

The sTeX3 Package Collection *

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

2022-05-01

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 3.0 (last revised 2022-05-01)

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
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	11
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	19
3.3.3	Argument Modes	19
	Mode-b Arguments	20
	Mode-a Arguments	20
	Mode-B Arguments	22
3.3.4	Type and Definiens Components	22
3.3.5	Precedences and Automated Bracketing	23
3.3.6	Variables	25
3.3.7	Variable Sequences	26
3.4	Module Inheritance and Structures	28
3.4.1	Multilinguality and Translations	28
3.4.2	Simple Inheritance and Namespaces	29
3.4.3	The mathstructure Environment	31
3.4.4	The copymodule Environment	34
3.4.5	The interpretmodule Environment	35
3.5	Primitive Symbols (The sTeX Metatheory)	35
4	Using sTeX Symbols	36
4.1	\symref and its variants	36
4.2	Marking Up Text and On-the-Fly Notations	37
4.3	Referencing Symbols and Statements	39

5	sTeX Statements	40
5.1	Definitions, Theorems, Examples, Paragraphs	40
6	Highlighting and Presentation Customizations	47
7	Additional Packages	49
7.1	Tikzinput: Treating TIKZ code as images	49
7.2	Modular Document Structuring	50
7.3	Slides and Course Notes	52
7.4	Representing Problems and Solutions	56
7.5	Homeworks, Quizzes and Exams	59
II	Documentation	62
8	sTeX-Basics	63
8.1	Macros and Environments	63
8.1.1	HTML Annotations	63
8.1.2	Babel Languages	64
8.1.3	Auxiliary Methods	64
9	sTeX-MathHub	65
9.1	Macros and Environments	65
9.1.1	Files, Paths, URIs	65
9.1.2	MathHub Archives	66
9.1.3	Using Content in Archives	67
10	sTeX-References	68
10.1	Macros and Environments	68
10.1.1	Setting Reference Targets	68
10.1.2	Using References	69
11	sTeX-Modules	70
11.1	Macros and Environments	70
11.1.1	The <code>smodule</code> environment	72
12	sTeX-Module Inheritance	74
12.1	Macros and Environments	74
12.1.1	SMS Mode	74
12.1.2	Imports and Inheritance	75
13	sTeX-Symbols	77
13.1	Macros and Environments	77
14	sTeX-Terms	79
14.1	Macros and Environments	79
15	sTeX-Structural Features	81
15.1	Macros and Environments	81
15.1.1	Structures	81

16	<code>sTeX</code>-Statements	82
16.1	Macros and Environments	82
17	<code>sTeX</code>-Proofs: Structural Markup for Proofs	83
18	<code>sTeX</code>-Metatheory	84
18.1	Symbols	84
III	Extensions	85
19	<code>Tikzinput</code>: Treating <code>TIKZ</code> code as images	86
19.1	Macros and Environments	86
20	<code>document-structure</code>: Semantic Markup for Open Mathematical Documents in <code>LaTeX</code>	87
21	<code>NotesSlides</code> – Slides and Course Notes	88
22	<code>problem.sty</code>: An Infrastructure for formatting Problems	89
23	<code>hwexam.sty/cls</code>: An Infrastructure for formatting Assignments and Exams	90
IV	Implementation	91
24	<code>sTeX</code>-Basics Implementation	92
24.1	The <code>sTeXDocument</code> Class	92
24.2	Preliminaries	93
24.3	Messages and logging	94
24.4	HTML Annotations	95
24.5	Babel Languages	96
24.6	Persistence	97
24.7	Auxiliary Methods	98
25	<code>sTeX</code>-MathHub Implementation	102
25.1	Generic Path Handling	102
25.2	PWD and <code>kpsewhich</code>	104
25.3	File Hooks and Tracking	105
25.4	MathHub Repositories	106
25.5	Using Content in Archives	111
26	<code>sTeX</code>-References Implementation	115
26.1	Document URIs and URLs	115
26.2	Setting Reference Targets	117
26.3	Using References	119
27	<code>sTeX</code>-Modules Implementation	122
27.1	The <code>smodule</code> environment	126
27.2	Invoking modules	132

28	sTeX-Module Inheritance Implementation	134
28.1	SMS Mode	134
28.2	Inheritance	138
29	sTeX-Symbols Implementation	143
29.1	Symbol Declarations	143
29.2	Notations	150
29.3	Variables	159
30	sTeX-Terms Implementation	165
30.1	Symbol Invocations	165
30.2	Terms	172
30.3	Notation Components	176
30.4	Variables	178
30.5	Sequences	180
31	sTeX-Structural Features Implementation	181
31.1	Imports with modification	182
31.2	The feature environment	190
31.3	Structure	190
32	sTeX-Statements Implementation	200
32.1	Definitions	200
32.2	Assertions	205
32.3	Examples	209
32.4	Logical Paragraphs	211
33	The Implementation	217
33.1	Proofs	217
33.2	Justifications	228
34	sTeX-Others Implementation	229
35	sTeX-Metatheory Implementation	230
36	Tikzinput Implementation	233
37	document-structure.sty Implementation	236
37.1	Package Options	236
37.2	Document Structure	237
37.3	Front and Backmatter	241
37.4	Global Variables	243
38	NotesSlides – Implementation	244
38.1	Class and Package Options	244
38.2	Notes and Slides	246
38.3	Header and Footer Lines	250
38.4	Frame Images	252
38.5	Colors and Highlighting	253
38.6	Sectioning	254
38.7	Excursions	256

39 The Implementation	258
39.1 Package Options	258
39.2 Problems and Solutions	259
39.3 Multiple Choice Blocks	266
39.4 Including Problems	267
39.5 Reporting Metadata	269
40 Implementation: The hwexam Package	271
40.1 Package Options	271
40.2 Assignments	272
40.3 Including Assignments	275
40.4 Typesetting Exams	276
40.5 Leftovers	278
41 References	279

Part I

Manual



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



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

Chapter 1

What is sTeX?

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

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

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

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

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

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

Chapter 2

Quickstart

2.1 Setup

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

1. way of writing $\text{\texttt{L\text{A}TeX}}$ 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 $\text{\texttt{TeXLive}}$ on your system as a $\text{\texttt{L\text{A}TeX}}$ enthusiast. If not now is the time to install it; see [\[TL\]](#). You can usually update $\text{\texttt{TeXLive}}$ via a package manager or the $\text{\texttt{TeXLive}}$ manager **tlmgr**.

Alternatively, you can install $\text{\texttt{sTeX}}$ from CTAN, the Comprehensive $\text{\texttt{TeX}}$ 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, you can 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. Then update your `TEXINPUTS` environment variable, e.g. by placing the following line in your `.bashrc`:

¹NEW PART: MK: reorganized, we do not need the full MKM tool chain

```
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 [\[Sla\]](#). 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](#)². 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](#).

²EdNOTE: For now, we require the `sTeX`-branch, requiring manually compiling the MMT sources

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*:

TODO: use some \LaTeX -archive instead of `smglom`, use a convergence-notion that includes the limit, mark-up the theorem properly

```

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

```

Compiling this document with `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 [chapter 6](#).

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

```
\begin{smodule}{GeometricSeries}
...
\end{smodule}
```

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

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

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

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

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

\symdef 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.

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

`\symname`

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 a local or 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, 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}`.

```
The \definame{geometricSeries} ...
```

`\definame`
`\definiendum`

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

```

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

```

The next snippet – set in a math environment – uses several semantic macros imported from (or recursively via) `series` and `realarithmetics`, such as `\defeq`, `\infinitesum`, etc. In math mode, using a semantic macro inserts its (default) definition. A semantic

macro can have several notations – in that case, we can explicitly choose a specific notation by providing its identifier as an optional argument; e.g. `\realdivide[frac]{a}{b}` will use the explicit notation named `frac` of the semantic macro `\realdivide`, which yields $\frac{a}{b}$ instead of a/b .

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

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

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?infinitiesum" property="stex:OMBIND">
    <munderover displaystyle="true">
      <mo resource="...?series?infinitiesum" 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?infinitiesum" property="stex:comp">=</mo>
        <mi resource="2" property="stex:arg">1</mi>
      </mrow>
      <mi resource="...?series?infinitiesum" 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>
```

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

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

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

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

Remark 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.

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.

sms (*<boolean>*) use *persisted* mode (not yet implemented).

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 sTeX **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. sTeX **expressions** finally are built up from usages of semantic macros.

 M

 M

 T

- sTeX archives are simultaneously MMT archives, and the same directory structure is consequently used.

- sTeX modules correspond to OMDoc/MMT *theories*. `\importmodules` (and

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

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, $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$ expressions are converted to OMDOC/MMT terms, which use the abstract syntax (and XML encoding) of OPENMATH [Bus+04].

3.2 $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$ 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, $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$ uses *archives* that determine the global namespaces for symbols and statements and make it possible for $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$ to find content referenced via such URIs.

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

1. If the $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$ package is loaded with the option `mathhub=/path/to/mathhub`, then $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$ 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 $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$ -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, $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$ 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, $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$ will look for a file `~/.stex/mathhub.path`. If this file exists, $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$ 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 $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$ Archives

An $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$ 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 $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$ system, it needs to be in `/user/foo/MathHub/smglom/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 $\text{\texttt{S\TeX}}$ 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 $\text{\texttt{S\TeX}}$ archive:

- `/source/mod/` – individual $\text{\texttt{S\TeX}}$ modules, containing symbol declarations, notations, and `\begin{sparagraph}[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

3.2.3 MANIFEST.MF-Files

The `MANIFEST.MF` in the `META-INF`-directory consists of key-value-pairs, informing $\text{\texttt{S\TeX}}$ (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/smgom/calculus
narration-base: http://mathhub.info/smgom/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 $\text{\texttt{S\TeX}}$, 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`),

³EdNOTE: MK: bisher habe ich immer PIC subdirs, soll ich das ändern?

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. $\text{\texttt{STeX}}$ ignores this field, but MMT can pick up on them to resolve dependencies, e.g. for `lmh install`.

3.2.4 Using Files in $\text{\texttt{STeX}}$ Archives Directly

Several macros provided by $\text{\texttt{STeX}}$ allow for directly including files in repositories. These are:

$\text{\texttt{\backslash mhinput}}$	$\text{\texttt{\backslash mhinput}}[\text{Some/Archive}]\{\text{some/file}\}$ directly inputs the file <code>some/file</code> in the <code>source-</code> folder of <code>Some/Archive</code> .
--------------------------------------	---

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

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

$\text{\texttt{\backslash ifinput}}$	Both $\text{\texttt{\backslash mhinput}}$ and $\text{\texttt{\backslash inputref}}$ set $\text{\texttt{\backslash ifinput}}$ 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.
--------------------------------------	---

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

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

$\text{\texttt{\backslash libinput}}$	$\text{\texttt{\backslash libinput}}\{\text{some/file}\}$ searches for a file <code>some/file</code> 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. $\text{\texttt{\backslash libinput}}\{\text{preamble}\}$ in a `.tex`-file in `smglom/calculus` will *first* input `.../smglom/meta-inf/lib/preamble.tex` and then `../smglom/calculus/lib/preamble.tex`.

$\text{\texttt{\backslash libinput}}$ 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` 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` (*<string>**) for use in customizations.

`deprecate` (*<module>*) if set, will throw a warning when loaded, urging to use *<module>* instead.

`id` (*<string>*) for cross-referencing.

`ns` (*<URI>*) 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` (*<language>*) if not set, computed from the current file name (e.g. `foo.en.tex`).

`sig` (*<language>*) 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 M \rightarrow An \LaTeX module corresponds to an MMT/OMDoc *theory*. As such it
 \hookrightarrow M \rightarrow gets assigned a module URI (*universal resource identifier*) of the form
 \hookrightarrow T \hookrightarrow $\langle namespace \rangle ? \langle module-name \rangle$.

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 `\smoduletitle`, `\smoduletype` and `\smoduleid`.

For example:

Example 2

Input:

```
1 \stexpatchmodule[display]
2   {\textbf{Module (\smoduletitle)}\par}
3   {\par\noindent\textbf{End of Module (\smoduletitle)}}
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

-  \rightarrow Applications of semantic macros, such as `\binarysymbol{a}{b}` are translated to
-  \rightarrow MMT/OMDOC as OMA-terms with head `<OMS name="...?binarysymbol"/>`.
-  \rightarrow 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 \LaTeX 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[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 \LaTeX 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 \LaTeX 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}[h1,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 `h1`. Since `h1` 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 \LaTeX allows to re-set notation defaults.

$\backslash\text{setnotation}$

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

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

Operator Notations

Once we have a semantic macro with arguments, such as $\backslash\text{newbinarysymbol}$, 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 $\backslash\text{notation}$ (or $\backslash\text{symdef}$) with an *operator notation*, indicated with the optional argument op= . We can then invoke the operator notation using $\backslash\text{symbolname}![\text{notation-identifier}]$. Since operator notations never take arguments, we do not need to use $\backslash\text{comp}$ in it, the whole notation is wrapped in a $\backslash\text{comp}$ 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 $\text{a:} \cdot \text{; b:} \cdot$

\hookrightarrow $\backslash\text{symbolname}!$ is translated to OMDoc/MMT as $\langle\text{OMS name}=\dots?\text{symbolname}\rangle$
 \hookrightarrow directly.

3.3.3 Argument Modes

The notations so far used *simple* arguments which we call *mode-i* arguments. Declaring a new symbol with $\backslash\text{symdecl}\{\text{foo}\}[\text{args}=3]$ is equivalent to writing $\backslash\text{symdecl}\{\text{foo}\}[\text{args}=iii]$, 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 T_EX, 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}{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=iai]
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 \LaTeX (or, rather, 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 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 – for those purposes, `\infprec` and `\neginfprec` exist (which are implemented as the maximal and minimal integer values accordingly).

More precisely, each notation takes

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

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

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

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

In the example above:



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

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;\neginfp
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
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 \TeX 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  $\text{\seqa!}$  is  $\text{\seqa}{i}$ .

```

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  $\text{\addition}{\text{\seqa}}$ 
```

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  $\text{\seqa!}$  and  $\text{\addition}{\text{\seqa}}$ 

```

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 sTeX 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 sTeX) 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 sTeX we will see a very simple application of modules: managing multilinguality modularly.

3.4.1 Multilinguality and Translations

If we load the sTeX document class or package with the option `lang=<lang>`, sTeX will load the appropriate `babel` language for you – e.g. `lang=de` will load the `babel` language `ngerman`. Additionally, it makes sTeX 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 sTeX package option is set that allows for inferring a language, sTeX 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 $\text{lcm}(a, b)$ in english, but is called *kleinstes gemeinsames Vielfaches* in german and consequently denoted as $\text{kgV}(a, b)$ 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`
`\usemodule`

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

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

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



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

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

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

In other words: outside of archives, the namespace corresponds to the file URI with the filename dropped iff it is equal to the module name, and ignoring the (optional) language suffix.



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



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

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

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

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

Since this is less compatible with a modular development, using full URIs directly is strongly discouraged, unless the module is declared in the current file directly.

`\STEXexport`

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



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` 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 \hookrightarrow (see [MRK18]):
 \hookrightarrow `mathstructure{<name>}` simply creates a nested theory with name `<name>-structure`. The *constant* `<name>` is defined as `Mod(<name>-structure)` – a *dependent record type with manifest fields*, the fields of which are generated

$\hookrightarrow M$ from (and correspond to) the constants in `<name>-structure`.
 $\hookrightarrow M$ `\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.
 $\rightsquigarrow T$

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 $\varM! := \mathstruct{\varM{universe}, \varM{op}!, \varM{unit}}$
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 $\varMb! := \mathstruct{\varMb{universe}, \varMb{op}!, \varMb{unit}}$
4 be a \symname{monoid} on $\Int$ ...

```

Output:

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

.

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

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   \syndef{Integers}{\comp{\mathbb Z}}
3   \syndef{plus}[args=2,op=+]{#1 \comp+ #2}
4   \syndef{zero}{\comp0}
5   \syndef{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 rather than encoding it in STEX itself⁴

⁴EdNOTE: MK: why? continue

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

`\symref`
`\symname`

We have already seen `\symname` and `\symref`, the latter being the more general.

`\symref{<symbolname>}{<code>}` marks-up `<code>` as referencing `<symbolname>`. Since quite often, the `<code>` should be (a variant of) the name of the symbol anyway, we also have `\symname{<symbolname>}`.

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 M \rightarrow As expected, the above example is translated to OMDoc/MMT as an
 \rightarrow M \rightarrow OMA with `<OMS name="...?addition"/>` as head and `<OMV name="n"/>` and
 \rightarrow T \rightarrow `<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).⁵

Example 36

Input:

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

Output:

⁵EdNOTE: MK: I do not understand why we have to/want to give the second `arg*`; I think this must be elaborated on.

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.⁶

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...

4.3 Referencing Symbols and Statements

TODO: references documentation

⁶EdNOTE: MK: I do not understand this at all.

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 [chapter 6](#) 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 [section 4.3](#)), `type=` for customization (see [chapter 6](#)) 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`
`\definame`
`\Definame`

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 present to users, e.g. when hovering over symbols.
 \hookrightarrow T \rightarrow

`\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:⁷

Example 39

Input:

⁷EdNOTE: MK: we should reference the example explicitly here.

```

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

```

Output:

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

Axiom 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...

.

The main difference to before⁸ 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 ]

```

⁸EDNOTE: 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

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.

We will go over the general intuition by way of a running example:

```

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

```

This yields the following result:

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

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**

1. For the induction we have to consider the following cases:

1.1. $n = 1$: then we compute $1 = 1^2$ □

1.2. $n = 2$: This case is not really necessary, but we do it for the fun of it (and to get more intuition). We compute $1 + 3 = 2^2 = 4$ □

1.3. $n > 1$:

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

1.3.2. We have to show that we can derive the assertion for $n = k + 1$ from this assumption, i.e. $\sum_{i=1}^{k+1} (2i - 1) = (k + 1)^2$.

1.3.3. We obtain $\sum_{i=1}^{k+1} (2i - 1) = \sum_{i=1}^k (2i - 1) + 2(k + 1) - 1$ by splitting the sum.

1.3.4. Thus we have $\sum_{i=1}^{k+1} (2i - 1) = k^2 + 2k + 1$ by inductive hypothesis.

1.3.5. We can simplify the right-hand side to $(k + 1)^2$, which proves the assertion. □

1.4. We have considered all the cases, so we have proven the assertion. □

spproof The **spproof** 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 **spfststep**, **spfstcomment**, and **spfstcases** environments that are used to markup the proof steps.

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

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

spfststep Regular proof steps are marked up with the **step** environment, 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.

<hr/> <hr/> <code>\spfjust</code>	This evidence is marked up with the <code>\spfjust</code> macro in the <code>stex-proofs</code> package. This environment totally invisible to the formatted result; it wraps the text in the proof step that corresponds to the evidence. The environment takes an optional <code>KeyVal</code> argument, which can have the <code>method</code> key, whose value is the name of a proof method (this will only need to mean something to the application that consumes the semantic annotations). Furthermore, the justification can contain “premises” (specifications to assertions that were used justify the step) and “arguments” (other information taken into account by the proof method).
<hr/> <hr/> <code>\premise</code>	The <code>\premise</code> macro allows to mark up part of the text as reference to an assertion that is used in the argumentation. In the running example we have used the <code>\premise</code> macro to identify the inductive hypothesis.
<hr/> <hr/> <code>\justarg</code>	<p>The <code>\justarg</code> macro is very similar to <code>\premise</code> with the difference that it is used to mark up arguments to the proof method. Therefore the content of the first argument is interpreted as a mathematical object rather than as an identifier as in the case of <code>\premise</code>. In our example, we specified that the simplification should take place on the right hand side of the equation. Other examples include proof methods that instantiate. Here we would indicate the substituted object in a <code>\justarg</code> macro.</p> <p>Note that both <code>\premise</code> and <code>\justarg</code> can be used with an empty second argument to mark up premises and arguments that are not explicitly mentioned in the text.</p>
<code>subproof</code>	The <code>spfcases</code> environment is used to mark up a subproof. This environment takes an optional <code>KeyVal</code> argument for semantic annotations and a second argument that allows to specify an introductory comment (just like in the <code>proof</code> environment). The <code>method</code> key can be used to give the name of the proof method executed to make this subproof.
<code>spfcases</code>	The <code>spfcases</code> environment is used to mark up a proof by cases. Technically it is a variant of the <code>subproof</code> where the <code>method</code> is <code>by-cases</code> . Its contents are <code>spfcase</code> environments that mark up the cases one by one.
<code>spfcase</code>	The content of a <code>spfcases</code> environment are a sequence of case proofs marked up in the <code>spfcase</code> environment, which takes an optional <code>KeyVal</code> argument for semantic annotations. The second argument is used to specify the the description of the case under consideration. The content of a <code>spfcase</code> environment is the same as that of a <code>sproof</code> , i.e. <code>spfsteps</code> , <code>spfcmmnts</code> , and <code>spfcases</code> environments.
<hr/> <hr/> <code>\spfcasesketch</code>	<code>\spfcasesketch</code> is a variant of the <code>spfcase</code> environment that takes the same arguments, but instead of the <code>spfsteps</code> in the body uses a third argument for a proof sketch.
<code>spfcmmnt</code>	The <code>spfcmmnt</code> environment is much like a <code>step</code> , only that it does not have an object-level assertion of its own. Rather than asserting some fact that is relevant for the proof, it is used to explain where the proof is going, what we are attempting to to, or what we have achieved so far. As such, it cannot be the target of a <code>\premise</code> .

`\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{}`.

Chapter 6

Highlighting and Presentation Customizations

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

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

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

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

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

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

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

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

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

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

```
1 \stexpatchassertion[theorem]
2   {\ifx\sassertiontitle\@empty
3     \begin{theorem}
```

```

4 \else
5   \begin{theorem}[\sassertiontitle]
6   \fi}
7 {\end{theorem}}

```

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

```

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

```

`\compemph`
`\varemp`
`\symrefemph`
`\defemph`

Apart from the environments, we can control how \TeX highlights variables, notation components, `\symrefs` and `\definiendums`, respectively.

To do so, we simply redefine these four macros. For example, to highlight notation components (i.e. everything in a `\comp`) in blue, as in this document, we can do `\def\compemph#1{\textcolor{blue}{#1}}`. By default, `\compemph` et al do nothing.

`\compemph@uri`
`\varemp@uri`
`\symrefemph@uri`
`\defemph@uri`

For each of the four macros, there exists an additional macro that takes the full URI of the relevant symbol currently being highlighted as a second argument. That allows us to e.g. use pdf tooltips and links. For example, this document uses⁹

```

1 \protected\def\symrefemph@uri#1#2{
2   \pdftooltip{
3     \srefsymuri{#2}{\symrefemph{#1}}
4   }{
5     URI:~\detokenize{#2}
6   }
7 }

```

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

Chapter 7

Additional Packages

7.1 Tikzinput: Treating TIKZ code as images

image

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

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

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

The `standalone` class is a minimal \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
\ctikzinput

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

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

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

`\mhtikzinput`
`\cmhtikzinput`

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

`\libusetikzlibrary`

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

7.2 Modular Document Structuring

The `document-structure` package supplies an infrastructure for writing OMDOC documents in L^AT_EX. This includes a simple structure sharing mechanism for S_TE_X 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 S_TE_X 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 S_TE_X sources, or after translation. Currently, trans-document referencing provided by this package can only be used in the S_TE_X 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.

