4683           \tl_head:N \l_tmpa_tl
4684         } \stex_invoke_symbol:n {
4685           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
4686         }{
4687           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4688         }
4689       } {
4690         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4691       }
4692     }{
4693       % argument is not a command name
4694       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4695       % \l_stex_all_symbols_seq
4696     }
4697   }
4698 }
4699
4700 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
4701   \str_set:Nn \l_tmpa_str { #1 }
4702   \bool_set_false:N \l_tmpa_bool
4703   \bool_if:NF \l_tmpa_bool {
4704     \tl_set:Nn \l_tmpa_tl {
4705       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
4706     }
4707     \str_set:Nn \l_tmpa_str { #1 }
4708     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4709     \seq_map_inline:Nn #2 {
4710       \str_set:Nn \l_tmpb_str { ##1 }
4711       \str_if_eq:eeT { \l_tmpa_str } {
4712         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4713       } {
4714         \seq_map_break:n {
4715           \tl_set:Nn \l_tmpa_tl {
4716             \str_set:Nn \l_stex_get_symbol_uri_str {
4717               ##1
4718             }
4719           }
4720         }
4721       }

```

```

4722     }
4723     \l_tmpa_tl
4724   }
4725 }
4726
4727 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
4728   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4729     { \tl_tail:N \l_tmpa_tl }
4730   \tl_if_single:NTF \l_tmpa_tl {
4731     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
4732       \exp_after:wN \str_set:Nn \exp_after:wN
4733         \l_stex_get_symbol_uri_str \l_tmpa_tl
4734       \__stex_copymodule_get_symbol_check:n { #1 }
4735     }{
4736       % TODO
4737       % tail is not a single group
4738     }
4739   }{
4740     % TODO
4741     % tail is not a single group
4742   }
4743 }
4744
4745 \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
4746   \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
4747     \msg_error:nxxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
4748       :~\seq_use:Nn #1 {,~}
4749     }
4750   }
4751 }
4752
4753 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
4754   % import module
4755   \stex_import_module_uri:nn { #1 } { #2 }
4756   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
4757   \stex_import_require_module:nnnn
4758     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4759     { \l_stex_import_path_str } { \l_stex_import_name_str }
4760
4761   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
4762   \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
4763
4764   % fields
4765   \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
4766   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4767     \seq_map_inline:cn {c_stex_module_##1_constants}{
4768       \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
4769         ##1 ? ####1
4770       }
4771     }
4772   }
4773
4774   % setup prop
4775   \seq_clear:N \l_tmpa_seq

```

```

4776 \exp_args:N Nx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
4777   name      = \l_stex_current_copymodule_name_str ,
4778   module    = \l_stex_current_module_str ,
4779   from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
4780   includes  = \l_tmpa_seq %,
4781 % fields    = \l_tmpa_seq
4782 }
4783 \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
4784   as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
4785 \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_seq {,
4786 \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,
4787
4788 \stex_if_do_html:T {
4789   \begin{stex_annotate_env} {#4} {
4790     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4791   }
4792   \stex_annotate_invisible:nnn{domain}{\l_stex_import_ns_str ?\l_stex_import_name_str}{}
4793 }
4794 }
4795
4796 \cs_new_protected:Nn \stex_copymodule_end:n {
4797   % apply to every field
4798   \def \l_tmpa_cs ##1 ##2 {#1}
4799
4800   \tl_clear:N \__stex_copymodule_module_tl
4801   \tl_clear:N \__stex_copymodule_exec_tl
4802
4803   %\prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
4804   \seq_clear:N \__stex_copymodule_fields_seq
4805
4806   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4807     \seq_map_inline:cn {c_stex_module_##1_constants}{
4808
4809       \tl_clear:N \__stex_copymodule_curr_symbol_tl % <- wrap in current symbol html
4810       \l_tmpa_cs{##1}{####1}
4811
4812       \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
4813         \str_set_eq:Nc \__stex_copymodule_curr_name_str {l__stex_copymodule_copymodule_##1?####1_name_str}
4814         \stex_if_do_html:T {
4815           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4816             \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?####1_name_str}}
4817           }
4818         }
4819       }{
4820         \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str /
4821       }
4822
4823       \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1_prop}
4824       \prop_put:Nnx \l_tmpa_prop { name } \__stex_copymodule_curr_name_str
4825       \prop_put:Nnx \l_tmpa_prop { module } \l_stex_current_module_str
4826
4827       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
4828         \stex_if_do_html:T {
4829           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {

```

```

4830         $\stex_annotate_invisible:nnn{definiens}{\exp_after:wN \exp_not:N\csname l__st
4831     }
4832 }
4833 \prop_put:Nnn \l_tmpa_prop { defined } { true }
4834 }
4835
4836 \stex_add_constant_to_current_module:n \__stex_copymodule_curr_name_str
4837 \tl_put_right:Nx \__stex_copymodule_module_tl {
4838     \seq_clear:c {l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_n
4839     \prop_set_from_keyval:cn {
4840         l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_name_str _prop
4841     }{
4842         \prop_to_keyval:N \l_tmpa_prop
4843     }
4844 }
4845
4846 \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
4847     \stex_if_do_html:T {
4848         \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4849             \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1
4850         }
4851     }
4852     \tl_put_right:Nx \__stex_copymodule_module_tl {
4853         \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
4854             \stex_invoke_symbol:n {
4855                 \l_stex_current_module_str ? \__stex_copymodule_curr_name_str
4856             }
4857         }
4858     }
4859 }
4860
4861 \seq_put_right:Nx \__stex_copymodule_fields_seq {\l_stex_current_module_str ? \__stex_
4862
4863 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4864     \stex_copy_notations:nn {\l_stex_current_module_str ? \__stex_copymodule_curr_name_s
4865 }
4866
4867 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4868     \stex_if_do_html:TF{
4869         \stex_annotate_invisible:nnn{assignment} {##1?####1} { \exp_after:wN \exp_not:n \e
4870     }{
4871         \exp_after:wN \exp_not:n \exp_after:wN {\__stex_copymodule_curr_symbol_tl}
4872     }
4873 }
4874 }
4875 }
4876
4877
4878 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \__stex_copymodule_fields_seq
4879 \tl_put_left:Nx \__stex_copymodule_module_tl {
4880     \prop_set_from_keyval:cn {
4881         l_stex_copymodule_ \l_stex_current_module_str? \l_stex_current_copymodule_name_str _pro
4882     }{
4883         \prop_to_keyval:N \l_stex_current_copymodule_prop

```

```

4884     }
4885 }
4886
4887 \seq_gput_right:cx{c_stex_module_\l_stex_current_module_str _copymodules}{
4888   \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4889 }
4890
4891 \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4892 \stex_debug:nn{copymodule}{result:\meaning \__stex_copymodule_module_tl}
4893 \stex_debug:nn{copymodule}{output:\meaning \__stex_copymodule_exec_tl}
4894
4895 \__stex_copymodule_exec_tl
4896 \stex_if_do_html:T {
4897   \end{stex_annotate_env}
4898 }
4899 }
4900
4901 \NewDocumentEnvironment {copymodule} { 0{} m m}{
4902   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ copymodule }
4903   \stex_deactivate_macro:Nn \symdecl {module~environments}
4904   \stex_deactivate_macro:Nn \symdef {module~environments}
4905   \stex_deactivate_macro:Nn \notation {module~environments}
4906   \stex_reactivate_macro:N \assign
4907   \stex_reactivate_macro:N \renamedekl
4908   \stex_reactivate_macro:N \donotcopy
4909   \stex_smsmode_do:
4910 }{
4911   \stex_copymodule_end:n {}
4912 }
4913
4914 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
4915   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ interpretmodule }
4916   \stex_deactivate_macro:Nn \symdecl {module~environments}
4917   \stex_deactivate_macro:Nn \symdef {module~environments}
4918   \stex_deactivate_macro:Nn \notation {module~environments}
4919   \stex_reactivate_macro:N \assign
4920   \stex_reactivate_macro:N \renamedekl
4921   \stex_reactivate_macro:N \donotcopy
4922   \stex_smsmode_do:
4923 }{
4924   \stex_copymodule_end:n {
4925     \tl_if_exist:cF {
4926       l__stex_copymodule_copymodule_##1?##2_def_tl
4927     }{
4928       \str_if_eq:eeF {
4929         \prop_item:cn{
4930           l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
4931         }{ true }{
4932           \msg_error:nxxx{stex}{error/interpretmodule/nodéfiniens}{
4933             ##1?##2
4934           }{\l_stex_current_copymodule_name_str}
4935         }
4936       }
4937     }

```

```

4938 }
4939
4940 \iffalse \begin{stex_annotate_env} \fi
4941 \NewDocumentEnvironment {realization} { 0 } { m } {
4942   \stex_copymodule_start:nnnn { #1 } { #2 } { #2 } { realize }
4943   \stex_deactivate_macro:Nn \symdecl {module~environments}
4944   \stex_deactivate_macro:Nn \symdef {module~environments}
4945   \stex_deactivate_macro:Nn \notation {module~environments}
4946   \stex_reactivate_macro:N \donotcopy
4947   \stex_reactivate_macro:N \assign
4948   \stex_smsmode_do:
4949 } {
4950   \stex_import_module_uri:nn { #1 } { #2 }
4951   \tl_clear:N \__stex_copymodule_exec_tl
4952   \tl_set:Nx \__stex_copymodule_module_tl {
4953     \stex_import_require_module:nnnn
4954     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4955     { \l_stex_import_path_str } { \l_stex_import_name_str }
4956   }
4957
4958   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4959     \seq_map_inline:cn {c_stex_module_##1_constants}{
4960       \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str / #1 }
4961       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?###1_def_tl}{
4962         \stex_if_do_html:T {
4963           \tl_put_right:Nx \__stex_copymodule_exec_tl {
4964             \stex_annotate_invisible:nnn{assignment} {##1?###1} {
4965               $\stex_annotate_invisible:nnn{definien}{}{\exp_after:wN \exp_not:N\csname l__
4966             }
4967           }
4968         }
4969         \tl_put_right:Nx \__stex_copymodule_module_tl {
4970           \prop_put:cnn {l_stex_symdecl_##1?###1_prop}{ defined }{ true }
4971         }
4972       }
4973     }
4974
4975     \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4976
4977     \__stex_copymodule_exec_tl
4978     \stex_if_do_html:T {\end{stex_annotate_env}}
4979   }
4980
4981   \NewDocumentCommand \donotcopy { m } {
4982     \str_clear:N \l_stex_import_name_str
4983     \str_set:Nn \l_tmpa_str { #1 }
4984     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4985     \seq_map_inline:Nn \l_stex_all_modules_seq {
4986       \str_set:Nn \l_tmpb_str { ##1 }
4987       \str_if_eq:eeT { \l_tmpa_str } {
4988         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4989       } {
4990         \seq_map_break:n {
4991           \stex_if_do_html:T {

```



```

4992         \stex_if_smsmode:F {
4993             \stex_annotate_invisible:nnn{donotcopy}{##1}{
4994                 \stex_annotate:nnn{domain}{##1}{}}
4995         }
4996     }
4997 }
4998 \str_set_eq:NN \l_stex_import_name_str \l_tmpb_str
4999 }
5000 }
5001 \seq_map_inline:cn {c_stex_module_###1_copymodules}{
5002     \str_set:Nn \l_tmpb_str { #####1 }
5003     \str_if_eq:eeT { \l_tmpa_str } {
5004         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
5005     } {
5006         \seq_map_break:n {\seq_map_break:n {
5007             \stex_if_do_html:T {
5008                 \stex_if_smsmode:F {
5009                     \stex_annotate_invisible:nnn{donotcopy}{#####1}{
5010                         \stex_annotate:nnn{domain}{
5011                             \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
5012                         }}
5013                 }
5014             }
5015         }
5016         \str_set:Nx \l_stex_import_name_str {
5017             \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
5018         }
5019     }}
5020 }
5021 }
5022 }
5023 \str_if_empty:NTF \l_stex_import_name_str {
5024     % TODO throw error
5025 }{
5026     \stex_collect_imports:n {\l_stex_import_name_str }
5027     \seq_map_inline:Nn \l_stex_collect_imports_seq {
5028         \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
5029         \seq_map_inline:cn {c_stex_module_###1_constants}{
5030             \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? #####1 }
5031             \bool_lazy_any:nT {
5032                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_name_str}}
5033                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_macroname_str}}
5034                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_def_tl}}
5035             }{
5036                 % TODO throw error
5037             }
5038         }
5039     }
5040     \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
5041     \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_name_str }
5042     \prop_put:Nno \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
5043 }
5044 \stex_smsmode_do:
5045 }

```

```

5046
5047 \NewDocumentCommand \assign { m m }{
5048   \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
5049   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
5050   \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_def_tl}{#2}
5051   \stex_smsmode_do:
5052 }
5053
5054 \keys_define:nn { stex / renamedecl } {
5055   name          .str_set_x:N = \l_stex_renamedecl_name_str
5056 }
5057 \cs_new_protected:Nn \__stex_copymodule_renamedecl_args:n {
5058   \str_clear:N \l_stex_renamedecl_name_str
5059   \keys_set:nn { stex / renamedecl } { #1 }
5060 }
5061
5062 \NewDocumentCommand \renamedecl { O{} m m }{
5063   \__stex_copymodule_renamedecl_args:n { #1 }
5064   \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
5065   \stex_debug:nn{renamedecl}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
5066   \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_macroname_str}{#3}
5067   \str_if_empty:NTF \l_stex_renamedecl_name_str {
5068     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
5069       \l_stex_get_symbol_uri_str
5070     } }
5071   } {
5072     \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_name_str}{\l_stex
5073       \stex_debug:nn{renamedecl}{@~\l_stex_current_module_str ? \l_stex_renamedecl_name_str}
5074       \prop_set_eq:cc {l_stex_symdecl_
5075         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
5076         _prop
5077       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str_prop}
5078       \seq_set_eq:cc {l_stex_symdecl_
5079         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
5080         _notations
5081       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str_notations}
5082       \prop_put:cnx {l_stex_symdecl_
5083         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
5084         _prop
5085       }{ name }{ \l_stex_renamedecl_name_str }
5086       \prop_put:cnx {l_stex_symdecl_
5087         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
5088         _prop
5089       }{ module }{ \l_stex_current_module_str }
5090       \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
5091         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
5092       }
5093       \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
5094         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
5095       } }
5096   }
5097   \stex_smsmode_do:
5098 }
5099

```

```

5100 \stex_deactivate_macro:Nn \assign {copymodules}
5101 \stex_deactivate_macro:Nn \renamedekl {copymodules}
5102 \stex_deactivate_macro:Nn \donotcopy {copymodules}
5103
5104

```

31.2 The feature environment

`structural@feature (env.)`

```

5105 <@@=stex_features>
5106
5107 \NewDocumentEnvironment{structural_feature_module}{m m m}{
5108   \stex_if_in_module:F {
5109     \msg_set:nnn{stex}{error/nomodule}{
5110       Structural~Feature~has~to~occur~in~a~module:\\
5111       Feature~#2~of~type~#1\\
5112       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
5113     }
5114     \msg_error:nn{stex}{error/nomodule}
5115   }
5116
5117   \str_set_eq:NN \l_stex_feature_parent_str \l_stex_current_module_str
5118
5119   \stex_module_setup:nn{meta=NONE}{#2 - #1}
5120
5121   \stex_if_do_html:T {
5122     \begin{stex_annotate_env}{feature:#1}{\l_stex_feature_parent_str ? #2 - #1}
5123     \stex_annotate_invisible:nnn{header}{\l_stex_feature_parent_str}{#3}
5124   }
5125 }{
5126   \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
5127   \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
5128   \stex_debug:nn{features}{
5129     Feature: \l_stex_last_feature_str
5130   }
5131   \stex_if_do_html:T {
5132     \end{stex_annotate_env}
5133   }
5134 }

```

31.3 Structure

`structure (env.)`

```

5135 <@@=stex_structures>
5136 \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
5137   \prop_if_exist:cF {c_stex_module_ \l_stex_current_module_str _structures}{
5138     \prop_new:c {c_stex_module_ \l_stex_current_module_str _structures}
5139   }
5140   \prop_gput:cxx{c_stex_module_ \l_stex_current_module_str _structures}
5141   {#1}{#2}
5142 }
5143

```

```

5144 \keys_define:nn { stex / features / structure } {
5145   name          .str_set_x:N = \l__stex_structures_name_str ,
5146 }
5147
5148 \cs_new_protected:Nn \__stex_structures_structure_args:n {
5149   \str_clear:N \l__stex_structures_name_str
5150   \keys_set:nn { stex / features / structure } { #1 }
5151 }
5152
5153 \NewDocumentEnvironment{mathstructure}{m O{}}{
5154   \__stex_structures_structure_args:n { #2 }
5155   \str_if_empty:NT \l__stex_structures_name_str {
5156     \str_set:Nx \l__stex_structures_name_str { #1 }
5157   }
5158   \stex_suppress_html:n {
5159     \bool_set_true:N \l_stex_symdecl_make_macro_bool
5160     \exp_args:Nx \stex_symdecl_do:nn {
5161       name = \l__stex_structures_name_str ,
5162       def = {\STEXsymbol{module-type}}{
5163         \STEXInternalTermMathOMSiiii {
5164           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
5165             { ns } ?
5166           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
5167             { name } / \l__stex_structures_name_str - structure
5168         }{}{}{}
5169       }}
5170     }{ #1 }
5171   }
5172   \exp_args:Nnnx
5173   \begin{structural_feature_module}{ structure }
5174     { \l__stex_structures_name_str }{}
5175   \stex_smsmode_do:
5176 }{
5177   \end{structural_feature_module}
5178   \stex_reset_up_to_module:n \l_stex_last_feature_str
5179   \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
5180   \seq_clear:N \l_tmpa_seq
5181   \seq_map_inline:Nn \l_stex_collect_imports_seq {
5182     \seq_map_inline:cn{c_stex_module_##1_constants}{
5183       \seq_put_right:Nn \l_tmpa_seq { ##1 ? ####1 }
5184     }
5185   }
5186   \exp_args:Nnno
5187   \prop_gput:cnn {c_stex_module_ \l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
5188   \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
5189   \stex_add_structure_to_current_module:nn
5190     \l__stex_structures_name_str
5191     \l_stex_last_feature_str
5192
5193   \stex_execute_in_module:x {
5194     \tl_set:cn { #1 }{
5195       \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures
5196     }
5197   }

```

```

5198 }
5199
5200 \cs_new:Nn \stex_invoke_structure:nn {
5201   \stex_invoke_symbol:n { #1?#2 }
5202 }
5203
5204 \cs_new_protected:Nn \stex_get_structure:n {
5205   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
5206     \tl_set:Nn \l_tmpa_tl { #1 }
5207     \__stex_structures_get_from_cs:
5208   }{
5209     \cs_if_exist:cTF { #1 }{
5210       \cs_set_eq:Nc \l_tmpa_cs { #1 }
5211       \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
5212       \str_if_empty:NTF \l_tmpa_str {
5213         \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs } \stex_invoke_structure:nn {
5214           \__stex_structures_get_from_cs:
5215         }{
5216           \__stex_structures_get_from_string:n { #1 }
5217         }
5218       }{
5219         \__stex_structures_get_from_string:n { #1 }
5220       }
5221     }{
5222       \__stex_structures_get_from_string:n { #1 }
5223     }
5224   }
5225 }
5226
5227 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
5228   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
5229     { \tl_tail:N \l_tmpa_tl }
5230   \str_set:Nx \l_tmpa_str {
5231     \exp_after:wN \use_i:nn \l_tmpa_tl
5232   }
5233   \str_set:Nx \l_tmpb_str {
5234     \exp_after:wN \use_ii:nn \l_tmpa_tl
5235   }
5236   \str_set:Nx \l_stex_get_structure_str {
5237     \l_tmpa_str ? \l_tmpb_str
5238   }
5239   \str_set:Nx \l_stex_get_structure_module_str {
5240     \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
5241   }
5242 }
5243
5244 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
5245   \tl_set:Nn \l_tmpa_tl {
5246     \msg_error:nnn{stex}{error/unknownstructure}{#1}
5247   }
5248   \str_set:Nn \l_tmpa_str { #1 }
5249   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
5250
5251   \seq_map_inline:Nn \l_stex_all_modules_seq {

```

```

5252 \prop_if_exist:cT {c_stex_module_##1_structures} {
5253 \prop_map_inline:cn {c_stex_module_##1_structures} {
5254 \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
5255 \prop_map_break:n{\seq_map_break:n{
5256 \tl_set:Nn \l_tmpa_tl {
5257 \str_set:Nn \l_stex_get_structure_str {##1?####1}
5258 \str_set:Nn \l_stex_get_structure_module_str {####2}
5259 }
5260 }}
5261 }
5262 }
5263 }
5264 }
5265 \l_tmpa_tl
5266 }

```

\instantiate