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

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

sfragment The structure of the document is given by nested `sfragment` environments. In the L^AT_EX 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, they need to be protected by `\protect`¹⁰, and we need to give the `loadmodules` key it needs no value. For instance we would have

```

1 \begin{smodule}{foo}
2   \symdef{bar}{B^a_r}
3   ...
4   \begin{sfragment}[id=sec.barderiv,loadmodules]
5     {Introducing $\protect\bar$ Derivations}
```

¹⁰EdNOTE: MK: still?

\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 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.¹¹

\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.

¹¹EDNOTE: MK: We need a substitute for the “Note that here the `display=flow` on the `sfragment` environment prevents numbering as is traditional for prefaces.”

`\currentsectionlevel`
`\CurrentSectionLevel`

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

`\prematurestop`
`\afterprematurestop`

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

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

`\setSGvar`
`\useSGvar`

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

`\ifSGvar`

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

7.3 Slides and Course Notes

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

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

slides
notes
sectocframes
frameimages
fiboxed

The notesslides class takes a variety of class options:

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

`frame,note` Slides are represented with the `frame` environment just like in the `beamer` class, see [Tanb] for details. The `notesslides` class adds the `note` environment for encapsulating the course note fragments.⁴



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 `\notestru` or `\notesfalse` is strongly discouraged however.

⁴MK: it would be very nice, if we did not need this environment, and this should be possible in principle, but not without intensive L^AT_EX trickery. Hints to the author are welcome.



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}`.

`nexample`, `nsproof`, `nassertion`

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

`\setslidelogo`

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

`\setsource`

The default footer line of the `notesslides` package mentions copyright and licensing. In the `beamer` class, `\source` stores the author's name as the copyright holder . By default it is *Michael Kohlhase* in the `notesslides` package since he is the main user and designer of this package. `\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.

Sometimes, we want to integrate slides as images after all – e.g. because we already have a PowerPoint presentation, to which we want to add `STEX` notes.

`\frameimage`
`\mhframeimage`

In this case we can use `\frameimage[⟨opt⟩]{⟨path⟩}`, where `⟨opt⟩` are the options of `\includegraphics` from the `graphicx` package [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: 

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}{../ex/founif}{We will cover first-order unification in}
2 ...
3 \begin{appendix}\printexcursions\end{appendix}
```

`\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`
`\printexcursion`
`\excursionref`

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

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

`\excursiongroup`

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

```
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

The `problem` package supplies an infrastructure that allows specify problem. Problems are text fragments that come with auxiliary functions: hints, notes, and solutions⁵. 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.

`solutions`
`notes`
`hints`
`gnotes`
`pts`
`min`
`boxed`
`test`

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

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

`problem` 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.

⁵for the moment multiple choice problems are not supported, but may well be in a future version

Example 40

Input:

```

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,height=3cm]
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 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,exnote,gnote The `hint` and `exnote` environments can be used in a `problem` environment to give hints and to make notes that elaborate certain aspects of the problem. The `gnote` (grading notes) environment can be used to document situations that may arise in grading.

\startsolutions
\stopsolutions

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

`\ifsolutions`

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

`mcb` 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.

If we start the solutions, then 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=Nooooooooo,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`
(**true**)
- ☐ `function`
(**false**) (*that is for C and C++*)
- ☐ `fun`
(**false**) (*that is for Standard ML*)
- ☐ `public static void`
(**false**) (*that is for Java*)

¹²EdNOTE: MK: that did not work!

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=Noooooooooooo,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
(true)
- ☐ function
(false) (that is for C and C++)
- ☐ fun
(false) (that is for Standard ML)
- ☐ public static void
(false) (that is for Java)

\includeproblem

The `\includeproblem` macro can be used to include a problem from another file. It takes an optional `KeyVal` argument and a second argument which is a path to the file containing the problem (the macro assumes that there is only one problem in the include file). The keys `title`, `min`, and `pts` specify the problem title, the estimated minutes for solving the problem and the points to be gained, and their values (if given) overwrite the ones specified in the `problem` environment in the included file.

The sum of the points and estimated minutes (that we specified in the `pts` and `min` keys to the `problem` environment or the `\includeproblem` macro) to the log file and the screen after each run. This is useful in preparing exams, where we want to make sure that the students can indeed solve the problems in an allotted time period.

The `\min` and `\pts` macros allow to specify (i.e. to print to the margin) the distribution of time and reward to parts of a problem, if the `pts` and `pts` options are set. This allows to give students hints about the estimated time and the points to be awarded.

7.5 Homeworks, Quizzes and Exams

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 `roblem` package. It is designed to be compatible with `problems.sty`, and inherits some of the functionality.

<code>solutions</code>	The <code>wexam</code> package and class take the options <code>solutions</code> , <code>notes</code> , <code>hints</code> , <code>gnotes</code> , <code>pts</code> , <code>min</code> , and <code>boxed</code> that are just passed on to the <code>problems</code> package (cf. its documentation for a description of the intended behavior).
<code>notes</code>	
<code>hints</code>	
<code>gnotes</code>	
<code>pts</code>	
<code>min</code>	
<code>assignment</code>	This package supplies the <code>assignment</code> environment that groups problems into assignment sheets. It takes an optional <code>KeyVal</code> argument with the keys <code>number</code> (for the assignment number; if none is given, 1 is assumed as the default or — in multi-assignment documents — the ordinal of the <code>assignment</code> environment), <code>title</code> (for the assignment title; this is referenced in the title of the assignment sheet), <code>type</code> (for the assignment type; e.g. “quiz”, or “homework”), <code>given</code> (for the date the assignment was given), and <code>due</code> (for the date the assignment is due).
<code>number</code>	
<code>title</code>	
<code>type</code>	
<code>given</code>	
<code>due</code>	
<code>multiple</code>	Furthermore, the <code>hwexam</code> package takes the option <code>multiple</code> 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.).
<code>test</code>	Finally, there is the option <code>test</code> that modifies the behavior to facilitate formatting tests. Only in <code>test</code> mode, the macros <code>\testspace</code> , <code>\testnewpage</code> , and <code>\testemptypage</code> have an effect: they generate space for the students to solve the given problems. Thus they can be left in the \LaTeX source.
<code>\testspace</code>	<code>\testspace</code> takes an argument that expands to a dimension, and leaves vertical space accordingly. <code>\testnewpage</code> makes a new page in <code>test</code> mode, and <code>\testemptypage</code> generates an empty page with the cautionary message that this page was intentionally left empty.
<code>\testnewpage</code>	
<code>\testemptypage</code>	
<code>testheading</code>	Finally, the <code>\testheading</code> takes an optional keyword argument where the keys <code>duration</code> specifies a string that specifies the duration of the test, <code>min</code> specifies the equivalent in number of minutes, and <code>reqpts</code> the points that are required for a perfect grade.
<code>duration</code>	
<code>min</code>	
<code>reqpts</code>	
	<pre> 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} </pre>

Will result in

Name:

Matriculation Number:

320101 General Computer Science (Fall 2010)

2022-05-01

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	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:13

13

`\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.

¹³EDNOTE: 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 sTeX 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</code>	<code>\begin{stex_annotate_env}{⟨property⟩}{⟨resource⟩}</code> $\langle content \rangle$ <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> <code>\stex_path_to_string:N</code>	The inverse; turns a path into a string and stores it in the second argument variable, or leaves it in the input stream.
---	--

<code>\stex_path_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 <i>absolute</i> , i.e. starts with an empty segment
---	---

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

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

9.1.2 MathHub Archives

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

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

9.1.3 Using Content in Archives

<hr/> <hr/> <code>\mhp</code> <hr/>	<code>\mhp{<i>archive-ID</i>}{<i>filename</i>}</code>
	Expands to the full path of file <i>filename</i> in repository <i>archive-ID</i> . Does not check whether the file or the repository exist.
<hr/> <hr/> <code>\inputref</code> <hr/> <code>\mhinput</code> <hr/>	<code>\inputref[<i>archive-ID</i>]{<i>filename</i>}</code> Both <code>\input</code> the file <i>filename</i> in archive <i>archive-ID</i> (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/> <hr/> <code>\addmhbibresource</code> <hr/>	<code>\inputref[<i>archive-ID</i>]{<i>filename</i>}</code> Adds a <code>.bib</code> -file <i>filename</i> in archive <i>archive-ID</i> (relative to the top-directory of the archive!).
<hr/> <hr/> <code>\libinput</code> <hr/>	<code>\libinput{<i>filename</i>}</code> Inputs <i>filename.tex</i> 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/> <hr/> <code>\libusepackage</code> <hr/>	<code>\libusepackage[<i>args</i>]{<i>filename</i>}</code> Like <code>\libinput</code> , but looks for <code>.sty</code> -files and calls <code>\usepackage[<i>meta</i>{<i>args</i>}]<i>Arg</i>{<i>filename</i>}</code> instead of <code>\input</code> . Throws an error, if none or more than one suitable package file is found.
<hr/> <hr/> <code>\mhgraphics</code> <hr/> <code>\cmhgraphics</code> <hr/>	<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/> <hr/> <code>\lstinputmhlisting</code> <hr/> <code>\clstinputmhlisting</code> <hr/>	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

\STEXreftitle**\STEXreftitle{*<some title>*}**

Sets the title of the current document to *<some title>*. A reference to the current document from *some other* document will then be displayed accordingly. e.g. if **\STEXreftitle{foo book}** is called, then referencing Definition 3.5 in this document in another document will display **Definition 3.5 in foo book**.

\stex_get_document_uri:

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.

\l_stex_current_docns_str

Stores its result in **\l_stex_current_docns_str**

\stex_get_document_url:

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.

\l_stex_current_docurl_str

Stores its result in **\l_stex_current_docurl_str**

10.1.1 Setting Reference Targets

\stex_ref_new_doc_target:n**\stex_ref_new_doc_target:n{*<id>*}**

Sets a new reference target with id *<id>*.

\stex_ref_new_sym_target:n**\stex_ref_new_sym_target:n{*<uri>*}**

Sets a new reference target for the symbol *<uri>*.

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:

`\c_stex_module_<URI>_prop`

A property list with the following fields:

name The *name* of the module,

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

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

lang the module's *language*,

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

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

meta the metatheory of the module.

`\c_stex_module_<URI>_code`

The code to execute when this module is activated (i.e. imported), e.g. to set all the semantic macros, notations, etc.

`\c_stex_module_<URI>_constants`

The names of all constants declared in the module

`\c_stex_module_<URI>_constants`

The full URIs of all modules imported in this module

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

\stex_modules_current_namespace:

Computes the current namespace as follows:

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

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

11.1.1 The smodule environment

module `\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 `\stex_invoke_module:n`

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

`\stex_activate_module:n`

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

Chapter 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: *`
`\stex_if_smsmode: TF *`

Tests whether SMS mode is currently active.

<hr/> <hr/> <code>\stex_file_in_smsmode:nn</code>	<code>\stex_in_smsmode:nn</code> $\{\langle filename \rangle\}$ $\{\langle code \rangle\}$
	Executes $\langle code \rangle$ in SMS mode, followed by the content of $\langle filename \rangle$. $\langle code \rangle$ 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.

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

12.1.2 Imports and Inheritance

<hr/> <hr/> <code>\importmodule</code>	<code>\importmodule</code> [$\langle archive-ID \rangle$] $\{\langle module-path \rangle\}$
	Imports a module by reading it from a file and “activating” it. $\text{\texttt{S\TeX}}$ determines the module and its containing file by passing its arguments on to <code>\stex_import_module_path:nn</code> .

<hr/> <hr/> <code>\usemodule</code>	<code>\importmodule</code> [$\langle archive-ID \rangle$] $\{\langle module-path \rangle\}$
	Like <code>\importmodule</code> , but does not export its contents; i.e. including the current module will not activate the used module

\stex_import_module_uri:nn

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

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

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

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

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

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

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

2. Otherwise:

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

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

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

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

\l_stex_import_name_str
\l_stex_import_archive_str
\l_stex_import_path_str
\l_stex_import_ns_str

stores the result in these four variables.

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

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

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

Chapter 13

STEX-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 STEX, but passed on to MMT for semantic services.
- **def**: An (ideally semantic) term, representing a *definiens*. Not used by STEX, but passed on to MMT for semantic services.
- **local**: A boolean (by default false). If set, this declaration will not be added to the module content, i.e. importing the current module will not make this declaration available.
- **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. $\backslash\text{symdecl}\{\text{plus}\}[\text{args}=\text{ii}]$ 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. $\backslash\text{symdecl}\{\text{plus}\}[\text{args}=\text{a}]$ allows for $\backslash\text{plus}\{2,2,2\}$.
 - b** a *variable* argument. Is treated by STEX like an *i*-argument, but an application is turned into an **OMBind** in OMDOC, binding the provided variable in the subsequent arguments of the operator; e.g. $\backslash\text{symdecl}\{\text{forall}\}[\text{args}=\text{bi}]$ allows for $\backslash\text{forall}\{x\in\text{Nat}\}\{x\geq 0\}$.

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

Chapter 14

STEX-Terms

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

14.1 Macros and Environments

<hr/> <hr/> <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>
<hr/> <hr/> <code>\symref</code>	<code>\symref{<symbol>}{<text>}</code> shortcut for <code>\STEXsymbol{<symbol>}! [<text>]</code>
<hr/> <hr/> <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 <code>!</code> , it will invoke the symbol <i>itself</i> rather than its application (and continue to <code>\stex_term_custom:nn</code>), i.e. it allows to refer to <code>\plus!</code> [addition] as an operation, rather than <code>\plus[addition of]{some}{terms}</code> .
<hr/> <hr/> <code>_stex_term_math_oms:nnnn</code> <code>_stex_term_math_oma:nnnn</code> <code>_stex_term_math_omb:nnnn</code>	<code><URI><fragment><precedence><body></code> Annotates <code><body></code> as an OMDOC-term (OMID, OMA or OMBIND, respectively) with head symbol <code><URI></code> , generated by the specific notation <code><fragment></code> with (upwards) operator precedence <code><precedence></code> . Inserts parentheses according to the current downwards precedence and operator precedence.
<hr/> <hr/> <code>_stex_term_math_arg:nnn</code>	<code>\stex_term_arg:nnn<int><prec><body></code> Annotates <code><body></code> as the <code><int></code> th argument of the current OMA or OMBIND, with (downwards) argument precedence <code><prec></code> .
<hr/> <hr/> <code>_stex_term_math_assoc_arg:nnnn</code>	<code>\stex_term_arg:nnn<int><prec><notation><body></code> Annotates <code><body></code> as the <code><int></code> th (associative) <i>sequence</i> argument (as comma-separated list of terms) of the current OMA or OMBIND, with (downwards) argument precedence <code><prec></code> and associative notation <code><notation></code> .

<hr/> <code>\infprec</code> <code>\neginfprec</code> <hr/>	Maximal and minimal notation precedences.
<hr/> <code>\dobrackets</code> <hr/>	<code>\dobrackets {⟨body⟩}</code> Puts <i>⟨body⟩</i> in parentheses; scaled if in display mode unscaled otherwise. Uses the current \S 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 <i>⟨body⟩</i>) sets the brackets used by \S TEX for automated bracketing (by default (and)) to <i>⟨left⟩</i> and <i>⟨right⟩</i> . Note that <i>⟨left⟩</i> and <i>⟨right⟩</i> 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 <i>⟨args⟩</i> 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` TODO

Chapter 16

sTeX-Statements

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

16.1 Macros and Environments

`symboldoc` `\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

LocalWords: bibfolder jobname.dtx tikzinput.dtx usetikzlibrary Gin@ewidth Gin@eheight
LocalWords: resizebox ctikzinput mhtikzinput Gin@mhrepos mhpath

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}{2022/03/03}{3.1.0}{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}{2022/03/03}{3.1.0}{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
57   *****^^J
58   *~This~is~sTeX~version~3.1.0*~^^J
59   *****^^J
60 ^^J}
61
62 %\RequirePackage{morewrites}
63 %\RequirePackage{amsmath}
64

```

Package options:

```

65 \keys_define:nn { stex } {
66   debug      .clist_set:N = \c_stex_debug_clist ,
67   lang       .clist_set:N = \c_stex_languages_clist ,
68   mathhub    .tl_set_x:N  = \mathhub ,
69   usesms     .bool_set:N  = \c_stex_persist_mode_bool ,
70   writesms   .bool_set:N  = \c_stex_persist_write_mode_bool ,
71   image      .bool_set:N  = \c_tikzinput_image_bool ,
72   unknown    .code:n      = {}
73 }
74 \ProcessKeysOptions { stex }

```

\stex The sTeX logo:

\sTeX

```

75 \RequirePackage{xspace}
76 \protected\def\stex{
77   \@ifundefined{texorpdfstring}{\let\texorpdfstring\@firstoftwo}{

```

```

78 \texorpdfstring{\raisebox{-.5ex}{S\kern-.5ex\TeX}}{sTeX}\xspace
79 }
80 \let\TeX\stex

```

(End definition for `\stex` and `\TeX`. These functions are documented on page 63.)

24.3 Messages and logging

```

81 <@@=stex_log>
    Warnings and error messages
82 \msg_new:nnn{stex}{error/unknownlanguage}{
83   Unknown~language:~#1
84 }
85 \msg_new:nnn{stex}{warning/nomathhub}{
86   MATHHUB~system~variable~not~found~and~no~
87   \detokenize{\mathhub}~value~set!
88 }
89 \msg_new:nnn{stex}{error/deactivated-macro}{
90   The~\detokenize{#1}~command~is~only~allowed~in~#2!
91 }

```

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

```

92 \cs_new_protected:Nn \stex_debug:nn {
93   \clist_if_in:NnTF \c_stex_debug_clist { all } {
94     \msg_set:nnn{stex}{debug / #1}{
95       \Debug~#1:~#2\
96     }
97     \msg_none:nn{stex}{debug / #1}
98   }{
99     \clist_if_in:NnT \c_stex_debug_clist { #1 } {
100       \msg_set:nnn{stex}{debug / #1}{
101         \Debug~#1:~#2\
102       }
103       \msg_none:nn{stex}{debug / #1}
104     }
105   }
106 }

```

(End definition for `\stex_debug:nn`. This function is documented on page 63.)

Redirecting messages:

```

107 \clist_if_in:NnTF \c_stex_debug_clist {all} {
108   \msg_redirect_module:nnn{ stex }{ none }{ term }
109 }{
110   \clist_map_inline:Nn \c_stex_debug_clist {
111     \msg_redirect_name:nnn{ stex }{ debug / ##1 }{ term }
112   }
113 }
114
115 \stex_debug:nn{log}{debug~mode~on}

```

24.4 HTML Annotations

116 `<@=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`

117 `\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`

```
118 \cs_new_protected:Nn \_stex_html_checkempty:n {
119   \tl_set:Nn \l_stex_html_arg_tl { #1 }
120   \tl_if_empty:NT \l_stex_html_arg_tl {
121     \tl_set_eq:NN \l_stex_html_arg_tl \c_stex_html_emptyarg_tl
122   }
123 }
```

(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`

```
124 \bool_new:N \_stex_html_do_output_bool
125 \bool_set_true:N \_stex_html_do_output_bool
126
127 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
128   \bool_if:nTF \_stex_html_do_output_bool
129     \prg_return_true: \prg_return_false:
130 }
```

(End definition for `\stex_if_do_html:TF`. This function is documented on page 63.)

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

```
131 \cs_new_protected:Nn \stex_suppress_html:n {
132   \exp_args:Nne \use:nn {
133     \bool_set_false:N \_stex_html_do_output_bool
134     #1
135   }{
136     \stex_if_do_html:T {
137       \bool_set_true:N \_stex_html_do_output_bool
138     }
139   }
140 }
```

(End definition for `\stex_suppress_html:n`. This function is documented on page 63.)

`\stex_annotate:bnx`

`\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.

```
141 \tl_if_exist:NF\stex@backend{
142   \ifcsname if@rustex\endcsname
143     \def\stex@backend{rustex}
144   \else
145     \ifcsname if@latexml\endcsname
```

```

146     \def\stex@backend{latexml}
147   \else
148     \def\stex@backend{pdflatex}
149   \fi
150 \fi
151 }
152 \input{stex-backend-\stex@backend.cfg}

```

(End definition for `\stex_annotate:nnn`, `\stex_annotate_invisible:n`, and `\stex_annotate_invisible:nnn`. These functions are documented on page 64.)

24.5 Babel Languages

```

153 <@@=stex_language>

```

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

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

```

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

```

(End definition for `\c_stex_languages_prop` and `\c_stex_language_abbrevs_prop`. These variables are documented on page 64.)

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

```

179 \cs_new_protected:Nn \stex_set_language:Nn {
180   \str_set:Nx \l_tmpa_str {#2}
181   \prop_get:NoNT \c_stex_languages_prop \l_tmpa_str #1 {
182     \ifx\@onlypreamble\@notprerr
183       \ltx@ifpackageloaded{babel}{
184         \exp_args:No \selectlanguage #1
185       }{}
186     \else
187       \exp_args:No \str_if_eq:nnTF #1 {turkish} {

```

```

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

```

24.6 Persistence

```

232 <@@=stex_persist>
233 \bool_if:NTF \c_stex_persist_mode_bool {
234   \def \stex_persist:n #1 {}
235   \def \stex_persist:x #1 {}
236 }{
237   \bool_if:NTF \c_stex_persist_write_mode_bool {

```

```

238 \iow_new:N \c__stex_persist_iow
239 \iow_open:Nn \c__stex_persist_iow{\jobname.sms}
240 \AtEndDocument{
241   \iow_close:N \c__stex_persist_iow
242 }
243 \cs_new_protected:Nn \stex_persist:n {
244   \tl_set:Nn \l_tmpa_tl { #1 }
245   \regex_replace_all:nnN { \cP\# } { \cO\# } \l_tmpa_tl
246   \exp_args:NNo \iow_now:Nn \c__stex_persist_iow \l_tmpa_tl
247 }
248 \cs_generate_variant:Nn \stex_persist:n {x}
249 }{
250   \def \stex_persist:n #1 {}
251   \def \stex_persist:x #1 {}
252 }
253 }

```

24.7 Auxiliary Methods

`\stex_deactivate_macro:Nn`

```

254 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
255   \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
256   \def#1{
257     \msg_error:nnnn{stex}{error/deactivated-macro}{\detokenize{#1}}{#2}
258   }
259 }

```

(End definition for `\stex_deactivate_macro:Nn`. This function is documented on page 64.)

`\stex_reactivate_macro:N`

```

260 \cs_new_protected:Nn \stex_reactivate_macro:N {
261   \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname
262 }

```

(End definition for `\stex_reactivate_macro:N`. This function is documented on page 64.)

`\ignorespacesandpars`

```

263 \protected\def\ignorespacesandpars{
264   \begingroup\catcode13=10\relax
265   \@ifnextchar\par{
266     \endgroup\expandafter\ignorespacesandpars\@gobble
267   }{
268     \endgroup
269   }
270 }
271
272 \cs_new_protected:Nn \stex_copy_control_sequence:NNN {
273   \tl_set:Nx \_tmp_args_tl {\cs_argument_spec:N #2}
274   \exp_args:NNo \tl_remove_all:Nn \_tmp_args_tl \c_hash_str
275   \int_set:Nn \l_tmpa_int {\tl_count:N \_tmp_args_tl}
276
277   \tl_clear:N \_tmp_args_tl
278   \int_step_inline:nn \l_tmpa_int {
279     \tl_put_right:Nx \_tmp_args_tl {\exp_not:n{####}\exp_not:n{##1}}

```



```

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

```

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

`\MMTrule`

```

330 \NewDocumentCommand \MMTrule {m m}{
331   \seq_set_split:Nnn \l_tmpa_seq , {#2}
332   \int_zero:N \l_tmpa_int
333   \stex_annotate_invisible:nnn{mmtrule}{scala://#1}{
334     \seq_if_empty:NF \l_tmpa_seq {
335       $\seq_map_inline:Nn \l_tmpa_seq {
336         \int_incr:N \l_tmpa_int
337         \stex_annotate:nnn{arg}{i\int_use:N \l_tmpa_int}{##1}
338       }$
339     }
340   }
341 }
342
343 \NewDocumentCommand \MMTinclude {m}{
344   \stex_annotate_invisible:nnn{import}{#1}{-}
345 }
346
347 \tl_new:N \g_stex_document_title
348 \cs_new_protected:Npn \STEXtitle #1 {
349   \tl_if_empty:NT \g_stex_document_title {
350     \tl_gset:Nn \g_stex_document_title { #1 }
351   }
352 }
353 \cs_new_protected:Nn \stex_document_title:n {
354   \tl_if_empty:NT \g_stex_document_title {
355     \tl_gset:Nn \g_stex_document_title { #1 }
356     \stex_annotate_invisible:n{\noindent
357       \stex_annotate:nnn{doctitle}{-}{ #1 }
358     \par}
359   }
360 }
361 \AtBeginDocument {
362   \let \STEXtitle \stex_document_title:n
363   \tl_if_empty:NF \g_stex_document_title {
364     \stex_annotate_invisible:n{\noindent
365       \stex_annotate:nnn{doctitle}{-}{ \g_stex_document_title }
366     \par}
367   }
368   \let \stex_maketitle:\maketitle
369   \def \maketitle{
370     \tl_if_empty:NF \@title {
371       \exp_args:No \stex_document_title:n \@title
372     }
373     \stex_maketitle:
374   }
375 }
376
377 \cs_new_protected:Nn \stex_par: {
378   \mode_if_vertical:F{
379     \if@minipage\else\if@nobreak\else\par\fi\fi
380   }
381 }
382
383 \</package>

```

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

Chapter 25

STEX -MathHub Implementation

```
384 <*package>
385
386 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
387
388 <@@=stex_path>
389
390 Warnings and error messages
391 \msg_new:nnn{stex}{error/norepository}{
392   No~archive~#1~found~in~#2
393 }
394 \msg_new:nnn{stex}{error/notinarchive}{
395   Not~currently~in~an~archive,~but~\detokenize{#1}~
396   needs~one!
397 }
398 \msg_new:nnn{stex}{error/nofile}{
399   \detokenize{#1}~could~not~find~file~#2
400 }
401 \msg_new:nnn{stex}{error/twofiles}{
402   \detokenize{#1}~found~two~candidates~for~#2
403 }
```

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`

```
402 \cs_new_protected:Nn \stex_path_from_string:Nn {
403   \str_set:Nx \l_tmpa_str { #2 }
404   \str_if_empty:NTF \l_tmpa_str {
405     \seq_clear:N #1
406   }{
407     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
408     \sys_if_platform_windows:T{
409       \seq_clear:N \l_tmpa_tl
```

```

410     \seq_map_inline:Nn #1 {
411       \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
412       \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
413     }
414     \seq_set_eq:NN #1 \l_tmpa_tl
415   }
416   \stex_path_canonicalize:N #1
417 }
418 }
419

```

(End definition for `\stex_path_from_string:Nn`. This function is documented on page 65.)

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

```

420 \cs_new_protected:Nn \stex_path_to_string:NN {
421   \exp_args:Nne \str_set:Nn #2 { \seq_use:Nn #1 / }
422 }
423
424 \cs_new:Nn \stex_path_to_string:N {
425   \seq_use:Nn #1 /
426 }

```

(End definition for `\stex_path_to_string:NN` and `\stex_path_to_string:N`. These functions are documented on page 65.)

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

```

427 \str_const:Nn \c__stex_path_dot_str {.}
428 \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.

```

429 \cs_new_protected:Nn \stex_path_canonicalize:N {
430   \seq_if_empty:NF #1 {
431     \seq_clear:N \l_tmpa_seq
432     \seq_get_left:NN #1 \l_tmpa_tl
433     \str_if_empty:NT \l_tmpa_tl {
434       \seq_put_right:Nn \l_tmpa_seq {}
435     }
436     \seq_map_inline:Nn #1 {
437       \str_set:Nn \l_tmpa_tl { ##1 }
438       \str_if_eq:NNF \l_tmpa_tl \c__stex_path_dot_str {
439         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
440           \seq_if_empty:NNTF \l_tmpa_seq {
441             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
442               \c__stex_path_up_str
443             }
444           }{
445             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
446             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
447               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
448                 \c__stex_path_up_str
449               }
450             }{

```

```

451         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
452     }
453 }
454 }{
455     \str_if_empty:NF \l_tmpa_tl {
456         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
457     }
458 }
459 }
460 }
461 \seq_gset_eq:NN #1 \l_tmpa_seq
462 }
463 }

```

(End definition for `\stex_path_canonicalize:N`. This function is documented on page 65.)

`\stex_path_if_absolute_p:N`
`\stex_path_if_absolute:N \underline{TF}`

```

464 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
465     \seq_if_empty:NTF #1 {
466         \prg_return_false:
467     }{
468         \seq_get_left:NN #1 \l_tmpa_tl
469         \sys_if_platform_windows:TF{
470             \str_if_in:NnTF \l_tmpa_tl {}:}{
471                 \prg_return_true:
472             }{
473                 \prg_return_false:
474             }
475         }{
476             \str_if_empty:NTF \l_tmpa_tl {
477                 \prg_return_true:
478             }{
479                 \prg_return_false:
480             }
481         }
482     }
483 }

```

(End definition for `\stex_path_if_absolute:N \underline{TF}` . This function is documented on page 65.)

25.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

484 \str_new:N\l_stex_kpsewhich_return_str
485 \cs_new_protected:Nn \stex_kpsewhich:n {
486     \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
487     \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
488     \tl_trim_spaces:N \l_stex_kpsewhich_return_str
489 }

```

(End definition for `\stex_kpsewhich:n`. This function is documented on page 65.)

We determine the PWD

`\c_stex_pwd_seq`
`\c_stex_pwd_str`

```

490 \sys_if_platform_windows:TF{
491   \begingroup\escapechar=-1\catcode'\=12
492   \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
493   \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
494   \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_
495   }}{
496   \stex_kpsewhich:n{-var-value~PWD}
497 }
498
499 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
500 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
501 \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 65.)

25.3 File Hooks and Tracking

502 `<@@=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

503 `\seq_gclear_new:N\g__stex_files_stack`

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`
`\c_stex_mainfile_str`

```

504 \str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
505 \stex_path_from_string:Nn \c_stex_mainfile_seq
506   \c_stex_mainfile_str

```

(End definition for `\c_stex_mainfile_seq` and `\c_stex_mainfile_str`. These variables are documented on page 65.)

`\g_stex_currentfile_seq`

507 `\seq_gclear_new:N\g_stex_currentfile_seq`

(End definition for `\g_stex_currentfile_seq`. This variable is documented on page 66.)

`\stex_filestack_push:n`

```

508 \cs_new_protected:Nn \stex_filestack_push:n {
509   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
510   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
511     \stex_path_from_string:Nn\g_stex_currentfile_seq{
512       \c_stex_pwd_str/#1
513     }
514   }
515   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
516   \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
517 }

```

(End definition for `\stex_filestack_push:n`. This function is documented on page 66.)

`\stex_filestack_pop:`

```

518 \cs_new_protected:Nn \stex_filestack_pop: {
519   \seq_if_empty:NF\g__stex_files_stack{
520     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
521   }
522   \seq_if_empty:NTF\g__stex_files_stack{
523     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
524   }{
525     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
526     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
527   }
528 }
```

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

Hooks for the current file:

```

529 \AddToHook{file/before}{
530   \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
531 }
532 \AddToHook{file/after}{
533   \stex_filestack_pop:
534 }
```

25.4 MathHub Repositories

535 `<@=stex_mathhub>`

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

```

536 \str_if_empty:NTF\mathhub{
537   \sys_if_platform_windows:TF{
538     \begingroup\escapechar=-1\catcode'\=12
539     \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str MATHHUB\c_percent_str}
540     \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
541     \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_kpsewhich_return_str}}
542   }{
543     \stex_kpsewhich:n{-var-value-MATHHUB}
544   }
545   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
546 }
547 \str_if_empty:NT \c_stex_mathhub_str {
548   \sys_if_platform_windows:TF{
549     \begingroup\escapechar=-1\catcode'\=12
550     \exp_args:Nx\stex_kpsewhich:n{-var-value-HOME}
551     \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
552     \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_kpsewhich_return_str}}
553   }{
554     \stex_kpsewhich:n{-var-value-HOME}
555   }
556   \ior_open:NnT \l_tmpa_ior{\l_stex_kpsewhich_return_str / .stex / mathhub.path}{
557     \begingroup\escapechar=-1\catcode'\=12
558     \ior_str_get:NN \l_tmpa_ior \l_tmpa_str
```



```

559     \sys_if_platform_windows:T{
560       \exp_args:NNx\str_replace_all:Nnn\l_tmpa_str{\c_backslash_str}/
561     }
562     \str_gset_eq:NN \c_stex_mathhub_str\l_tmpa_str
563     \endgroup
564     \ior_close:N \l_tmpa_ior
565   }
566 }
567 \str_if_empty:NTF\c_stex_mathhub_str{
568   \msg_warning:nn{stex}{warning/nomathhub}
569 }{
570   \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
571   \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
572 }
573 }{
574   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
575   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
576     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
577       \c_stex_pwd_str/\mathhub
578     }
579   }
580   \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
581   \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
582 }

```

(End definition for `\mathhub`, `\c_stex_mathhub_seq`, and `\c_stex_mathhub_str`. These variables are documented on page 66.)

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

```

583 \cs_new_protected:Nn \_stex_mathhub_do_manifest:n {
584   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
585     \str_set:Nx \l_tmpa_str { #1 }
586     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
587     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
588     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
589     \_stex_mathhub_find_manifest:N \l_tmpa_seq
590     \seq_if_empty:NTF \_stex_mathhub_manifest_file_seq {
591       \msg_error:nnxx{stex}{error/norepository}{#1}{
592         \stex_path_to_string:N \c_stex_mathhub_str
593       }
594     } {
595       \exp_args:No \_stex_mathhub_parse_manifest:n { \l_tmpa_str }
596     }
597   }
598 }

```

(End definition for `_stex_mathhub_do_manifest:n`.)

`\l_stex_mathhub_manifest_file_seq`

```

599 \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`:

```

600 \cs_new_protected:Nn \_stex_mathhub_find_manifest:N {
601   \seq_set_eq:NN \l_tmpa_seq #1
602   \bool_set_true:N \l_tmpa_bool
603   \bool_while_do:Nn \l_tmpa_bool {
604     \seq_if_empty:NTF \l_tmpa_seq {
605       \bool_set_false:N \l_tmpa_bool
606     }{
607       \file_if_exist:nTF{
608         \stex_path_to_string:N \l_tmpa_seq/MANIFEST.MF
609       }{
610         \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
611         \bool_set_false:N \l_tmpa_bool
612       }{
613         \file_if_exist:nTF{
614           \stex_path_to_string:N \l_tmpa_seq/META-INF/MANIFEST.MF
615         }{
616           \seq_put_right:Nn \l_tmpa_seq{META-INF}
617           \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
618           \bool_set_false:N \l_tmpa_bool
619         }{
620           \file_if_exist:nTF{
621             \stex_path_to_string:N \l_tmpa_seq/meta-inf/MANIFEST.MF
622           }{
623             \seq_put_right:Nn \l_tmpa_seq{meta-inf}
624             \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
625             \bool_set_false:N \l_tmpa_bool
626           }{
627             \seq_pop_right:NN \l_tmpa_seq \l_tmpa_tl
628           }
629         }
630       }
631     }
632   }
633   \seq_set_eq:NN \l__stex_mathhub_manifest_file_seq \l_tmpa_seq
634 }

```

(End definition for `_stex_mathhub_find_manifest:N`.)

`\c_stex_mathhub_manifest_ior` File variable used for MANIFEST-files

```

635 \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:

```

636 \cs_new_protected:Nn \_stex_mathhub_parse_manifest:n {
637   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
638   \ior_open:Nn \c_stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
639   \ior_map_inline:Nn \c_stex_mathhub_manifest_ior {
640     \str_set:Nn \l_tmpa_str {##1}
641     \exp_args:NNoo \seq_set_split:Nnn
642       \l_tmpb_seq \c_colon_str \l_tmpa_str
643     \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {

```

```

644 \exp_args:NNe \str_set:Nn \l_tmpb_tl {
645 \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
646 }
647 \exp_args:No \str_case:nnTF \l_tmpa_tl {
648 {id} {
649 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
650 { id } \l_tmpb_tl
651 }
652 {narration-base} {
653 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
654 { narr } \l_tmpb_tl
655 }
656 {url-base} {
657 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
658 { docurl } \l_tmpb_tl
659 }
660 {source-base} {
661 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
662 { ns } \l_tmpb_tl
663 }
664 {ns} {
665 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
666 { ns } \l_tmpb_tl
667 }
668 {dependencies} {
669 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
670 { deps } \l_tmpb_tl
671 }
672 }{}{}
673 }{}
674 }
675 \ior_close:N \c__stex_mathhub_manifest_ior
676 \stex_persist:x {
677 \prop_set_from_keyval:cn{ c_stex_mathhub_#1_manifest_prop }{
678 \exp_after:wN \prop_to_keyval:N \csname c_stex_mathhub_#1_manifest_prop\endcsname
679 }
680 }
681 }

```

(End definition for `__stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

682 \cs_new_protected:Nn \stex_set_current_repository:n {
683 \stex_require_repository:n { #1 }
684 \prop_set_eq:Nc \l_stex_current_repository_prop {
685 c_stex_mathhub_#1_manifest_prop
686 }
687 }

```

(End definition for `\stex_set_current_repository:n`. This function is documented on page 66.)

`\stex_require_repository:n`

```

688 \cs_new_protected:Nn \stex_require_repository:n {
689 \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
690 \stex_debug:nn{mathhub}{Opening~archive:~#1}

```

```

691   \_stex_mathhub_do_manifest:n { #1 }
692   }
693 }

```

(End definition for `\stex_require_repository:n`. This function is documented on page 66.)

`\l_stex_current_repository_prop` Current MathHub repository

```

694 %\prop_new:N \l_stex_current_repository_prop
695 \bool_if:NF \c_stex_persist_mode_bool {
696   \_stex_mathhub_find_manifest:N \c_stex_pwd_seq
697   \seq_if_empty:NTF \l_stex_mathhub_manifest_file_seq {
698     \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
699   } {
700     \_stex_mathhub_parse_manifest:n { main }
701     \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
702     \l_tmpa_str
703     \prop_set_eq:cN { c_stex_mathhub\_l_tmpa_str_manifest_prop }
704     \c_stex_mathhub_main_manifest_prop
705     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
706     \stex_debug:nn{mathhub}{Current~repository:~
707     \prop_item:Nn \l_stex_current_repository_prop {id}
708   }
709 }
710 }

```

(End definition for `\l_stex_current_repository_prop`. This variable is documented on page 66.)

`\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.

```

711 \cs_new_protected:Nn \stex_in_repository:nn {
712   \str_set:Nx \l_tmpa_str { #1 }
713   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
714   \str_if_empty:NTF \l_tmpa_str {
715     \prop_if_exist:NTF \l_stex_current_repository_prop {
716       \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
717       \exp_args:Ne \l_tmpa_cs{
718         \prop_item:Nn \l_stex_current_repository_prop { id }
719     }
720   }{
721     \l_tmpa_cs{}
722   }
723 }{
724   \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}
725   \stex_require_repository:n \l_tmpa_str
726   \str_set:Nx \l_tmpa_str { #1 }
727   \exp_args:Nne \use:nn {
728     \stex_set_current_repository:n \l_tmpa_str
729     \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
730   }{
731     \stex_debug:nn{mathhub}{switching~back~to:~
732     \prop_if_exist:NTF \l_stex_current_repository_prop {
733       \prop_item:Nn \l_stex_current_repository_prop { id }::~
734     \meaning\l_stex_current_repository_prop
735   }{

```

```

736         no~repository
737     }
738 }
739 \prop_if_exist:NTF \l_stex_current_repository_prop {
740     \stex_set_current_repository:n {
741         \prop_item:Nn \l_stex_current_repository_prop { id }
742     }
743 }{
744     \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
745 }
746 }
747 }
748 }

```

(End definition for `\stex_in_repository:nn`. This function is documented on page 66.)

25.5 Using Content in Archives

`\mhpath`

```

749 \def \mhpath #1 #2 {
750     \exp_args:Ne \tl_if_empty:nTF{#1}{
751         \c_stex_mathhub_str /
752         \prop_item:Nn \l_stex_current_repository_prop { id }
753         / source / #2
754     }{
755         \c_stex_mathhub_str / #1 / source / #2
756     }
757 }

```

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

`\inputref`

`\mhinput`

```

758 \newif \ifinputref \inputreffalse
759
760 \cs_new_protected:Nn \__stex_mathhub_mhinput:nn {
761     \stex_in_repository:nn {#1} {
762         \ifinputref
763             \input{ \c_stex_mathhub_str / ##1 / source / #2 }
764         \else
765             \inputreftrue
766             \input{ \c_stex_mathhub_str / ##1 / source / #2 }
767         \inputreffalse
768     \fi
769 }
770 }
771 \NewDocumentCommand \mhinput { 0{} m }{
772     \__stex_mathhub_mhinput:nn{ #1 }{ #2 }
773 }
774
775 \cs_new_protected:Nn \__stex_mathhub_inputref:nn {
776     \stex_in_repository:nn {#1} {
777         \stex_html_backend:TF {
778             \str_clear:N \l_tmpa_str

```

```

779     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
780       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
781     }
782     \stex_annotate_invisible:nnn{inputref}{
783       \l_tmpa_str / #2
784     }{}
785   }{
786     \begingroup
787     \inputreftrue
788     \tl_if_empty:nTF{ ##1 }{
789       \input{#2}
790     }{
791       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
792     }
793     \endgroup
794   }
795 }
796 }
797 \NewDocumentCommand \inputref { 0{} m}{
798   \__stex_mathhub_inputref:nn{ #1 }{ #2 }
799 }

```

(End definition for `\inputref` and `\mhinput`. These functions are documented on page 67.)

`\addmhbibresource`

```

800 \cs_new_protected:Nn \__stex_mathhub_mhbibresource:nn {
801   \stex_in_repository:nn {#1} {
802     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
803   }
804 }
805 \newcommand\addmhbibresource[2][ ]{
806   \__stex_mathhub_mhbibresource:nn{ #1 }{ #2 }
807 }

```

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

`\libinput`

```

808 \cs_new_protected:Npn \libinput #1 {
809   \prop_if_exist:NF \l_stex_current_repository_prop {
810     \msg_error:nnn{stex}{error/notinarchive}\libinput
811   }
812   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
813     \msg_error:nnn{stex}{error/notinarchive}\libinput
814   }
815   \seq_clear:N \l__stex_mathhub_libinput_files_seq
816   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
817   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
818
819   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
820     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #1.tex}
821     \IfFileExists{ \l_tmpa_str }{
822       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
823     }{}
824     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
825     \seq_put_right:No \l_tmpa_seq \l_tmpa_str

```

```

826 }
827
828 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}
829 \IfFileExists{ \l_tmpa_str }{
830   \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
831 }{}
832
833 \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
834   \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
835 }{
836   \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
837     \input{ ##1 }
838   }
839 }
840 }

```

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

`\libusepackage`

```

841 \NewDocumentCommand \libusepackage {0{ } m} {
842   \prop_if_exist:NF \l_stex_current_repository_prop {
843     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
844   }
845   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
846     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
847   }
848   \seq_clear:N \l__stex_mathhub_libinput_files_seq
849   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
850   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
851
852   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
853     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #2}
854     \IfFileExists{ \l_tmpa_str.sty }{
855       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
856     }{}
857     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
858     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
859   }
860
861   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2}
862   \IfFileExists{ \l_tmpa_str.sty }{
863     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
864   }{}
865
866   \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
867     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
868   }{
869     \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
870       \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
871         \usepackage[#1]{ ##1 }
872       }
873     }{
874       \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
875     }
876   }
877 }

```

```

876 }
877 }

```

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

```

\mhgraphics
\cmhgraphics

```

```

878
879 \AddToHook{begindocument}{
880 \ltx@ifpackageloaded{graphicx}{
881   \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
882   \newcommand\mhgraphics[2][]{\%
883     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
884     \includegraphics[#1]{\mhp@th\Gin@mhrepos{#2}}}
885   \newcommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
886 }{}

```

(End definition for `\mhgraphics` and `\cmhgraphics`. These functions are documented on page 67.)

```

\lstinputmhlisting
\clstinputmhlisting

```

```

887 \ltx@ifpackageloaded{listings}{
888   \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
889   \newcommand\lstinputmhlisting[2][]{\%
890     \def\lst@mhrepos{}\setkeys{lst}{#1}%
891     \lstinputlisting[#1]{\mhp@th\lst@mhrepos{#2}}}
892   \newcommand\clstinputmhlisting[2][]{\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
893 }{}
894 }
895
896 \end{package}

```

(End definition for `\lstinputmhlisting` and `\clstinputmhlisting`. These functions are documented on page 67.)

Chapter 26

STEX -References Implementation

```
897 <*package>
898
899 %%%%%%%%%% references.dtx %%%%%%%%%%
900
901 <@@=stex_refs>
    Warnings and error messages
902
```

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

```
903 %\iow_new:N \c__stex_refs_refs_iow
904 \AtBeginDocument{
905 % \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
906 }
907 \AtEndDocument{
908 % \iow_close:N \c__stex_refs_refs_iow
909 }
```

`\STEXreftitle`

```
910 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
911
912 \NewDocumentCommand \STEXreftitle { m } {
913   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
914 }
```

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

26.1 Document URIs and URLs

`\l_stex_current_docns_str`

```
915 \str_new:N \l_stex_current_docns_str
```

(End definition for `\l_stex_current_docns_str`. This variable is documented on page 68.)

`\stex_get_document_uri:`

```
916 \cs_new_protected:Nn \stex_get_document_uri: {
917   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
918   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
919   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
920   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
921   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
922
923   \str_clear:N \l_tmpa_str
924   \prop_if_exist:NT \l_stex_current_repository_prop {
925     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
926       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
927     }
928   }
929
930   \str_if_empty:NTF \l_tmpa_str {
931     \str_set:Nx \l_stex_current_docns_str {
932       file:/\stex_path_to_string:N \l_tmpa_seq
933     }
934   }{
935     \bool_set_true:N \l_tmpa_bool
936     \bool_while_do:Nn \l_tmpa_bool {
937       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
938       \exp_args:No \str_case:nnTF { \l_tmpb_str } {
939         {source} { \bool_set_false:N \l_tmpa_bool }
940       }{}{
941         \seq_if_empty:NT \l_tmpa_seq {
942           \bool_set_false:N \l_tmpa_bool
943         }
944       }
945     }
946
947     \seq_if_empty:NTF \l_tmpa_seq {
948       \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
949     }{
950       \str_set:Nx \l_stex_current_docns_str {
951         \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
952       }
953     }
954   }
955 }
```

(End definition for `\stex_get_document_uri:`. This function is documented on page 68.)

`\l_stex_current_docurl_str`

```
956 \str_new:N \l_stex_current_docurl_str
```

(End definition for `\l_stex_current_docurl_str`. This variable is documented on page 68.)