```

5267
5268 \NewDocumentEnvironment{usestructure}{m}{
5269 \stex_get_structure:n {#1}
5270 \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5271 }{}
5272
5273 \keys_define:nn { stex / instantiate } {
5274 name .str_set_x:N = \l__stex_structures_name_str
5275 }
5276 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
5277 \str_clear:N \l__stex_structures_name_str
5278 \keys_set:nn { stex / instantiate } { #1 }
5279 }
5280
5281 \NewDocumentCommand \instantiate {m O{} m m O{}}{
5282 \begingroup
5283 \stex_get_structure:n {#3}
5284 \__stex_structures_instantiate_args:n { #2 }
5285 \str_if_empty:NT \l__stex_structures_name_str {
5286 \str_set:Nn \l__stex_structures_name_str { #1 }
5287 }
5288 \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5289 \seq_clear:N \l__stex_structures_fields_seq
5290 \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
5291 \seq_map_inline:Nn \l_stex_collect_imports_seq {
5292 \seq_map_inline:cn {c_stex_module_##1_constants}{
5293 \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
5294 }
5295 }
5296
5297 \tl_if_empty:nF{#5}{
5298 \seq_set_split:Nnn \l_tmpa_seq , {#5}
5299 \prop_clear:N \l_tmpa_prop
5300 \seq_map_inline:Nn \l_tmpa_seq {
5301 \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
5302 \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
5303 \msg_error:nnn{stex}{error/keyval}{##1}

```

```

5304     }
5305     \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
5306     \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
5307     \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
5308     \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}
5309     \exp_args:Nxx \str_if_eq:nnF
5310         {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5311         {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
5312         \msg_error:nnxxxx{stex}{error/incompatible}
5313         {\l__stex_structures_dom_str}
5314         {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5315         {\l_stex_get_symbol_uri_str}
5316         {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
5317     }
5318     \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str
5319 }
5320 }
5321
5322 \seq_map_inline:Nn \l__stex_structures_fields_seq {
5323     \str_set:Nx \l_tmpa_str {field:\l__stex_structures_name_str . \prop_item:cn {l_stex_sy
5324     \stex_debug:nn{instantiate}{Field~\l_tmpa_str :~##1}
5325
5326     \stex_add_constant_to_current_module:n {\l_tmpa_str}
5327     \stex_execute_in_module:x {
5328         \prop_set_from_keyval:cn { l_stex_symdecl_ \l_stex_current_module_str?\l_tmpa_str _p
5329             name = \l_tmpa_str ,
5330             args = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
5331             arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
5332             assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs} ,
5333             argnames = {\prop_item:cn {l_stex_symdecl_##1_prop}{argnames}}
5334         }
5335         \seq_clear:c {l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _notations}
5336     }
5337
5338     \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
5339         \stex_find_notation:nn{##1}{}
5340         \stex_execute_in_module:x {
5341             \seq_put_right:cn {l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _notation
5342         }
5343
5344         \stex_copy_control_sequence_ii:ccN
5345             {stex_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_notation_
5346             {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5347             \l_tmpa_tl
5348         \exp_args:No \stex_execute_in_module:n \l_tmpa_tl
5349
5350
5351         \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
5352             \tl_set_eq:Nc \l_tmpa_cs {stex_op_notation_##1\c_hash_str \l_stex_notation_variant
5353             \stex_execute_in_module:x {
5354                 \tl_set:cn
5355                     {stex_op_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_not
5356                     { \exp_args:No \exp_not:n \l_tmpa_cs}
5357             }

```

```

5358     }
5359
5360   }
5361
5362   \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\l_stex_cur
5363 }
5364
5365   \stex_execute_in_module:x {
5366     \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
5367       domain = \l_stex_get_structure_module_str ,
5368     \prop_to_keyval:N \l_tmpa_prop
5369   }
5370   \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
5371 }
5372   \stex_debug:nn{instantiate}{
5373     Instance~\l_stex_current_module_str?\l__stex_structures_name_str \
5374     \prop_to_keyval:N \l_tmpa_prop
5375   }
5376   \exp_args:Nxx \stex_symdecl_do:nn {
5377     type={\STEXsymbol{module-type}}{
5378       \STEXInternalTermMathOMSiiii {
5379         \l_stex_get_structure_module_str
5380       }{}{0}{}
5381     }}
5382   }{\l__stex_structures_name_str}
5383   % {
5384     \str_set:Nx \l_stex_get_symbol_uri_str {\l_stex_current_module_str?\l__stex_structures
5385     \tl_set:Nn \l_stex_notation_after_do_tl {\__stex_notation_final:}
5386     \stex_notation_do:nnnnn{}{}{}{}{\comp{#4}}
5387   % }
5388   %\exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}
5389   \endgroup
5390   \stex_smsmode_do:\ignorespacesandpars
5391 }
5392
5393 \cs_new_protected:Nn \stex_symbol_or_var:n {
5394   \cs_if_exist:cTF{#1}{
5395     \cs_set_eq:Nc \l_tmpa_tl { #1 }
5396     \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
5397     \str_if_empty:NTF \l_tmpa_str {
5398       \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
5399       \stex_invoke_variable:n {
5400         \bool_set_true:N \l_stex_symbol_or_var_bool
5401         \bool_set_false:N \l_stex_instance_or_symbol_bool
5402         \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
5403         \tl_set:Nx \l_tmpa_tl {\exp_after:wN \use:n \l_tmpa_tl}
5404         \str_set:Nx \l_stex_get_symbol_uri_str {
5405           \exp_after:wN \use:n \l_tmpa_tl
5406         }
5407       }{ % TODO \stex_invoke_varinstance:n
5408         \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl } \stex_invoke_varinstance:n {
5409           \bool_set_true:N \l_stex_symbol_or_var_bool
5410           \bool_set_true:N \l_stex_instance_or_symbol_bool
5411           \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}

```



```

5412         \tl_set:Nx \l_tmpa_tl {\exp_after:wN \use:n \l_tmpa_tl}
5413         \str_set:Nx \l_stex_get_symbol_uri_str {
5414             \exp_after:wN \use:n \l_tmpa_tl
5415         }
5416     }{
5417         \bool_set_false:N \l_stex_symbol_or_var_bool
5418         \stex_get_symbol:n{#1}
5419     }
5420 }
5421 }{
5422     \__stex_structures_symbolorvar_from_string:n{ #1 }
5423 }
5424 }{
5425     \__stex_structures_symbolorvar_from_string:n{ #1 }
5426 }
5427 }
5428
5429 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
5430     \prop_if_exist:cTF {l_stex_symdecl_var://#1 _prop}{
5431         \bool_set_true:N \l_stex_symbol_or_var_bool
5432         \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
5433     }{
5434         \bool_set_false:N \l_stex_symbol_or_var_bool
5435         \stex_get_symbol:n{#1}
5436     }
5437 }
5438
5439 \keys_define:nn { stex / varinstantiate } {
5440     name          .str_set_x:N = \__stex_structures_name_str,
5441     bind          .choices:nn =
5442         {forall,exists}
5443         {\str_set:Nx \l__stex_structures_bind_str {\l_keys_choice_tl}}
5444 }
5445 }
5446 \cs_new_protected:Nn \__stex_structures_varinstantiate_args:n {
5447     \str_clear:N \l__stex_structures_name_str
5448     \str_clear:N \l__stex_structures_bind_str
5449     \keys_set:nn { stex / varinstantiate } { #1 }
5450 }
5451
5452 \NewDocumentCommand \varinstantiate {m O{} m m O{}}{
5453     \begin{group}
5454         \stex_get_structure:n {#3}
5455         \__stex_structures_varinstantiate_args:n { #2 }
5456         \str_if_empty:NT \l__stex_structures_name_str {
5457             \str_set:Nn \l__stex_structures_name_str { #1 }
5458         }
5459         \stex_if_do_html:TF{
5460             \stex_annotate:nnn{varinstance}{\l__stex_structures_name_str}
5461         }{\use:n}
5462         {
5463             \stex_if_do_html:T{
5464                 \stex_annotate_invisible:nnn{domain}{\l_stex_get_structure_module_str}{}
5465             }

```

```

5466 \seq_clear:N \l__stex_structures_fields_seq
5467 \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
5468 \seq_map_inline:Nn \l_stex_collect_imports_seq {
5469     \seq_map_inline:cn {c_stex_module_##1_constants}{
5470         \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
5471     }
5472 }
5473 \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5474 \prop_clear:N \l_tmpa_prop
5475 \tl_if_empty:nF {#5} {
5476     \seq_set_split:Nnn \l_tmpa_seq , {#5}
5477     \seq_map_inline:Nn \l_tmpa_seq {
5478         \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
5479         \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
5480             \msg_error:nnn{stex}{error/keyval}{##1}
5481         }
5482         \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_structures_fields_seq
5483         \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
5484         \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_uri_str
5485         \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
5486         \stex_if_do_html:T{
5487             \stex_annotate:nnn{assign}{\l__stex_structures_dom_str,
5488                 \bool_if:NTF\l_stex_symbol_or_var_bool{var://}{\l_stex_get_symbol_uri_str}{}}
5489         }
5490         \bool_if:NTF \l_stex_symbol_or_var_bool {
5491             \exp_args:Nxx \str_if_eq:nnF
5492                 {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
5493                 {\prop_item:cn{l_stex_symdecl_var://\l_stex_get_symbol_uri_str _prop}{args}}}{
5494                 \msg_error:nnxxx{stex}{error/incompatible}
5495                 {\l__stex_structures_dom_str}
5496                 {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
5497                 {\l_stex_get_symbol_uri_str}
5498                 {\prop_item:cn{l_stex_symdecl_var://\l_stex_get_symbol_uri_str _prop}{args}}}
5499         }
5500         \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n {##1}}
5501     }{
5502         \exp_args:Nxx \str_if_eq:nnF
5503             {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
5504             {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}{
5505                 \msg_error:nnxxx{stex}{error/incompatible}
5506                 {\l__stex_structures_dom_str}
5507                 {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
5508                 {\l_stex_get_symbol_uri_str}
5509                 {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}}
5510     }
5511     \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n {##1}}
5512 }
5513 }
5514 }
5515 \tl_gclear:N \g__stex_structures_aftergroup_tl
5516 \seq_map_inline:Nn \l__stex_structures_fields_seq {
5517     \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdecl_
5518         \stex_debug:nn{varinstantiate}{Field~\l_tmpa_str :~##1}}
5519     \seq_if_empty:cF{l_stex_symdecl_##1_notations}{

```

```

5520         \stex_find_notation:nn{##1}{}
5521         \cs_gset_eq:cc{g__stex_structures_tmpa_\l_tmpa_str_cs}
5522             {stex_notation_##1\c_hash_str \l_stex_notation_variant_str_cs}
5523         \stex_debug:nn{varinstantiate}{Notation:~\cs_meaning:c{g__stex_structures_tmpa_\l_
5524         \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str_cs}{
5525             \cs_gset_eq:cc {g__stex_structures_tmpa_op_\l_tmpa_str_cs}
5526             {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str_cs}
5527             \stex_debug:nn{varinstantiate}{Operator~Notation:~\cs_meaning:c{g__stex_struct
5528         }
5529     }
5530
5531     \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
5532         \prop_set_from_keyval:cn {l_stex_symdecl_ var://\l_tmpa_str_prop}{
5533             name      = \l_tmpa_str ,
5534             args      = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
5535             arity     = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
5536             assocs    = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs} ,
5537             argnames  = {\prop_item:cn {l_stex_symdecl_##1_prop}{argnames}} ,
5538         }
5539         \cs_set_eq:cc {stex_var_notation_\l_tmpa_str_cs}
5540             {g__stex_structures_tmpa_\l_tmpa_str_cs}
5541         \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str_cs}
5542             {g__stex_structures_tmpa_op_\l_tmpa_str_cs}
5543     }
5544     \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\stex_inv
5545 }
5546     \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
5547         \prop_set_from_keyval:cn {l_stex_varinstance_\l__stex_structures_name_str_prop }{
5548             domain = \l_stex_get_structure_module_str ,
5549             \prop_to_keyval:N \l_tmpa_prop
5550         }
5551         \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
5552         \tl_set:cn {l_stex_varinstance_\l__stex_structures_name_str_op_tl}{
5553             \exp_args:Nnx \exp_not:N \use:nn {
5554                 \str_set:Nn \exp_not:N \STEXInternalCurrentSymbolStr {var://\l__stex_structures_
5555                 \stex_term_omv:nn {var://\l__stex_structures_name_str}{
5556                     \exp_not:n{
5557                         \varcomp{#4}
5558                     }
5559                 }
5560             }{
5561                 \exp_not:n{\stex_reset:N \STEXInternalCurrentSymbolStr}
5562             }
5563         }
5564     }
5565 }
5566     \stex_debug:nn{varinstantiate}{\expandafter\detokenize\expandafter{\g__stex_structures_a
5567     \aftergroup\g__stex_structures_aftergroup_tl
5568 \endgroup
5569 \stex_smsmode_do:\ignorespacesandpars
5570 }
5571
5572 \cs_new_protected:Nn \stex_invoke_instance:n {
5573     \peek_charcode_remove:NTF ! {

```

```

5574     \stex_invoke_symbol:n{#1}
5575   }{
5576     \_stex_invoke_instance:nn {#1}
5577   }
5578 }
5579
5580
5581 \cs_new_protected:Nn \stex_invoke_varinstance:n {
5582   \peek_charcode_remove:NTF ! {
5583     \exp_args:Nnx \use:nn {
5584       \def\comp{\_varcomp}
5585       \use:c{l_stex_varinstance_#1_op_tl}
5586     }{
5587       \_stex_reset:N \comp
5588     }
5589   }{
5590     \_stex_invoke_varinstance:nn {#1}
5591   }
5592 }
5593
5594 \cs_new_protected:Nn \_stex_invoke_instance:nn {
5595   \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
5596     \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
5597   }{
5598     \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
5599     \msg_error:nnxxx{stex}{error/unknownfield}{#2}{#1}{
5600       \prop_to_keyval:N \l_tmpa_prop
5601     }
5602   }
5603 }
5604
5605 \cs_new_protected:Nn \_stex_invoke_varinstance:nn {
5606   \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
5607     \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
5608     \l_tmpa_tl
5609   }{
5610     \msg_error:nnnnn{stex}{error/unknownfield}{#2}{#1}{
5611     }
5612   }

```

(End definition for \instantiate. This function is documented on page 34.)

\stex_invoke_structure:nnn

```

5613 % #1: URI of the instance
5614 % #2: URI of the instantiated module
5615 \cs_new_protected:Nn \stex_invoke_structure:nnn {
5616   \tl_if_empty:nTF{ #3 }{
5617     \prop_set_eq:Nc \l__stex_structures_structure_prop {
5618       c_stex_feature_ #2 _prop
5619     }
5620     \tl_clear:N \l_tmpa_tl
5621     \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
5622     \seq_map_inline:Nn \l_tmpa_seq {
5623       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }

```

```

5624 \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
5625 \cs_if_exist:cT {
5626   stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
5627 }{
5628   \tl_if_empty:NF \l_tmpa_tl {
5629     \tl_put_right:Nn \l_tmpa_tl {,}
5630   }
5631   \tl_put_right:Nx \l_tmpa_tl {
5632     \stex_invoke_symbol:n {#1/\l_tmpa_str}!
5633   }
5634 }
5635 }
5636 \exp_args:No \mathstruct \l_tmpa_tl
5637 }{
5638   \stex_invoke_symbol:n{#1/#3}
5639 }
5640 }

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

5641 </package>

```

Chapter 32

STEX -Statements Implementation

```
5642 <*package>
5643
5644 %%%%%%%%%%% features.dtx %%%%%%%%%%%
5645
5646 <@@=stex_statements>
    Warnings and error messages
5647

\titleemph
5648 \def\titleemph#1{\textbf{#1}}

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

32.1 Definitions

definiendum

```
5649 \keys_define:nn {stex / definiendum }{
5650   pre      .tl_set:N      = \l__stex_statements_definiendum_pre_tl,
5651   post     .tl_set:N      = \l__stex_statements_definiendum_post_tl,
5652   root     .str_set_x:N    = \l__stex_statements_definiendum_root_str,
5653   gfa      .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
5654 }
5655 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
5656   \str_clear:N \l__stex_statements_definiendum_root_str
5657   \tl_clear:N \l__stex_statements_definiendum_post_tl
5658   \str_clear:N \l__stex_statements_definiendum_gfa_str
5659   \keys_set:nn { stex / definiendum }{ #1 }
5660 }
5661 \NewDocumentCommand \definiendum { O{} m m } {
5662   \__stex_statements_definiendum_args:n { #1 }
5663   \stex_get_symbol:n { #2 }
5664   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5665   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
5666     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

```

5667     \tl_set:Nn \l_tmpa_tl { #3 }
5668   } {
5669     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
5670     \tl_set:Nn \l_tmpa_tl {
5671       \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
5672     }
5673   }
5674 } {
5675   \tl_set:Nn \l_tmpa_tl { #3 }
5676 }
5677
5678 % TODO root
5679 \stex_html_backend:TF {
5680   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
5681 } {
5682   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
5683 }
5684 }
5685 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

(End definition for definiendum. This function is documented on page 44.)

definame

```

5686
5687 \NewDocumentCommand \definame { 0{ } m } {
5688   \__stex_statements_definiendum_args:n { #1 }
5689   % TODO: root
5690   \stex_get_symbol:n { #2 }
5691   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5692   \str_set:Nx \l_tmpa_str {
5693     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5694   }
5695   \str_replace_all:Nnn \l_tmpa_str {-} {~}
5696   \stex_html_backend:TF {
5697     \stex_if_do_html:T {
5698       \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5699         \l_tmpa_str\l__stex_statements_definiendum_post_tl
5700       }
5701     }
5702   } {
5703     \exp_args:Nnx \defemph@uri {
5704       \l_tmpa_str\l__stex_statements_definiendum_post_tl
5705     } { \l_stex_get_symbol_uri_str }
5706   }
5707 }
5708 \stex_deactivate_macro:Nn \definame {definition~environments}
5709
5710 \NewDocumentCommand \Definame { 0{ } m } {
5711   \__stex_statements_definiendum_args:n { #1 }
5712   \stex_get_symbol:n { #2 }
5713   \str_set:Nx \l_tmpa_str {
5714     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5715   }
5716   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}

```

```

5717 \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5718 \stex_html_backend:TF {
5719   \stex_if_do_html:T {
5720     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5721       \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5722     }
5723   }
5724 } {
5725   \exp_args:Nnx \defemph@uri {
5726     \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5727   } { \l_stex_get_symbol_uri_str }
5728 }
5729 }
5730 \stex_deactivate_macro:Nn \Definame {definition-environments}
5731
5732 \NewDocumentCommand \premise { m }{
5733   \noindent\stex_annotate:nnn{ premise }{}{\ignorespaces #1 }
5734 }
5735 \NewDocumentCommand \conclusion { m }{
5736   \noindent\stex_annotate:nnn{ conclusion }{}{\ignorespaces #1 }
5737 }
5738 \NewDocumentCommand \definiens { 0{} m }{
5739   \str_clear:N \l_stex_get_symbol_uri_str
5740   \tl_if_empty:nF {#1} {
5741     \stex_get_symbol:n { #1 }
5742   }
5743   \str_if_empty:NT \l_stex_get_symbol_uri_str {
5744     \int_compare:nNnTF {\clist_count:N \l__stex_statements_sdefinition_for_clist} = 1 {
5745       \str_set:Nx \l_stex_get_symbol_uri_str {\clist_item:Nn \l__stex_statements_sdefinition
5746     }{
5747       % TODO throw error
5748     }
5749   }
5750   \str_if_eq:eeT {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{module}}
5751   {\l_stex_current_module_str}{
5752     \str_if_eq:eeF {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defin
5753   }{true}{
5754     \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5755     \exp_args:Nx \stex_add_to_current_module:n {
5756       \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5757     }
5758   }
5759 }
5760 \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
5761 }
5762
5763 \NewDocumentCommand \varbindforall {m}{
5764   \stex_symbol_or_var:n {#1}
5765   \bool_if:NTF\l_stex_symbol_or_var_bool{
5766     \stex_if_do_html:T {
5767       \stex_annotate_invisible:nnn {bindtype}{forall,\l_stex_get_symbol_uri_str}{}
5768     }
5769   }{
5770     % todo throw error

```



```

5771 }
5772 }
5773
5774 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
5775 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
5776 \stex_deactivate_macro:Nn \definiens {definition~environments}
5777 \stex_deactivate_macro:Nn \varbindforall {definition~or~assertion~environments}
5778

```

(End definition for definame. This function is documented on page 44.)

sdefinition (env.)

```

5779
5780 \keys_define:nn {stex / sdefinition }{
5781   type      .str_set_x:N = \sdefinitiontype,
5782   id        .str_set_x:N = \sdefinitionid,
5783   name      .str_set_x:N = \sdefinitionname,
5784   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5785   title     .tl_set:N    = \sdefinitiontitle
5786 }
5787 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
5788   \str_clear:N \sdefinitiontype
5789   \str_clear:N \sdefinitionid
5790   \str_clear:N \sdefinitionname
5791   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5792   \tl_clear:N \sdefinitiontitle
5793   \keys_set:nn { stex / sdefinition }{ #1 }
5794 }
5795
5796 \NewDocumentEnvironment{sdefinition}{0{}}{
5797   \__stex_statements_sdefinition_args:n{ #1 }
5798   \stex_reactivate_macro:N \definiendum
5799   \stex_reactivate_macro:N \definame
5800   \stex_reactivate_macro:N \Definame
5801   \stex_reactivate_macro:N \premise
5802   \stex_reactivate_macro:N \definiens
5803   \stex_reactivate_macro:N \varbindforall
5804   \stex_if_smsmode:F{
5805     \seq_clear:N \l_tmpb_seq
5806     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5807       \tl_if_empty:nF{ ##1 }{
5808         \stex_get_symbol:n { ##1 }
5809         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5810           \l_stex_get_symbol_uri_str
5811         }
5812       }
5813     }
5814     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5815     \exp_args:Nnnx
5816     \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpb_seq {,}}
5817     \str_if_empty:NF \sdefinitiontype {
5818       \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{ }
5819     }
5820     \str_if_empty:NF \sdefinitionname {

```

```

5821     \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5822   }
5823   \clist_set:No \l_tmpa_clist \sdefinitiontype
5824   \tl_clear:N \l_tmpa_tl
5825   \clist_map_inline:Nn \l_tmpa_clist {
5826     \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
5827       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
5828     }
5829   }
5830   \tl_if_empty:NTF \l_tmpa_tl {
5831     \__stex_statements_sdefinition_start:
5832   }{
5833     \l_tmpa_tl
5834   }
5835 }
5836 \stex_ref_new_doc_target:n \sdefinitionid
5837 \stex_smsmode_do:
5838 ){
5839   \stex_suppress_html:n {
5840     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5841   }
5842   \stex_if_smsmode:F {
5843     \clist_set:No \l_tmpa_clist \sdefinitiontype
5844     \tl_clear:N \l_tmpa_tl
5845     \clist_map_inline:Nn \l_tmpa_clist {
5846       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
5847         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
5848       }
5849     }
5850     \tl_if_empty:NTF \l_tmpa_tl {
5851       \__stex_statements_sdefinition_end:
5852     }{
5853       \l_tmpa_tl
5854     }
5855     \end{stex_annotate_env}
5856   }
5857 }

```

\stexpatchdefinition

```

5858 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
5859   \stex_par:\noindent\titleemph{Definition}\tl_if_empty:NF \sdefinitiontitle {
5860     ~(\sdefinitiontitle)
5861   }~}
5862 }
5863 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\stex_par:\medskip}
5864
5865 \newcommand\stexpatchdefinition[3]{} {
5866   \str_set:Nx \l_tmpa_str{ #1 }
5867   \str_if_empty:NTF \l_tmpa_str {
5868     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
5869     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
5870   }{
5871     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
5872     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }

```

```

5873     }
5874 }

```

(End definition for `\stexpatchdefinition`. This function is documented on page 51.)