`\stex_get_document_url:`

```
957 \cs_new_protected:Nn \stex_get_document_url: {
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 { docurl } \l_tmpa_str {
967     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
968       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
969     }
970   }
971 }
972
973 \str_if_empty:NTF \l_tmpa_str {
974   \str_set:Nx \l_stex_current_docurl_str {
975     file:/\stex_path_to_string:N \l_tmpa_seq
976   }
977 }{
978   \bool_set_true:N \l_tmpa_bool
979   \bool_while_do:Nn \l_tmpa_bool {
980     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
981     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
982       {source} { \bool_set_false:N \l_tmpa_bool }
983     }{}{
984       \seq_if_empty:NT \l_tmpa_seq {
985         \bool_set_false:N \l_tmpa_bool
986       }
987     }
988   }
989
990   \seq_if_empty:NTF \l_tmpa_seq {
991     \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
992   }{
993     \str_set:Nx \l_stex_current_docurl_str {
994       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
995     }
996   }
997 }
998 }

```

(End definition for `\stex_get_document_url`:. This function is documented on page 68.)

26.2 Setting Reference Targets

```

999 \str_const:Nn \c__stex_refs_url_str{URL}
1000 \str_const:Nn \c__stex_refs_ref_str{REF}
1001 \str_new:N \l__stex_refs_curr_label_str
1002 % @currentlabel -> number
1003 % @currentlabelname -> title
1004 % @currentHref -> name.number <- id of some kind
1005 % \theH# -> \arabic{section}
1006 % \the# -> number
1007 % \hyper@makecurrent{#}
1008 \int_new:N \l__stex_refs_unnamed_counter_int

```

`\stex_ref_new_doc_target:n`

```

1009 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
1010   \stex_get_document_uri:
1011   \str_clear:N \l__stex_refs_curr_label_str
1012   \str_set:Nx \l_tmpa_str { #1 }
1013   \str_if_empty:NT \l_tmpa_str {
1014     \int_incr:N \l__stex_refs_unnamed_counter_int
1015     \str_set:Nx \l_tmpa_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
1016   }
1017   \str_set:Nx \l__stex_refs_curr_label_str {
1018     \l_stex_current_docns_str?\l_tmpa_str
1019   }
1020   \seq_if_exist:cF{g__stex_refs_labels_\l_tmpa_str_seq}{
1021     \seq_new:c {g__stex_refs_labels_\l_tmpa_str_seq}
1022   }
1023   \seq_if_in:coF{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str {
1024     \seq_gput_right:co{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str
1025   }
1026   \stex_if_smsmode:TF {
1027     \stex_get_document_url:
1028     \str_gset_eq:cN {sref_url_\l__stex_refs_curr_label_str_str}\l_stex_current_docurl_str
1029     \str_gset_eq:cN {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_url_str
1030   }{
1031     %\iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter\unexpanded\expandafter{
1032     \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
1033     \immediate\write\@auxout{\stexauxadddocref{\l_stex_current_docns_str}{\l_tmpa_str}}
1034     \str_gset:cx {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_ref_str
1035   }
1036 }

```

(End definition for `\stex_ref_new_doc_target:n`. This function is documented on page 68.)

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

```

1037 \cs_new_protected:Npn \stexauxadddocref #1 #2 {
1038   \str_set:Nn \l_tmpa_str {#1?#2}
1039   \str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
1040   \seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
1041     \seq_new:c {g__stex_refs_labels_#2_seq}
1042   }
1043   \seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
1044     \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
1045   }
1046 }

```

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

```

1047 \AtEndDocument{
1048   \def\stexauxadddocref#1 #2 {}{}
1049 }

```

`\stex_ref_new_sym_target:n`

```

1050 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
1051   \stex_if_smsmode:TF {
1052     \str_if_exist:cF{sref_sym_#1_type}{
1053       \stex_get_document_url:
1054       \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str

```

```

1055     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
1056   }
1057   ){
1058     \str_if_empty:NF \l__stex_refs_curr_label_str {
1059       \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
1060       \immediate\write\@auxout{
1061         \exp_not:N\expandafter\def\exp_not:N\csname \exp_not:N\detokenize{sref_sym_#1_label_
1062           \l__stex_refs_curr_label_str
1063         }
1064       }
1065     }
1066   }
1067 }

```

(End definition for `\stex_ref_new_sym_target:n`. This function is documented on page 68.)

26.3 Using References

```

1068 \str_new:N \l__stex_refs_indocument_str

```

\sref Optional arguments:

```

1069
1070 \keys_define:nn { stex / sref } {
1071   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
1072   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
1073   pre           .tl_set:N = \l__stex_refs_pre_tl ,
1074   post          .tl_set:N = \l__stex_refs_post_tl ,
1075 }
1076 \cs_new_protected:Nn \__stex_refs_args:n {
1077   \tl_clear:N \l__stex_refs_linktext_tl
1078   \tl_clear:N \l__stex_refs_fallback_tl
1079   \tl_clear:N \l__stex_refs_pre_tl
1080   \tl_clear:N \l__stex_refs_post_tl
1081   \str_clear:N \l__stex_refs_repo_str
1082   \keys_set:nn { stex / sref } { #1 }
1083 }

```

The actual macro:

```

1084 \NewDocumentCommand \sref { 0{} m}{
1085   \__stex_refs_args:n { #1 }
1086   \str_if_empty:NTF \l__stex_refs_indocument_str {
1087     \str_set:Nx \l_tmpa_str { #2 }
1088     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
1089     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
1090       \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
1091         \seq_get_left:cNF {g__stex_refs_labels_\l_tmpa_str _seq} \l_tmpa_str {
1092           \str_clear:N \l_tmpa_str
1093         }
1094       }{
1095         \str_clear:N \l_tmpa_str
1096       }
1097     }{
1098       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1099       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str

```

```

1100 \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
1101 \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str_seq}{
1102   \str_set_eq:NN \l_tmpc_str \l_tmpa_str
1103   \str_clear:N \l_tmpa_str
1104   \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str_seq} {
1105     \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
1106       \str_range:nnn { ##1 }{-\l_tmpa_int}{ -1 }
1107     }{
1108       \seq_map_break:n {
1109         \str_set:Nn \l_tmpa_str { ##1 }
1110       }
1111     }
1112   }
1113 }{
1114   \str_clear:N \l_tmpa_str
1115 }
1116 }
1117 \str_if_empty:NTF \l_tmpa_str {
1118   \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_lin
1119 }{
1120   \str_if_eq:cNTF {sref_\l_tmpa_str_type} \c__stex_refs_ref_str {
1121     \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1122       \cs_if_exist:cTF{autoref}{
1123         \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1124       }{
1125         \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1126       }
1127     }{
1128       \ltx@ifpackageloaded{hyperref}{
1129         \hyperref[sref_\l_tmpa_str]\l__stex_refs_linktext_tl
1130       }{
1131         \l__stex_refs_linktext_tl
1132       }
1133     }
1134   }{
1135     \ltx@ifpackageloaded{hyperref}{
1136       \href{\use:c{sref_url_\l_tmpa_str_str}}{\tl_if_empty:NTF \l__stex_refs_linktext_t
1137     }{
1138       \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs
1139     }
1140   }
1141 }
1142 }{
1143   % TODO
1144 }
1145 }

```

(End definition for `\sref`. This function is documented on page 69.)

`\srefsym`

```

1146 \NewDocumentCommand \srefsym { 0{} m}{
1147   \stex_get_symbol:n { #2 }
1148   \__stex_refs_sym_aux:nn{##1}{\l_stex_get_symbol_uri_str}
1149 }

```

```

1150
1151 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1152   \str_if_exist:cTF {sref_sym_#2 _label_str }{
1153     \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1154   }{
1155     \__stex_refs_args:n { #1 }
1156     \str_if_empty:NTF \l__stex_refs_indocument_str {
1157       \tl_if_exist:cTF{sref_sym_#2 _type}{
1158         % doc uri in \l_tmpb_str
1159         \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1160         \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1161           % reference
1162           \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1163             \cs_if_exist:cTF{autoref}{
1164               \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1165             }{
1166               \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1167             }
1168           }{
1169             \ltx@ifpackageloaded{hyperref}{
1170               \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1171             }{
1172               \l__stex_refs_linktext_tl
1173             }
1174           }
1175         }{
1176           % URL
1177           \ltx@ifpackageloaded{hyperref}{
1178             \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl \
1179           }{
1180             \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_re
1181           }
1182         }
1183       }{
1184         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_l
1185       }
1186     }{
1187       % TODO
1188     }
1189   }
1190 }

```

(End definition for \srefsym. This function is documented on page 69.)

\srefsymuri

```

1191 \cs_new_protected:Npn \srefsymuri #1 #2 {
1192   \__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1193 }

```

(End definition for \srefsymuri. This function is documented on page 69.)

```

1194 \</package>

```

Chapter 27

STEX -Modules Implementation

```
1195 <*package>
1196
1197 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1198
1199 <@@=stex_modules>
1200
1201   Warnings and error messages
1202   \msg_new:nnn{stex}{error/unknownmodule}{
1203     No~module~#1~found
1204   }
1205   \msg_new:nnn{stex}{error/syntax}{
1206     Syntax~error:~#1
1207   }
1208   \msg_new:nnn{stex}{error/siglanguage}{
1209     Module~#1~declares~signature~#2,~but~does~not~
1210     declare~its~language
1211   }
1212   \msg_new:nnn{stex}{warning/deprecated}{
1213     #1~is~deprecated;~please~use~#2~instead!
1214   }
1215   \msg_new:nnn{stex}{error/conflictingmodules}{
1216     Conflicting~imports~for~module~#1
1217   }
```

`\l_stex_current_module_str` The current module:

```
1217 \str_new:N \l_stex_current_module_str
```

(End definition for \l_stex_current_module_str. This variable is documented on page 71.)

`\l_stex_all_modules_seq` Stores all available modules

```
1218 \seq_new:N \l_stex_all_modules_seq
```

(End definition for \l_stex_all_modules_seq. This variable is documented on page 71.)


```

\stex_if_in_module_p:
\stex_if_in_module:TF
1219 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1220   \str_if_empty:NTF \l_stex_current_module_str
1221   \prg_return_false: \prg_return_true:
1222 }

(End definition for \stex_if_in_module:TF. This function is documented on page 71.)

```

```

\stex_if_module_exists_p:n
\stex_if_module_exists:nTF
1223 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1224   \prop_if_exist:cTF { c_stex_module_#1_prop }
1225   \prg_return_true: \prg_return_false:
1226 }

(End definition for \stex_if_module_exists:nTF. This function is documented on page 71.)

```

```

\stex_add_to_current_module:n
\STEXexport
1227 \cs_new_protected:Nn \stex_execute_in_module:n { \stex_if_in_module:T {
1228   \stex_add_to_current_module:n { #1 }
1229   \stex_do_up_to_module:n { #1 }
1230 }}
1231 \cs_generate_variant:Nn \stex_execute_in_module:n {x}
1232
1233 \cs_new_protected:Nn \stex_add_to_current_module:n {
1234   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1235 }
1236 \cs_generate_variant:Nn \stex_add_to_current_module:n {x}
1237 \cs_new_protected:Npn \STEXexport {
1238   \begingroup
1239   \newlinechar=-1\relax
1240   \endlinechar=-1\relax
1241   %\catcode'\ = 9\relax
1242   \expandafter\endgroup\__stex_modules_export:n
1243 }
1244 \cs_new_protected:Nn \__stex_modules_export:n {
1245   \ignorespaces #1
1246   \stex_add_to_current_module:n { \ignorespaces #1 }
1247   \stex_smsmode_do:
1248 }
1249 \stex_deactivate_macro:Nn \STEXexport {module~environments}

(End definition for \stex_add_to_current_module:n and \STEXexport. These functions are documented
on page 71.)

```

```

\stex_add_constant_to_current_module:n
1250 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1251   \str_set:Nx \l_tmpa_str { #1 }
1252   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
1253 }

(End definition for \stex_add_constant_to_current_module:n. This function is documented on page
71.)

```

`\stex_add_import_to_current_module:n`

```

1254 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1255   \str_set:Nx \l_tmpa_str { #1 }
1256   \exp_args:Nno
1257   \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1258     \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1259   }
1260 }

```

(End definition for `\stex_add_import_to_current_module:n`. This function is documented on page 71.)

`\stex_collect_imports:n`

```

1261 \cs_new_protected:Nn \stex_collect_imports:n {
1262   \seq_clear:N \l_stex_collect_imports_seq
1263   \__stex_modules_collect_imports:n {#1}
1264 }
1265 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1266   \seq_map_inline:cn {c_stex_module_#1_imports} {
1267     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1268       \__stex_modules_collect_imports:n { ##1 }
1269     }
1270   }
1271   \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1272     \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1273   }
1274 }

```

(End definition for `\stex_collect_imports:n`. This function is documented on page 71.)

`\stex_do_up_to_module:n`

```

1275 \int_new:N \l__stex_modules_group_depth_int
1276 \cs_new_protected:Nn \stex_do_up_to_module:n {
1277   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1278     #1
1279   }{
1280     #1
1281     \expandafter \tl_gset:Nn
1282     \csname l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1283     \expandafter\expandafter\expandafter\endcsname
1284     \expandafter\expandafter\expandafter { \csname
1285       l__stex_modules_aftergroup_\l_stex_current_module_str _tl\endcsname #1 }
1286     \aftergroup\__stex_modules_aftergroup_do:
1287   }
1288 }
1289 \cs_generate_variant:Nn \stex_do_up_to_module:n {x}
1290 \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1291   \stex_debug:nn{aftergroup}{\cs_meaning:c{
1292     l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1293   }}
1294   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1295     \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1296     \tl_gclear:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1297   }{
1298     \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}

```

```

1299     \aftergroup\__stex_modules_aftergroup_do:
1300   }
1301 }
1302 \cs_new_protected:Nn \stex_reset_up_to_module:n {
1303   \expandafter\let\csname l__stex_modules_aftergroup_#1_tl\endcsname\undefined
1304 }

```

(End definition for `\stex_do_up_to_module:n`. This function is documented on page 71.)

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

```

1305

```

(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.

```

1306 \str_new:N \l_stex_module_ns_str
1307 \str_new:N \l_stex_module_subpath_str
1308 \cs_new_protected:Nn \__stex_modules_compute_namespace:nN {
1309   \seq_set_eq:NN \l_tmpa_seq #2
1310   % split off file extension
1311   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str % <- filename
1312   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1313   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str % <- filename without suffixes
1314   \seq_put_right:No \l_tmpa_seq \l_tmpb_str % <- file path including name without suffixes
1315
1316   \bool_set_true:N \l_tmpa_bool
1317   \bool_while_do:Nn \l_tmpa_bool {
1318     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1319     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1320       {source} { \bool_set_false:N \l_tmpa_bool }
1321     }{}{
1322       \seq_if_empty:NT \l_tmpa_seq {
1323         \bool_set_false:N \l_tmpa_bool
1324       }
1325     }
1326   }
1327
1328   \stex_path_to_string:NN \l_tmpa_seq \l_stex_module_subpath_str
1329   % \l_tmpa_seq <- sub-path relative to archive
1330   \str_if_empty:NTF \l_stex_module_subpath_str {
1331     \str_set:Nx \l_stex_module_ns_str {#1}
1332   }{
1333     \str_set:Nx \l_stex_module_ns_str {
1334       #1/\l_stex_module_subpath_str
1335     }
1336   }
1337 }
1338
1339 \cs_new_protected:Nn \stex_modules_current_namespace: {
1340   \str_clear:N \l_stex_module_subpath_str
1341   \prop_if_exist:NTF \l_stex_current_repository_prop {
1342     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str

```

```

1343     \__stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1344   }{
1345     % split off file extension
1346     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1347     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1348     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1349     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1350     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1351     \str_set:Nx \l_stex_module_ns_str {
1352       file:/\stex_path_to_string:N \l_tmpa_seq
1353     }
1354   }
1355 }

```

(End definition for `\stex_modules_current_namespace:.` This function is documented on page [72](#).)

27.1 The smodule environment

smodule arguments:

```

1356 \keys_define:nn { stex / module } {
1357   title      .tl_set:N      = \smodulename ,
1358   type       .str_set_x:N   = \smodulename ,
1359   id         .str_set_x:N   = \smoduleid ,
1360   deprecate  .str_set_x:N   = \l_stex_module_deprecate_str ,
1361   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1362   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1363   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1364   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1365   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1366   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1367   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1368 }
1369
1370 \cs_new_protected:Nn \__stex_modules_args:n {
1371   \str_clear:N \smodulename
1372   \str_clear:N \smodulename
1373   \str_clear:N \smoduleid
1374   \str_clear:N \l_stex_module_ns_str
1375   \str_clear:N \l_stex_module_deprecate_str
1376   \str_clear:N \l_stex_module_lang_str
1377   \str_clear:N \l_stex_module_sig_str
1378   \str_clear:N \l_stex_module_creators_str
1379   \str_clear:N \l_stex_module_contributors_str
1380   \str_clear:N \l_stex_module_meta_str
1381   \str_clear:N \l_stex_module_srccite_str
1382   \keys_set:nn { stex / module } { #1 }
1383 }
1384
1385 % module parameters here? In the body?
1386

```

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

```

1387 \cs_new_protected:Nn \stex_module_setup:nn {

```

```

1388 \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1389 \str_set:Nx \l_stex_module_name_str { #2 }
1390 \__stex_modules_args:n { #1 }

```

First, we set up the name and namespace of the module.
Are we in a nested module?

```

1391 \stex_if_in_module:TF {
1392   % Nested module
1393   \prop_get:cnN {c_stex_module\l_stex_current_module_str _prop}
1394   { ns } \l_stex_module_ns_str
1395   \str_set:Nx \l_stex_module_name_str {
1396     \prop_item:cn {c_stex_module\l_stex_current_module_str _prop}
1397     { name } / \l_stex_module_name_str
1398   }
1399   \str_if_empty:NT \l_stex_module_lang_str {
1400     \str_set:Nx \l_stex_module_lang_str {
1401       \prop_item:cn {c_stex_module\l_stex_current_module_str _prop}
1402       { lang }
1403     }
1404   }
1405 }{
1406   % not nested:
1407   \str_if_empty:NT \l_stex_module_ns_str {
1408     \stex_modules_current_namespace:
1409     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1410     / {\l_stex_module_ns_str}
1411     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1412     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1413       \str_set:Nx \l_stex_module_ns_str {
1414         \stex_path_to_string:N \l_tmpa_seq
1415       }
1416     }
1417   }
1418 }

```

Next, we determine the language of the module:

```

1419 \str_if_empty:NT \l_stex_module_lang_str {
1420   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1421   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1422   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1423   \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
1424     \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
1425       \exp_args:NNNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
1426     }
1427   }
1428   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1429   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
1430     \seq_pop_right:NN \l_tmpa_seq \l_stex_module_lang_str
1431     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1432       inferred~from~file~name}
1433   }
1434 }
1435
1436 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {

```

```

1437     \exp_args:NNo \stex_set_language:Nn \l_tmpa_str \l_stex_module_lang_str
1438   }}

```

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

```

1439   \str_if_empty:NTF \l_stex_module_sig_str {
1440     \exp_args:Nnx \prop_gset_from_keyval:cn {
1441       c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1442     } {
1443       name      = \l_stex_module_name_str ,
1444       ns        = \l_stex_module_ns_str ,
1445       file      = \exp_not:o { \g_stex_currentfile_seq } ,
1446       lang      = \l_stex_module_lang_str ,
1447       sig       = \l_stex_module_sig_str ,
1448       deprecate = \l_stex_module_deprecate_str ,
1449       meta      = \l_stex_module_meta_str
1450     }
1451     \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1452     \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1453     \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _copymodules}
1454     \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1455     \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1456   \str_if_empty:NT \l_stex_module_meta_str {
1457     \str_set:Nx \l_stex_module_meta_str {
1458       \c_stex_metatheory_ns_str ? Metatheory
1459     }
1460   }
1461   \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1462     \bool_set_true:N \l_stex_in_meta_bool
1463     \exp_args:Nx \stex_add_to_current_module:n {
1464       \bool_set_true:N \l_stex_in_meta_bool
1465       \stex_activate_module:n {\l_stex_module_meta_str}
1466       \bool_set_false:N \l_stex_in_meta_bool
1467     }
1468     \stex_activate_module:n {\l_stex_module_meta_str}
1469     \bool_set_false:N \l_stex_in_meta_bool
1470   }
1471   }{
1472     \str_if_empty:NT \l_stex_module_lang_str {
1473       \msg_error:nnxx{stex}{error/siglanguage}{
1474         \l_stex_module_ns_str?\l_stex_module_name_str
1475       }\l_stex_module_sig_str}
1476     }
1477     \stex_debug:nn{modules}{Signature~\l_stex_module_sig_str~for~\l_stex_module_ns_str?\l_st
1478     \stex_if_module_exists:nTF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1479       \stex_debug:nn{modules}{(already exists)}
1480     }{
1481       \stex_debug:nn{modules}{(needs loading)}
1482       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1483       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1484       \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1485       \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex

```

```

1486 \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1487 \str_set:Nx \l_tmpa_str {
1488   \stex_path_to_string:N \l_tmpa_seq /
1489   \l_tmpa_str . \l_stex_module_sig_str .tex
1490 }
1491 \IfFileExists \l_tmpa_str {
1492   \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1493     \str_clear:N \l_stex_current_module_str
1494     \seq_clear:N \l_stex_all_modules_seq
1495     \stex_debug:nn{modules}{Loading~signature}
1496   }
1497 }{
1498   \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1499 }
1500 }
1501 \stex_if_smsmode:F {
1502   \stex_activate_module:n {
1503     \l_stex_module_ns_str ? \l_stex_module_name_str
1504   }
1505 }
1506 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1507 }
1508 \str_if_empty:NF \l_stex_module_deprecate_str {
1509   \msg_warning:nnxx{stex}{warning/deprecated}{
1510     Module~\l_stex_current_module_str
1511   }{
1512     \l_stex_module_deprecate_str
1513   }
1514 }
1515 \seq_put_right:Nx \l_stex_all_modules_seq {
1516   \l_stex_module_ns_str ? \l_stex_module_name_str
1517 }
1518 \tl_clear:c{l_stex_modules_aftergroup\l_stex_module_ns_str ? \l_stex_module_name_str _tl}
1519 }

```

(End definition for `\stex_module_setup:nn`. This function is documented on page [72](#).)

smodule The module environment.

```

\__stex_modules_begin_module: implements \begin{smodule}
1520 \cs_new_protected:Nn \__stex_modules_begin_module: {
1521   \stex_reactivate_macro:N \STEXexport
1522   \stex_reactivate_macro:N \importmodule
1523   \stex_reactivate_macro:N \symdecl
1524   \stex_reactivate_macro:N \notation
1525   \stex_reactivate_macro:N \symdef
1526
1527   \stex_debug:nn{modules}{
1528     New~module:\\
1529     Namespace:~\l_stex_module_ns_str\\
1530     Name:~\l_stex_module_name_str\\
1531     Language:~\l_stex_module_lang_str\\
1532     Signature:~\l_stex_module_sig_str\\
1533     Metatheory:~\l_stex_module_meta_str\\

```

```

1534   File:~\stex_path_to_string:N \g_stex_currentfile_seq
1535 }
1536
1537 \stex_if_do_html:T{
1538   \begin{stex_annotate_env} {theory} {
1539     \l_stex_module_ns_str ? \l_stex_module_name_str
1540   }
1541
1542   \stex_annotate_invisible:nnn{header}{} {
1543     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1544     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1545     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1546       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1547     }
1548     \str_if_empty:NF \smoduletype {
1549       \stex_annotate:nnn{type}{\smoduletype}{}
1550     }
1551   }
1552 }
1553 % TODO: Inherit metatheory for nested modules?
1554 }
1555 \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}

```

1556 \cs_new_protected:Nn \_stex_modules_end_module: {
1557   \stex_debug:nn{modules}{Closing~module~\prop_item:cn {c_stex_module_\l_stex_current_module_str}
1558   \stex_reset_up_to_module:n \l_stex_current_module_str
1559   \stex_if_smsmode:T {
1560     \stex_persist:x {
1561       \prop_set_from_keyval:cn{c_stex_module_\l_stex_current_module_str _prop}{
1562         \exp_after:wN \prop_to_keyval:N \csname c_stex_module_\l_stex_current_module_str _pr
1563       }
1564       \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _constants}{
1565         \seq_use:cn{c_stex_module_\l_stex_current_module_str _constants},
1566       }
1567       \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _imports}{
1568         \seq_use:cn{c_stex_module_\l_stex_current_module_str _imports},
1569       }
1570       \tl_set:cn {c_stex_module_\l_stex_current_module_str _code}
1571     }
1572     \exp_after:wN \let \exp_after:wN \l_tmpa_tl \csname c_stex_module_\l_stex_current_module_str
1573     \exp_after:wN \stex_persist:n \exp_after:wN { \exp_after:wN { \l_tmpa_tl } }
1574   }
1575 }

```

(End definition for _stex_modules_end_module:.)

The core environment

```

1576 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1577 \NewDocumentEnvironment { smodule } { 0 } { m } {
1578   \stex_module_setup:nn{#1}{#2}
1579   %\par
1580   \stex_if_smsmode:F{

```



```

1581 \tl_if_empty:NF \smodulename {
1582 \exp_args:No \stex_document_title:n \smodulename
1583 }
1584 \tl_clear:N \l_tmpa_tl
1585 \clist_map_inline:Nn \smodulename {
1586 \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1587 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1588 }
1589 }
1590 \tl_if_empty:NTF \l_tmpa_tl {
1591 \__stex_modules_smodule_start:
1592 }{
1593 \l_tmpa_tl
1594 }
1595 }
1596 \__stex_modules_begin_module:
1597 \str_if_empty:NF \smoduleid {
1598 \stex_ref_new_doc_target:n \smoduleid
1599 }
1600 \stex_smsmode_do:
1601 } {
1602 \__stex_modules_end_module:
1603 \stex_if_smsmode:F {
1604 \end{stex_annotate_env}
1605 \clist_set:No \l_tmpa_clist \smodulename
1606 \tl_clear:N \l_tmpa_tl
1607 \clist_map_inline:Nn \l_tmpa_clist {
1608 \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1609 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1610 }
1611 }
1612 \tl_if_empty:NTF \l_tmpa_tl {
1613 \__stex_modules_smodule_end:
1614 }{
1615 \l_tmpa_tl
1616 }
1617 }
1618 }

```

\stexpatchmodule

```

1619 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1620 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1621
1622 \newcommand\stexpatchmodule[3] [] {
1623 \str_set:Nx \l_tmpa_str{ #1 }
1624 \str_if_empty:NTF \l_tmpa_str {
1625 \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1626 \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1627 }{
1628 \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1629 \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1630 }
1631 }

```

(End definition for \stexpatchmodule. This function is documented on page 72.)

27.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n 1632 \NewDocumentCommand \STEXModule { m } {
1633   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1634   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1635   \tl_set:Nn \l_tmpa_tl {
1636     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1637   }
1638   \seq_map_inline:Nn \l_stex_all_modules_seq {
1639     \str_set:Nn \l_tmpb_str { ##1 }
1640     \str_if_eq:eeT { \l_tmpa_str } {
1641       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1642     } {
1643       \seq_map_break:n {
1644         \tl_set:Nn \l_tmpa_tl {
1645           \stex_invoke_module:n { ##1 }
1646         }
1647       }
1648     }
1649   }
1650   \l_tmpa_tl
1651 }
1652
1653 \cs_new_protected:Nn \stex_invoke_module:n {
1654   \stex_debug:nn{modules}{Invoking~module~#1}
1655   \peek_charcode_remove:NTF ! {
1656     \__stex_modules_invoke_uri:nN { #1 }
1657   } {
1658     \peek_charcode_remove:NTF ? {
1659       \__stex_modules_invoke_symbol:nn { #1 }
1660     } {
1661       \msg_error:nnx{stex}{error/syntax}{
1662         ?~or~!~expected~after~
1663         \c_backslash_str STEXModule{#1}
1664       }
1665     }
1666   }
1667 }
1668
1669 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1670   \str_set:Nn #2 { #1 }
1671 }
1672
1673 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1674   \stex_invoke_symbol:n{#1?#2}
1675 }

```

(End definition for `\STEXModule` and `\stex_invoke_module:n`. These functions are documented on page 72.)

```

\stex_activate_module:n
1676 \bool_new:N \l_stex_in_meta_bool
1677 \bool_set_false:N \l_stex_in_meta_bool

```

```

1678 \cs_new_protected:Nn \stex_activate_module:n {
1679   \stex_debug:nn{modules}{Activating~module~#1}
1680   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1681     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1682     \use:c{ c_stex_module_#1_code }
1683   }
1684 }

```

(End definition for \stex_activate_module:n. This function is documented on page 73.)

```

1685 \endpackage

```

Chapter 28

STEX -Module Inheritance Implementation

```
1686 <*package>
1687
1688 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1689
```

28.1 SMS Mode

```
1690 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1691 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1692 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1693 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1694
1695 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1696   \makeatletter
1697   \makeatother
1698   \ExplSyntaxOn
1699   \ExplSyntaxOff
1700   \rustexBREAK
1701 }
1702
1703 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1704   \symdef
1705   \importmodule
1706   \notation
1707   \symdecl
1708   \STEXexport
1709   \inlineass
1710   \inlinedef
1711   \inlineex
1712   \endinput
1713   \setnotation
```

```

1714 \copynotation
1715 \assign
1716 \renamedekl
1717 \donotcopy
1718 \instantiate
1719 }
1720
1721 \exp_args:Nn \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1722   \tl_to_str:n {
1723     smodule,
1724     copymodule,
1725     interpretmodule,
1726     sdefinition,
1727     sexample,
1728     sassertion,
1729     sparagraph,
1730     mathstructure
1731   }
1732 }

```

(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 74.)

```

\stex_if_smsmode_p:
\stex_if_smsmode:TF
1733 \bool_new:N \g__stex_smsmode_bool
1734 \bool_set_false:N \g__stex_smsmode_bool
1735 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1736   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1737 }

```

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

```

\__stex_smsmode_in_smsmode:nn
1738 \cs_new_protected:Nn \__stex_smsmode_in_smsmode:nn { \stex_suppress_html:n {
1739   \vbox_set:Nn \l_tmpa_box {
1740     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1741     \bool_gset_true:N \g__stex_smsmode_bool
1742     #2
1743     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1744   }
1745   \box_clear:N \l_tmpa_box
1746 } }

```

(End definition for `__stex_smsmode_in_smsmode:nn`.)

```

\stex_file_in_smsmode:nn
1747 \quark_new:N \q__stex_smsmode_break
1748
1749 \NewDocumentCommand \__stex_smsmode_importmodule: { 0{} m } {
1750   \seq_gput_right:Nn \l__stex_smsmode_importmodules_seq { {#1}{#2}}
1751   \stex_smsmode_do:
1752 }
1753
1754 \cs_new_protected:Nn \__stex_smsmode_module:nn {
1755   \__stex_modules_args:n{#1}

```

```

1756 \stex_if_in_module:F {
1757   \str_if_empty:NF \l_stex_module_sig_str {
1758     \stex_modules_current_namespace:
1759     \str_set:Nx \l_stex_module_name_str { #2 }
1760     \stex_if_module_exists:nF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1761       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1762       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1763       \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1764       \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1765       \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1766       \str_set:Nx \l_tmpa_str {
1767         \stex_path_to_string:N \l_tmpa_seq /
1768         \l_tmpa_str . \l_stex_module_sig_str .tex
1769       }
1770       \IfFileExists \l_tmpa_str {
1771         \exp_args:NNx \seq_gput_right:Nn \l__stex_smsmode_sigmodules_seq \l_tmpa_str
1772       }{
1773         \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1774       }
1775     }
1776   }
1777 }
1778 }
1779
1780 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1781   \stex_filestack_push:n{#1}
1782   \seq_gclear:N \l__stex_smsmode_importmodules_seq
1783   \seq_gclear:N \l__stex_smsmode_sigmodules_seq
1784   % ----- new -----
1785   \__stex_smsmode_in_smsmode:nn{#1}{
1786     \let\importmodule\__stex_smsmode_importmodule:
1787     \let\stex_module_setup:nn\__stex_smsmode_module:nn
1788     \let\__stex_modules_begin_module:\relax
1789     \let\__stex_modules_end_module:\relax
1790     \seq_clear:N \g_stex_smsmode_allowedenvs_seq
1791     \exp_args:NNx \seq_put_right:Nn \g_stex_smsmode_allowedenvs_seq {\tl_to_str:n{smodule}}
1792     \tl_clear:N \g_stex_smsmode_allowedmacros_tl
1793     \tl_clear:N \g_stex_smsmode_allowedmacros_escape_tl
1794     \tl_put_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\importmodule}
1795     \everyeof{\q__stex_smsmode_break\noexpand}
1796     \expandafter\expandafter\expandafter
1797     \stex_smsmode_do:
1798     \csname @ @ input\endcsname "#1"\relax
1799
1800     \seq_map_inline:Nn \l__stex_smsmode_sigmodules_seq {
1801       \stex_filestack_push:n{##1}
1802       \expandafter\expandafter\expandafter
1803       \stex_smsmode_do:
1804       \csname @ @ input\endcsname "##1"\relax
1805       \stex_filestack_pop:
1806     }
1807   }
1808   % ----- new -----
1809   \__stex_smsmode_in_smsmode:nn{#1} {

```

```

1810 #2
1811 % ----- new -----
1812 \begingroup
1813 %\stex_debug:nn{smsmode}{Here:~\seq_use:Nn\l__stex_smsmode_importmodules_seq, }
1814 \seq_map_inline:Nn \l__stex_smsmode_importmodules_seq {
1815   \stex_import_module_uri:nn ##1
1816   \stex_import_require_module:nnnn
1817   \l_stex_import_ns_str
1818   \l_stex_import_archive_str
1819   \l_stex_import_path_str
1820   \l_stex_import_name_str
1821 }
1822 \endgroup
1823 \stex_debug:nn{smsmode}{Actually~loading~file~#1}
1824 % ----- new -----
1825 \everyeof{\q__stex_smsmode_break\noexpand}
1826 \expandafter\expandafter\expandafter
1827 \stex_smsmode_do:
1828 \csname @ @ input\endcsname "#1"\relax
1829 }
1830 \stex_filestack_pop:
1831 }

```

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

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

```

1832 \cs_new_protected:Npn \stex_smsmode_do: {
1833   \stex_if_smsmode:T {
1834     \__stex_smsmode_do:w
1835   }
1836 }
1837 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1838   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
1839     \expandafter\if\expandafter\relax\noexpand#1
1840     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1841     \else\expandafter\__stex_smsmode_do:w\fi
1842   }{
1843     \__stex_smsmode_do:w %#1
1844   }
1845 }
1846 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1847   \cs_if_eq:NNTF #1 \q__stex_smsmode_break {
1848     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1849       #1\__stex_smsmode_do:w
1850     }{
1851       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1852         #1
1853       }{
1854         \cs_if_eq:NNTF \begin #1 {
1855           \__stex_smsmode_check_begin:n
1856         }{
1857           \cs_if_eq:NNTF \end #1 {
1858             \__stex_smsmode_check_end:n

```

```

1859         }{
1860         \__stex_smsmode_do:w
1861         }
1862     }
1863 }
1864 }
1865 }
1866 }
1867
1868 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1869 \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1870 \begin{#1}
1871 }{
1872 \__stex_smsmode_do:w
1873 }
1874 }
1875 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
1876 \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1877 \end{#1}\__stex_smsmode_do:w
1878 }{
1879 \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1880 }
1881 }

```

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

28.2 Inheritance

```

1882 <@@=stex_importmodule>

\stex_import_module_uri:nn

1883 \cs_new_protected:Nn \stex_import_module_uri:nn {
1884 \str_set:Nx \l_stex_import_archive_str { #1 }
1885 \str_set:Nn \l_stex_import_path_str { #2 }
1886
1887 \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1888 \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1889 \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1890
1891 \stex_modules_current_namespace:
1892 \bool_lazy_all:nTF {
1893   {\str_if_empty_p:N \l_stex_import_archive_str}
1894   {\str_if_empty_p:N \l_stex_import_path_str}
1895   {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1896 }{
1897   \str_set_eq:NN \l_stex_import_path_str \l_stex_module_subpath_str
1898   \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1899 }{
1900   \str_if_empty:NT \l_stex_import_archive_str {
1901     \prop_if_exist:NT \l_stex_current_repository_prop {
1902       \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1903     }
1904   }
1905   \str_if_empty:NTF \l_stex_import_archive_str {

```



```

1906     \str_if_empty:NF \l_stex_import_path_str {
1907         \str_set:Nx \l_stex_import_ns_str {
1908             \l_stex_module_ns_str / \l_stex_import_path_str
1909         }
1910     }
1911 }{
1912     \stex_require_repository:n \l_stex_import_archive_str
1913     \prop_get:cnN { c_stex_mathhub_\l_stex_import_archive_str _manifest_prop } { ns }
1914     \l_stex_import_ns_str
1915     \str_if_empty:NF \l_stex_import_path_str {
1916         \str_set:Nx \l_stex_import_ns_str {
1917             \l_stex_import_ns_str / \l_stex_import_path_str
1918         }
1919     }
1920 }
1921 }
1922 }

```

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

```

\l_stex_import_name_str Store the return values of \stex_import_module_uri:nn.
\l_stex_import_archive_str
\l_stex_import_path_str
\l_stex_import_ns_str
1923 \str_new:N \l_stex_import_name_str
1924 \str_new:N \l_stex_import_archive_str
1925 \str_new:N \l_stex_import_path_str
1926 \str_new:N \l_stex_import_ns_str

```

(End definition for `\l_stex_import_name_str` and others. These variables are documented on page 76.)

```

\stex_import_require_module:nnnnn {<ns>} {<archive-ID>} {<path>} {<name>}
1927 \cs_new_protected:Nn \stex_import_require_module:nnnnn {
1928     \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1929
1930         %\stex_debug:nn{requiremodule}{Here:\\~1::~~1\\~2::~~2\\~3::~~3\\~4::~~4}
1931
1932         \exp_args:NNxx \seq_set_split:Nnn \l_tmpa_seq {\tl_to_str:n{/}} {#4}
1933         \seq_get_left:NN \l_tmpa_seq \l_tmpc_str
1934
1935         %\stex_debug:nn{requiremodule}{Top~module:\l_tmpc_str}
1936
1937         % archive
1938         \str_set:Nx \l_tmpa_str { #2 }
1939         \str_if_empty:NTF \l_tmpa_str {
1940             \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1941         } {
1942             \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1943             \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1944             \seq_put_right:Nn \l_tmpa_seq { source }
1945         }
1946
1947         % path
1948         \str_set:Nx \l_tmpb_str { #3 }
1949         \str_if_empty:NTF \l_tmpb_str {
1950             \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / \l_tmpc_str }
1951

```

```

1952 \ltx@ifpackageloaded{babel} {
1953   \exp_args:NnX \prop_get:NnNF \c_stex_language_abbrevs_prop
1954     { \language } \l_tmpb_str {
1955       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1956     }
1957 } {
1958   \str_clear:N \l_tmpb_str
1959 }
1960
1961 %\stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1962 \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1963   \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1964 }{
1965   %\stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1966   \IfFileExists{ \l_tmpa_str.tex }{
1967     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1968   }{
1969     % try english as default
1970     %\stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1971     \IfFileExists{ \l_tmpa_str.en.tex }{
1972       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1973     }{
1974       \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1975     }
1976   }
1977 }
1978
1979 } {
1980   \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1981   \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1982
1983   \ltx@ifpackageloaded{babel} {
1984     \exp_args:NnX \prop_get:NnNF \c_stex_language_abbrevs_prop
1985       { \language } \l_tmpb_str {
1986         \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1987       }
1988   } {
1989     \str_clear:N \l_tmpb_str
1990   }
1991
1992   \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1993
1994   %\stex_debug:nn{modules}{Checking~\l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex}
1995   \IfFileExists{ \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }{
1996     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }
1997   }{
1998     %\stex_debug:nn{modules}{Checking~\l_tmpa_str/\l_tmpc_str.tex}
1999     \IfFileExists{ \l_tmpa_str/\l_tmpc_str.tex }{
2000       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.tex }
2001     }{
2002       % try english as default
2003       %\stex_debug:nn{modules}{Checking~\l_tmpa_str/\l_tmpc_str.en.tex}
2004       \IfFileExists{ \l_tmpa_str/\l_tmpc_str.en.tex }{
2005         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.en.tex }

```

```

2006     }{
2007     %\stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
2008     \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2009         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2010     }{
2011         %\stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
2012         \IfFileExists{ \l_tmpa_str.tex }{
2013             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2014         }{
2015             % try english as default
2016             %\stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
2017             \IfFileExists{ \l_tmpa_str.en.tex }{
2018                 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2019             }{
2020                 \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2021             }
2022         }
2023     }
2024 }
2025 }
2026 }
2027 }
2028
2029 \str_if_eq:eeF{\g__stex_importmodule_file_str}{\seq_use:Nn \g_stex_currentfile_seq /}{
2030     \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
2031         \seq_clear:N \l_stex_all_modules_seq
2032         \str_clear:N \l_stex_current_module_str
2033         \str_set:Nx \l_tmpb_str { #2 }
2034         \str_if_empty:NF \l_tmpb_str {
2035             \stex_set_current_repository:n { #2 }
2036         }
2037         \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
2038     }
2039
2040     \stex_if_module_exists:nF { #1 ? #4 } {
2041         \msg_error:nnx{stex}{error/unknownmodule}{
2042             #1?#4~(in~file~\g__stex_importmodule_file_str)
2043         }
2044     }
2045 }
2046
2047 }
2048 \stex_activate_module:n { #1 ? #4 }
2049 }

```

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

`\importmodule`

```

2050 \NewDocumentCommand \importmodule { 0{} m } {
2051     \stex_import_module_uri:nn { #1 } { #2 }
2052     \stex_debug:nn{modules}{Importing~module:~
2053         \l_stex_import_ns_str ? \l_stex_import_name_str
2054     }
2055     \stex_import_require_module:nnnn

```

```

2056 { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2057 { \l_stex_import_path_str } { \l_stex_import_name_str }
2058 \stex_if_smsmode:F {
2059   \stex_annotate_invisible:nnn
2060   {import} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
2061 }
2062 \exp_args:Nx \stex_add_to_current_module:n {
2063   \stex_import_require_module:nnnn
2064   { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2065   { \l_stex_import_path_str } { \l_stex_import_name_str }
2066 }
2067 \exp_args:Nx \stex_add_import_to_current_module:n {
2068   \l_stex_import_ns_str ? \l_stex_import_name_str
2069 }
2070 \stex_smsmode_do:
2071 \ignorespacesandpars
2072 }
2073 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

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

`\usemodule`

```

2074 \NewDocumentCommand \usemodule { 0{} m } {
2075   \stex_if_smsmode:F {
2076     \stex_import_module_uri:nn { #1 } { #2 }
2077     \stex_import_require_module:nnnn
2078     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2079     { \l_stex_import_path_str } { \l_stex_import_name_str }
2080     \stex_annotate_invisible:nnn
2081     {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
2082   }
2083   \stex_smsmode_do:
2084   \ignorespacesandpars
2085 }

```

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

```

2086 \cs_new_protected:Nn \stex_csl_to_imports:Nn {
2087   \tl_if_empty:nF{#2}{
2088     \clist_set:Nn \l_tmpa_clist {#2}
2089     \clist_map_inline:Nn \l_tmpa_clist {
2090       \tl_if_head_eq_charcode:nNTF {##1}[{
2091         #1 ##1
2092       }{
2093         #1{##1}
2094       }
2095     }
2096   }
2097 }
2098 \cs_generate_variant:Nn \stex_csl_to_imports:Nn {No}
2099
2100
2101 </package>

```

Chapter 29

STEX -Symbols Implementation

```
2102 <*package>
2103
2104 %%%%%%%%%%%%% symbols.dtx %%%%%%%%%%%%%
2105
      Warnings and error messages
2106 \msg_new:nnn{stex}{error/wrongargs}{
2107   args~value~in~symbol~declaration~for~#1~
2108   needs~to~be~i,~a,~b~or~B,~but~#2~given
2109 }
2110 \msg_new:nnn{stex}{error/unknownsymbol}{
2111   No~symbol~#1~found!
2112 }
2113 \msg_new:nnn{stex}{error/seqlength}{
2114   Expected~#1~arguments;~got~#2!
2115 }
2116 \msg_new:nnn{stex}{error/unknownnotation}{
2117   Unknown~notation~#1~for~#2!
2118 }
```

29.1 Symbol Declarations

```
2119 <@@=stex_symdecl>
\stex_all_symbols:n Map over all available symbols
2120 \cs_new_protected:Nn \stex_all_symbols:n {
2121   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
2122   \seq_map_inline:Nn \l_stex_all_modules_seq {
2123     \seq_map_inline:cn{c_stex_module_##1_constants}{
2124       \__stex_symdecl_all_symbols_cs{##1?####1}
2125     }
2126   }
2127 }
```

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

\STEXsymbol

```
2128 \NewDocumentCommand \STEXsymbol { m } {
2129   \stex_get_symbol:n { #1 }
2130   \exp_args:No
2131   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
2132 }
```

(End definition for \STEXsymbol. This function is documented on page 79.)

symdecl arguments:

```
2133 \keys_define:nn { stex / symdecl } {
2134   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2135   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2136   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2137   type      .tl_set:N = \l_stex_symdecl_type_tl ,
2138   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,
2139   align     .str_set:N = \l_stex_symdecl_align_str , % TODO(?)
2140   gfc       .str_set:N = \l_stex_symdecl_gfc_str , % TODO(?)
2141   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
2142   def       .tl_set:N = \l_stex_symdecl_definiens_tl ,
2143   reorder   .str_set_x:N = \l_stex_symdecl_reorder_str ,
2144   assoc     .choices:nn =
2145     {bin,binl,binr,pre,conj,pwconj}
2146     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}
2147 }
2148
2149 \bool_new:N \l_stex_symdecl_make_macro_bool
2150
2151 \cs_new_protected:Nn \__stex_symdecl_args:n {
2152   \str_clear:N \l_stex_symdecl_name_str
2153   \str_clear:N \l_stex_symdecl_args_str
2154   \str_clear:N \l_stex_symdecl_deprecate_str
2155   \str_clear:N \l_stex_symdecl_reorder_str
2156   \str_clear:N \l_stex_symdecl_assoctype_str
2157   \bool_set_false:N \l_stex_symdecl_local_bool
2158   \tl_clear:N \l_stex_symdecl_type_tl
2159   \tl_clear:N \l_stex_symdecl_definiens_tl
2160
2161   \keys_set:nn { stex / symdecl } { #1 }
2162 }
```

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

```
2163
2164 \NewDocumentCommand \symdecl { s m O{} } {
2165   \__stex_symdecl_args:n { #3 }
2166   \IfBooleanTF #1 {
2167     \bool_set_false:N \l_stex_symdecl_make_macro_bool
2168   } {
2169     \bool_set_true:N \l_stex_symdecl_make_macro_bool
2170   }
2171   \stex_symdecl_do:n { #2 }
2172   \stex_smsmode_do:
2173 }
```