`\inlinedef` inline:

```

5875 \keys_define:nn {stex / inlinedef }{
5876   type      .str_set_x:N = \sdefinitiontype,
5877   id        .str_set_x:N = \sdefinitionid,
5878   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5879   name      .str_set_x:N = \sdefinitionname
5880 }
5881 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
5882   \str_clear:N \sdefinitiontype
5883   \str_clear:N \sdefinitionid
5884   \str_clear:N \sdefinitionname
5885   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5886   \keys_set:nn { stex / inlinedef }{ #1 }
5887 }
5888 \NewDocumentCommand \inlinedef { 0{} m } {
5889   \begingroup
5890   \__stex_statements_inlinedef_args:n{ #1 }
5891   \stex_reactivate_macro:N \definiendum
5892   \stex_reactivate_macro:N \definame
5893   \stex_reactivate_macro:N \Definame
5894   \stex_reactivate_macro:N \premise
5895   \stex_reactivate_macro:N \definiens
5896   \stex_reactivate_macro:N \varbindforall
5897   \stex_ref_new_doc_target:n \sdefinitionid
5898   \stex_if_smsmode:TF{\stex_suppress_html:n {
5899     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5900   }}{
5901     \seq_clear:N \l_tmpb_seq
5902     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5903       \tl_if_empty:nF{ ##1 }{
5904         \stex_get_symbol:n { ##1 }
5905         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5906           \l_stex_get_symbol_uri_str
5907         }
5908       }
5909     }
5910     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5911     \exp_args:Nnx
5912     \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpb_seq {,}}{
5913       \str_if_empty:NF \sdefinitiontype {
5914         \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5915       }
5916       #2
5917       \str_if_empty:NF \sdefinitionname {
5918         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sdefinitionname}}
5919         \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5920       }
5921     }
5922   }

```

```

5923 \endgroup
5924 \stex_smsmode_do:
5925 }

```

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

32.2 Assertions

sassertion (*env.*)

```

5926 \keys_define:nn {stex / sassertion }{
5927   type      .str_set_x:N = \sassertiontype,
5928   id        .str_set_x:N = \sassertionid,
5929   title     .tl_set:N     = \sassertiontitle ,
5930   for       .clist_set:N  = \l__stex_statements_sassertion_for_clist ,
5931   name      .str_set_x:N  = \sassertionname
5932 }
5933
5934 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
5935   \str_clear:N \sassertiontype
5936   \str_clear:N \sassertionid
5937   \str_clear:N \sassertionname
5938   \clist_clear:N \l__stex_statements_sassertion_for_clist
5939   \tl_clear:N \sassertiontitle
5940   \keys_set:nn { stex / sassertion }{ #1 }
5941 }
5942
5943 %\tl_new:N \g__stex_statements_aftergroup_tl
5944
5945 \NewDocumentEnvironment{sassertion}{0}{}{
5946   \__stex_statements_sassertion_args:n{ #1 }
5947   \stex_reactivate_macro:N \premise
5948   \stex_reactivate_macro:N \conclusion
5949   \stex_reactivate_macro:N \varbindforall
5950   \stex_if_smsmode:F {
5951     \seq_clear:N \l_tmpb_seq
5952     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5953       \tl_if_empty:nF{ ##1 }{
5954         \stex_get_symbol:n { ##1 }
5955         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5956           \l_stex_get_symbol_uri_str
5957         }
5958       }
5959     }
5960     \exp_args:Nnnx
5961     \begin{sassertion_env}{sassertion}{\seq_use:Nn \l_tmpb_seq {},}}
5962     \str_if_empty:NF \sassertiontype {
5963       \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
5964     }
5965     \str_if_empty:NF \sassertionname {
5966       \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
5967     }
5968     \clist_set:N \l_tmpa_clist \sassertiontype
5969     \tl_clear:N \l_tmpa_tl

```

```

5970 \clist_map_inline:Nn \l_tmpa_clist {
5971   \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
5972     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
5973   }
5974 }
5975 \tl_if_empty:NTF \l_tmpa_tl {
5976   \__stex_statements_sassertion_start:
5977 }{
5978   \l_tmpa_tl
5979 }
5980 }
5981 \str_if_empty:NTF \sassertionid {
5982   \str_if_empty:NF \sassertionname {
5983     \stex_ref_new_doc_target:n {}
5984   }
5985 } {
5986   \stex_ref_new_doc_target:n \sassertionid
5987 }
5988 \stex_smsmode_do:
5989 ){
5990   \str_if_empty:NF \sassertionname {
5991     \stex_suppress_html:n{\stex_symdecl_do:nn}{\sassertionname}}
5992     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5993   }
5994   \stex_if_smsmode:F {
5995     \clist_set:No \l_tmpa_clist \sassertiontype
5996     \tl_clear:N \l_tmpa_tl
5997     \clist_map_inline:Nn \l_tmpa_clist {
5998       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
5999         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
6000       }
6001     }
6002     \tl_if_empty:NTF \l_tmpa_tl {
6003       \__stex_statements_sassertion_end:
6004     }{
6005       \l_tmpa_tl
6006     }
6007     \end{stex_annotate_env}
6008   }
6009 }

```

\stexpatchassertion

```

6010
6011 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
6012   \stex_par:\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {
6013     (\sassertiontitle)
6014   }~}
6015 }
6016 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\stex_par:\medskip}
6017
6018 \newcommand\stexpatchassertion[3] [] {
6019   \str_set:Nx \l_tmpa_str{ #1 }
6020   \str_if_empty:NTF \l_tmpa_str {
6021     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }

```

```

6022     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
6023   }{
6024     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
6025     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
6026   }
6027 }

```

(End definition for `\stexpatchassertion`. This function is documented on page 51.)

`\inlineass` inline:

```

6028 \keys_define:nn {stex / inlineass }{
6029   type      .str_set_x:N = \sassertiontype,
6030   id        .str_set_x:N = \sassertionid,
6031   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
6032   name      .str_set_x:N = \sassertionname
6033 }
6034 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
6035   \str_clear:N \sassertiontype
6036   \str_clear:N \sassertionid
6037   \str_clear:N \sassertionname
6038   \clist_clear:N \l__stex_statements_sassertion_for_clist
6039   \keys_set:nn { stex / inlineass }{ #1 }
6040 }
6041 \NewDocumentCommand \inlineass { 0{} m } {
6042   \begingroup
6043   \stex_reactivate_macro:N \premise
6044   \stex_reactivate_macro:N \conclusion
6045   \stex_reactivate_macro:N \varbindforall
6046   \__stex_statements_inlineass_args:n{ #1 }
6047   \str_if_empty:NTF \sassertionid {
6048     \str_if_empty:NF \sassertionname {
6049       \stex_ref_new_doc_target:n {}
6050     }
6051   } {
6052     \stex_ref_new_doc_target:n \sassertionid
6053   }
6054
6055   \stex_if_smsmode:TF{
6056     \str_if_empty:NF \sassertionname {
6057       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
6058       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
6059     }
6060   }{
6061     \seq_clear:N \l_tmpb_seq
6062     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
6063       \tl_if_empty:nF{ ##1 }{
6064         \stex_get_symbol:n { ##1 }
6065         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6066           \l_stex_get_symbol_uri_str
6067         }
6068       }
6069     }
6070     \exp_args:Nnx
6071     \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpb_seq {,}}{

```

```

6072     \str_if_empty:NF \sassassertiontype {
6073       \stex_annotate_invisible:nnn{typestrings}{\sassassertiontype}{ }
6074     }
6075     #2
6076     \str_if_empty:NF \sassassertionname {
6077       \stex_suppress_html:n{\stex_symdecl_do:nn{ }{\sassassertionname}}
6078       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassassertionname}
6079       \stex_annotate_invisible:nnn{statementname}{\sassassertionname}{ }
6080     }
6081   }
6082 }
6083 \endgroup
6084 \stex_smsmode_do:
6085 }

```

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

32.3 Examples

`sexample (env.)`

```

6086
6087 \keys_define:nn {stex / sexample }{
6088   type      .str_set_x:N = \exampletype,
6089   id        .str_set_x:N = \sexampleid,
6090   title     .tl_set:N    = \sexamplename,
6091   name      .str_set_x:N = \sexamplename ,
6092   for       .clist_set:N = \l__stex_statements_sexample_for_clist,
6093 }
6094 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
6095   \str_clear:N \sexampletype
6096   \str_clear:N \sexampleid
6097   \str_clear:N \sexamplename
6098   \tl_clear:N \sexamplename
6099   \clist_clear:N \l__stex_statements_sexample_for_clist
6100   \keys_set:nn { stex / sexample }{ #1 }
6101 }
6102
6103 \NewDocumentEnvironment{sexample}{0{}}{
6104   \__stex_statements_sexample_args:n{ #1 }
6105   \stex_reactivate_macro:N \premise
6106   \stex_reactivate_macro:N \conclusion
6107   \stex_if_smsmode:F {
6108     \seq_clear:N \l_tmpb_seq
6109     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
6110       \tl_if_empty:nF{ ##1 }{
6111         \stex_get_symbol:n { ##1 }
6112         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6113           \l_stex_get_symbol_uri_str
6114         }
6115       }
6116     }
6117     \exp_args:Nnnx
6118     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpb_seq {,}}

```

```

6119 \str_if_empty:NF \sexamplotype {
6120 \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{}}
6121 }
6122 \str_if_empty:NF \sexamplename {
6123 \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}}
6124 }
6125 \clist_set:No \l_tmpa_clist \sexamplotype
6126 \tl_clear:N \l_tmpa_tl
6127 \clist_map_inline:Nn \l_tmpa_clist {
6128 \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
6129 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
6130 }
6131 }
6132 \tl_if_empty:NTF \l_tmpa_tl {
6133 \__stex_statements_sexample_start:
6134 }{
6135 \l_tmpa_tl
6136 }
6137 }
6138 \str_if_empty:NF \sexampleid {
6139 \stex_ref_new_doc_target:n \sexampleid
6140 }
6141 \stex_smsmode_do:
6142 }{
6143 \str_if_empty:NF \sexamplename {
6144 \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
6145 }
6146 \stex_if_smsmode:F {
6147 \clist_set:No \l_tmpa_clist \sexamplotype
6148 \tl_clear:N \l_tmpa_tl
6149 \clist_map_inline:Nn \l_tmpa_clist {
6150 \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
6151 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
6152 }
6153 }
6154 \tl_if_empty:NTF \l_tmpa_tl {
6155 \__stex_statements_sexample_end:
6156 }{
6157 \l_tmpa_tl
6158 }
6159 \end{stex_annotate_env}
6160 }
6161 }

```

\stexpatchexample

```

6162
6163 \cs_new_protected:Nn \__stex_statements_sexample_start: {
6164 \stex_par:\noindent\titllemph{Example~\tl_if_empty:NF \sexampltitle {
6165 (\sexampltitle)
6166 }~}
6167 }
6168 \cs_new_protected:Nn \__stex_statements_sexample_end: {\stex_par:\medskip}
6169
6170 \newcommand\stexpatchexample[3][ ] {

```



```

6171 \str_set:Nx \l_tmpa_str{ #1 }
6172 \str_if_empty:NTF \l_tmpa_str {
6173   \tl_set:Nn \__stex_statements_sexample_start: { #2 }
6174   \tl_set:Nn \__stex_statements_sexample_end: { #3 }
6175 }{
6176   \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
6177   \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
6178 }
6179 }

```

(End definition for `\stexpatchexample`. This function is documented on page 51.)

`\inlineex` inline:

```

6180 \keys_define:nn {stex / inlineex }{
6181   type      .str_set_x:N = \sexamplotype,
6182   id        .str_set_x:N = \sexampleid,
6183   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
6184   name      .str_set_x:N = \sexamplename
6185 }
6186 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
6187   \str_clear:N \sexamplotype
6188   \str_clear:N \sexampleid
6189   \str_clear:N \sexamplename
6190   \clist_clear:N \l__stex_statements_sexample_for_clist
6191   \keys_set:nn { stex / inlineex }{ #1 }
6192 }
6193 \NewDocumentCommand \inlineex { 0{} m } {
6194   \begingroup
6195   \stex_reactivate_macro:N \premise
6196   \stex_reactivate_macro:N \conclusion
6197   \__stex_statements_inlineex_args:n{ #1 }
6198   \str_if_empty:NF \sexampleid {
6199     \stex_ref_new_doc_target:n \sexampleid
6200   }
6201   \stex_if_smsmode:TF{
6202     \str_if_empty:NF \sexamplename {
6203       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sexamplename}}
6204     }
6205   }{
6206     \seq_clear:N \l_tmpb_seq
6207     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
6208       \tl_if_empty:nF{ ##1 }{
6209         \stex_get_symbol:n { ##1 }
6210         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6211           \l_stex_get_symbol_uri_str
6212         }
6213       }
6214     }
6215     \exp_args:Nnx
6216     \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpb_seq {,}}{
6217       \str_if_empty:NF \sexamplotype {
6218         \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{ }
6219       }
6220     } #2

```

```

6221     \str_if_empty:NF \sexamplename {
6222       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sexamplename}}
6223       \stex_annotate_invisible:nnn{statementname}{\sexamplename}{ }
6224     }
6225   }
6226 }
6227 \endgroup
6228 \stex_smsmode_do:
6229 }

```

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

32.4 Logical Paragraphs

`sparagraph` (*env.*)

```

6230 \keys_define:nn { stex / sparagraph } {
6231   id      .str_set:x:N = \sparagraphid ,
6232   title   .tl_set:N    = \l_stex_sparagraph_title_tl ,
6233   type    .str_set:x:N = \sparagraphtype ,
6234   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
6235   from    .tl_set:N    = \sparagraphfrom ,
6236   to      .tl_set:N    = \sparagraphto ,
6237   start   .tl_set:N    = \l_stex_sparagraph_start_tl ,
6238   name    .str_set:N    = \sparagraphname ,
6239   imports .tl_set:N    = \l__stex_statements_sparagraph_imports_tl
6240 }
6241
6242 \cs_new_protected:Nn \stex_sparagraph_args:n {
6243   \tl_clear:N \l_stex_sparagraph_title_tl
6244   \tl_clear:N \sparagraphfrom
6245   \tl_clear:N \sparagraphto
6246   \tl_clear:N \l_stex_sparagraph_start_tl
6247   \tl_clear:N \l__stex_statements_sparagraph_imports_tl
6248   \str_clear:N \sparagraphid
6249   \str_clear:N \sparagraphtype
6250   \clist_clear:N \l__stex_statements_sparagraph_for_clist
6251   \str_clear:N \sparagraphname
6252   \keys_set:nn { stex / sparagraph } { #1 }
6253 }
6254 \newif\if@in@omtext\@in@omtextfalse
6255
6256 \NewDocumentEnvironment {sparagraph} { 0{ } } {
6257   \stex_sparagraph_args:n { #1 }
6258   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
6259     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
6260   }{
6261     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
6262   }
6263   \@in@omtexttrue
6264   \stex_if_smsmode:F {
6265     \seq_clear:N \l_tmpb_seq
6266     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
6267       \tl_if_empty:nF{ ##1 }{

```

```

6268     \stex_get_symbol:n { ##1 }
6269     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6270         \l_stex_get_symbol_uri_str
6271     }
6272 }
6273 }
6274 \exp_args:Nnnx
6275 \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}
6276 \str_if_empty:NF \sparagraphtype {
6277     \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
6278 }
6279 \str_if_empty:NF \sparagraphfrom {
6280     \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
6281 }
6282 \str_if_empty:NF \sparagraphto {
6283     \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
6284 }
6285 \str_if_empty:NF \sparagraphname {
6286     \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
6287 }
6288 \clist_set:No \l_tmpa_clist \sparagraphtype
6289 \tl_clear:N \l_tmpa_tl
6290 \clist_map_inline:Nn \sparagraphtype {
6291     \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
6292         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
6293     }
6294 }
6295 \stex_csl_to_imports:No \usemodule \l__stex_statements_sparagraph_imports_tl
6296 \tl_if_empty:NTF \l_tmpa_tl {
6297     \__stex_statements_sparagraph_start:
6298 }{
6299     \l_tmpa_tl
6300 }
6301 }
6302 \clist_set:No \l_tmpa_clist \sparagraphtype
6303 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
6304     {
6305         \stex_reactivate_macro:N \definiendum
6306         \stex_reactivate_macro:N \definame
6307         \stex_reactivate_macro:N \Definame
6308         \stex_reactivate_macro:N \premise
6309         \stex_reactivate_macro:N \definiens
6310     }
6311 \str_if_empty:NTF \sparagraphid {
6312     \str_if_empty:NTF \sparagraphname {
6313         \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
6314             \stex_ref_new_doc_target:n {}
6315         }
6316     } {
6317         \stex_ref_new_doc_target:n {}
6318     }
6319 } {
6320     \stex_ref_new_doc_target:n \sparagraphid
6321 }

```

```

6322 \exp_args:NNx
6323 \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
6324   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
6325     \tl_if_empty:nF{ ##1 }{
6326       \stex_get_symbol:n { ##1 }
6327       \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
6328     }
6329   }
6330 }
6331 \stex_smsmode_do:
6332 \ignorespacesandpars
6333 }{
6334   \str_if_empty:NF \sparagraphname {
6335     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
6336     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
6337   }
6338   \stex_if_smsmode:F {
6339     \clist_set:No \l_tmpa_clist \sparagraphtype
6340     \tl_clear:N \l_tmpa_tl
6341     \clist_map_inline:Nn \l_tmpa_clist {
6342       \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
6343         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
6344       }
6345     }
6346     \tl_if_empty:NTF \l_tmpa_tl {
6347       \__stex_statements_sparagraph_end:
6348     }{
6349       \l_tmpa_tl
6350     }
6351     \end{stex_annotate_env}
6352   }
6353 }

```

\stexpatchparagraph

```

6354
6355 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
6356   \stex_par:\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
6357     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
6358       \titleemph{\l_stex_sparagraph_title_tl}:~
6359     }
6360   }{
6361     \titleemph{\l_stex_sparagraph_start_tl}~
6362   }
6363 }
6364 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\stex_par:\medskip}
6365
6366 \newcommand\stexpatchparagraph[3] [] {
6367   \str_set:Nx \l_tmpa_str{ #1 }
6368   \str_if_empty:NTF \l_tmpa_str {
6369     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
6370     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
6371   }{
6372     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
6373     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }

```

```

6374     }
6375 }
6376
6377 \keys_define:nn { stex / inlinepara } {
6378   id      .str_set:N = \sparagraphid ,
6379   type    .str_set:N = \sparagraphtype ,
6380   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
6381   from    .tl_set:N   = \sparagraphfrom ,
6382   to      .tl_set:N   = \sparagraphto ,
6383   name    .str_set:N   = \sparagraphname
6384 }
6385 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
6386   \tl_clear:N \sparagraphfrom
6387   \tl_clear:N \sparagraphto
6388   \str_clear:N \sparagraphid
6389   \str_clear:N \sparagraphtype
6390   \clist_clear:N \l__stex_statements_sparagraph_for_clist
6391   \str_clear:N \sparagraphname
6392   \keys_set:nn { stex / inlinepara } { #1 }
6393 }
6394 \NewDocumentCommand \inlinepara { O{} m } {
6395   \begingroup
6396   \__stex_statements_inlinepara_args:n{ #1 }
6397   \clist_set:Nn \l_tmpa_clist \sparagraphtype
6398   \str_if_empty:NTF \sparagraphid {
6399     \str_if_empty:NTF \sparagraphname {
6400       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
6401         \stex_ref_new_doc_target:n {}
6402       }
6403     } {
6404       \stex_ref_new_doc_target:n {}
6405     }
6406   } {
6407     \stex_ref_new_doc_target:n \sparagraphid
6408   }
6409   \stex_if_smsmode:TF{
6410     \str_if_empty:NF \sparagraphname {
6411       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
6412     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
6413   }
6414 }{
6415   \seq_clear:N \l_tmpb_seq
6416   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
6417     \tl_if_empty:nF{ ##1 }{
6418       \stex_get_symbol:n { ##1 }
6419       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6420         \l_stex_get_symbol_uri_str
6421       }
6422     }
6423   }
6424   \exp_args:Nnx
6425   \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}{
6426     \str_if_empty:NF \sparagraphtype {
6427       \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{

```

```

6428     }
6429     \str_if_empty:NF \sparagraphfrom {
6430         \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
6431     }
6432     \str_if_empty:NF \sparagraphto {
6433         \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
6434     }
6435     \str_if_empty:NF \sparagraphname {
6436         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
6437         \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
6438         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
6439     }
6440     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
6441         \clist_map_inline:Nn \l_tmpb_seq {
6442             \stex_ref_new_sym_target:n {##1}
6443         }
6444     }
6445     #2
6446 }
6447 }
6448 \endgroup
6449 \stex_smsmode_do:
6450 }
6451

```

(End definition for `\stexpatchparagraph`. This function is documented on page 51.)

```

6452 </package>

```

Chapter 33

The Implementation

```
6453 <*package>
6454 <@@=stex_sproof>
6455
6456 %%%%%%%%%%    sproof.dtx    %%%%%%%%%%
6457
```

33.1 Proofs

We first define some keys for the proof environment.

```
6458 \keys_define:nn { stex / spf } {
6459   id          .str_set_x:N = \spfid,
6460   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,
6461   from        .tl_set:N    = \l__stex_sproof_spf_from_tl ,
6462   proofend    .tl_set:N    = \l__stex_sproof_spf_proofend_tl,
6463   type        .str_set_x:N = \spftype,
6464   title       .tl_set:N    = \spftitle,
6465   continues   .tl_set:N    = \l__stex_sproof_spf_continues_tl,
6466   functions   .tl_set:N    = \l__stex_sproof_spf_functions_tl,
6467   term        .tl_set:N    = \l__stex_sproof_spf_term_tl,
6468   method      .tl_set:N    = \l__stex_sproof_spf_method_tl,
6469   hide        .bool_set:N  = \l__stex_sproof_spf_hide_bool
6470 }
6471 \cs_new_protected:Nn \l__stex_sproof_spf_args:n {
6472   \str_clear:N \spfid
6473   \tl_clear:N \l__stex_sproof_spf_for_tl
6474   \tl_clear:N \l__stex_sproof_spf_from_tl
6475   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
6476   \str_clear:N \spftype
6477   \tl_clear:N \spftitle
6478   \tl_clear:N \l__stex_sproof_spf_continues_tl
6479   \tl_clear:N \l__stex_sproof_spf_term_tl
6480   \tl_clear:N \l__stex_sproof_spf_functions_tl
6481   \tl_clear:N \l__stex_sproof_spf_method_tl
6482   \bool_set_false:N \l__stex_sproof_spf_hide_bool
6483   \keys_set:nn { stex / spf }{ #1 }
6484 }
6485 \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`

```
6486 \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.

```
6487 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
6488 \cs_new_protected:Npn\sproofnumber{
6489   \int_set:Nn\l_tmpa_int{1}
6490   \bool_while_do:nn{
6491     \int_compare_p:nNn{
6492       \intarray_item:Nn\l__stex_sproof_counter_intarray\l_tmpa_int
6493     }>0
6494   }{
6495     \intarray_item:Nn\l__stex_sproof_counter_intarray\l_tmpa_int.
6496     \int_incr:N\l_tmpa_int
6497   }
6498 }
6499 \cs_new_protected:Npn\__stex_sproof_inc_counter:{
6500   \int_set:Nn\l_tmpa_int{1}
6501   \bool_while_do:nn{
6502     \int_compare_p:nNn{
6503       \intarray_item:Nn\l__stex_sproof_counter_intarray\l_tmpa_int
6504     }>0
6505   }{
6506     \int_incr:N\l_tmpa_int
6507   }
6508   \int_compare:nNnF\l_tmpa_int=1{
6509     \int_decr:N\l_tmpa_int
6510   }
6511   \intarray_gset:Nnn\l__stex_sproof_counter_intarray\l_tmpa_int{
6512     \intarray_item:Nn\l__stex_sproof_counter_intarray\l_tmpa_int+1
6513   }
6514 }
6515
6516 \cs_new_protected:Npn\__stex_sproof_add_counter:{
6517   \int_set:Nn\l_tmpa_int{1}
6518   \bool_while_do:nn{
6519     \int_compare_p:nNn{
6520       \intarray_item:Nn\l__stex_sproof_counter_intarray\l_tmpa_int
6521     }>0
6522   }{
6523     \int_incr:N\l_tmpa_int
6524   }
6525   \intarray_gset:Nnn\l__stex_sproof_counter_intarray\l_tmpa_int{1}
6526 }
6527
```



```

6528 \cs_new_protected:Npn \__stex_sproof_remove_counter: {
6529   \int_set:Nn \l_tmpa_int {1}
6530   \bool_while_do:nn {
6531     \int_compare_p:nNn {
6532       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6533     } > 0
6534   }{
6535     \int_incr:N \l_tmpa_int
6536   }
6537   \int_decr:N \l_tmpa_int
6538   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 0 }
6539 }

```

\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

```

6540 \def\sproof@box{
6541   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
6542 }
6543 \def\sproofend{
6544   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
6545     \hfil\hfill\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
6546   }
6547 }

```

(End definition for \sproofend. This function is documented on page 51.)

spf@*kw

```

6548 \def\spf@proofsketch@kw{Proof~Sketch}
6549 \def\spf@proof@kw{Proof}
6550 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

6551 \AddToHook{begindocument}{
6552   \ltx@ifpackageloaded{babel}{
6553     \makeatletter
6554     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6555     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6556       \input{sproof-ngerman.ldf}
6557     }
6558     \clist_if_in:NnT \l_tmpa_clist {finnish}{
6559       \input{sproof-finnish.ldf}
6560     }
6561     \clist_if_in:NnT \l_tmpa_clist {french}{
6562       \input{sproof-french.ldf}
6563     }
6564     \clist_if_in:NnT \l_tmpa_clist {russian}{
6565       \input{sproof-russian.ldf}
6566     }
6567     \makeatother
6568   }{}
6569 }

```

spfsketch

```

6570 \newcommand\spfsketch[2] [] {
6571   \begin{group}
6572   \let \premise \stex_proof_premise:
6573   \stex_sproof_spf_args:n{#1}
6574   \stex_if_smsmode:TF {
6575     \str_if_empty:NF \spfid {
6576       \stex_ref_new_doc_target:n \spfid
6577     }
6578   }{
6579     \seq_clear:N \l_tmpa_seq
6580     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6581       \tl_if_empty:nF{ ##1 }{
6582         \stex_get_symbol:n { ##1 }
6583         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6584           \l_stex_get_symbol_uri_str
6585         }
6586       }
6587     }
6588     \exp_args:Nnx
6589     \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {},}{ }{
6590       \str_if_empty:NF \spftype {
6591         \stex_annotate_invisible:nnn{type}{\spftype}{ }{
6592         }
6593       }
6594       \clist_set:Nn \l_tmpa_clist \spftype
6595       \tl_set:Nn \l_tmpa_tl {
6596         \titleemph{
6597           \tl_if_empty:NTF \spftitle {
6598             \spf@proofsketch@kw
6599           }{
6600             \spftitle
6601           }
6602         }
6603       }
6604       \clist_map_inline:Nn \l_tmpa_clist {
6605         \exp_args:Nn \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6606           \tl_clear:N \l_tmpa_tl
6607         }
6608       }
6609       \str_if_empty:NF \spfid {
6610         \stex_ref_new_doc_target:n \spfid
6611       }
6612       \l_tmpa_tl #2 \sproofend
6613     }
6614   }
6615   \endgroup
6616 }
6617

```

(End definition for *spfsketch*. This function is documented on page 50.)

```

\__stex_sproof_maybe_comment:
\__stex_sproof_maybe_comment_end:
\__stex_sproof_start_comment:
6618 \bool_set_false:N \l__stex_sproof_in_spfblock_bool

```

```

6619
6620 \cs_new_protected:Nn \__stex_sproof_maybe_comment: {
6621   \bool_if:NF \l__stex_sproof_in_spfblock_bool {
6622     \par \setbox \l_tmpa_box \vbox \bgroup \everypar{\__stex_sproof_start_comment:}
6623   }
6624 }
6625 \cs_new_protected:Nn \__stex_sproof_maybe_comment_end: {
6626   \bool_if:NF \l__stex_sproof_in_spfblock_bool { \egroup }
6627 }
6628 \cs_new_protected:Nn \__stex_sproof_start_comment: {
6629   \csname @ @ par\endcsname\egroup\item[]\bgroup\stexcommentfont
6630 }
6631

```

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

\stexcommentfont

```

6632 \cs_new_protected:Npn \stexcommentfont {
6633   \small\itshape
6634 }

```

(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.

```

6635 \cs_new_protected:Nn \__stex_sproof_start_env:nnn {
6636   \seq_clear:N \l_tmpa_seq
6637   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6638     \tl_if_empty:NF{ ##1 }{
6639       \stex_get_symbol:n { ##1 }
6640       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6641         \l_stex_get_symbol_uri_str
6642       }
6643     }
6644   }
6645   \exp_args:Nnnx
6646   \begin{stex_annotate_env}{#1}{\seq_use:Nn \l_tmpa_seq {,}}
6647   \str_if_empty:NF \spftype {
6648     \stex_annotate_invisible:nnn{type}{\spftype}{}
6649   }
6650   #3 {\~\stex_annotate:nnn{spftitle}{}{#2}}
6651   \str_if_empty:NF \spfid {
6652     \stex_ref_new_doc_target:n \spfid
6653   }
6654   \begin{stex_annotate_env}{spfbody}{\bool_if:NTF \l__stex_sproof_spf_hide_bool {false}{true}
6655   \bool_if:NT \l__stex_sproof_spf_hide_bool{
6656     \stex_html_backend:F{\setbox\l_tmpa_box\vbox\bgroup}
6657   }
6658   \begin{list}{}{
6659     \setlength\topsep{0pt}
6660     \setlength\parsep{0pt}
6661     \setlength\rightmargin{0pt}

```

```

6662 } \_stex_sproof_maybe_comment:
6663 }
6664 \cs_new_protected:Nn \_stex_sproof_end_env:n {
6665   \stex_if_smsmode:F{
6666     \_stex_sproof_maybe_comment_end:
6667     \end{list}
6668     \bool_if:NT \l__stex_sproof_spf_hide_bool{
6669       \stex_html_backend:F{\egroup}
6670     }
6671     \clist_set:No \l_tmpa_clist \spftype
6672     #1
6673     \end{stex_annotate_env}
6674     \end{stex_annotate_env}
6675   }
6676 }
6677 }
6678 \NewDocumentEnvironment{sproof}{s O{} m}{
6679   \intarray_gzero:N \l__stex_sproof_counter_intarray
6680   \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
6681   \stex_reactivate_macro:N \yield
6682   \stex_reactivate_macro:N \eqstep
6683   \stex_reactivate_macro:N \assumption
6684   \stex_reactivate_macro:N \conclude
6685   \stex_reactivate_macro:N \spfstep
6686   \_stex_sproof_spf_args:n{#2}
6687   \stex_if_smsmode:TF {
6688     \str_if_empty:NF \spfid {
6689       \stex_ref_new_doc_target:n \spfid
6690     }
6691   }{
6692     \_stex_sproof_start_env:nnn{sproof}{#3}{
6693       \clist_set:No \l_tmpa_clist \spftype
6694       \tl_clear:N \l_tmpa_tl
6695       \clist_map_inline:Nn \l_tmpa_clist {
6696         \tl_if_exist:cT {\_stex_sproof_sproof_##1_start:}{
6697           \tl_set:Nn \l_tmpa_tl {\use:c{\_stex_sproof_sproof_##1_start:}}
6698         }
6699         \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6700           \tl_set:Nn \l_tmpa_tl {\use:n{}}
6701         }
6702       }
6703       \tl_if_empty:NTF \l_tmpa_tl {
6704         \_stex_sproof_sproof_start:
6705       }{
6706         \l_tmpa_tl
6707       }
6708     }
6709   }
6710   \stex_smsmode_do:
6711 }{\_stex_sproof_end_env:n{
6712   \tl_clear:N \l_tmpa_tl
6713   \clist_map_inline:Nn \l_tmpa_clist {
6714     \tl_if_exist:cT {\_stex_sproof_sproof_##1_end:}{
6715       \tl_set:Nn \l_tmpa_tl {\use:c{\_stex_sproof_sproof_##1_end:}}

```

```

6716     }
6717   }
6718   \tl_if_empty:NTF \l_tmpa_tl {
6719     \__stex_sproof_sproof_end:
6720   }{
6721     \l_tmpa_tl
6722   }
6723 }}
6724 \NewDocumentEnvironment{subproof}{s O{} m}{
6725   \__stex_sproof_spf_args:n{#2}
6726   \stex_if_smsmode:TF {
6727     \str_if_empty:NF \spfid {
6728       \stex_ref_new_doc_target:n \spfid
6729     }
6730   }{
6731     \__stex_sproof_start_env:nnn{subproof}{\item[\sproofnumber]\ignorespacesandpars #3}{}
6732   }
6733   \__stex_sproof_add_counter:
6734   \stex_smsmode_do:
6735 }{\__stex_sproof_remove_counter:\__stex_sproof_end_env:n{
6736   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6737     \__stex_sproof_inc_counter:
6738   }
6739   \aftergroup\__stex_sproof_maybe_comment:
6740 }
6741 \AddToHook{env/subproof/before}{\__stex_sproof_maybe_comment_end:}
6742
6743 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
6744   \par\noindent\titleemph{
6745     \tl_if_empty:NTF \spftype {
6746       \spf@proof@kw
6747     }{
6748       \spftype
6749     }
6750   }:
6751 }
6752 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
6753
6754 \newcommand\stexpatchproof[3] [] {
6755   \str_set:Nx \l_tmpa_str{ #1 }
6756   \str_if_empty:NTF \l_tmpa_str {
6757     \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
6758     \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
6759   }{
6760     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
6761     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
6762   }
6763 }

\pstep
\conclude
\assumption
\have
\eqstep

```

```

6764
6765 \keys_define:nn { stex / spfsteps } {
6766   id          .str_set_x:N = \spfstepid,
6767   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,

```

```

6768 type .str_set_x:N = \spftype,
6769 title .tl_set:N = \spftitle,
6770 method .tl_set:N = \l__stex_sproof_spf_method_tl,
6771 term .tl_set:N = \l__stex_sproof_spf_term_tl
6772 }
6773 \cs_new_protected:Nn \__stex_sproof_spfstep_args:n {
6774 \str_clear:N \spfstepid
6775 \clist_clear:N \l__stex_sproof_spf_for_clist
6776 \str_clear:N \spftype
6777 \tl_clear:N \l__stex_sproof_spf_method_tl
6778 \tl_clear:N \l__stex_sproof_spf_term_tl
6779 %\bool_set_false:N \l__stex_sproof_inc_counter_bool
6780 \keys_set:nn { stex / spfsteps }{ #1 }
6781 }
6782
6783 \cs_new_protected:Nn \__stex_sproof_make_step_macro:Nnnnn {
6784 \NewDocumentCommand #1 {s O{} +m} {
6785 \__stex_sproof_maybe_comment_end:
6786
6787 \__stex_sproof_spfstep_args:n{##2}
6788 \stex_annotate:nnn{spfstep}{#2}{
6789 \tl_if_empty:NF \l__stex_sproof_spf_term_tl {
6790 \stex_annotate_invisible:nnn{spfyield}{}{\l__stex_sproof_spf_term_tl$}
6791 }
6792 \bool_if:NTF \l__stex_sproof_in_spfblock_bool {
6793 #4
6794 }{
6795 \item[\IfBooleanTF ##1 {}{#3}]
6796 }
6797 \ignorespacesandpars ##3
6798 }
6799 \bool_if:NF \l__stex_sproof_in_spfblock_bool { \IfBooleanTF ##1 {}{ #5 } }
6800 \__stex_sproof_maybe_comment:
6801 }
6802 \stex_deactivate_macro:Nn #1 {sproof~environments}
6803 }
6804
6805 \__stex_sproof_make_step_macro:Nnnnn \assumption {assumption} \sproofnumber {} \__stex_sproof
6806 \__stex_sproof_make_step_macro:Nnnnn \conclude {conclusion} {\$ \Rightarrow$} {} {}
6807 \__stex_sproof_make_step_macro:Nnnnn \spfstep {} \sproofnumber {} \__stex_sproof_inc_counter
6808
6809 \NewDocumentCommand \eqstep {s m}{
6810 \__stex_sproof_maybe_comment_end:
6811 \bool_if:NTF \l__stex_sproof_in_spfblock_bool {
6812 $=$
6813 }{
6814 \item[$=$]
6815 }
6816 $\stex_annotate:nnn{spfstep}{eq}{ #2 }$
6817 \__stex_sproof_maybe_comment:
6818 }
6819 \stex_deactivate_macro:Nn \eqstep {sproof~environments}
6820
6821 \NewDocumentCommand \yield {+m}{

```

```

6822 \stex_annotate:nnn{spfyield}{\}{ #1 }
6823 }
6824 \stex_deactivate_macro:Nn \yield {sproof~environments}
6825
6826 \NewDocumentEnvironment{spfblock}{\}{\{
6827 \item[]
6828 \bool_set_true:N \l__stex_sproof_in_spfblock_bool
6829 }{
6830 \aftergroup\__stex_sproof_maybe_comment:
6831 }
6832 \AddToHook{env/spfblock/before}{\__stex_sproof_maybe_comment_end:}
6833

```

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

\spfidea

```

6834 \NewDocumentCommand\spfidea{0{\} +m}{
6835 \__stex_sproof_spf_args:n{#1}
6836 \titleemph{
6837 \tl_if_empty:NTF \spftype {Proof~Idea}{
6838 \spftype
6839 }:
6840 }~#2
6841 \sproofend
6842 }

```

(End definition for \spfidea. This function is documented on page 50.)

```

6843 \newcommand\spfjust[1]{
6844 #1
6845 }
6846 \</package>

```

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

Chapter 34

STEX -Others Implementation

```
6847 <*package>
6848
6849 %%%%%%%%%%% others.dtx %%%%%%%%%%%
6850
6851 <@@=stex_others>
        Warnings and error messages
6852 % None

\MSC Math subject classifier

6853 \NewDocumentCommand \MSC {m} {
6854 % TODO
6855 }

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

6856 \@ifpackageloaded{tikzinput}{
6857 \RequirePackage{stex-tikzinput}
6858 }{}
6859
6860 \bool_if:NT \c_stex_persist_mode_bool {
6861 \let__stex_notation_restore_notation_old:nnnnn
6862 \__stex_notation_restore_notation:nnnnn
6863 \def__stex_notation_restore_notation_new:nnnnn#1#2#3#4#5{
6864 \__stex_notation_restore_notation_old:nnnnn{#1}{#2}{#3}{#4}{#5}
6865 \ExplSyntaxOn
6866 }
6867 \def__stex_notation_restore_notation:nnnnn{
6868 \ExplSyntaxOff
6869 \catcode'\sim10
6870 \__stex_notation_restore_notation_new:nnnnn
6871 }
6872 \input{\jobname.sms}
6873 \let__stex_notation_restore_notation:nnnnn
6874 \__stex_notation_restore_notation_old:nnnnn
6875 \prop_if_exist:NT\c_stex_mathhub_main_manifest_prop{
```



```

6876 \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
6877 \l_tmpa_str
6878 \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str _manifest_prop }
6879 \c_stex_mathhub_main_manifest_prop
6880 \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
6881 }
6882 }
6883
6884 \stex_get_document_uri:
6885 </package>