```

2174
2175 \cs_new_protected:Nn \stex_symdecl_do:nn {
2176   \__stex_symdecl_args:n{#1}
2177   \bool_set_false:N \l_stex_symdecl_make_macro_bool
2178   \stex_symdecl_do:n{#2}
2179 }
2180
2181 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

\stex_symdecl_do:n

```

2182 \cs_new_protected:Nn \stex_symdecl_do:n {
2183   \stex_if_in_module:F {
2184     % TODO throw error? some default namespace?
2185   }
2186
2187   \str_if_empty:NT \l_stex_symdecl_name_str {
2188     \str_set:Nx \l_stex_symdecl_name_str { #1 }
2189   }
2190
2191   \prop_if_exist:cT { l_stex_symdecl_
2192     \l_stex_current_module_str ?
2193     \l_stex_symdecl_name_str
2194     _prop
2195   }{
2196     % TODO throw error (beware of circular dependencies)
2197   }
2198
2199   \prop_clear:N \l_tmpa_prop
2200   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
2201   \seq_clear:N \l_tmpa_seq
2202   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
2203   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
2204
2205   \str_if_empty:NT \l_stex_symdecl_deprecate_str {
2206     \str_if_empty:NF \l_stex_module_deprecate_str {
2207       \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
2208     }
2209   }
2210   \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
2211
2212   \exp_args:No \stex_add_constant_to_current_module:n {
2213     \l_stex_symdecl_name_str
2214   }
2215
2216   % arity/args
2217   \int_zero:N \l_tmpb_int
2218
2219   \bool_set_true:N \l_tmpa_bool
2220   \str_map_inline:Nn \l_stex_symdecl_args_str {
2221     \token_case_meaning:NnF ##1 {
2222       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2223       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }

```

```

2224     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2225     {\tl_to_str:n a} {
2226         \bool_set_false:N \l_tmpa_bool
2227         \int_incr:N \l_tmpb_int
2228     }
2229     {\tl_to_str:n B} {
2230         \bool_set_false:N \l_tmpa_bool
2231         \int_incr:N \l_tmpb_int
2232     }
2233 }{
2234     \msg_error:nnxx{stex}{error/wrongargs}{
2235         \l_stex_current_module_str ?
2236         \l_stex_symdecl_name_str
2237     }{##1}
2238 }
2239 }
2240 \bool_if:NTF \l_tmpa_bool {
2241     % possibly numeric
2242     \str_if_empty:NTF \l_stex_symdecl_args_str {
2243         \prop_put:Nnn \l_tmpa_prop { args } {}
2244         \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2245     }{
2246         \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
2247         \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2248         \str_clear:N \l_tmpa_str
2249         \int_step_inline:nn \l_tmpa_int {
2250             \str_put_right:Nn \l_tmpa_str i
2251         }
2252         \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
2253     }
2254 } {
2255     \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
2256     \prop_put:Nnx \l_tmpa_prop { arity }
2257     { \str_count:N \l_stex_symdecl_args_str }
2258 }
2259 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2260
2261 \tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
2262     \prop_put:Nnx \l_tmpa_prop { defined }{ false }
2263 }{
2264     \prop_put:Nnx \l_tmpa_prop { defined }{ true }
2265 }
2266
2267 % semantic macro
2268
2269 \bool_if:NT \l_stex_symdecl_make_macro_bool {
2270     \exp_args:Nx \stex_do_up_to_module:n {
2271         \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2272             \l_stex_current_module_str ? \l_stex_symdecl_name_str
2273         }}
2274     }
2275 }
2276
2277 \stex_debug:nn{symbols}{New~symbol:~

```



```

2278 \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2279 Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2280 Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
2281 Definiens:~\exp_not:o {\l_stex_symdecl_definiens_tl}
2282 }
2283
2284 % circular dependencies require this:
2285 \stex_if_do_html:T {
2286   \stex_annotate_invisible:nnn {symdecl} {
2287     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2288   } {
2289     \tl_if_empty:NF \l_stex_symdecl_type_tl {
2290       \stex_annotate_invisible:nnn{type}{}{\l_stex_symdecl_type_tl$}
2291     }
2292     \stex_annotate_invisible:nnn{args}{}{
2293       \prop_item:Nn \l_tmpa_prop { args }
2294     }
2295     \stex_annotate_invisible:nnn{macroname}{#1}{}
2296     \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2297       \stex_annotate_invisible:nnn{definiens}{}
2298         {\l_stex_symdecl_definiens_tl$}
2299     }
2300     \str_if_empty:NF \l_stex_symdecl_assoc_type_str {
2301       \stex_annotate_invisible:nnn{assoc_type}{\l_stex_symdecl_assoc_type_str}{}
2302     }
2303     \str_if_empty:NF \l_stex_symdecl_reorder_str {
2304       \stex_annotate_invisible:nnn{reorderargs}{\l_stex_symdecl_reorder_str}{}
2305     }
2306   }
2307 }
2308 \prop_if_exist:cF {
2309   \l_stex_symdecl_
2310   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2311   _prop
2312 } {
2313   \bool_if:NTF \l_stex_symdecl_local_bool \stex_do_up_to_module:x \stex_execute_in_module:
2314     \__stex_symdecl_restore_symbol:nnnnnnn
2315       {\l_stex_symdecl_name_str}
2316       { \prop_item:Nn \l_tmpa_prop {args} }
2317       { \prop_item:Nn \l_tmpa_prop {arity} }
2318       { \prop_item:Nn \l_tmpa_prop {assoc} }
2319       { \prop_item:Nn \l_tmpa_prop {defined} }
2320       {\bool_if:NT \l_stex_symdecl_make_macro_bool {#1} }
2321       {\l_stex_current_module_str}
2322   }
2323 }
2324 }
2325 \cs_new_protected:Nn \__stex_symdecl_restore_symbol:nnnnnnn {
2326   \prop_clear:N \l_tmpa_prop
2327   \prop_put:Nnn \l_tmpa_prop { module } { #7 }
2328   \prop_put:Nnn \l_tmpa_prop { name } { #1 }
2329   \prop_put:Nnn \l_tmpa_prop { args } { #2 }
2330   \prop_put:Nnn \l_tmpa_prop { arity } { #3 }
2331   \prop_put:Nnn \l_tmpa_prop { assoc } { #4 }

```

```

2332 \prop_put:Nnn \l_tmpa_prop { defined } { #5 }
2333 \tl_if_empty:nF{#6}{
2334   \tl_set:cx{#6}{\stex_invoke_symbol:n{\detokenize{#7 ? #1}}}
2335 }
2336 \prop_set_eq:cN{l_stex_symdecl_ \detokenize{#7 ? #1} _prop}\l_tmpa_prop
2337 \seq_clear:c{l_stex_symdecl_ \detokenize{#7 ? #1} _notations}
2338 }

```

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

`\stex_get_symbol:n`

```

2339 \str_new:N \l_stex_get_symbol_uri_str
2340
2341 \cs_new_protected:Nn \stex_get_symbol:n {
2342   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2343     \tl_set:Nn \l_tmpa_tl { #1 }
2344     \__stex_symdecl_get_symbol_from_cs:
2345   }{
2346     % argument is a string
2347     % is it a command name?
2348     \cs_if_exist:cTF { #1 }{
2349       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2350       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2351       \str_if_empty:NTF \l_tmpa_str {
2352         \exp_args:Nx \cs_if_eq:NNTF {
2353           \tl_head:N \l_tmpa_tl
2354         } \stex_invoke_symbol:n {
2355           \__stex_symdecl_get_symbol_from_cs:
2356         }{
2357           \__stex_symdecl_get_symbol_from_string:n { #1 }
2358         }
2359       } {
2360         \__stex_symdecl_get_symbol_from_string:n { #1 }
2361       }
2362     }{
2363       % argument is not a command name
2364       \__stex_symdecl_get_symbol_from_string:n { #1 }
2365       % \l_stex_all_symbols_seq
2366     }
2367   }
2368   \str_if_eq:eeF {
2369     \prop_item:cn {
2370       l_stex_symdecl_\l_stex_get_symbol_uri_str _prop
2371     }{ deprecate }
2372   }{
2373     \msg_warning:nxxx{stex}{warning/deprecated}{
2374       Symbol~\l_stex_get_symbol_uri_str
2375     }{
2376       \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{ deprecate }
2377     }
2378   }
2379 }
2380
2381 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {

```

```

2382 \tl_set:Nn \l_tmpa_tl {
2383   \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2384 }
2385 \str_set:Nn \l_tmpa_str { #1 }
2386
2387 %\int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2388
2389 \str_if_in:NnTF \l_tmpa_str ? {
2390   \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
2391   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
2392   \str_set:Nx \l_tmpb_str {\seq_use:Nn \l_tmpa_seq ?}
2393 }{
2394   \str_clear:N \l_tmpb_str
2395 }
2396 \str_if_empty:NNTF \l_tmpb_str {
2397   \seq_map_inline:Nn \l_stex_all_modules_seq {
2398     \seq_map_inline:cn{c_stex_module_###1_constants}{
2399       \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2400         \seq_map_break:n{\seq_map_break:n{
2401           \tl_set:Nn \l_tmpa_tl {
2402             \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2403           }
2404         }}
2405       }
2406     }
2407   }
2408 }{
2409   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpb_str }
2410   \seq_map_inline:Nn \l_stex_all_modules_seq {
2411     \str_if_eq:eeT{ \l_tmpb_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2412       \seq_map_inline:cn{c_stex_module_###1_constants}{
2413         \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2414           \seq_map_break:n{\seq_map_break:n{
2415             \tl_set:Nn \l_tmpa_tl {
2416               \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2417             }
2418           }}
2419         }
2420       }
2421     }
2422   }
2423 }
2424
2425 \l_tmpa_tl
2426 }
2427
2428 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
2429   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2430   { \tl_tail:N \l_tmpa_tl }
2431   \tl_if_single:NNTF \l_tmpa_tl {
2432     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2433       \exp_after:wN \str_set:Nn \exp_after:wN
2434       \l_stex_get_symbol_uri_str \l_tmpa_tl
2435     }{

```

```

2436     % TODO
2437     % tail is not a single group
2438   }
2439 }{
2440   % TODO
2441   % tail is not a single group
2442 }
2443 }

```

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

29.2 Notations

```

2444 <@@=stex_notation>

notation arguments:
2445 \keys_define:nn { stex / notation } {
2446   % lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2447   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2448   prec     .str_set_x:N = \l__stex_notation_prec_str ,
2449   op       .tl_set:N = \l__stex_notation_op_tl ,
2450   primary  .bool_set:N = \l__stex_notation_primary_bool ,
2451   primary  .default:n = {true} ,
2452   unknown  .code:n = \str_set:Nx
2453             \l__stex_notation_variant_str \l_keys_key_str
2454 }
2455
2456 \cs_new_protected:Nn \stex_notation_args:n {
2457   % \str_clear:N \l__stex_notation_lang_str
2458   \str_clear:N \l__stex_notation_variant_str
2459   \str_clear:N \l__stex_notation_prec_str
2460   \tl_clear:N \l__stex_notation_op_tl
2461   \bool_set_false:N \l__stex_notation_primary_bool
2462
2463   \keys_set:nn { stex / notation } { #1 }
2464 }

\notation

2465 \NewDocumentCommand \notation { s m O{} } {
2466   \stex_notation_args:n { #3 }
2467   \tl_clear:N \l_stex_symdecl_definiens_tl
2468   \stex_get_symbol:n { #2 }
2469   \tl_set:Nn \l_stex_notation_after_do_tl {
2470     \__stex_notation_final:
2471     \IfBooleanTF#1{
2472       \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2473     }{}
2474     \stex_smsmode_do:\ignorespacesandpars
2475   }
2476   \stex_notation_do:nnnnn
2477   { \prop_item:cn {\l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } }
2478   { \prop_item:cn { \l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } }
2479   { \l__stex_notation_variant_str }
2480   { \l__stex_notation_prec_str }

```

```

2481 }
2482 \stex_deactivate_macro:Nn \notation {module-environments}

```

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

\stex_notation_do:nnnnn

```

2483 \seq_new:N \l__stex_notation_precedences_seq
2484 \tl_new:N \l__stex_notation_opprec_tl
2485 \int_new:N \l__stex_notation_currarg_int
2486 \tl_new:N \stex_symbol_after_invokation_tl
2487
2488 \cs_new_protected:Nn \stex_notation_do:nnnnn {
2489   \let\l_stex_current_symbol_str\relax
2490   \seq_clear:N \l__stex_notation_precedences_seq
2491   \tl_clear:N \l__stex_notation_opprec_tl
2492   \str_set:Nx \l__stex_notation_args_str { #1 }
2493   \str_set:Nx \l__stex_notation_arity_str { #2 }
2494   \str_set:Nx \l__stex_notation_suffix_str { #3 }
2495   \str_set:Nx \l__stex_notation_prec_str { #4 }
2496
2497   % precedences
2498   \str_if_empty:NTF \l__stex_notation_prec_str {
2499     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2500       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2501     }{
2502       \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2503     }
2504   } {
2505     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2506       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2507       \int_step_inline:nn { \l__stex_notation_arity_str } {
2508         \exp_args:NNo
2509         \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2510       }
2511     }{
2512       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2513       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2514         \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2515         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2516           \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2517             \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2518           \seq_map_inline:Nn \l_tmpa_seq {
2519             \seq_put_right:Nn \l_tmpb_seq { ##1 }
2520           }
2521         }
2522       }{
2523         \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2524           \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2525         }{
2526           \tl_set:No \l__stex_notation_opprec_tl { 0 }
2527         }
2528       }
2529     }
2530   }

```

```

2531 \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2532 \int_step_inline:nn { \l__stex_notation_arity_str } {
2533   \seq_pop_left:NNTF \l_tmpa_seq \l_tmpb_str {
2534     \exp_args:NNo
2535     \seq_put_right:No \l__stex_notation_precedences_seq {
2536       \l__stex_notation_opprec_tl
2537     }
2538   }
2539 }
2540 }
2541 \tl_clear:N \l_stex_notation_dummyargs_tl
2542
2543 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2544   \exp_args:NNe
2545   \cs_set:Npn \l_stex_notation_macrocode_cs {
2546     \_stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2547     { \l__stex_notation_suffix_str }
2548     { \l__stex_notation_opprec_tl }
2549     { \exp_not:n { #5 } }
2550   }
2551   \l_stex_notation_after_do_tl
2552 }{
2553   \str_if_in:NnTF \l__stex_notation_args_str b {
2554     \exp_args:Nne \use:nn
2555     {
2556       \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2557       \cs_set:Npn \l__stex_notation_arity_str } { {
2558         \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2559         { \l__stex_notation_suffix_str }
2560         { \l__stex_notation_opprec_tl }
2561         { \exp_not:n { #5 } }
2562       }}
2563   }{
2564     \str_if_in:NnTF \l__stex_notation_args_str B {
2565       \exp_args:Nne \use:nn
2566       {
2567         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2568         \cs_set:Npn \l__stex_notation_arity_str } { {
2569           \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2570           { \l__stex_notation_suffix_str }
2571           { \l__stex_notation_opprec_tl }
2572           { \exp_not:n { #5 } }
2573         } }
2574     }{
2575       \exp_args:Nne \use:nn
2576       {
2577         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2578         \cs_set:Npn \l__stex_notation_arity_str } { {
2579           \_stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2580           { \l__stex_notation_suffix_str }
2581           { \l__stex_notation_opprec_tl }
2582           { \exp_not:n { #5 } }
2583         } }
2584     }

```

```

2585     }
2586
2587     \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2588     \int_zero:N \l__stex_notation_currarg_int
2589     \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2590     \__stex_notation_arguments:
2591   }
2592 }

```

(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

```

2593 \cs_new_protected:Nn \__stex_notation_arguments: {
2594   \int_incr:N \l__stex_notation_currarg_int
2595   \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2596     \l_stex_notation_after_do_tl
2597   }{
2598     \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2599     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2600     \str_if_eq:VnTF \l_tmpa_str a {
2601       \__stex_notation_argument_assoc:nn{a}
2602     }{
2603       \str_if_eq:VnTF \l_tmpa_str B {
2604         \__stex_notation_argument_assoc:nn{B}
2605       }{
2606         \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpb_str
2607         \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2608           { \stex_term_math_arg:nnn
2609             { \l_tmpa_str\int_use:N \l__stex_notation_currarg_int }
2610             { \l_tmpb_str }
2611             { ###\int_use:N \l__stex_notation_currarg_int }
2612           }
2613         }
2614         \__stex_notation_arguments:
2615       }
2616     }
2617   }
2618 }

```

(End definition for __stex_notation_arguments:.)

__stex_notation_argument_assoc:nn

```

2619 \cs_new_protected:Nn \__stex_notation_argument_assoc:nn {
2620
2621   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2622     {\l__stex_notation_arity_str}{
2623       #2
2624     }
2625   \int_zero:N \l_tmpa_int
2626   \tl_clear:N \l_tmpa_tl
2627   \str_map_inline:Nn \l__stex_notation_args_str {
2628     \int_incr:N \l_tmpa_int
2629     \tl_put_right:Nx \l_tmpa_tl {
2630       \str_if_eq:nnTF {##1}{a}{ { } }{ }

```

```

2631 \str_if_eq:nnTF {##1}{B}{ } }{
2632   {\_stex_term_arg:nn{##1\int_use:N \l_tmpa_int}{##### \int_use:N \l_tmpa
2633   }
2634 }
2635 }
2636 }
2637 \exp_after:wN\exp_after:wN\exp_after:wN \def
2638 \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2639 \exp_after:wN\exp_after:wN\exp_after:wN ##
2640 \exp_after:wN\exp_after:wN\exp_after:wN 1
2641 \exp_after:wN\exp_after:wN\exp_after:wN ##
2642 \exp_after:wN\exp_after:wN\exp_after:wN 2
2643 \exp_after:wN\exp_after:wN\exp_after:wN {
2644   \exp_after:wN \exp_after:wN \exp_after:wN
2645   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2646     \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2647   }
2648 }
2649
2650 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2651 \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
2652   \_stex_term_math_assoc_arg:nnnn
2653   { #1\int_use:N \l__stex_notation_currarg_int }
2654   { \l_tmpa_str }
2655   { #####\int_use:N \l__stex_notation_currarg_int }
2656   { \l_tmpa_cs {####1} {####2} }
2657 } }
2658 \__stex_notation_arguments:
2659 }

```

(End definition for _stex_notation_argument_assoc:nn.)

_stex_notation_final: Called after processing all notation arguments

```

2660 \cs_new_protected:Nn \_stex_notation_restore_notation:nnnnn {
2661   \cs_generate_from_arg_count:cNnn{stex_notation_\detokenize{#1} \c_hash_str \detokenize{#2}}
2662   \cs_set_nopar:Npn {#3}{#4}
2663   \tl_if_empty:nF {#5}{
2664     \tl_set:cn{stex_op_notation_\detokenize{#1} \c_hash_str \detokenize{#2}_cs}{\comp{ #5 }
2665   }
2666   \seq_if_exist:cT { l_stex_symdecl_\detokenize{#1} _notations }{
2667     \seq_put_right:cx { l_stex_symdecl_\detokenize{#1} _notations } { \detokenize{#2} }
2668   }
2669 }
2670
2671 \cs_new_protected:Nn \_stex_notation_final: {
2672
2673   \stex_execute_in_module:x {
2674     \_stex_notation_restore_notation:nnnnn
2675     {\l_stex_get_symbol_uri_str}
2676     {\l_stex_notation_suffix_str}
2677     {\l__stex_notation_arity_str}
2678     {
2679       \exp_after:wN \exp_after:wN \exp_after:wN
2680       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN

```



```

2681     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2682   }
2683   {\exp_args:No \exp_not:n \l__stex_notation_op_tl }
2684 }
2685
2686 \stex_debug:nn{symbols}{
2687   Notation~\l__stex_notation_suffix_str
2688   ~for~\l_stex_get_symbol_uri_str^^J
2689   Operator~precedence:~\l__stex_notation_opprec_tl^^J
2690   Argument~precedences:~
2691     \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2692   Notation: \cs_meaning:c {
2693     stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2694     \l__stex_notation_suffix_str
2695     _cs
2696   }
2697 }
2698 % HTML annotations
2699 \stex_if_do_html:T {
2700   \stex_annotate_invisible:nnn { notation }
2701   { \l_stex_get_symbol_uri_str } {
2702     \stex_annotate_invisible:nnn { notationfragment }
2703     { \l__stex_notation_suffix_str }{}
2704     \stex_annotate_invisible:nnn { precedence }
2705     { \l__stex_notation_prec_str }{}
2706
2707     \int_zero:N \l_tmpa_int
2708     \str_set_eq:NN \l__stex_notation_remaining_args_str \l_stex_notation_args_str
2709     \tl_clear:N \l_tmpa_tl
2710     \int_step_inline:nn { \l__stex_notation_arity_str }{
2711       \int_incr:N \l_tmpa_int
2712       \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2713       \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_rema
2714       \str_if_eq:VnTF \l_tmpb_str a {
2715         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2716           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2717           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2718         } }
2719       }{
2720         \str_if_eq:VnTF \l_tmpb_str B {
2721           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2722             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2723             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2724           } }
2725         }{
2726           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2727             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
2728           } }
2729       }
2730     }
2731   }
2732   \stex_annotate_invisible:nnn { notationcomp }{}{
2733     \str_set:Nx \l_stex_current_symbol_str {\l_stex_get_symbol_uri_str }
2734     $ \exp_args:Nno \use:nn { \use:c {

```

```

2735         stex_notation_ \l_stex_current_symbol_str
2736         \c_hash_str \l__stex_notation_suffix_str _cs
2737     } } { \l_tmpa_tl } $
2738 }
2739 }
2740 }
2741 }

```

(End definition for _stex_notation_final:.)