```

Chapter 35

STEX -Metatheory Implementation

```
6886 <*package>
6887 <@@=stex_modules>
6888
6889 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
6890
6891 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX/meta}
6892 \begingroup
6893 \stex_module_setup:nn{
6894   ns=\c_stex_metatheory_ns_str,
6895   meta=NONE
6896 }{Metatheory}
6897 \stex_reactivate_macro:N \symdecl
6898 \stex_reactivate_macro:N \notation
6899 \stex_reactivate_macro:N \symdef
6900 \ExplSyntaxOff
6901 \csname stex_suppress_html:n\endcsname{
6902   % is-a (a:A, a \in A, a is an A, etc.)
6903   \symdecl{isa}[args=ai]
6904   \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
6905   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
6906   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
6907
6908   % bind (\forall, \Pi, \lambda etc.)
6909   \symdecl{bind}[args=Bi,assoc=pre]
6910   \notation{bind}[depfun,prec=nobrackets,op={(\cdot)\;\to\;\cdot}]{\comp( #1 \comp{}\;\to\;)}
6911   \notation{bind}[forall]{\comp\forall #1.\;#2}{##1 \comp, ##2}
6912   \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6913
6914   % implicit bind
6915   \symdecl{implicitbind}[args=Bi,assoc=pre]
6916   \notation{implicitbind}[braces,prec=nobrackets,op={(\cdot\_I\;\cdot)}]{\comp\{ #1 \comp{
6917   \notation{implicitbind}[depfun,prec=nobrackets]{\comp( #1 \comp{}\;\to\_I\; } #2}{##1 \comp,
6918   \notation{implicitbind}[Pi]{\comp\prod^I_{#1}#2}{##1\comp,##2}
6919
6920   % dummy variable
```

```

6921 \symdecl{dummyvar}
6922 \notation{dummyvar}[underscore]{\comp\_}
6923 \notation{dummyvar}[dot]{\comp\cdot}
6924 \notation{dummyvar}[dash]{\comp{\rm --}}
6925
6926 %fromto (function space, Hom-set, implication etc.)
6927 \symdecl{fromto}[args=ai]
6928 \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
6929 \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
6930
6931 % mapto (lambda etc.)
6932 \symdecl{mapto}[args=Bi]
6933 %\notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
6934 %\notation{mapto}[lambda]{\comp\lambda #1 \comp.\; #2}{#1 \comp, #2}
6935 %\notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.\; #2}{#1 \comp, #2}
6936
6937 % function/operator application
6938 \symdecl{apply}[args=ia]
6939 \notation{apply}[prec=0;0x\infpres,parens,op=\cdot(\cdot)]{#1 \comp( #2 \comp)}{##1 \comp,
6940 \notation{apply}[prec=0;0x\infpres,lambda]{#1 \; #2 }{##1 \; ; ##2}
6941
6942 % collection of propositions/booleans/truth values
6943 \symdecl{prop}[name=proposition]
6944 \notation{prop}[prop]{\comp{\rm prop}}
6945 \notation{prop}[BOOL]{\comp{\rm BOOL}}
6946
6947 \symdecl{judgmentholds}[args=1]
6948 \notation{judgmentholds}[vdash,op=\vdash]{\comp\vdash\; #1}
6949
6950 % sequences
6951 \symdecl{seqtype}[args=1]
6952 \notation{seqtype}[kleene]{#1^{\comp\ast}}
6953
6954 \symdecl{seqexpr}[args=a]
6955 \notation{seqexpr}[angle,prec=nobrackets]{\comp\langle #1\comp\rangle}{##1\comp,##2}
6956
6957 \symdef{seqmap}[args=abi,setlike]{\comp\{#3 \comp| #2\comp\in \dobrackets{#1} \comp\}}{##1\comp,##2}
6958 \symdef{seqprepend}[args=ia]{#1 \comp{::} #2}{##1 \comp, ##2}
6959 \symdef{seqappend}[args=ai]{#1 \comp{::} #2}{##1 \comp, ##2}
6960 \symdef{seqfoldleft}[args=iabbi]{ \comp{foldl}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\comp}
6961 \symdef{seqfoldright}[args=iabbi,op=foldr]{ \comp{foldr}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\comp}
6962 \symdef{seqhead}[args=a]{\comp{head}\dobrackets{#1}}{##1 \comp, ##2}
6963 \symdef{seqtail}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6964 \symdef{seqlast}[args=a]{\comp{last}\dobrackets{#1}}{##1 \comp, ##2}
6965 \symdef{seqinit}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6966
6967 \symdef{sequence-index}[args=2,li,prec=nobrackets]{\comp{#1}_{#2}}
6968 \notation{sequence-index}[ui,prec=nobrackets]{\comp{#1}^{\comp{#2}}}
6969
6970 \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{\,\ellipses\,}}{##1\comp,##2}
6971 \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{\,\ellipses\,}#2}{##1\comp,##2}
6972 \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{\,\ellipses\,}#2\comp{\,\ellipses\,}#3}{##1\comp,##2,##3}
6973
6974 % nat literals

```

```

6975 \symdef{natliteral}{\comp{\mathtt{Ord}}}
6976
6977 % letin (''let'', local definitions, variable substitution)
6978 \symdecl{letin}[args=bii]
6979 \notation{letin}{let}{\comp{\rm let}}\;#1\comp{=}\;#2\;\comp{\rm in}}\;#3}
6980 \notation{letin}{subst}{#3 \comp[ #1 \comp/ #2 \comp]}
6981 \notation{letin}{frac}{#3 \comp[ \frac{#2}{#1} \comp]}
6982
6983 % structures
6984 \symdecl*{module-type}[args=1]
6985 \notation{module-type}{\comp{\mathtt{MOD}}} #1}
6986 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
6987 \notation{mathstruct}[angle,prec=nobrackets]{\comp\angle #1 \comp\angle}{##1 \comp, ##2}
6988
6989 % objects
6990 \symdecl{object}
6991 \notation{object}{\comp{\mathtt{OBJECT}}}
6992
6993 }
6994
6995 % The following are abbreviations in the sTeX corpus that are left over from earlier
6996 % developments. They will eventually be phased out.
6997
6998 \ExplSyntaxOn
6999 \stex_add_to_current_module:n{
7000   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
7001   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
7002   \def\livar{\csname sequence-index\endcsname[li]}
7003   \def\uivar{\csname sequence-index\endcsname[ui]}
7004   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
7005   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
7006 }
7007 \__stex_modules_end_module:
7008 \endgroup
7009 \</package>

```

Chapter 36

Tikzinput Implementation

```
7010 <@@=tikzinput>
7011 <*package>
7012
7013 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
7014
7015 \ProvidesExplPackage{tikzinput}{2022/08/08}{3.2.0}{tikzinput package}
7016 \RequirePackage{l3keys2e}
7017
7018 \keys_define:nn { tikzinput } {
7019   image .bool_set:N = \c_tikzinput_image_bool,
7020   image .default:n = false ,
7021   unknown .code:n = {}
7022 }
7023
7024 \ProcessKeysOptions { tikzinput }
7025
7026 \bool_if:NTF \c_tikzinput_image_bool {
7027   \RequirePackage{graphicx}
7028
7029   \providecommand\usetikzlibrary[]{}
7030   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
7031 }{
7032   \RequirePackage{tikz}
7033   \RequirePackage{standalone}
7034
7035   \newcommand \tikzinput [2] [] {
7036     \setkeys{Gin}{#1}
7037     \ifx \Gin@ewidth \Gin@exclamation
7038       \ifx \Gin@eheight \Gin@exclamation
7039         \input { #2 }
7040       \else
7041         \resizebox{!}{ \Gin@eheight }{
7042           \input { #2 }
7043         }
7044       \fi
7045     \else
7046       \ifx \Gin@eheight \Gin@exclamation
7047         \resizebox{ \Gin@ewidth }{!}{
```

```

7048         \input { #2 }
7049     }
7050     \else
7051         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
7052             \input { #2 }
7053         }
7054     \fi
7055 \fi
7056 }
7057 }
7058
7059 \newcommand \ctikzinput [2] [] {
7060     \begin{center}
7061         \tikzinput [#1] {#2}
7062     \end{center}
7063 }
7064
7065 \@ifpackageloaded{stex}{
7066     \RequirePackage{stex-tikzinput}
7067 }{}
7068
7069 </package>
7070 <*stex>
7071 \ProvidesExplPackage{stex-tikzinput}{2022/08/08}{3.2.0}{stex-tikzinput}
7072 \RequirePackage{stex}
7073 \RequirePackage{tikzinput}
7074
7075 \newcommand\mhtikzinput[2] []{%
7076     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
7077     \stex_in_repository:nn\Gin@mhrepos{
7078         \tikzinput[#1]{\mhp@hpath{##1}{#2}}
7079     }
7080 }
7081 \newcommand\cmhtikzinput[2] [] {\begin{center}\mhtikzinput[#1]{#2}\end{center}}
7082
7083 \cs_new_protected:Nn \__tikzinput_usetikzlibrary:nn {
7084     \pgfkeys@spdef\pgf@temp{#1}
7085     \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
7086     \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgf@temp
7087     \expandafter\edef\csname tikz@library@#1@atcode\endcsname{\the\catcode'\@}
7088     \expandafter\edef\csname tikz@library@#1@barcode\endcsname{\the\catcode'\|}
7089     \expandafter\edef\csname tikz@library@#1@dollarcode\endcsname{\the\catcode'\$}
7090     \catcode'\@=11
7091     \catcode'\|=12
7092     \catcode'\$=3
7093     \pgfutil@InputIfFileExists{#2}{-}{-}
7094     \catcode'\@=\csname tikz@library@#1@atcode\endcsname
7095     \catcode'\|=\csname tikz@library@#1@barcode\endcsname
7096     \catcode'\$=\csname tikz@library@#1@dollarcode\endcsname
7097 }
7098
7099
7100 \newcommand\libusetikzlibrary[1]{

```

```

7101 \prop_if_exist:NF \l_stex_current_repository_prop {
7102   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
7103 }
7104 \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
7105   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
7106 }
7107 \seq_clear:N \l__tikzinput_libinput_files_seq
7108 \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
7109 \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
7110
7111 \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
7112   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / tikzlibrary}
7113   \IfFileExists{ \l_tmpa_str }{
7114     \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
7115   }{}
7116   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
7117   \seq_put_right:No \l_tmpa_seq \l_tmpa_str
7118 }
7119
7120 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / tikzlibrary #1 .code.tex}
7121 \IfFileExists{ \l_tmpa_str }{
7122   \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
7123 }{}
7124
7125 \seq_if_empty:NTF \l__tikzinput_libinput_files_seq {
7126   \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .code.tex}
7127 }{
7128   \int_compare:nNnTF {\seq_count:N \l__tikzinput_libinput_files_seq} = 1 {
7129     \seq_map_inline:Nn \l__tikzinput_libinput_files_seq {
7130       \__tikzinput_usetikzlibrary:nn{#1}{ ##1 }
7131     }
7132   }{
7133     \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .code.tex}
7134   }
7135 }
7136 }
7137 </stex>

```

Chapter 37

document-structure.sty Implementation

```
7138 <*package>
7139 <@@=document_structure>
7140 \ProvidesExplPackage{document-structure}{2022/08/08}{3.2.0}{Modular Document Structure}
7141 \RequirePackage{13keys2e}
```

37.1 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).

```
7142
7143 \keys_define:nn{ document-structure }{
7144   class      .str_set_x:N = \c_document_structure_class_str,
7145   topsect    .str_set_x:N = \c_document_structure_topsect_str,
7146   unknown    .code:n      = {
7147     \PassOptionsToClass{\CurrentOption}{stex}
7148     \PassOptionsToClass{\CurrentOption}{tikzinput}
7149   }
7150   % showignores .bool_set:N = \c_document_structure_showignores_bool,
7151 }
7152 \ProcessKeysOptions{ document-structure }
7153 \str_if_empty:NT \c_document_structure_class_str {
7154   \str_set:Nn \c_document_structure_class_str {article}
7155 }
7156 \str_if_empty:NT \c_document_structure_topsect_str {
7157   \str_set:Nn \c_document_structure_topsect_str {section}
7158 }
```

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

```
7159 \RequirePackage{xspace}
7160 \RequirePackage{comment}
7161 \RequirePackage{stex}
7162 \AddToHook{begindocument}{
```



```

7163 \ltx@ifpackageloaded{babel}{
7164   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7165   \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
7166     \makeatletter\input{document-structure-ngerman.ldf}\makeatother
7167   }
7168 }{}
7169 }

```

`\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.

```

7170 \int_new:N \l_document_structure_section_level_int
7171 \str_case:NnF \c_document_structure_topsect_str {
7172   {part}}{
7173     \int_set:Nn \l_document_structure_section_level_int {0}
7174   }
7175   {chapter}{
7176     \int_set:Nn \l_document_structure_section_level_int {1}
7177   }
7178 }{
7179   \str_case:NnF \c_document_structure_class_str {
7180     {book}{
7181       \int_set:Nn \l_document_structure_section_level_int {0}
7182     }
7183     {report}{
7184       \int_set:Nn \l_document_structure_section_level_int {0}
7185     }
7186   }{
7187     \int_set:Nn \l_document_structure_section_level_int {2}
7188   }
7189 }

```

37.2 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

```

7190 \def\current@section@level{document}%
7191 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
7192 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%

```

(End definition for `\currentsectionlevel`. This function is documented on page 58.)

`\skipfragment`

```

7193 \cs_new_protected:Npn \skipfragment {

```

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

```

7194 \ifcase\l_document_structure_section_level_int
7195 \or\stepcounter{part}
7196 \or\stepcounter{chapter}
7197 \or\stepcounter{section}
7198 \or\stepcounter{subsection}
7199 \or\stepcounter{subsubsection}
7200 \or\stepcounter{paragraph}
7201 \or\stepcounter{subparagraph}
7202 \fi
7203 }

```

(End definition for `\skipfragment`. This function is documented on page 57.)

`blindfragment (env.)`

```

7204 \newcommand\at@begin@blindsfragment[1]{
7205 \newenvironment{blindfragment}
7206 {
7207 \int_incr:N\l_document_structure_section_level_int
7208 \at@begin@blindsfragment\l_document_structure_section_level_int
7209 }{}

```

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

```

7210 \newcommand\sfragment@nonum[2]{
7211 \ifx\hyper@anchor\@undefined\else\phantomsection\fi
7212 \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
7213 }

```

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

`\sfragment@num` convenience macro: `\sfragment@num{<level>}{<title>}` makes numbered sectioning with title `<title>` at level `<level>`. We have to check the `short` key was given in the `sfragment` environment and – if it is use it. But how to do that depends on whether the `rdmeta` package has been loaded. In the end we call `\sref@label@id` to enable crossreferencing.

```

7214 \newcommand\sfragment@num[2]{
7215 \tl_if_empty:NTF \l__document_structure_sfragment_short_tl {
7216 \@nameuse{#1}{#2}
7217 }{
7218 \cs_if_exist:NTF\rdmeta@sectioning{
7219 \@nameuse{rdmeta@#1@old}[\l__document_structure_sfragment_short_tl]{#2}
7220 }{
7221 \@nameuse{#1}[\l__document_structure_sfragment_short_tl]{#2}
7222 }
7223 }
7224 %\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\sfragment@id
7225 }

```

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

`sfragment (env.)`

```

7226 \keys_define:nn { document-structure / sfragment }{
7227 id .str_set_x:N = \l__document_structure_sfragment_id_str,
7228 date .str_set_x:N = \l__document_structure_sfragment_date_str,

```

```

7229 creators      .clist_set:N = \l__document_structure_sfragment_creators_clist,
7230 contributors   .clist_set:N = \l__document_structure_sfragment_contributors_clist,
7231 srccite        .tl_set:N     = \l__document_structure_sfragment_srccite_tl,
7232 type          .tl_set:N     = \l__document_structure_sfragment_type_tl,
7233 short         .tl_set:N     = \l__document_structure_sfragment_short_tl,
7234 intro         .tl_set:N     = \l__document_structure_sfragment_intro_tl,
7235 imports       .tl_set:N     = \l__document_structure_sfragment_imports_tl,
7236 loadmodules   .bool_set:N   = \l__document_structure_sfragment_loadmodules_bool
7237 }
7238 \cs_new_protected:Nn \l__document_structure_sfragment_args:n {
7239   \str_clear:N \l__document_structure_sfragment_id_str
7240   \str_clear:N \l__document_structure_sfragment_date_str
7241   \clist_clear:N \l__document_structure_sfragment_creators_clist
7242   \clist_clear:N \l__document_structure_sfragment_contributors_clist
7243   \tl_clear:N \l__document_structure_sfragment_srccite_tl
7244   \tl_clear:N \l__document_structure_sfragment_type_tl
7245   \tl_clear:N \l__document_structure_sfragment_short_tl
7246   \tl_clear:N \l__document_structure_sfragment_imports_tl
7247   \tl_clear:N \l__document_structure_sfragment_intro_tl
7248   \bool_set_false:N \l__document_structure_sfragment_loadmodules_bool
7249   \keys_set:nn { document-structure / sfragment } { #1 }
7250 }

```

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.

```

7251 \newif\if@mainmatter\@mainmattertrue
7252 \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.

```

7253 \keys_define:nn { document-structure / sectioning }{
7254   name      .str_set_x:N = \l__document_structure_sect_name_str   ,
7255   ref       .str_set_x:N = \l__document_structure_sect_ref_str    ,
7256   clear     .bool_set:N  = \l__document_structure_sect_clear_bool ,
7257   clear     .default:n   = {true}                                ,
7258   num       .bool_set:N  = \l__document_structure_sect_num_bool   ,
7259   num       .default:n   = {true}
7260 }
7261 \cs_new_protected:Nn \l__document_structure_sect_args:n {
7262   \str_clear:N \l__document_structure_sect_name_str
7263   \str_clear:N \l__document_structure_sect_ref_str
7264   \bool_set_false:N \l__document_structure_sect_clear_bool
7265   \bool_set_false:N \l__document_structure_sect_num_bool
7266   \keys_set:nn { document-structure / sectioning } { #1 }
7267 }
7268 \newcommand\omdoc@sectioning[3][]{
7269   \l__document_structure_sect_args:n {#1 }
7270   \let\omdoc@sect@name\l__document_structure_sect_name_str
7271   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
7272   \if@mainmatter% numbering not overridden by frontmatter, etc.
7273     \bool_if:NTF \l__document_structure_sect_num_bool {
7274       \sfragment@num{#2}{#3}
7275     }{

```

```

7276     \sfragment@nonum{#2}{#3}
7277   }
7278   \def\current@section@level{\omdoc@sect@name}
7279   \else
7280     \sfragment@nonum{#2}{#3}
7281   \fi
7282 }% 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.

```

7283 \newcommand\sfragment@redefine@addtocontents[1]{%
7284 %\edef\__document_structureimport{#1}%
7285 %\@for\@I:=\__document_structureimport\do{%
7286 %\edef\@path{\csname module@\@I @path\endcsname}%
7287 %\@ifundefined{tf@toc}\relax%
7288 %   {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
7289 %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
7290 %\def\addcontentsline##1##2##3{%
7291 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
7292 %\else% hyperref.sty not loaded
7293 %\def\addcontentsline##1##2##3{%
7294 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
7295 %\fi
7296 }% 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.

```

7297 \newenvironment{sfragment}[2][ ]% keys, title
7298 {
7299   \__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.

```

7300   \stex_csl_to_imports:No \usemodule \l__document_structure_sfragment_imports_tl
7301
7302   \bool_if:NT \l__document_structure_sfragment_loadmodules_bool {
7303     \sfragment@redefine@addtocontents{
7304       %\@ifundefined{module@id}\used@modules%
7305       %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}
7306     }
7307   }

```

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

```

7308
7309   \stex_document_title:n { #2 }
7310
7311   \int_incr:N\l__document_structure_section_level_int
7312   \ifcase\l__document_structure_section_level_int
7313     \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
7314     \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
7315     \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
7316     \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}

```

```

7317 \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
7318 \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
7319 \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
7320 \fi
7321 \at@begin@sfragment[#1]\l__document_structure_section_level_int{#2}
7322 \str_if_empty:NF \l__document_structure_sfragment_id_str {
7323   \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
7324 }
7325 }% for customization
7326 {}

```

and finally, we localize the sections

```

7327 \newcommand\omdoc@part@kw{Part}
7328 \newcommand\omdoc@chapter@kw{Chapter}
7329 \newcommand\omdoc@section@kw{Section}
7330 \newcommand\omdoc@subsection@kw{Subsection}
7331 \newcommand\omdoc@subsubsection@kw{Subsubsection}
7332 \newcommand\omdoc@paragraph@kw{paragraph}
7333 \newcommand\omdoc@subparagraph@kw{subparagraph}

```

37.3 Front and Backmatter

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

`\printindex`

```

7334 \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).

```

7335 \cs_if_exist:NTF\frontmatter{
7336   \let\__document_structure_orig_frontmatter\frontmatter
7337   \let\frontmatter\relax
7338 }{
7339   \tl_set:Nn\__document_structure_orig_frontmatter{
7340     \clearpage
7341     \@mainmatterfalse
7342     \pagenumbering{roman}
7343   }
7344 }
7345 \cs_if_exist:NTF\backmatter{
7346   \let\__document_structure_orig_backmatter\backmatter
7347   \let\backmatter\relax
7348 }{
7349   \tl_set:Nn\__document_structure_orig_backmatter{
7350     \clearpage
7351     \@mainmatterfalse
7352     \pagenumbering{roman}
7353   }

```

7354 }

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.

```
7355 \newenvironment{frontmatter}{
7356   \__document_structure_orig_frontmatter
7357 }{
7358   \cs_if_exist:NTF\mainmatter{
7359     \mainmatter
7360   }{
7361     \clearpage
7362     \@mainmattertrue
7363     \pagenumbering{arabic}
7364   }
7365 }
```

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

```
7366 \newenvironment{backmatter}{
7367   \__document_structure_orig_backmatter
7368 }{
7369   \cs_if_exist:NTF\mainmatter{
7370     \mainmatter
7371   }{
7372     \clearpage
7373     \@mainmattertrue
7374     \pagenumbering{arabic}
7375   }
7376 }
```

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

```
7377 \@mainmattertrue\pagenumbering{arabic}
```

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

```
7378 \def \c__document_structure_document_str{document}
7379 \newcommand\afterprematurestop{}
7380 \def\prematurestop@endsfragment{
7381   \unless\ifx\@currenvir\c__document_structure_document_str
7382     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
7383     \expandafter\prematurestop@endsfragment
7384   \fi
7385 }
7386 \providecommand\prematurestop{
7387   \message{Stopping~sTeX~processing~prematurely}
7388   \prematurestop@endsfragment
7389   \afterprematurestop
7390   \end{document}
7391 }
```

(End definition for `\prematurestop`. This function is documented on page 58.)

37.4 Global Variables

\setSGvar set a global variable

```
7392 \RequirePackage{etoolbox}
7393 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}
```

(End definition for \setSGvar. This function is documented on page 58.)

\useSGvar use a global variable

```
7394 \newrobustcmd\useSGvar[1]{%
7395   \@ifundefined{sTeX@Gvar@#1}
7396   {\PackageError{document-structure}
7397     {The sTeX Global variable #1 is undefined}
7398     {set it with \protect\setSGvar}}
7399   \@nameuse{sTeX@Gvar@#1}}
```

(End definition for \useSGvar. This function is documented on page 58.)

\ifSGvar execute something conditionally based on the state of the global variable.

```
7400 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
7401   \@ifundefined{sTeX@Gvar@#1}
7402   {\PackageError{document-structure}
7403     {The sTeX Global variable #1 is undefined}
7404     {set it with \protect\setSGvar}}
7405   {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}
```

(End definition for \ifSGvar. This function is documented on page 58.)

Chapter 38

NotesSlides – Implementation

38.1 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.

```
7406 \*cls)
7407 \@@=notesslides)
7408 \ProvidesExplClass{notesslides}{2022/08/08}{3.2.0}{notesslides Class}
7409 \RequirePackage{13keys2e}
7410
7411 \keys_define:nn{notesslides / cls}{
7412   class .str_set_x:N = \c__notesslides_class_str,
7413   notes .bool_set:N = \c__notesslides_notes_bool ,
7414   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
7415   docopt .str_set_x:N = \c__notesslides_docopt_str,
7416   unknown .code:n = {
7417     \PassOptionsToPackage{\CurrentOption}{document-structure}
7418     \PassOptionsToClass{\CurrentOption}{beamer}
7419     \PassOptionsToPackage{\CurrentOption}{notesslides}
7420     \PassOptionsToPackage{\CurrentOption}{stex}
7421   }
7422 }
7423 \ProcessKeysOptions{ notesslides / cls }
7424
7425 \str_if_empty:NF \c__notesslides_class_str {
7426   \PassOptionsToPackage{class=\c__notesslides_class_str}{document-structure}
7427 }
7428
7429 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{book}{
7430   \PassOptionsToPackage{defaultttopsect=part}{notesslides}
7431 }
7432 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{report}{
7433   \PassOptionsToPackage{defaultttopsect=part}{notesslides}
7434 }
7435
7436 \RequirePackage{stex}
```



```

7437 \stex_html_backend:T {
7438   \bool_set_true:N\c__notesslides_notes_bool
7439 }
7440
7441 \bool_if:NTF \c__notesslides_notes_bool {
7442   \PassOptionsToPackage{notes=true}{notesslides}
7443   \message{notesslides.cls:~Formatting~course~materials~in~notes~mode}
7444 }{
7445   \PassOptionsToPackage{notes=false}{notesslides}
7446   \message{notesslides.cls:~Formatting~course~materials~in~slides~mode}
7447 }
7448 \</cls>

```

now we do the same for the notesslides package.

```

7449 <*package>
7450 \ProvidesExplPackage{notesslides}{2022/08/08}{3.2.0}{notesslides Package}
7451 \RequirePackage{l3keys2e}
7452
7453 \keys_define:nn{notesslides / pkg}{
7454   topsect          .str_set_x:N = \c__notesslides_topsect_str,
7455   defaulttopsect   .str_set_x:N = \c__notesslides_defaulttopsec_str,
7456   notes            .bool_set:N = \c__notesslides_notes_bool ,
7457   slides           .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
7458   sectocframes     .bool_set:N = \c__notesslides_sectocframes_bool ,
7459   frameimages      .bool_set:N = \c__notesslides_frameimages_bool ,
7460   fiboxed          .bool_set:N = \c__notesslides_fiboxed_bool ,
7461   nopproblems      .bool_set:N = \c__notesslides_nopproblems_bool,
7462   unknown          .code:n      = {
7463     \PassOptionsToClass{\CurrentOption}{stex}
7464     \PassOptionsToClass{\CurrentOption}{tikzinput}
7465   }
7466 }
7467 \ProcessKeysOptions{ notesslides / pkg }
7468
7469 \RequirePackage{stex}
7470 \stex_html_backend:T {
7471   \bool_set_true:N\c__notesslides_notes_bool
7472 }
7473
7474 \newif\ifnotes
7475 \bool_if:NTF \c__notesslides_notes_bool {
7476   \notesttrue
7477 }{
7478   \notestfalse
7479 }
7480

```

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

```

7481 \str_if_empty:NTF \c__notesslides_topsect_str {
7482   \str_set_eq:NN \__notesslides_topsect \c__notesslides_defaulttopsec_str
7483 }{
7484   \str_set_eq:NN \__notesslides_topsect \c__notesslides_topsect_str
7485 }
7486 \PassOptionsToPackage{topsect=\__notesslides_topsect}{document-structure}

```

```
7487 </package>
```

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

```
7488 <*cls>
7489 \bool_if:NTF \c__notesslides_notes_bool {
7490   \str_if_empty:NT \c__notesslides_class_str {
7491     \str_set:Nn \c__notesslides_class_str {article}
7492   }
7493   \exp_after:wN\LoadClass\exp_after:wN[\c__notesslides_dcopt_str]
7494     {\c__notesslides_class_str}
7495 }{
7496   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
7497   \newcounter{Item}
7498   \newcounter{paragraph}
7499   \newcounter{subparagraph}
7500   \newcounter{Hfootnote}
7501 }
7502 \RequirePackage{document-structure}
```

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

```
7503 \RequirePackage{notesslides}
7504 </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.

```
7505 <*package>
7506 \bool_if:NT \c__notesslides_notes_bool {
7507   \RequirePackage{a4wide}
7508   \RequirePackage{marginnote}
7509   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
7510   \RequirePackage{mdframed}
7511   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
7512   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
7513 }
7514 \RequirePackage{stex-tikzinput}
7515 \RequirePackage{comment}
7516 \RequirePackage{url}
7517 \RequirePackage{graphicx}
7518 \RequirePackage{pgf}
7519 \RequirePackage{bookmark}
```

38.2 Notes and Slides

For the lecture notes cases, we also provide the `\usetheme` macro that would otherwise come from the `beamer` class.

```
7520 \bool_if:NT \c__notesslides_notes_bool {
7521   \renewcommand\usetheme[2][\usepackage{#1}{beamertheme#2}]
7522 }
```

```

7523 \NewDocumentCommand \libusetheme {0{} m} {
7524   \libusepackage[#1]{beamertheme#2}
7525 }
7526

```

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

```

7527 \newcounter{slide}
7528 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
7529 \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.

```

7530 \bool_if:NTF \c__notesslides_notes_bool {
7531   \renewenvironment{note}{\ignorespaces}{}
7532 }{
7533   \excludecomment{note}
7534 }

```

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.

```

7535 \bool_if:NT \c__notesslides_notes_bool {
7536   \newlength{\slideframewidth}
7537   \setlength{\slideframewidth}{1.5pt}

```

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

```

7538 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
7539   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
7540     \bool_set_true:N #1
7541   }{
7542     \bool_set_false:N #1
7543   }
7544 }
7545 \keys_define:nn{notesslides / frame}{
7546   label .str_set_x:N = \l__notesslides_frame_label_str,
7547   allowframebreaks .code:n = {
7548     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
7549   },
7550   allowdisplaybreaks .code:n = {
7551     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
7552   },
7553   fragile .code:n = {
7554     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
7555   },
7556   shrink .code:n = {
7557     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
7558   },
7559   squeeze .code:n = {
7560     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
7561   },
7562   t .code:n = {

```

```

7563     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
7564   },
7565   unknown    .code:n      = {}
7566 }
7567 \cs_new_protected:Nn \__notesslides_frame_args:n {
7568   \str_clear:N \l__notesslides_frame_label_str
7569   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
7570   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
7571   \bool_set_true:N \l__notesslides_frame_fragile_bool
7572   \bool_set_true:N \l__notesslides_frame_shrink_bool
7573   \bool_set_true:N \l__notesslides_frame_squeeze_bool
7574   \bool_set_true:N \l__notesslides_frame_t_bool
7575   \keys_set:nn { notesslides / frame }{ #1 }
7576 }

```

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

```

7577 \renewenvironment{frame}[1][]{
7578   \__notesslides_frame_args:n{#1}
7579   \sffamily
7580   \stepcounter{slide}
7581   \def\@currentlabel{\theslide}
7582   \str_if_empty:NF \l__notesslides_frame_label_str {
7583     \label{\l__notesslides_frame_label_str}
7584   }

```

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

```

7585   \def\itemize@level{outer}
7586   \def\itemize@outer{outer}
7587   \def\itemize@inner{inner}
7588   \renewcommand\newpage{\addtocounter{framenum}{1}}
7589   %\newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
7590   \renewenvironment{itemize}{
7591     \ifx\itemize@level\itemize@outer
7592       \def\itemize@label{$\rhd$}
7593     \fi
7594     \ifx\itemize@level\itemize@inner
7595       \def\itemize@label{$\scriptstyle\rhd$}
7596     \fi
7597     \begin{list}
7598       {\itemize@label}
7599       {\setlength{\labelsep}{.3em}
7600        \setlength{\labelwidth}{.5em}
7601        \setlength{\leftmargin}{1.5em}
7602       }
7603     \edef\itemize@level{\itemize@inner}
7604   }{
7605     \end{list}
7606   }

```

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

```

7607   \stex_html_backend:TF {
7608     \begin{stex_annotate_env}{frame}{}\vbox\bgroup
7609     \mdf@patchamsthm
7610   }{
7611     \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwid

```

```

7612     }
7613   }{
7614     \stex_html_backend:TF {
7615       \miko@slidelabel\egroup\end{stex_annotate_env}
7616     }\medskip\miko@slidelabel\end{mdframed}}
7617   }

```

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

`\frametitle`

```

7618   \renewcommand{\frametitle}[1]{
7619     \stex_document_title:n { #1 }
7620     {\Large\bf\sf\color{blue}{#1}}\medskip
7621   }
7622 }

```

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

EdN:10

`\pause` 10

```

7623 \bool_if:NT \c__notesslides_notes_bool {
7624   \newcommand\pause{
7625   }

```

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

`nparagraph (env.)`

```

7626 \bool_if:NTF \c__notesslides_notes_bool {
7627   \newenvironment{nparagraph}[1] [] {\begin{sparagraph}[#1]}\end{sparagraph}}
7628 }{
7629   \excludecomment{nparagraph}
7630 }

```

`nfragment (env.)`

```

7631 \bool_if:NTF \c__notesslides_notes_bool {
7632   \newenvironment{nfragment}[2] [] {\begin{sfragment}[#1][#2]}\end{sfragment}}
7633 }{
7634   \excludecomment{nfragment}
7635 }

```

`ndefinition (env.)`

```

7636 \bool_if:NTF \c__notesslides_notes_bool {
7637   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}\end{sdefinition}}
7638 }{
7639   \excludecomment{ndefinition}
7640 }

```

`nassertion (env.)`

```

7641 \bool_if:NTF \c__notesslides_notes_bool {
7642   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}\end{sassertion}}
7643 }{
7644   \excludecomment{nassertion}
7645 }

```

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

`nsproof (env.)`

```
7646 \bool_if:NTF \c__notesslides_notes_bool {
7647   \newenvironment{nproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
7648 }{
7649   \excludecomment{nproof}
7650 }
```

`nexample (env.)`

```
7651 \bool_if:NTF \c__notesslides_notes_bool {
7652   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}{\end{sexample}}
7653 }{
7654   \excludecomment{nexample}
7655 }
```

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

```
7656 \def\inputref@preskip{\smallskip}
7657 \def\inputref@postskip{\medskip}
```

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

`\inputref*`

```
7658 \let\orig@inputref\inputref
7659 \def\inputref{\@ifstar\ninputref\orig@inputref}
7660 \newcommand\ninputref[2] []{
7661   \bool_if:NT \c__notesslides_notes_bool {
7662     \orig@inputref[#1]{#2}
7663   }
7664 }
```

(End definition for \inputref. This function is documented on page 60.)*

38.3 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 \TeX logo. Customization can be done by `\setslidelogo{<logo name>}`.

```
7665 \newlength{\slidelogoheight}
7666
7667 \RequirePackage{graphicx}
7668
7669 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7670 \providecommand\mhgraphics[2] []{
7671   \def\Gin@mhrepos{}\setkeys{Gin}{#1}
7672   \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}
7673 }
7674
7675 \bool_if:NTF \c__notesslides_notes_bool {
7676   \setlength{\slidelogoheight}{.4cm}
7677 }{
7678   \setlength{\slidelogoheight}{.25cm}
7679 }
```

```

7680 \ifcsname slidelogo\endcsname\else
7681 \newsavebox{\slidelogo}
7682 \sbox{\slidelogo}{\sTeX}
7683 \fi
7684 \newrobustcmd{\setslidelogo}[2][]{
7685   \tl_if_empty:nTF{#1}{
7686     \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#2}}
7687   }{
7688     \sbox{\slidelogo}{\mhgraphics[height=\slidelogoheight,mhrepos=#1]{#2}}
7689   }
7690 }

```

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

\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.

```

7691 \bool_if:NT \c__notesslides_notes_bool {
7692   \def\author{\@dblarg\@ns@author}
7693   \long\def\@ns@author[#1]#2{%
7694     \def\c__notesslides_shortauthor{#1}%
7695     \def\@author{#2}
7696   }
7697 }

```

(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.

```

7698 \newrobustcmd{\setsource}[1]{\def\source{#1}}

```

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

\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.

```

7699 \def\copyrightnotice{%
7700   \footnotesize\copyright : \hspace{.3ex}%
7701   \ifcsname source\endcsname\source\else%
7702   \ifcsname c__notesslides_shortauthor\endcsname\c__notesslides_shortauthor\else%
7703   \PackageWarning{notesslides}{Author/Source-undefined-in-copyright-notice}%
7704   ?source/author?\fi%
7705   \fi}
7706 \newsavebox{\cclogo}
7707 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{stex-cc_somerights}}
7708 \newif\ifcchref\cchreffalse
7709 \AtBeginDocument{
7710   \@ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
7711 }
7712 \def\licensing{
7713   \ifcchref
7714     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
7715   \else
7716     {\usebox{\cclogo}}

```

```

7717 \fi
7718 }
7719 \newrobustcmd{\setlicensing}[2][]{
7720   \def\@url{#1}
7721   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
7722   \ifx\@url\@empty
7723     \def\licensing{\usebox{\cclogo}}
7724   \else
7725     \def\licensing{
7726       \ifcchref
7727         \href{#1}{\usebox{\cclogo}}
7728       \else
7729         {\usebox{\cclogo}}
7730       \fi
7731     }
7732   \fi
7733 }

```

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

EdN:11

```

\slidelabel Now, we set up the slide label for the article mode.11
7734 \newrobustcmd\miko@slidelabel{
7735   \vbox to \slidelogoheight{
7736     \vss\hbox to \slidewidth
7737       {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
7738   }
7739 }

```

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

38.4 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.

```

7740 \def\Gin@mhrepos{}
7741 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7742 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
7743 \newrobustcmd\frameimage[2][]{
7744   \stepcounter{slide}
7745   \bool_if:NT \c__notesslides_frameimages_bool {
7746     \def\Gin@ewidth{}\setkeys{Gin}{#1}
7747     \bool_if:NF \c__notesslides_notes_bool { \vfill }
7748     \begin{center}
7749       \bool_if:NTF \c__notesslides_fiboxed_bool {
7750         \fbox{
7751           \ifx\Gin@ewidth\@empty
7752             \ifx\Gin@mhrepos\@empty
7753               \mhgraphics[width=\slidewidth,#1]{#2}
7754             \else
7755               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7756             \fi
7757           \else% Gin@ewidth empty

```

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


```

7758         \ifx\Gin@mhrepos\@empty
7759             \mhgraphics[#1]{#2}
7760         \else
7761             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7762         \fi
7763     \fi% Gin@ewidth empty
7764 }
7765 }{
7766     \ifx\Gin@ewidth\@empty
7767         \ifx\Gin@mhrepos\@empty
7768             \mhgraphics[width=\slidewidth,#1]{#2}
7769         \else
7770             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7771         \fi
7772         \ifx\Gin@mhrepos\@empty
7773             \mhgraphics[#1]{#2}
7774         \else
7775             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7776         \fi
7777     \fi% Gin@ewidth empty
7778 }
7779 \end{center}
7780 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
7781 \bool_if:NF \c__notesslides_notes_bool { \vfill }
7782 }
7783 } % ifmks@sty@frameimages

```

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

38.5 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`.