\setnotation

```

2742 \keys_define:nn { stex / setnotation } {
2743   % lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2744   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2745   unknown .code:n      = \str_set:Nx
2746     \l__stex_notation_variant_str \l_keys_key_str
2747 }
2748
2749 \cs_new_protected:Nn \stex_setnotation_args:n {
2750   % \str_clear:N \l__stex_notation_lang_str
2751   \str_clear:N \l__stex_notation_variant_str
2752   \keys_set:nn { stex / setnotation } { #1 }
2753 }
2754
2755 \cs_new_protected:Nn \__stex_notation_setnotation:nn {
2756   \seq_if_exist:cT{l_stex_symdecl_#1_notations}{
2757     \seq_remove_all:cn { l_stex_symdecl_#1_notations }{ #2 }
2758     \seq_put_left:cn { l_stex_symdecl_#1_notations }{ #2 }
2759   }
2760 }
2761
2762 \cs_new_protected:Nn \stex_setnotation:n {
2763   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1_notations }
2764     { \l__stex_notation_variant_str }{
2765     \stex_execute_in_module:x{ \__stex_notation_setnotation:nn {#1}{\l__stex_notation_vari
2766     \stex_debug:nn {notations}{
2767       Setting~default~notation~
2768       {\l__stex_notation_variant_str }~for~
2769       #1 \\
2770       \expandafter\meaning\csname
2771       l_stex_symdecl_#1_notations\endcsname
2772     }
2773   }{
2774     \msg_error:nnxx{stex}{unknownnotation}{\l__stex_notation_variant_str}{#1}
2775   }
2776 }
2777
2778 \NewDocumentCommand \setnotation {m m} {
2779   \stex_get_symbol:n { #1 }
2780   \stex_setnotation_args:n { #2 }
2781   \stex_setnotation:n{\l_stex_get_symbol_uri_str}
2782   \stex_smsmode_do:\ignorespacesandpars
2783 }
2784

```

```

2785 \cs_new_protected:Nn \stex_copy_notations:nn {
2786   \stex_debug:nn {notations}{
2787     Copying~notations~from~#2~to~#1\\
2788     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
2789   }
2790   \tl_clear:N \l_tmpa_tl
2791   \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2792     \tl_put_right:Nn \l_tmpa_tl { {##### #1} }
2793   }
2794   \seq_map_inline:cn {l_stex_symdecl_#2_notations}{
2795     \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2796     \edef \l_tmpa_tl {
2797       \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2798       \exp_after:wN\exp_after:wN\exp_after:wN {
2799         \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2800       }
2801     }
2802
2803     \exp_after:wN \def \exp_after:wN \l_tmpa_tl
2804     \exp_after:wN #####\exp_after:wN 1 \exp_after:wN #####\exp_after:wN 2
2805     \exp_after:wN { \l_tmpa_tl }
2806
2807     \edef \l_tmpa_tl {
2808       \exp_after:wN \exp_not:n \exp_after:wN {
2809         \l_tmpa_tl {##### 1}{##### 2}
2810       }
2811     }
2812
2813     \stex_execute_in_module:x {
2814       \__stex_notation_restore_notation:nnnnn
2815       {#1}{##1}
2816       { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }
2817       { \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl} }
2818       {
2819         \cs_if_exist:cT{stex_op_notation_ #2\c_hash_str ##1 _cs}{
2820           \exp_args:NNo\exp_args:No\exp_not:n{\csname stex_op_notation_ #2\c_hash_str ##1
2821             }
2822         }
2823       }
2824     }
2825   }
2826
2827   \NewDocumentCommand \copynotation {m m} {
2828     \stex_get_symbol:n { #1 }
2829     \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2830     \stex_get_symbol:n { #2 }
2831     \exp_args:Noo
2832     \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2833     \stex_smsmode_do:\ignorespacesandpars
2834   }
2835

```

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

\symdef

```
2836 \keys_define:nn { stex / symdef } {
2837   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2838   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2839   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2840   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
2841   def       .tl_set:N    = \l_stex_symdecl_definiens_tl ,
2842   reorder   .str_set_x:N = \l_stex_symdecl_reorder_str ,
2843   op        .tl_set:N    = \l__stex_notation_op_tl ,
2844   % lang     .str_set_x:N = \l__stex_notation_lang_str ,
2845   variant   .str_set_x:N = \l__stex_notation_variant_str ,
2846   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2847   assoc     .choices:nn =
2848     {bin,binl,binr,pre,conj,pwconj}
2849     {\str_set:Nx \l_stex_symdecl_assoc_type_str {\l_keys_choice_tl}},
2850   unknown   .code:n      = \str_set:Nx
2851     \l__stex_notation_variant_str \l_keys_key_str
2852 }
2853
2854 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2855   \str_clear:N \l_stex_symdecl_name_str
2856   \str_clear:N \l_stex_symdecl_args_str
2857   \str_clear:N \l_stex_symdecl_assoc_type_str
2858   \str_clear:N \l_stex_symdecl_reorder_str
2859   \bool_set_false:N \l_stex_symdecl_local_bool
2860   \tl_clear:N \l_stex_symdecl_type_tl
2861   \tl_clear:N \l_stex_symdecl_definiens_tl
2862   % \str_clear:N \l__stex_notation_lang_str
2863   \str_clear:N \l__stex_notation_variant_str
2864   \str_clear:N \l__stex_notation_prec_str
2865   \tl_clear:N \l__stex_notation_op_tl
2866
2867   \keys_set:nn { stex / symdef } { #1 }
2868 }
2869
2870 \NewDocumentCommand \symdef { m O{} } {
2871   \__stex_notation_symdef_args:n { #2 }
2872   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2873   \stex_symdecl_do:n { #1 }
2874   \tl_set:Nn \l_stex_notation_after_do_tl {
2875     \__stex_notation_final:
2876     \stex_smsmode_do:\ignorespacesandpars
2877   }
2878   \str_set:Nx \l_stex_get_symbol_uri_str {
2879     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2880   }
2881   \exp_args:Nx \stex_notation_do:nnnnn
2882     { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } }
2883     { \prop_item:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } }
2884     { \l__stex_notation_variant_str }
2885     { \l__stex_notation_prec_str }
2886 }
2887 \stex_deactivate_macro:Nn \symdef {module~environments}
```

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

29.3 Variables

```

2888 <@@=stex_variables>
2889
2890 \keys_define:nn { stex / vardef } {
2891   name      .str_set_x:N = \l__stex_variables_name_str ,
2892   args      .str_set_x:N = \l__stex_variables_args_str ,
2893   type      .tl_set:N    = \l__stex_variables_type_tl ,
2894   def       .tl_set:N    = \l__stex_variables_def_tl ,
2895   op        .tl_set:N    = \l__stex_variables_op_tl ,
2896   prec      .str_set_x:N = \l__stex_variables_prec_str ,
2897   assoc     .choices:nn =
2898     {bin,binl,binr,pre,conj,pwconj}
2899     {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
2900   bind      .choices:nn =
2901     {forall,exists}
2902     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2903 }
2904
2905 \cs_new_protected:Nn \__stex_variables_args:n {
2906   \str_clear:N \l__stex_variables_name_str
2907   \str_clear:N \l__stex_variables_args_str
2908   \str_clear:N \l__stex_variables_prec_str
2909   \str_clear:N \l__stex_variables_assoctype_str
2910   \str_clear:N \l__stex_variables_bind_str
2911   \tl_clear:N \l__stex_variables_type_tl
2912   \tl_clear:N \l__stex_variables_def_tl
2913   \tl_clear:N \l__stex_variables_op_tl
2914
2915   \keys_set:nn { stex / vardef } { #1 }
2916 }
2917
2918 \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{} } {
2919   \__stex_variables_args:n {#2}
2920   \str_if_empty:NT \l__stex_variables_name_str {
2921     \str_set:Nx \l__stex_variables_name_str { #1 }
2922   }
2923   \prop_clear:N \l_tmpa_prop
2924   \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
2925
2926   \int_zero:N \l_tmpb_int
2927   \bool_set_true:N \l_tmpa_bool
2928   \str_map_inline:Nn \l__stex_variables_args_str {
2929     \token_case_meaning:NnF ##1 {
2930       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2931       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2932       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2933       {\tl_to_str:n a} {
2934         \bool_set_false:N \l_tmpa_bool
2935         \int_incr:N \l_tmpb_int
2936       }

```

```

2937     {\tl_to_str:n B} {
2938         \bool_set_false:N \l_tmpa_bool
2939         \int_incr:N \l_tmpb_int
2940     }
2941 }{
2942     \msg_error:nxxx{stex}{error/wrongargs}{
2943         variable~\l__stex_variables_name_str
2944     }{##1}
2945 }
2946 }
2947 \bool_if:NTF \l_tmpa_bool {
2948     % possibly numeric
2949     \str_if_empty:NTF \l__stex_variables_args_str {
2950         \prop_put:Nnn \l_tmpa_prop { args } {}
2951         \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2952     }{
2953         \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
2954         \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2955         \str_clear:N \l_tmpa_str
2956         \int_step_inline:nn \l_tmpa_int {
2957             \str_put_right:Nn \l_tmpa_str i
2958         }
2959         \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
2960         \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2961     }
2962 } {
2963     \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2964     \prop_put:Nnx \l_tmpa_prop { arity }
2965         { \str_count:N \l__stex_variables_args_str }
2966 }
2967 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2968 \tl_set:cx { #1 }{ \stex_invoke_variable:n { \l__stex_variables_name_str } }
2969
2970 \prop_set_eq:cN { l_stex_variable_\l__stex_variables_name_str _prop } \l_tmpa_prop
2971
2972 \tl_if_empty:NF \l__stex_variables_op_tl {
2973     \cs_set:cpx {
2974         stex_var_op_notation_\l__stex_variables_name_str_cs
2975     } { \exp_not:N\comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
2976 }
2977
2978 \tl_set:Nn \l_stex_notation_after_do_tl {
2979     \exp_args:Nne \use:nn {
2980         \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str_cs }
2981         \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
2982     } {{
2983         \exp_after:wN \exp_after:wN \exp_after:wN
2984         \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2985         { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_symbol
2986     }}
2987 \stex_if_do_html:T {
2988     \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
2989         \stex_annotate_invisible:nnn { precedence }
2990         { \l__stex_variables_prec_str }{}

```

```

2991 \tl_if_empty:NF \l__stex_variables_type_tl {\stex_annotate_invisible:nnn{type}{}}{\$ \l
2992 \stex_annotate_invisible:nnn{args}{}}{\l__stex_variables_args_str }
2993 \stex_annotate_invisible:nnn{macroname}{#1}{}
2994 \tl_if_empty:NF \l__stex_variables_def_tl {
2995   \stex_annotate_invisible:nnn{definiens}{}
2996   {\$ \l__stex_variables_def_tl$}
2997 }
2998 \str_if_empty:NF \l__stex_variables_assoctype_str {
2999   \stex_annotate_invisible:nnn{assoctype}{\l__stex_variables_assoctype_str}{}
3000 }
3001 \str_if_empty:NF \l__stex_variables_bind_str {
3002   \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{}
3003 }
3004 \int_zero:N \l_tmpa_int
3005 \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
3006 \tl_clear:N \l_tmpa_tl
3007 \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
3008   \int_incr:N \l_tmpa_int
3009   \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
3010   \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables
3011   \str_if_eq:VnTF \l_tmpb_str a {
3012     \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3013       \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3014       \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3015     } }
3016   }{
3017     \str_if_eq:VnTF \l_tmpb_str B {
3018       \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3019         \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3020         \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3021       } }
3022     }{
3023       \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3024         \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
3025       } }
3026     }
3027   }
3028 }
3029 \stex_annotate_invisible:nnn { notationcomp }{}{
3030   \str_set:Nx \l_stex_current_symbol_str {var://\l__stex_variables_name_str }
3031   $ \exp_args:Nno \use:nn { \use:c {
3032     stex_var_notation_\l__stex_variables_name_str_cs
3033   } } { \l_tmpa_tl } $
3034 }
3035 }
3036 }\ignorespacesandpars
3037 }
3038
3039 \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
3040 }
3041
3042 \cs_new:Nn \_stex_reset:N {
3043   \tl_if_exist:NTF #1 {
3044     \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }

```

```

3045 }{
3046   \let \exp_not:N #1 \exp_not:N \undefined
3047 }
3048 }
3049
3050 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
3051   \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
3052   \exp_args:Nnx \use:nn {
3053     % TODO
3054     \stex_annotate_invisible:nnn {vardecl}{\clist_use:Nn\l__stex_variables_names,}{
3055       #2
3056     }
3057   }{
3058     \stex_reset:N \varnot
3059     \stex_reset:N \vartype
3060     \stex_reset:N \vardefi
3061   }
3062 }
3063
3064 \NewDocumentCommand \vardef { s } {
3065   \IfBooleanTF#1 {
3066     \__stex_variables_do_complex:nn
3067   }{
3068     \__stex_variables_do_simple:nnn
3069   }
3070 }
3071
3072 \NewDocumentCommand \svar { 0{} m }{
3073   \tl_if_empty:nTF {#1}{
3074     \str_set:Nn \l_tmpa_str { #2 }
3075   }{
3076     \str_set:Nn \l_tmpa_str { #1 }
3077   }
3078   \stex_term_omv:nn {
3079     var://\l_tmpa_str
3080   }{
3081     \exp_args:Nnx \use:nn {
3082       \def\comp{\_varcomp}
3083       \str_set:Nx \l_stex_current_symbol_str { var://\l_tmpa_str }
3084       \comp{ #2 }
3085     }{
3086       \stex_reset:N \comp
3087       \stex_reset:N \l_stex_current_symbol_str
3088     }
3089   }
3090 }
3091
3092
3093
3094 \keys_define:nn { stex / varseq } {
3095   name .str_set_x:N = \l__stex_variables_name_str ,
3096   args .int_set:N = \l__stex_variables_args_int ,
3097   type .tl_set:N = \l__stex_variables_type_tl ,
3098   mid .tl_set:N = \l__stex_variables_mid_tl ,

```



```

3099   bind      .choices:nn      =
3100           {forall,exists}
3101           {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
3102   }
3103
3104   \cs_new_protected:Nn \__stex_variables_seq_args:n {
3105     \str_clear:N \l__stex_variables_name_str
3106     \int_set:Nn \l__stex_variables_args_int 1
3107     \tl_clear:N \l__stex_variables_type_tl
3108     \str_clear:N \l__stex_variables_bind_str
3109
3110     \keys_set:nn { stex / varseq } { #1 }
3111   }
3112
3113   \NewDocumentCommand \varseq {m O{}} m m m m){
3114     \__stex_variables_seq_args:n { #2 }
3115     \str_if_empty:NT \l__stex_variables_name_str {
3116       \str_set:Nx \l__stex_variables_name_str { #1 }
3117     }
3118     \prop_clear:N \l_tmpa_prop
3119     \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
3120
3121     \seq_set_from_clist:Nn \l_tmpa_seq {#3}
3122     \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
3123       \msg_error:nnxx{stex}{error/seqlength}
3124       {\int_use:N \l__stex_variables_args_int}
3125       {\seq_count:N \l_tmpa_seq}
3126     }
3127     \seq_set_from_clist:Nn \l_tmpb_seq {#4}
3128     \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
3129       \msg_error:nnxx{stex}{error/seqlength}
3130       {\int_use:N \l__stex_variables_args_int}
3131       {\seq_count:N \l_tmpb_seq}
3132     }
3133     \prop_put:Nnn \l_tmpa_prop {starts} {#3}
3134     \prop_put:Nnn \l_tmpa_prop {ends} {#4}
3135
3136     \cs_generate_from_arg_count:cNnn {stex_varseq\l__stex_variables_name_str _cs}
3137     \cs_set:Npn {\int_use:N \l__stex_variables_args_int} { #5 }
3138
3139     \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}
3140     \int_step_inline:nn \l__stex_variables_args_int {
3141       \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
3142     }
3143     \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
3144     \tl_put_right:Nn \l_tmpa_tl {\,\ellipses,}
3145     \tl_if_empty:NF \l__stex_variables_mid_tl {
3146       \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
3147       \tl_put_right:Nn \l_tmpa_tl {\,\ellipses,}
3148     }
3149     \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}
3150     \int_step_inline:nn \l__stex_variables_args_int {
3151       \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
3152     }

```

```

3153 \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
3154 \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
3155
3156
3157 \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl
3158
3159 \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}
3160
3161 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}
3162
3163 \int_step_inline:nn \l__stex_variables_args_int {
3164   \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
3165     \stex_term_math_arg:nnn{i##1}{0}{\exp_not:n{####}##1}
3166   }}
3167 }
3168
3169 \tl_set:Nx \l_tmpa_tl {
3170   \stex_term_math_oma:nnnn { varseq://\l__stex_variables_name_str}{0}{
3171     \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
3172   }
3173 }
3174
3175 \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \stex_symbol_after_invokation_tl} }
3176
3177 \exp_args:Nno \use:nn {
3178   \cs_generate_from_arg_count:cNnn {stex_varseq\l__stex_variables_name_str _cs}
3179   \cs_set:Npn {\int_use:N \l__stex_variables_args_int}{\l_tmpa_tl}
3180
3181   \stex_debug:nn{sequences}{New~Sequence:~
3182     \expandafter\meaning\csname stex_varseq\l__stex_variables_name_str _cs\endcsname\\~\\
3183     \prop_to_keyval:N \l_tmpa_prop
3184   }
3185   \stex_if_do_html:T{\stex_annotate_invisible:nnn{varseq}{\l__stex_variables_name_str}{
3186     \tl_if_empty:NF \l__stex_variables_type_tl {
3187       \stex_annotate:nnn {type}{}{\${seqtype\l__stex_variables_type_tl$}
3188     }
3189     \stex_annotate:nnn {args}{\int_use:N \l__stex_variables_args_int}{}
3190     \str_if_empty:NF \l__stex_variables_bind_str {
3191       \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{}
3192     }
3193   }}
3194
3195   \prop_set_eq:cN {stex_varseq\l__stex_variables_name_str _prop}\l_tmpa_prop
3196   \ignorespacesandpars
3197 }
3198
3199 </package>

```

Chapter 30

STEX -Terms Implementation

```
3200 <*package>
3201
3202 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
3203
3204 <@@=stex_terms>
3205
3206   Warnings and error messages
3207 \msg_new:nnn{stex}{error/nonotation}{
3208   Symbol~#1~invoked,~but~has~no~notation#2!
3209 }
3210 \msg_new:nnn{stex}{error/notationarg}{
3211   Error~in~parsing~notation~#1
3212 }
3213 \msg_new:nnn{stex}{error/noop}{
3214   Symbol~#1~has~no~operator~notation~for~notation~#2
3215 }
3216 \msg_new:nnn{stex}{error/notallowed}{
3217   Symbol~invocation~#1~not~allowed~in~notation~component~of~#2
3218 }
3219 \msg_new:nnn{stex}{error/doubleargument}{
3220   Argument~#1~of~symbol~#2~already~assigned
3221 }
3222 \msg_new:nnn{stex}{error/overarity}{
3223   Argument~#1~invalid~for~symbol~#2~with~arity~#3
3224 }
```

30.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
3224
3225
3226 \bool_new:N \l_stex_allow_semantic_bool
3227 \bool_set_true:N \l_stex_allow_semantic_bool
3228
```

```

3229 \cs_new_protected:Nn \stex_invoke_symbol:n {
3230   \bool_if:NTF \l_stex_allow_semantic_bool {
3231     \str_if_eq:eeF {
3232       \prop_item:cn {
3233         l_stex_symdecl_#1_prop
3234       }{ deprecate }
3235     }{}{
3236       \msg_warning:nxxx{stex}{warning/deprecated}{
3237         Symbol~#1
3238       }{
3239         \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
3240       }
3241     }
3242     \if_mode_math:
3243       \exp_after:wN \__stex_terms_invoke_math:n
3244     \else:
3245       \exp_after:wN \__stex_terms_invoke_text:n
3246     \fi: { #1 }
3247   }{
3248     \msg_error:nxxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
3249   }
3250 }
3251
3252 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
3253   \peek_charcode_remove:NTF ! {
3254     \__stex_terms_invoke_op_custom:nn {#1}
3255   }{
3256     \__stex_terms_invoke_custom:nn {#1}
3257   }
3258 }
3259
3260 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
3261   \peek_charcode_remove:NTF ! {
3262     % operator
3263     \peek_charcode_remove:NTF * {
3264       % custom op
3265       \__stex_terms_invoke_op_custom:nn {#1}
3266     }{
3267       % op notation
3268       \peek_charcode:NTF [ {
3269         \__stex_terms_invoke_op_notation:nw {#1}
3270       }{
3271         \__stex_terms_invoke_op_notation:nw {#1}[]
3272       }
3273     }
3274   }{
3275     \peek_charcode_remove:NTF * {
3276       \__stex_terms_invoke_custom:nn {#1}
3277       % custom
3278     }{
3279       % normal
3280       \peek_charcode:NTF [ {
3281         \__stex_terms_invoke_notation:nw {#1}
3282       }{

```

```

3283     \__stex_terms_invoke_notation:nw {#1}[]
3284   }
3285 }
3286 }
3287 }
3288
3289
3290 \cs_new_protected:Nn \__stex_terms_invoke_op_custom:nn {
3291   \exp_args:Nnx \use:nn {
3292     \def\comp{\_comp}
3293     \str_set:Nn \l_stex_current_symbol_str { #1 }
3294     \bool_set_false:N \l_stex_allow_semantic_bool
3295     \stex_term_oms:nnn {#1}{#1 \c_hash_str CUSTOM-}{
3296       \comp{ #2 }
3297     }
3298   }{
3299     \stex_reset:N \comp
3300     \stex_reset:N \l_stex_current_symbol_str
3301     \bool_set_true:N \l_stex_allow_semantic_bool
3302   }
3303 }
3304
3305 \keys_define:nn { stex / terms } {
3306   % lang .tl_set_x:N = \l_stex_notation_lang_str ,
3307   variant .tl_set_x:N = \l_stex_notation_variant_str ,
3308   unknown .code:n = \str_set:Nx
3309     \l_stex_notation_variant_str \l_keys_key_str
3310 }
3311
3312 \cs_new_protected:Nn \__stex_terms_args:n {
3313   % \str_clear:N \l_stex_notation_lang_str
3314   \str_clear:N \l_stex_notation_variant_str
3315
3316   \keys_set:nn { stex / terms } { #1 }
3317 }
3318
3319 \cs_new_protected:Nn \stex_find_notation:nn {
3320   \__stex_terms_args:n { #2 }
3321   \seq_if_empty:cTF {
3322     l_stex_symdecl_ #1 _notations
3323   } {
3324     \msg_error:nxxx{stex}{error/nonotation}{#1}{s}
3325   } {
3326     \str_if_empty:NTF \l_stex_notation_variant_str {
3327       \seq_get_left:cN {l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
3328     }{
3329       \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
3330         \l_stex_notation_variant_str
3331       }{
3332         % \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3333       }{
3334         \msg_error:nxxx{stex}{error/nonotation}{#1}{
3335           ~\l_stex_notation_variant_str
3336         }

```

```

3337     }
3338   }
3339 }
3340 }
3341
3342 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3343   \exp_args:Nnx \use:nn {
3344     \def\comp{\_comp}
3345     \str_set:Nn \l_stex_current_symbol_str { #1 }
3346     \stex_find_notation:nn { #1 }{ #2 }
3347     \bool_set_false:N \l_stex_allow_semantic_bool
3348     \cs_if_exist:cTF {
3349       stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3350     }{
3351       \_stex_term_oms:nnn { #1 }{
3352         #1 \c_hash_str \l_stex_notation_variant_str
3353       }{
3354         \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3355       }
3356     }{
3357       \int_compare:nNnTF {\prop_item:cn {\l_stex_symdecl_#1_prop}{arity}} = 0{
3358         \cs_if_exist:cTF {
3359           stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3360         }{
3361           \tl_set:Nx \stex_symbol_after_invokation_tl {
3362             \_stex_reset:N \comp
3363             \_stex_reset:N \stex_symbol_after_invokation_tl
3364             \_stex_reset:N \l_stex_current_symbol_str
3365             \bool_set_true:N \l_stex_allow_semantic_bool
3366           }
3367           \def\comp{\_comp}
3368           \str_set:Nn \l_stex_current_symbol_str { #1 }
3369           \bool_set_false:N \l_stex_allow_semantic_bool
3370           \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3371         }{
3372           \msg_error:nnxx{stex}{error/nonotation}{#1}{
3373             ~\l_stex_notation_variant_str
3374           }
3375         }
3376       }{
3377         \msg_error:nnxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3378       }
3379     }
3380   }{
3381     \_stex_reset:N \comp
3382     \_stex_reset:N \l_stex_current_symbol_str
3383     \bool_set_true:N \l_stex_allow_semantic_bool
3384   }
3385 }
3386
3387 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {
3388   \stex_find_notation:nn { #1 }{ #2 }
3389   \cs_if_exist:cTF {
3390     stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs

```