```

7784 \stex_html_backend:F {
7785     \bool_if:NT \c__notesslides_sectocframes_bool {
7786         \str_if_eq:VnTF \__notesslidestopsect{part}{
7787             \newcounter{chapter}\counterwithin*{section}{chapter}
7788         }{
7789             \str_if_eq:VnT\__notesslidestopsect{chapter}{
7790                 \newcounter{chapter}\counterwithin*{section}{chapter}
7791             }
7792         }
7793     }
7794 }

```

`\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`

```

7795 \def\part@prefix{}
7796 \@ifpackageloaded{document-structure}{
7797     \str_case:VnF \__notesslidestopsect {

```

```

7798 {part}{
7799   \int_set:Nn \l_document_structure_section_level_int {0}
7800   \def\thesection{\arabic{chapter}.\arabic{section}}
7801   \def\part@prefix{\arabic{chapter}.}
7802 }
7803 {chapter}{
7804   \int_set:Nn \l_document_structure_section_level_int {1}
7805   \def\thesection{\arabic{chapter}.\arabic{section}}
7806   \def\part@prefix{\arabic{chapter}.}
7807 }
7808 }{
7809   \int_set:Nn \l_document_structure_section_level_int {2}
7810   \def\part@prefix{}
7811 }
7812 }
7813
7814 \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.)`

```

7815 \renewenvironment{sfragment}[2][]{
7816   \__document_structure_sfragment_args:n { #1 }
7817   \int_incr:N \l_document_structure_section_level_int
7818   \bool_if:NT \c__notesslides_sectocframes_bool {
7819     \stepcounter{slide}
7820     \begin{frame}[noframenumbering]
7821     \vfill\Large\centering
7822     \red{
7823       \ifcase\l_document_structure_section_level_int\or
7824         \stepcounter{part}
7825         \def\__notesslideslabel{\omdoc@part@kw}~\Roman{part}}
7826         \addcontentsline{toc}{part}{\protect\numberline{\thepart}#2}
7827         \pdfbookmark[0]{\thepart\ #2}{part.\thepart}
7828         \def\currentsectionlevel{\omdoc@part@kw}
7829       \or
7830         \stepcounter{chapter}
7831         \def\__notesslideslabel{\omdoc@chapter@kw}~\arabic{chapter}}
7832         \addcontentsline{toc}{chapter}{\protect\numberline{\thechapter}#2}
7833         \pdfbookmark[1]{\thechapter\ #2}{chapter.\cs_if_exist:cT{\thepart}\thepart.\thechapter}
7834         \def\currentsectionlevel{\omdoc@chapter@kw}
7835       \or
7836         \stepcounter{section}
7837         \def\__notesslideslabel{\part@prefix\arabic{section}}
7838         \addcontentsline{toc}{section}{\protect\numberline{\thesection}#2}
7839         \pdfbookmark[2]{\cs_if_exist:cT{\thechapter}{\thechapter}.\thesection\ #2}
7840         {section.\cs_if_exist:cT{\thepart}{\thepart}.\cs_if_exist:cT{\thechapter}{\thechapter}.\thepart}
7841         \def\currentsectionlevel{\omdoc@section@kw}
7842       \or
7843         \stepcounter{subsection}
7844         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
7845         \addcontentsline{toc}{subsection}{\protect\numberline{\thesubsection}#2}

```

```

7846         \pdfbookmark[3]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection
7847         {subsection.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection}
7848         \def\currentsectionlevel{\omdoc@subsection@kw}
7849     \or
7850         \stepcounter{subsubsection}
7851         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
7852         \addcontentsline{toc}{subsubsection}{\protect\numberline{\thesubsubsection}#2}
7853         \pdfbookmark[4]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection.\thesubsubsection}
7854         {subsubsection.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsubsection}
7855         \def\currentsectionlevel{\omdoc@subsubsection@kw}
7856     \or
7857         \stepcounter{paragraph}
7858         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{paragraph}}
7859         \addcontentsline{toc}{paragraph}{\protect\numberline{\theparagraph}#2}
7860         \pdfbookmark[5]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection.\theparagraph}
7861         {paragraph.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\theparagraph}
7862         \def\currentsectionlevel{\omdoc@paragraph@kw}
7863     \else
7864         \def\__notesslideslabel{}
7865         \def\currentsectionlevel{\omdoc@paragraph@kw}
7866     \fi% end ifcase
7867     \__notesslideslabel\quad #2%
7868 }%
7869 \vfill%
7870 \end{frame}%
7871 }
7872 \str_if_empty:NF \l__document_structure_sfragment_id_str {
7873     \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
7874 }
7875 }{}
7876 }

```

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

```

7877 \def\inserttheorembodyfont{\normalfont}
7878 %\bool_if:NF \c__notesslides_notes_bool {
7879 % \defbeamertemplate{theorem begin}{miko}
7880 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
7881 % \ifx\inserttheoremaddition@empty\else\ (\inserttheoremaddition)\fi%
7882 % \inserttheorempunctuation\inserttheorembodyfont\xspace}
7883 % \defbeamertemplate{theorem end}{miko}{\fi}

```

and we set it as the default one.

```

7884 % \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.

```

7885 % \expandafter\def\csname Parent2\endcsname{}
7886 %}
7887
7888 \AddToHook{begindocument}{% this does not work for some reason
7889     \setbeamertemplate{theorems}[ams style]
7890 }
7891 \bool_if:NT \c__notesslides_notes_bool {
7892     \renewenvironment{columns}[1][\fi]{}{}

```

```

7893     \par\noindent%
7894     \begin{minipage}%
7895     \slidewidth\centering\leavevmode%
7896   }{%
7897     \end{minipage}\par\noindent%
7898   }%
7899   \newsavebox\columnbox%
7900   \renewenvironment<>{column}[2][]{%
7901     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
7902   }{%
7903     \end{minipage}\end{lrbox}\usebox\columnbox%
7904   }%
7905 }

7906 \bool_if:NTF \c__notesslides_noproblems_bool {
7907   \newenvironment{problems}{}{}
7908 }{
7909   \excludacomment{problems}
7910 }

```

38.6 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.

```

7911 \gdef\printexcursions{}
7912 \newcommand\excursionref[2]{% label, text
7913   \bool_if:NT \c__notesslides_notes_bool {
7914     \begin{sparagraph}[title=Excursion]
7915       #2 \sref[fallback=the appendix]{#1}.
7916     \end{sparagraph}
7917   }
7918 }
7919 \newcommand\activate@excursion[2][{}{
7920   \gappto\printexcursions{\inputref{#1}{#2}}
7921 }
7922 \newcommand\excursion[4][{}{ repos, label, path, text
7923   \bool_if:NT \c__notesslides_notes_bool {
7924     \activate@excursion{#1}{#3}\excursionref{#2}{#4}
7925   }
7926 }

```

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

\excursiongroup

```

7927 \keys_define:nn{notesslides / excursiongroup }{
7928   id          .str_set_x:N = \l__notesslides_excursion_id_str,
7929   intro       .tl_set:N   = \l__notesslides_excursion_intro_tl,
7930   mhrepos     .str_set_x:N = \l__notesslides_excursion_mhrepos_str
7931 }
7932 \cs_new_protected:Nn \__notesslides_excursion_args:n {
7933   \tl_clear:N \l__notesslides_excursion_intro_tl
7934   \str_clear:N \l__notesslides_excursion_id_str

```

```

7935 \str_clear:N \l__notesslides_excursion_mhrepos_str
7936 \keys_set:nn {notesslides / excursionsgroup }{ #1 }
7937 }
7938 \newcommand\excursionsgroup[1][]{
7939   \__notesslides_excursion_args:n{ #1 }
7940   \ifdefempty\printexcursions{}% only if there are excursions
7941   {\begin{note}
7942     \begin{sfragment}[#1]{Excursions}%
7943     \ifdefempty\l__notesslides_excursion_intro_tl}{
7944       \inputref[\l__notesslides_excursion_mhrepos_str]{
7945         \l__notesslides_excursion_intro_tl
7946       }
7947     }
7948     \printexcursions%
7949     \end{sfragment}
7950   \end{note}}
7951 }
7952 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
7953 \end{package}

```

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

Chapter 39

The Implementation

39.1 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.

```
7954 <*package>
7955 <@@=problems>
7956 \ProvidesExplPackage{problem}{2022/08/08}{3.2.0}{Semantic Markup for Problems}
7957 \RequirePackage{13keys2e}
7958 \RequirePackage{amssymb}% for \Box
7959
7960 \keys_define:nn { problem / pkg }{
7961   notes      .default:n    = { true },
7962   notes      .bool_set:N   = \c__problems_notes_bool,
7963   gnotes     .default:n    = { true },
7964   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
7965   hints      .default:n    = { true },
7966   hints      .bool_set:N   = \c__problems_hints_bool,
7967   solutions  .default:n    = { true },
7968   solutions  .bool_set:N   = \c__problems_solutions_bool,
7969   pts        .default:n    = { true },
7970   pts        .bool_set:N   = \c__problems_pts_bool,
7971   min        .default:n    = { true },
7972   min        .bool_set:N   = \c__problems_min_bool,
7973   boxed      .default:n    = { true },
7974   boxed      .bool_set:N   = \c__problems_boxed_bool,
7975   test       .default:n    = { true },
7976   test       .bool_set:N   = \c__problems_test_bool,
7977   unknown    .code:n       = {
7978     \PassOptionsToPackage{\CurrentOption}{stex}
7979   }
7980 }
7981 \newif\ifsolutions
7982
7983 \ProcessKeysOptions{ problem / pkg }
7984 \bool_if:NTF \c__problems_solutions_bool {
7985   \solutionstrue
```

```

7986 }{
7987   \solutionsfalse
7988 }
7989 \RequirePackage{stex}

```

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

```

7990 \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.

```

7991 \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.

```

7992 \def\prob@problem@kw{Problem}
7993 \def\prob@solution@kw{Solution}
7994 \def\prob@hint@kw{Hint}
7995 \def\prob@note@kw{Note}
7996 \def\prob@grade@kw{Grading}
7997 \def\prob@pt@kw{pt}
7998 \def\prob@min@kw{min}
7999 \def\prob@correct@kw{Correct}
8000 \def\prob@wrong@kw{Wrong}

```

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

For the other languages, we set up triggers

```

8001 \AddToHook{begindocument}{
8002   \ltx@ifpackageloaded{babel}{
8003     \makeatletter
8004     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
8005     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
8006       \input{problem-ngerman.ldf}
8007     }
8008     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
8009       \input{problem-finnish.ldf}
8010     }
8011     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
8012       \input{problem-french.ldf}
8013     }
8014     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
8015       \input{problem-russian.ldf}
8016     }
8017     \makeatother
8018   }{ }
8019 }

```

39.2 Problems and Solutions

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

```

8020 \keys_define:nn{ problem / problem }{
8021   id .str_set_x:N = \l__problems_prob_id_str,

```

```

8022 pts      .tl_set:N      = \l__problems_prob_pts_tl,
8023 min      .tl_set:N      = \l__problems_prob_min_tl,
8024 title     .tl_set:N      = \l__problems_prob_title_tl,
8025 type      .tl_set:N      = \l__problems_prob_type_tl,
8026 imports   .tl_set:N      = \l__problems_prob_imports_tl,
8027 name       .str_set_x:N   = \l__problems_prob_name_str,
8028 refnum     .int_set:N     = \l__problems_prob_refnum_int
8029 }
8030 \cs_new_protected:Nn \__problems_prob_args:n {
8031   \str_clear:N \l__problems_prob_id_str
8032   \str_clear:N \l__problems_prob_name_str
8033   \tl_clear:N \l__problems_prob_pts_tl
8034   \tl_clear:N \l__problems_prob_min_tl
8035   \tl_clear:N \l__problems_prob_title_tl
8036   \tl_clear:N \l__problems_prob_type_tl
8037   \tl_clear:N \l__problems_prob_imports_tl
8038   \int_zero_new:N \l__problems_prob_refnum_int
8039   \keys_set:nn { problem / problem }{ #1 }
8040   \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
8041     \let\l__problems_prob_refnum_int\undefined
8042   }
8043 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

8044 \newcounter{problem}[section]
8045 \newcommand\numberproblemsin[1]{\@addtoreset{problem}{#1}}
8046 \def\theplainsproblem{\arabic{problem}}
8047 \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.

```

8048 \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

```

8049 \newcommand\prob@number{
8050   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
8051     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
8052   }{
8053     \int_if_exist:NTF \l__problems_prob_refnum_int {
8054       \prob@label{\int_use:N \l__problems_prob_refnum_int }
8055     }{
8056       \prob@label\theplainsproblem
8057     }
8058   }
8059 }
8060 \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.

```

8061 \newcommand\prob@title[3]{%
8062   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
8063     #2 \l__problems_inclprob_title_tl #3
8064   }{
8065     \tl_if_empty:NTF \l__problems_prob_title_tl {
8066       #1
8067     }{
8068       #2 \l__problems_prob_title_tl #3
8069     }
8070   }
8071 }

```

(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.

```

8072 \def\prob@heading{
8073   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
8074   %\sref@label@id{\prob@problem@kw~\prob@number}{~}
8075 }

```

(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.)`

```

8076 \newenvironment{sproblem}[1][]{
8077   \__problems_prob_args:n{#1}%\sref@target%
8078   \@in@omtexttrue% we are in a statement (for inline definitions)
8079   \refstepcounter{sproblem}\record@problem
8080   \def\current@section@level{\prob@problem@kw}
8081
8082   \str_if_empty:NT \l__problems_prob_name_str {
8083     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
8084     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
8085     \seq_get_left:NN \l_tmpa_seq \l__problems_prob_name_str
8086   }
8087
8088   \stex_if_do_html:T{
8089     \tl_if_empty:NF \l__problems_prob_title_tl {
8090       \exp_args:No \stex_document_title:n \l__problems_prob_title_tl
8091     }
8092   }
8093
8094   \exp_args:Nno\stex_module_setup:nn{type=problem}\l__problems_prob_name_str
8095
8096   \stex_reactivate_macro:N \STEXexport
8097   \stex_reactivate_macro:N \importmodule

```

```

8098 \stex_reactivate_macro:N \symdecl
8099 \stex_reactivate_macro:N \notation
8100 \stex_reactivate_macro:N \symdef
8101
8102 \stex_if_do_html:T{
8103   \begin{stex_annotate_env} {problem} {
8104     \l_stex_module_ns_str ? \l_stex_module_name_str
8105   }
8106
8107   \stex_annotate_invisible:nnn{header}{} {
8108     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
8109     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
8110     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
8111       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
8112     }
8113   }
8114 }
8115
8116 \stex_csl_to_imports:No \importmodule \l__problems_prob_imports_tl
8117
8118
8119 \tl_if_exist:NTF \l__problems_inclprob_type_tl {
8120   \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
8121 }{
8122   \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
8123 }
8124 \str_if_exist:NTF \l__problems_inclprob_id_str {
8125   \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
8126 }{
8127   \str_set_eq:NN \sproblemid \l__problems_prob_id_str
8128 }
8129
8130
8131 \stex_if_smsmode:F {
8132   \clist_set:No \l_tmpa_clist \sproblemtype
8133   \tl_clear:N \l_tmpa_tl
8134   \clist_map_inline:Nn \l_tmpa_clist {
8135     \tl_if_exist:cT {__problems_sproblem_##1_start:}{
8136       \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_start:}}
8137     }
8138   }
8139   \tl_if_empty:NTF \l_tmpa_tl {
8140     \__problems_sproblem_start:
8141   }{
8142     \l_tmpa_tl
8143   }
8144 }
8145 \stex_ref_new_doc_target:n \sproblemid
8146 \stex_if_smsmode:TF \stex_smsmode_do: \ignorespacesandpars
8147 }{
8148   \__stex_modules_end_module:
8149   \stex_if_smsmode:F{
8150     \clist_set:No \l_tmpa_clist \sproblemtype
8151     \tl_clear:N \l_tmpa_tl

```

```

8152 \clist_map_inline:Nn \l_tmpa_clist {
8153   \tl_if_exist:cT {__problems_sproblem_##1_end:}{
8154     \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_end:}}
8155   }
8156 }
8157 \tl_if_empty:NTF \l_tmpa_tl {
8158   \__problems_sproblem_end:
8159 }{
8160   \l_tmpa_tl
8161 }
8162 }
8163 \stex_if_do_html:T{
8164   \end{stex_annotate_env}
8165 }
8166
8167 \smallskip
8168 }
8169
8170 \seq_put_right:Nx\g_stex_smsmode_allowedenvs_seq{\tl_to_str:n{sproblem}}
8171
8172
8173
8174 \cs_new_protected:Nn \__problems_sproblem_start: {
8175   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
8176 }
8177 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
8178
8179 \newcommand\stexpatchproblem[3][] {
8180   \str_set:Nx \l_tmpa_str{ #1 }
8181   \str_if_empty:NTF \l_tmpa_str {
8182     \tl_set:Nn \__problems_sproblem_start: { #2 }
8183     \tl_set:Nn \__problems_sproblem_end: { #3 }
8184   }{
8185     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
8186     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
8187   }
8188 }
8189
8190
8191 \bool_if:NT \c__problems_boxed_bool {
8192   \surroundwithmdframed{problem}
8193 }

```

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

```

8194 \def\record@problem{
8195   \protected@write\@auxout{}
8196   {
8197     \string\@problem{\prob@number}
8198     {
8199       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
8200         \l__problems_inclprob_pts_tl
8201       }{
8202         \l__problems_prob_pts_tl
8203       }

```

```

8204     }%
8205     {
8206         \tl_if_exist:NTF \l__problems_inclprob_min_tl {
8207             \l__problems_inclprob_min_tl
8208         }{
8209             \l__problems_prob_min_tl
8210         }
8211     }
8212 }
8213 }

```

(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).

```

8214 \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.

```

8215 \keys_define:nn { problem / solution }{
8216     id          .str_set_x:N = \l__problems_solution_id_str ,
8217     for         .str_set_x:N = \l__problems_solution_for_str ,
8218     type        .str_set_x:N = \l__problems_solution_type_str ,
8219     title       .tl_set:N     = \l__problems_solution_title_tl
8220 }
8221 \cs_new_protected:Nn \__problems_solution_args:n {
8222     \str_clear:N \l__problems_solution_id_str
8223     \str_clear:N \l__problems_solution_type_str
8224     \str_clear:N \l__problems_solution_for_str
8225     \tl_clear:N \l__problems_solution_title_tl
8226     \keys_set:nn { problem / solution }{ #1 }
8227 }

```

`\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.

```

8228 \box_new:N \l__problems_solution_box
8229 \newenvironment{solution}[1][{}]{
8230     \__problems_solution_args:n{#1}
8231     \stex_html_backend:TF{
8232         \stex_if_do_html:T{
8233             \begin{stex_annotate_env}{solution}{}
8234             \str_if_empty:NF \l__problems_solution_type_str {
8235                 \par\noindent
8236                 \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{}
8237             }
8238             \noindent\textbf{Solution}\tl_if_empty:NF\l__problems_solution_title_tl{~(\l__problem
8239         }
8240     }{
8241         \setbox\l__problems_solution_box\vbox\bgroup
8242         \par\smallskip\hrule\smallskip
8243         \noindent\textbf{Solution}\tl_if_empty:NF\l__problems_solution_title_tl{~(\l__problems
8244     }

```

```

8245 }{
8246   \stex_html_backend:TF{
8247     \stex_if_do_html:T{
8248       \end{stex_annotate_env}
8249     }
8250   }{
8251     \smallskip\hrule
8252     \egroup
8253     \bool_if:NT \c__problems_solutions_bool {
8254       \strut\par\noindent
8255       \box\l__problems_solution_box
8256     }
8257   }
8258 }
8259
8260 \newcommand\startsolutions{
8261   \bool_set_true:N \c__problems_solutions_bool
8262   \solutionstrue
8263   % \specialcomment{solution}{\@startsolution}{
8264   %   \bool_if:NF \c__problems_boxed_bool {
8265   %     \hrule\medskip
8266   %   }
8267   %   \end{small}%
8268   % }
8269   % \bool_if:NT \c__problems_boxed_bool {
8270   %   \surroundwithmdframed{solution}
8271   % }
8272 }

```

(End definition for \startsolutions. This function is documented on page 64.)

\stopsolutions

```

8273 \newcommand\stopsolutions{\bool_set_false:N \c__problems_solutions_bool \solutionsfalse}%\ex

```

(End definition for \stopsolutions. This function is documented on page 64.)

exnote (env.)

```

8274 \bool_if:NTF \c__problems_notes_bool {
8275   \newenvironment{exnote}[1][]{
8276     \par\smallskip\hrule\smallskip
8277     \noindent\textbf{\prob@note@kw :~ }\small
8278   }{
8279     \smallskip\hrule
8280   }
8281   }{
8282     \excludacomment{exnote}
8283   }

```

hint (env.)

```

8284 \bool_if:NTF \c__problems_notes_bool {
8285   \newenvironment{hint}[1][]{
8286     \par\smallskip\hrule\smallskip
8287     \noindent\textbf{\prob@hint@kw :~ }\small
8288   }{

```

```

8289     \smallskip\hrule
8290   }
8291   \newenvironment{exhint}[1][ ]{
8292     \par\smallskip\hrule\smallskip
8293     \noindent\textbf{\prob@hint@kw :~ }\small
8294   }{
8295     \smallskip\hrule
8296   }
8297   }{
8298     \excludecomment{hint}
8299     \excludecomment{exhint}
8300   }

```

gnote (*env.*)

```

8301 \bool_if:NTF \c__problems_notes_bool {
8302   \newenvironment{gnote}[1][ ]{
8303     \par\smallskip\hrule\smallskip
8304     \noindent\textbf{\prob@gnote@kw :~ }\small
8305   }{
8306     \smallskip\hrule
8307   }
8308   }{
8309     \excludecomment{gnote}
8310   }

```

39.3 Markup for Added Value Services

39.4 Multiple Choice Blocks

EdN:12

mcb (*env.*)¹²

```

8311 \newenvironment{mcb}{
8312   \begin{enumerate}
8313 }{
8314   \end{enumerate}
8315 }

```

we define the keys for the mcb macro

```

8316 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
8317   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
8318     \bool_set_true:N #1
8319   }{
8320     \bool_set_false:N #1
8321   }
8322 }
8323 \keys_define:nn { problem / mcb }{
8324   id .str_set_x:N = \l__problems_mcc_id_str ,
8325   feedback .tl_set:N = \l__problems_mcc_feedback_tl ,
8326   T .default:n = { false } ,
8327   T .bool_set:N = \l__problems_mcc_t_bool ,
8328   F .default:n = { false } ,

```

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

```

8329 F .bool_set:N = \l__problems_mcc_f_bool ,
8330 Ttext .tl_set:N = \l__problems_mcc_Ttext_tl ,
8331 Ftext .tl_set:N = \l__problems_mcc_Ftext_tl
8332 }
8333 \cs_new_protected:Nn \l__problems_mcc_args:n {
8334 \str_clear:N \l__problems_mcc_id_str
8335 \tl_clear:N \l__problems_mcc_feedback_tl
8336 \bool_set_false:N \l__problems_mcc_t_bool
8337 \bool_set_false:N \l__problems_mcc_f_bool
8338 \tl_clear:N \l__problems_mcc_Ttext_tl
8339 \tl_clear:N \l__problems_mcc_Ftext_tl
8340 \str_clear:N \l__problems_mcc_id_str
8341 \keys_set:nn { problem / mcc }{ #1 }
8342 }

```

\mcc

```

8343 \def\mccTrueText{\textbf{\prob@correct@kw!~}}
8344 \def\mccFalseText{\textbf{\prob@wrong@kw!~}}
8345 \newcommand\mcc[2][] {
8346 \l__problems_mcc_args:n{ #1 }
8347 \item[{$\Box$}] #2
8348 \bool_if:NT \c__problems_solutions_bool {
8349 \
8350 \bool_if:NT \l__problems_mcc_t_bool {
8351 \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccTrueText\l__problems_mcc_Ttext_tl
8352 }
8353 \bool_if:NT \l__problems_mcc_f_bool {
8354 \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccFalseText\l__problems_mcc_Ftext_tl
8355 }
8356 \tl_if_empty:NF \l__problems_mcc_feedback_tl {
8357 \emph{\l__problems_mcc_feedback_tl}
8358 }
8359 }
8360 } %solutions

```

(End definition for \mcc. This function is documented on page 65.)

39.5 Filling in Concrete Solutions

\includeproblem This is embarrassingly simple, but can grow over time.

```

8361 \newcommand\fillinsol[2][] {%
8362 \def\@test{#1}
8363 \quad%
8364 \ifsolutions\textcolor{red}{\@test!}\else%
8365 \fbox{\ifx\@test\empty\phantom{\huge{21}}\else\hspace{#1}\fi}%
8366 \fi}

```

(End definition for \includeproblem. This function is documented on page 67.)

39.6 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.

```

8367
8368 \keys_define:nn{ problem / inclproblem }{
8369   id      .str_set_x:N = \l__problems_inclprob_id_str,
8370   pts     .tl_set:N    = \l__problems_inclprob_pts_tl,
8371   min     .tl_set:N    = \l__problems_inclprob_min_tl,
8372   title   .tl_set:N    = \l__problems_inclprob_title_tl,
8373   refnum  .int_set:N    = \l__problems_inclprob_refnum_int,
8374   type    .tl_set:N    = \l__problems_inclprob_type_tl,
8375   mhrepos .str_set_x:N = \l__problems_inclprob_mhrepos_str
8376 }
8377 \cs_new_protected:Nn \l__problems_inclprob_args:n {
8378   \str_clear:N \l__problems_prob_id_str
8379   \tl_clear:N \l__problems_inclprob_pts_tl
8380   \tl_clear:N \l__problems_inclprob_min_tl
8381   \tl_clear:N \l__problems_inclprob_title_tl
8382   \tl_clear:N \l__problems_inclprob_type_tl
8383   \int_zero_new:N \l__problems_inclprob_refnum_int
8384   \str_clear:N \l__problems_inclprob_mhrepos_str
8385   \keys_set:nn { problem / inclproblem }{ #1 }
8386   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
8387     \let\l__problems_inclprob_pts_tl\undefined
8388   }
8389   \tl_if_empty:NT \l__problems_inclprob_min_tl {
8390     \let\l__problems_inclprob_min_tl\undefined
8391   }
8392   \tl_if_empty:NT \l__problems_inclprob_title_tl {
8393     \let\l__problems_inclprob_title_tl\undefined
8394   }
8395   \tl_if_empty:NT \l__problems_inclprob_type_tl {
8396     \let\l__problems_inclprob_type_tl\undefined
8397   }
8398   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
8399     \let\l__problems_inclprob_refnum_int\undefined
8400   }
8401 }
8402
8403 \cs_new_protected:Nn \l__problems_inclprob_clear: {
8404   \let\l__problems_inclprob_id_str\undefined
8405   \let\l__problems_inclprob_pts_tl\undefined
8406   \let\l__problems_inclprob_min_tl\undefined
8407   \let\l__problems_inclprob_title_tl\undefined
8408   \let\l__problems_inclprob_type_tl\undefined
8409   \let\l__problems_inclprob_refnum_int\undefined
8410   \let\l__problems_inclprob_mhrepos_str\undefined
8411 }
8412 \l__problems_inclprob_clear:
8413
8414 \newcommand\includeproblem[2][ ]{
8415   \l__problems_inclprob_args:n{ #1 }

```



```

8416 \exp_args:No \stex_in_repository:nn\l__problems_inclprob_mhrepos_str{
8417   \stex_html_backend:TF {
8418     \str_clear:N \l_tmpa_str
8419     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
8420       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
8421     }
8422     \stex_annotate_invisible:nnn{includeproblem}{
8423       \l_tmpa_str / #2
8424     }{}
8425   }{
8426     \begingroup
8427     \inputreftrue
8428     \tl_if_empty:nTF{ ##1 }{
8429       \input{#2}
8430     }{
8431       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
8432     }
8433     \endgroup
8434   }
8435 }
8436 \__problems_inclprob_clear:
8437 }

```

(End definition for `\includeproblem`. This function is documented on page 67.)

39.7 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.

```

8438 \AddToHook{enddocument}{
8439   \bool_if:NT \c__problems_pts_bool {
8440     \message{Total:~\arabic{pts}~points}
8441   }
8442   \bool_if:NT \c__problems_min_bool {
8443     \message{Total:~\arabic{min}~minutes}
8444   }
8445 }

```

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

```

8446 \def\pts#1{
8447   \bool_if:NT \c__problems_pts_bool {
8448     \marginpar{#1~\prob@pt@kw}
8449   }
8450 }
8451 \def\min#1{
8452   \bool_if:NT \c__problems_min_bool {
8453     \marginpar{#1~\prob@min@kw}
8454   }
8455 }

```

`\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.

```

8456 \newcounter{pts}
8457 \def\show@pts{
8458   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
8459     \bool_if:NT \c__problems_pts_bool {
8460       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
8461       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
8462     }
8463   }{
8464     \tl_if_exist:NT \l__problems_prob_pts_tl {
8465       \bool_if:NT \c__problems_pts_bool {
8466         \tl_if_empty:NT\l__problems_prob_pts_tl{
8467           \tl_set:Nn \l__problems_prob_pts_tl {0}
8468         }
8469         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
8470         \addtocounter{pts}{\l__problems_prob_pts_tl}
8471       }
8472     }
8473   }
8474 }

```

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

\show@min

```

8475 \newcounter{min}
8476 \def\show@min{
8477   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
8478     \bool_if:NT \c__problems_min_bool {
8479       \marginpar{\l__problems_inclprob_min_tl\ min}
8480       \addtocounter{min}{\l__problems_inclprob_min_tl}
8481     }
8482   }{
8483     \tl_if_exist:NT \l__problems_prob_min_tl {
8484       \bool_if:NT \c__problems_min_bool {
8485         \tl_if_empty:NT\l__problems_prob_min_tl{
8486           \tl_set:Nn \l__problems_prob_min_tl {0}
8487         }
8488         \marginpar{\l__problems_prob_min_tl\ min}
8489         \addtocounter{min}{\l__problems_prob_min_tl}
8490       }
8491     }
8492   }
8493 }
8494 \</package>

```

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

39.8 Testing and Spacing

\testspace

```

8495 \newcommand\testspace[1]{\bool_if:NT \c__problems_boxed_bool {\vspace*{#1}}}

```

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

`\testnewpage`

```
8496 \newcommand\testnewpage{\bool_if:NT \c__problems_boxed_bool {\newpage}}
```

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

`\testemptypage`

```
8497 \newcommand\testemptypage[1] [] {%
```

```
8498 \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`

```
8499 \newcommand\testsmallspace{\testspace{1cm}}
```

```
8500 \newcommand\testmedspace{\testspace{2cm}}
```

```
8501 \newcommand\testbigspace{\testspace{3cm}}
```

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

Chapter 40

Implementation: The hwexam Package

40.1 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.

```
8502 \*package>
8503 \ProvidesExplPackage{hwexam}{2022/08/08}{3.2.0}{homework assignments and exams}
8504 \RequirePackage{13keys2e}
8505
8506 \newif\iftest\testfalse
8507 \DeclareOption{test}{\testtrue\PassOptionsToPackage{\CurrentOption}{problem}}
8508 \newif\ifmultiple\multiplefalse
8509 \DeclareOption{multiple}{\multipletrue}
8510 \DeclareOption{lang}{\PassOptionsToPackage{\CurrentOption}{problem}}
8511 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
8512 \ProcessOptions
```

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

```
8513 \RequirePackage{keyval}[1997/11/10]
8514 \RequirePackage{problem}
```

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

```
8515 \newcommand\hwexam@assignment@kw{Assignment}
8516 \newcommand\hwexam@given@kw{Given}
8517 \newcommand\hwexam@due@kw{Due}
8518 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~blank~for~extra~space}
8519 \newcommand\hwexam@minutes@kw{minutes}
8520 \newcommand\correction@probs@kw{prob.}
8521 \newcommand\correction@pts@kw{total}
8522 \newcommand\correction@reached@kw{reached}
8523 \newcommand\correction@sum@kw{Sum}
8524 \newcommand\correction@grade@kw{grade}
8525 \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

```

8526 \AddToHook{begindocument}{
8527 \ltx@ifpackageloaded{babel}{
8528 \makeatletter
8529 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
8530 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
8531 \input{hwexam-ngerman.ldf}
8532 }
8533 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
8534 \input{hwexam-finnish.ldf}
8535 }
8536 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
8537 \input{hwexam-french.ldf}
8538 }
8539 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
8540 \input{hwexam-russian.ldf}
8541 }
8542 \makeatother
8543 }{}
8544 }
8545

```

40.2 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.

```

8546 \newcounter{assignment}
8547 %\numberproblemsin{assignment}

```

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

```

8548 \keys_define:nn { hwexam / assignment } {
8549 id .str_set:N = \l_@@_assign_id_str,
8550 number .int_set:N = \l_@@_assign_number_int,
8551 title .tl_set:N = \l_@@_assign_title_tl,
8552 type .tl_set:N = \l_@@_assign_type_tl,
8553 given .tl_set:N = \l_@@_assign_given_tl,
8554 due .tl_set:N = \l_@@_assign_due_tl,
8555 loadmodules .code:n = {
8556 \bool_set_true:N \l_@@_assign_loadmodules_bool
8557 }
8558 }
8559 \cs_new_protected:Nn \_@@_assignment_args:n {
8560 \str_clear:N \l_@@_assign_id_str
8561 \int_set:Nn \l_@@_assign_number_int {-1}
8562 \tl_clear:N \l_@@_assign_title_tl
8563 \tl_clear:N \l_@@_assign_type_tl
8564 \tl_clear:N \l_@@_assign_given_tl
8565 \tl_clear:N \l_@@_assign_due_tl
8566 \bool_set_false:N \l_@@_assign_loadmodules_bool
8567 \keys_set:nn { hwexam / assignment }{ #1 }
8568 }

```

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.

```

8569 \newcommand\given@due[2]{
8570 \bool_lazy_all:nF {
8571 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
8572 { \tl_if_empty_p:V \l_@@_assign_given_tl }
8573 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
8574 { \tl_if_empty_p:V \l_@@_assign_due_tl }
8575 }{ #1 }
8576
8577 \tl_if_empty:NTF \l_@@_inclasssign_given_tl {
8578 \tl_if_empty:NF \l_@@_assign_given_tl {
8579 \hwexam@given@kw\xspace\l_@@_assign_given_tl
8580 }
8581 }{
8582 \hwexam@given@kw\xspace\l_@@_inclasssign_given_tl
8583 }
8584
8585 \bool_lazy_or:nnF {
8586 \bool_lazy_and_p:nn {
8587 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
8588 }{
8589 \tl_if_empty_p:V \l_@@_assign_due_tl
8590 }
8591 }{
8592 \bool_lazy_and_p:nn {
8593 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
8594 }{
8595 \tl_if_empty_p:V \l_@@_assign_due_tl
8596 }
8597 }{ ,~ }
8598
8599 \tl_if_empty:NTF \l_@@_inclasssign_due_tl {
8600 \tl_if_empty:NF \l_@@_assign_due_tl {
8601 \hwexam@due@kw\xspace \l_@@_assign_due_tl
8602 }
8603 }{
8604 \hwexam@due@kw\xspace \l_@@_inclasssign_due_tl
8605 }
8606
8607 \bool_lazy_all:nF {
8608 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
8609 { \tl_if_empty_p:V \l_@@_assign_given_tl }
8610 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
8611 { \tl_if_empty_p:V \l_@@_assign_due_tl }
8612 }{ #2 }
8613 }

```

`\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.

```

8614 \newcommand\assignment@title[3]{
8615 \tl_if_empty:NTF \l_@@_inclasssign_title_tl {
8616 \tl_if_empty:NTF \l_@@_assign_title_tl {
8617 #1
8618 }{
8619 #2\l_@@_assign_title_tl#3
8620 }
8621 }{
8622 #2\l_@@_inclasssign_title_tl#3
8623 }
8624 }

```

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

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

```

8625 \newcommand\assignment@number{
8626 \int_compare:nNnTF \l_@@_inclasssign_number_int = {-1} {
8627 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8628 \arabic{assignment}
8629 } {
8630 \int_use:N \l_@@_assign_number_int
8631 }
8632 }{
8633 \int_use:N \l_@@_inclasssign_number_int
8634 }
8635 }

```

(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.

```

8636 \newenvironment{assignment}[1][]{
8637 \_@@_assignment_args:n { #1 }
8638 %\sref@target
8639 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8640 \global\stepcounter{assignment}
8641 }{
8642 \global\setcounter{assignment}{\int_use:N\l_@@_assign_number_int}
8643 }
8644 \setcounter{sproblem}{0}
8645 \renewcommand\prob@label[1]{\assignment@number.##1}
8646 \def\current@section@level{\document@hwexamtype}
8647 %\sref@label{id}{\document@hwexamtype \thesection}
8648 \begin{@assignment}
8649 }{
8650 \end{@assignment}
8651 }

```

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

```

8652 \def\ass@title{
8653 {\protect\document@hwexamtype}\arabic{assignment}
8654 \assignment@title{}\{;\}\{;\} -- \given@due{}\{;\}
8655 }
8656 \ifmultiple
8657 \newenvironment{@assignment}{
8658 \bool_if:NTF \l_@@_assign_loadmodules_bool {
8659 \begin{sfragment}[loadmodules]{\ass@title}
8660 }{
8661 \begin{sfragment}{\ass@title}
8662 }
8663 }{
8664 \end{sfragment}
8665 }

```

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

```

8666 \else
8667 \newenvironment{@assignment}{
8668 \begin{center}\bf
8669 \Large@title\strut\
8670 \document@hwexamtype}\arabic{assignment}\assignment@title{}\{;\}\{;\}\{;\}
8671 \large\given@due{--;\}\{;\}--}
8672 \end{center}
8673 }{}
8674 \fi% multiple

```

40.3 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.

```

8675 \keys_define:nn { hwexam / inclassignment } {
8676 %id .str_set_x:N = \l_@@_assign_id_str,
8677 number .int_set:N = \l_@@_inclassign_number_int,
8678 title .tl_set:N = \l_@@_inclassign_title_tl,
8679 type .tl_set:N = \l_@@_inclassign_type_tl,
8680 given .tl_set:N = \l_@@_inclassign_given_tl,
8681 due .tl_set:N = \l_@@_inclassign_due_tl,
8682 mhrepos .str_set_x:N = \l_@@_inclassign_mhrepos_str
8683 }
8684 \cs_new_protected:Nn \l_@@_inclassignment_args:n {
8685 \int_set:Nn \l_@@_inclassign_number_int {-1}
8686 \tl_clear:N \l_@@_inclassign_title_tl
8687 \tl_clear:N \l_@@_inclassign_type_tl
8688 \tl_clear:N \l_@@_inclassign_given_tl
8689 \tl_clear:N \l_@@_inclassign_due_tl
8690 \str_clear:N \l_@@_inclassign_mhrepos_str
8691 \keys_set:nn { hwexam / inclassignment }{ #1 }
8692 }
8693 \l_@@_inclassignment_args:n {}
8694
8695 \newcommand\inputassignment[2][]{

```



```

8696 \_@@_inclassignment_args:n { #1 }
8697 \str_if_empty:NTF \l_@@_inclassign_mhrepos_str {
8698   \input{#2}
8699 }{
8700   \stex_in_repository:nn{\l_@@_inclassign_mhrepos_str}{
8701     \input{\mhpath{\l_@@_inclassign_mhrepos_str}{#2}}
8702   }
8703 }
8704 \_@@_inclassignment_args:n {}
8705 }
8706 \newcommand\includeassignment[2][ ]{
8707   \newpage
8708   \inputassignment[#1]{#2}
8709 }

```

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

40.4 Typesetting Exams

\quizheading

```

8710 \ExplSyntaxOff
8711 \newcommand\quizheading[1]{%
8712   \def\@tas{#1}%
8713   \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
8714   \ifx\@tas\@empty\else%
8715     \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
8716   \fi%
8717 }
8718 \ExplSyntaxOn

```

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

\testheading

```

8719
8720 \def\hwexamheader{\input{hwexam-default.header}}
8721
8722 \def\hwexamminutes{
8723   \tl_if_empty:NTF \testheading@duration {
8724     {\testheading@min}~\hwexam@minutes@kw
8725   }{
8726     \testheading@duration
8727   }
8728 }
8729
8730 \keys_define:nn { hwexam / testheading } {
8731   min .tl_set:N = \testheading@min,
8732   duration .tl_set:N = \testheading@duration,
8733   reqpts .tl_set:N = \testheading@reqpts,
8734   tools .tl_set:N = \testheading@tools
8735 }
8736 \cs_new_protected:Nn \_@@_testheading_args:n {
8737   \tl_clear:N \testheading@min
8738   \tl_clear:N \testheading@duration

```

```

8739 \tl_clear:N \testheading@reqpts
8740 \tl_clear:N \testheading@tools
8741 \keys_set:nn { hwexam / testheading }{ #1 }
8742 }
8743 \newenvironment{testheading}[1][ ]{
8744 \_@@_testheading_args:n{ #1 }
8745 \newcount\check@time\check@time=\testheading@min
8746 \advance\check@time by -\theassignment@totalmin
8747 \newif\if@bonuspoints
8748 \tl_if_empty:NTF \testheading@reqpts {
8749 \@bonuspointsfalse
8750 }{
8751 \newcount\bonus@pts
8752 \bonus@pts=\theassignment@totalpts
8753 \advance\bonus@pts by -\testheading@reqpts
8754 \edef\bonus@pts{\the\bonus@pts}
8755 \@bonuspointstrue
8756 }
8757 \edef\check@time{\the\check@time}
8758
8759 \makeatletter\hwexamheader\makeatother
8760 }{
8761 \newpage
8762 }

```

(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.

```

8763 <@@=problems>
8764 \renewcommand\@problem[3]{
8765 \stepcounter{assignment@probs}
8766 \def\__problemspts{#2}
8767 \ifx\__problemspts\empty\else
8768 \addtocounter{assignment@totalpts}{#2}
8769 \fi
8770 \def\__problemsmin{#3}\ifx\__problemsmin\empty\else\addtocounter{assignment@totalmin}{#3}\fi
8771 \xdef\correction@probs{\correction@probs & #1}%
8772 \xdef\correction@pts{\correction@pts & #2}
8773 \xdef\correction@reached{\correction@reached &}
8774 }
8775 <@@=hwexam>

```

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

\correction@table This macro generates the correction table

```

8776 \newcounter{assignment@probs}
8777 \newcounter{assignment@totalpts}
8778 \newcounter{assignment@totalmin}
8779 \def\correction@probs{\correction@probs@kw}
8780 \def\correction@pts{\correction@pts@kw}
8781 \def\correction@reached{\correction@reached@kw}
8782 \stepcounter{assignment@probs}
8783 \newcommand\correction@table{

```

```

8784 \resizebox{\textwidth}{!}{%
8785 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
8786 &\multicolumn{\theassignment@probs}{c|}{%|
8787 {\footnotesize\correction@forgrading@kw} &\\ \hline
8788 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
8789 \correction@pts & \theassignment@totalpts & \\ \hline
8790 \correction@reached & & \[.7cm]\hline
8791 \end{tabular}}
8792 \end{package}

```

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

40.5 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 41

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/graphicx.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*

- [SIa] *sLaTeX/sTeX-IDE*. URL: <https://github.com/slatex/sTeX-IDE> (visited on 04/22/2022).
- [SIb] *sLaTeX/stexls-vscode-plugin*. URL: <https://github.com/slatex/stexls-vscode-plugin> (visited on 04/22/2022).
- [SLS] *sLaTeX/stexls*. URL: <https://github.com/slatex/stexls> (visited on 04/22/2022).
- [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).