```

3391 }{
3392   \tl_set:Nx \stex_symbol_after_invokation_tl {
3393     \_stex_reset:N \comp
3394     \_stex_reset:N \stex_symbol_after_invokation_tl
3395     \_stex_reset:N \l_stex_current_symbol_str
3396     \bool_set_true:N \l_stex_allow_semantic_bool
3397   }
3398   \def\comp{\_comp}
3399   \str_set:Nn \l_stex_current_symbol_str { #1 }
3400   \bool_set_false:N \l_stex_allow_semantic_bool
3401   \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3402 }{
3403   \msg_error:nnxx{stex}{error/nonotation}{#1}{
3404     ~\l_stex_notation_variant_str
3405   }
3406 }
3407 }
3408
3409 \prop_new:N \l__stex_terms_custom_args_prop
3410
3411 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3412   \exp_args:Nnx \use:nn {
3413     \bool_set_false:N \l_stex_allow_semantic_bool
3414     \def\comp{\_comp}
3415     \str_set:Nn \l_stex_current_symbol_str { #1 }
3416     \prop_clear:N \l__stex_terms_custom_args_prop
3417     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3418     \prop_get:cnN {
3419       l_stex_symdecl_#1 _prop
3420     }{ args } \l_tmpa_str
3421     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3422     \tl_set:Nn \arg { \__stex_terms_arg: }
3423     \str_if_empty:NTF \l_tmpa_str {
3424       \_stex_term_oms:nnn {#1}{#1\c_hash_str CUSTOM-}{#2}
3425     }{
3426       \str_if_in:NnTF \l_tmpa_str b {
3427         \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3428       }{
3429         \str_if_in:NnTF \l_tmpa_str B {
3430           \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3431         }{
3432           \_stex_term_oma:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3433         }
3434       }
3435     }
3436     % TODO check that all arguments exist
3437   }{
3438     \_stex_reset:N \l_stex_current_symbol_str
3439     \_stex_reset:N \arg
3440     \_stex_reset:N \comp
3441     \_stex_reset:N \l__stex_terms_custom_args_prop
3442     \bool_set_true:N \l_stex_allow_semantic_bool
3443   }
3444 }

```

```

3445 \NewDocumentCommand \__stex_terms_arg: { s O{} m}{
3446   \tl_if_empty:nTF {#2}{
3447     \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
3448     \bool_set_true:N \l_tmpa_bool
3449     \bool_do_while:Nn \l_tmpa_bool {
3450       \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int
3451         \int_incr:N \l_tmpa_int
3452       }{
3453         \bool_set_false:N \l_tmpa_bool
3454       }
3455     }
3456   }{
3457     \int_set:Nn \l_tmpa_int { #2 }
3458   }
3459   \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
3460   \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
3461     \msg_error:nnxxx{stex}{error/overarity}
3462     {\int_use:N \l_tmpa_int}
3463     {\l_stex_current_symbol_str}
3464     {\str_count:N \l_tmpa_str}
3465   }
3466   \str_set:Nx \l_tmpa_str {\str_item:Nn \l_tmpa_str \l_tmpa_int}
3467   \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
3468     \bool_lazy_any:nF {
3469       {\str_if_eq_p:Vn \l_tmpa_str {a}}
3470       {\str_if_eq_p:Vn \l_tmpa_str {B}}
3471     }{
3472       \msg_error:nnxx{stex}{error/doubleargument}
3473       {\int_use:N \l_tmpa_int}
3474       {\l_stex_current_symbol_str}
3475     }
3476   }
3477   \exp_args:NNx \prop_put:Nnn \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {#3}
3478   \bool_set_true:N \l_stex_allow_semantic_bool
3479   \IfBooleanTF#1{
3480     \stex_annotate_invisible:n { %TODO
3481       \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{#3}
3482     }
3483   }{ %TODO
3484     \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{#3}
3485   }
3486   \bool_set_false:N \l_stex_allow_semantic_bool
3487 }
3488
3489
3490 \cs_new_protected:Nn \_stex_term_arg:nn {
3491   \bool_set_true:N \l_stex_allow_semantic_bool
3492   \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3493   \bool_set_false:N \l_stex_allow_semantic_bool
3494 }
3495
3496 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
3497   \exp_args:Nnx \use:nn

```



```

3499 { \int_set:Nn \l__stex_terms_downprec { #2 }
3500   \stex_term_arg:nn { #1 }{ #3 }
3501 }
3502 { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3503 }

```

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

`\stex_term_math_assoc_arg:nnnn`

```

3504 \cs_new_protected:Nn \stex_term_math_assoc_arg:nnnn {
3505   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3506   \tl_set:Nn \l_tmpb_tl {\stex_term_math_arg:nnn{#1}{#2}}
3507   \tl_if_empty:NTF { #3 }{
3508     \stex_term_math_arg:nnn{#1}{#2}{#3}
3509   }{
3510     \exp_args:Nx \tl_if_empty:NTF { \tl_tail:n{ #3 } }{
3511       \expandafter\if\expandafter\relax\noexpand#3
3512       \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_maybe_sequence:Nn#3{#1}}
3513     }else
3514       \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_simple:nn{#1}{#3}}
3515     \fi
3516     \l_tmpa_tl
3517   }{
3518     \__stex_terms_math_assoc_arg_simple:nn{#1}{#3}
3519   }
3520 }
3521 }
3522
3523 \cs_new_protected:Nn \__stex_terms_math_assoc_arg_maybe_sequence:Nn {
3524   \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
3525   \str_if_empty:NTF \l_tmpa_str {
3526     \exp_args:Nx \cs_if_eq:NNTF {
3527       \tl_head:N #1
3528     } \stex_invoke_sequence:n {
3529       \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
3530       \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
3531       \tl_set:Nx \l_tmpa_tl {\prop_item:cn {stex_varseq_\l_tmpa_str _prop}{notation}}
3532       \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
3533       \tl_set:Nx \l_tmpa_tl {\exp_not:N \exp_not:n{
3534         \exp_not:n{\exp_args:Nnx \use:nn} {
3535           \exp_not:n {
3536             \def\comp{\_varcomp}
3537             \str_set:Nn \l_stex_current_symbol_str
3538             } {varseq://\l_tmpa_str}
3539           \exp_not:n{ ##1 }
3540         }{
3541           \exp_not:n {
3542             \_stex_reset:N \comp
3543             \_stex_reset:N \l_stex_current_symbol_str
3544           }
3545         }
3546       }}}
3547   \exp_args:Nno \use:nn {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
3548   \seq_reverse:N \l_tmpa_seq

```

```

3549 \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
3550 \seq_map_inline:Nn \l_tmpa_seq {
3551   \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3552     \exp_args:Nno
3553     \l_tmpa_cs { ##1 } \l_tmpa_tl
3554   }
3555 }
3556 \tl_set:Nx \l_tmpa_tl {
3557   \stex_term_omv:nn {varseq://\l_tmpa_str}{
3558     \exp_args:No \exp_not:n \l_tmpa_tl
3559   }
3560 }
3561 \exp_args:No\l_tmpb_tl\l_tmpa_tl
3562 }{
3563   \stex_terms_math_assoc_arg_simple:nn{#2} { #1 }
3564 }
3565 } {
3566   \stex_terms_math_assoc_arg_simple:nn{#2} { #1 }
3567 }
3568 }
3569 }
3570
3571 \cs_new_protected:Nn \stex_terms_math_assoc_arg_simple:nn {
3572   \clist_set:Nn \l_tmpa_clist{ #2 }
3573   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3574     \tl_set:Nn \l_tmpa_tl { \stex_term_arg:nn{A#1}{ #2 } }
3575   }{
3576     \clist_reverse:N \l_tmpa_clist
3577     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3578     \tl_set:Nx \l_tmpa_tl { \stex_term_arg:nn{A#1}{
3579       \exp_args:No \exp_not:n \l_tmpa_tl
3580     }}
3581     \clist_map_inline:Nn \l_tmpa_clist {
3582       \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3583         \exp_args:Nno
3584         \l_tmpa_cs { \stex_term_arg:nn{A#1}{##1} } \l_tmpa_tl
3585       }
3586     }
3587   }
3588   \exp_args:No\l_tmpb_tl\l_tmpa_tl
3589 }

```

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

30.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
3590 \tl_const:Nx \infprec {\int_use:N \c_max_int}
3591 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
3592 \int_new:N \l__stex_terms_downprec
3593 \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 80.)

Bracketing:

```
\l_stex_terms_left_bracket_str
\l_stex_terms_right_bracket_str
```

```
3594 \tl_set:Nn \l__stex_terms_left_bracket_str (
3595 \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

```
3596 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
3597   \bool_if:NTF \l__stex_terms_brackets_done_bool {
3598     \bool_set_false:N \l__stex_terms_brackets_done_bool
3599     #2
3600   } {
3601     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
3602       \bool_if:NTF \l_stex_inarray_bool { #2 } {
3603         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
3604         \dobrackets { #2 }
3605       }
3606     }{ #2 }
3607   }
3608 }
```

(End definition for `__stex_terms_maybe_brackets:nn`.)

\dobrackets

```
3609 \bool_new:N \l__stex_terms_brackets_done_bool
3610 %\RequirePackage{scalerel}
3611 \cs_new_protected:Npn \dobrackets #1 {
3612   %\ThisStyle{\if D\m@switch
3613   %   \exp_args:Nnx \use:nn
3614   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
3615   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
3616   %   \else
3617   \exp_args:Nnx \use:nn
3618   {
3619     \bool_set_true:N \l__stex_terms_brackets_done_bool
3620     \int_set:Nn \l__stex_terms_downprec \infprec
3621     \l__stex_terms_left_bracket_str
3622     #1
3623   }
3624   {
3625     \bool_set_false:N \l__stex_terms_brackets_done_bool
3626     \l__stex_terms_right_bracket_str
3627     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3628   }
3629   %\fi}
3630 }
```

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

`\withbrackets`

```
3631 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
3632   \exp_args:Nnx \use:nn
3633   {
3634     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
3635     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
3636     #3
3637   }
3638   {
3639     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
3640     {\l__stex_terms_left_bracket_str}
3641     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
3642     {\l__stex_terms_right_bracket_str}
3643   }
3644 }
```

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

`\STEXinvisible`

```
3645 \cs_new_protected:Npn \STEXinvisible #1 {
3646   \stex_annotate_invisible:n { #1 }
3647 }
```

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

OMDoc terms:

`_stex_term_math_oms:nnnn`

```
3648 \cs_new_protected:Nn \_stex_term_oms:nnn {
3649   \stex_annotate:nnn{ OMID }{ #2 }{
3650     #3
3651   }
3652 }
3653
3654 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
3655   \__stex_terms_maybe_brackets:nn { #3 }{
3656     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3657   }
3658 }
```

(End definition for `_stex_term_math_oms:nnnn`. This function is documented on page 79.)

`_stex_term_math_omv:nn`

```
3659 \cs_new_protected:Nn \_stex_term_omv:nn {
3660   \stex_annotate:nnn{ OMV }{ #1 }{
3661     #2
3662   }
3663 }
```

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

`_stex_term_math_oma:nnnn`

```
3664 \cs_new_protected:Nn \_stex_term_oma:nnn {
3665   \stex_annotate:nnn{ OMA }{ #2 }{
3666     #3
3667   }
```

```

3668 }
3669
3670 \cs_new_protected:Nn \stex_term_math_oma:nnnn {
3671   \__stex_terms_maybe_brackets:nn { #3 }{
3672     \stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3673   }
3674 }

```

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

`\stex_term_math_omb:nnnn`

```

3675 \cs_new_protected:Nn \stex_term_ombind:nnn {
3676   \stex_annotate:nnn{ OMBIND }{ #2 }{
3677     #3
3678   }
3679 }
3680
3681 \cs_new_protected:Nn \stex_term_math_omb:nnnn {
3682   \__stex_terms_maybe_brackets:nn { #3 }{
3683     \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3684   }
3685 }

```

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

`\symref`

`\symname`

```

3686 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3687
3688 \keys_define:nn { stex / symname } {
3689   pre      .tl_set_x:N      = \l__stex_terms_pre_tl ,
3690   post     .tl_set_x:N      = \l__stex_terms_post_tl ,
3691   root     .tl_set_x:N      = \l__stex_terms_root_tl
3692 }
3693
3694 \cs_new_protected:Nn \stex_symname_args:n {
3695   \tl_clear:N \l__stex_terms_post_tl
3696   \tl_clear:N \l__stex_terms_pre_tl
3697   \tl_clear:N \l__stex_terms_root_str
3698   \keys_set:nn { stex / symname } { #1 }
3699 }
3700
3701 \NewDocumentCommand \symref { m m }{
3702   \let\compemph_uri_prev:\compemph@uri
3703   \let\compemph@uri\symrefemph@uri
3704   \STEXsymbol{#1}!\{ #2 }
3705   \let\compemph@uri\compemph_uri_prev:
3706 }
3707
3708 \NewDocumentCommand \synonym { 0{} m m }{
3709   \stex_symname_args:n { #1 }
3710   \let\compemph_uri_prev:\compemph@uri
3711   \let\compemph@uri\symrefemph@uri
3712   % TODO
3713   \STEXsymbol{#2}!\{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
3714   \let\compemph@uri\compemph_uri_prev:

```

```

3715 }
3716
3717 \NewDocumentCommand \symname { 0{} m }{
3718   \stex_symname_args:n { #1 }
3719   \stex_get_symbol:n { #2 }
3720   \str_set:Nx \l_tmpa_str {
3721     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3722   }
3723   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3724
3725   \let\compemph_uri_prev:\compemph@uri
3726   \let\compemph@uri\symrefemph@uri
3727   \exp_args:NNx \use:nn
3728   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmmode*\fi{
3729     \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
3730   } }
3731   \let\compemph@uri\compemph_uri_prev:
3732 }
3733
3734 \NewDocumentCommand \Symname { 0{} m }{
3735   \stex_symname_args:n { #1 }
3736   \stex_get_symbol:n { #2 }
3737   \str_set:Nx \l_tmpa_str {
3738     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3739   }
3740   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3741   \let\compemph_uri_prev:\compemph@uri
3742   \let\compemph@uri\symrefemph@uri
3743   \exp_args:NNx \use:nn
3744   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmmode*\fi{
3745     \exp_after:wN \stex_capitalize:n \l_tmpa_str
3746     \l__stex_terms_post_tl
3747   } }
3748   \let\compemph@uri\compemph_uri_prev:
3749 }

```

(End definition for `\symref` and `\symname`. These functions are documented on page 79.)

30.3 Notation Components

```

3750 <@@=stex_notationcomps>

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

3751 \cs_new_protected:Npn \_comp #1 {
3752   \str_if_empty:NF \l_stex_current_symbol_str {
3753     \stex_html_backend:TF {
3754       \stex_annotate:nnn { comp }{\l_stex_current_symbol_str }{ #1 }
3755     }{
3756       \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3757     }
3758   }
3759 }
3760
3761 \cs_new_protected:Npn \_varcomp #1 {

```

```

3762 \str_if_empty:NF \l_stex_current_symbol_str {
3763   \stex_html_backend:TF {
3764     \stex_annotate:nnn { varcomp }{ \l_stex_current_symbol_str }{ #1 }
3765   }{
3766     \exp_args:Nnx \varemp@uri { #1 } { \l_stex_current_symbol_str }
3767   }
3768 }
3769 }
3770
3771 \def\comp{\_comp}
3772
3773 \cs_new_protected:Npn \compemph@uri #1 #2 {
3774   \compemph{ #1 }
3775 }
3776
3777
3778 \cs_new_protected:Npn \compemph #1 {
3779   #1
3780 }
3781
3782 \cs_new_protected:Npn \defemph@uri #1 #2 {
3783   \defemph{#1}
3784 }
3785
3786 \cs_new_protected:Npn \defemph #1 {
3787   \textbf{#1}
3788 }
3789
3790 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3791   \symrefemph{#1}
3792 }
3793
3794 \cs_new_protected:Npn \symrefemph #1 {
3795   \emph{#1}
3796 }
3797
3798 \cs_new_protected:Npn \varemp@uri #1 #2 {
3799   \varemp{#1}
3800 }
3801
3802 \cs_new_protected:Npn \varemp #1 {
3803   #1
3804 }

```

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

\ellipses

```

3805 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix 3806 \bool_new:N \l_stex_inparray_bool
\parrayline 3807 \bool_set_false:N \l_stex_inparray_bool
\parraylineh 3808 \NewDocumentCommand \parray { m m } {
\parraycell

```

```

3809 \begingroup
3810 \bool_set_true:N \l_stex_inarray_bool
3811 \begin{array}{#1}
3812 #2
3813 \end{array}
3814 \endgroup
3815 }
3816
3817 \NewDocumentCommand \prmatrix { m } {
3818 \begingroup
3819 \bool_set_true:N \l_stex_inarray_bool
3820 \begin{matrix}
3821 #1
3822 \end{matrix}
3823 \endgroup
3824 }
3825
3826 \def \maybepline {
3827 \bool_if:NT \l_stex_inarray_bool {\hline}
3828 }
3829
3830 \def \parrayline #1 #2 {
3831 #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3832 }
3833
3834 \def \pmrow #1 { \parrayline{}{ #1 } }
3835
3836 \def \parraylineh #1 #2 {
3837 #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3838 }
3839
3840 \def \parraycell #1 {
3841 #1 \bool_if:NT \l_stex_inarray_bool {&}
3842 }

```

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

30.4 Variables

```

3843 <@@=stex_variables>

```

\stex_invoke_variable:n Invokes a variable

```

3844 \cs_new_protected:Nn \stex_invoke_variable:n {
3845 \if_mode_math:
3846 \exp_after:wN \__stex_variables_invoke_math:n
3847 \else:
3848 \exp_after:wN \__stex_variables_invoke_text:n
3849 \fi: {#1}
3850 }
3851
3852 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
3853 %TODO
3854 }
3855

```



```

3856
3857 \cs_new_protected:Nn \__stex_variables_invoke_math:n {
3858   \peek_charcode_remove:NTF ! {
3859     \peek_charcode_remove:NTF ! {
3860       \peek_charcode:NTF [ {
3861         \__stex_variables_invoke_op_custom:nw
3862       }{
3863         % TODO throw error
3864       }
3865     }{
3866       \__stex_variables_invoke_op:n { #1 }
3867     }
3868   }{
3869     \peek_charcode_remove:NTF * {
3870       \__stex_variables_invoke_text:n { #1 }
3871     }{
3872       \__stex_variables_invoke_math_ii:n { #1 }
3873     }
3874   }
3875 }
3876
3877 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
3878   \cs_if_exist:cTF {
3879     stex_var_op_notation_ #1 _cs
3880   }{
3881     \exp_args:Nnx \use:nn {
3882       \def\comp{\_varcomp}
3883       \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3884       \_stex_term_omv:nn { var://#1 }{
3885         \use:c{stex_var_op_notation_ #1 _cs }
3886       }
3887     }{
3888       \_stex_reset:N \comp
3889       \_stex_reset:N \l_stex_current_symbol_str
3890     }
3891   }{
3892     \int_compare:nNnTF {\prop_item:cn {l_stex_variable_#1_prop}{arity}} = 0{
3893       \__stex_variables_invoke_math_ii:n {#1}
3894     }{
3895       \msg_error:nxxx{stex}{error/noop}{variable~#1}{}
3896     }
3897   }
3898 }
3899
3900 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n #1 {
3901   \cs_if_exist:cTF {
3902     stex_var_notation_#1_cs
3903   }{
3904     \tl_set:Nx \stex_symbol_after_invokation_tl {
3905       \_stex_reset:N \comp
3906       \_stex_reset:N \stex_symbol_after_invokation_tl
3907       \_stex_reset:N \l_stex_current_symbol_str
3908       \bool_set_true:N \l_stex_allow_semantic_bool
3909     }

```

```

3910     \def\comp{\_varcomp}
3911     \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3912     \bool_set_false:N \l_stex_allow_semantic_bool
3913     \use:c{stex_var_notation_#1_cs}
3914   }{
3915     \msg_error:nnxx{stex}{error/nonotation}{variable-#1}{s}
3916   }
3917 }

```

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

30.5 Sequences

```

3918 <@@=stex_sequences>
3919
3920 \cs_new_protected:Nn \stex_invoke_sequence:n {
3921   \peek_charcode_remove:NTF ! {
3922     \_stex_term_omv:nn {varseq://#1}{
3923       \exp_args:Nnx \use:nn {
3924         \def\comp{\_varcomp}
3925         \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3926         \prop_item:cn{stex_varseq_#1_prop}{notation}
3927       }{
3928         \_stex_reset:N \comp
3929         \_stex_reset:N \l_stex_current_symbol_str
3930       }
3931     }
3932   }{
3933     \bool_set_false:N \l_stex_allow_semantic_bool
3934     \def\comp{\_varcomp}
3935     \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3936     \tl_set:Nx \stex_symbol_after_invokation_tl {
3937       \_stex_reset:N \comp
3938       \_stex_reset:N \stex_symbol_after_invokation_tl
3939       \_stex_reset:N \l_stex_current_symbol_str
3940       \bool_set_true:N \l_stex_allow_semantic_bool
3941     }
3942     \use:c { stex_varseq_#1_cs }
3943   }
3944 }
3945 </package>

```

Chapter 31

STEX -Structural Features Implementation

```
3946 ⟨*package⟩
3947
3948 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3949
3950 Warnings and error messages
3951 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3952   Symbol~#1~can~not~be~assigned~in~copymodule~#2
3953 }
3954 \msg_new:nnn{stex}{error/interpretmodule/nodfiniens}{
3955   Symbol~#1~not~assigned~in~interpretmodule~#2
3956 }
3957 \msg_new:nnn{stex}{error/unknownstructure}{
3958   No~structure~#1~found!
3959 }
3960
3961 \msg_new:nnn{stex}{error/unknownfield}{
3962   No~field~#1~in~instance~#2~found!\#3
3963 }
3964
3965 \msg_new:nnn{stex}{error/keyval}{
3966   Invalid~key=value~pair:#1
3967 }
3968 \msg_new:nnn{stex}{error/instantiate/missing}{
3969   Assignments~missing~in~instantiate:~#1
3970 }
3971 \msg_new:nnn{stex}{error/incompatible}{
3972   Incompatible~signature:~#1~(#2)~and~#3~(#4)
3973 }
3974
```

31.1 Imports with modification

```

3975 <@@=stex_copymodule>
3976 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
3977   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3978     \tl_set:Nn \l_tmpa_tl { #1 }
3979     \__stex_copymodule_get_symbol_from_cs:
3980   }{
3981     % argument is a string
3982     % is it a command name?
3983     \cs_if_exist:cTF { #1 }{
3984       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3985       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3986       \str_if_empty:NNTF \l_tmpa_str {
3987         \exp_args:Nx \cs_if_eq:NNTF {
3988           \tl_head:N \l_tmpa_tl
3989         } \stex_invoke_symbol:n {
3990           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
3991         }{
3992           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3993         }
3994       } {
3995         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3996       }
3997     }{
3998       % argument is not a command name
3999       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4000       % \l_stex_all_symbols_seq
4001     }
4002   }
4003 }
4004
4005 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
4006   \str_set:Nn \l_tmpa_str { #1 }
4007   \bool_set_false:N \l_tmpa_bool
4008   \bool_if:NF \l_tmpa_bool {
4009     \tl_set:Nn \l_tmpa_tl {
4010       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
4011     }
4012     \str_set:Nn \l_tmpa_str { #1 }
4013     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4014     \seq_map_inline:Nn #2 {
4015       \str_set:Nn \l_tmpb_str { ##1 }
4016       \str_if_eq:eeT { \l_tmpa_str } {
4017         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4018       } {
4019         \seq_map_break:n {
4020           \tl_set:Nn \l_tmpa_tl {
4021             \str_set:Nn \l_stex_get_symbol_uri_str {
4022               ##1
4023             }
4024           }
4025         }
4026       }

```

```

4027     }
4028     \l_tmpa_tl
4029   }
4030 }
4031
4032 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
4033   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4034     { \tl_tail:N \l_tmpa_tl }
4035   \tl_if_single:NTF \l_tmpa_tl {
4036     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
4037       \exp_after:wN \str_set:Nn \exp_after:wN
4038         \l_stex_get_symbol_uri_str \l_tmpa_tl
4039       \__stex_copymodule_get_symbol_check:n { #1 }
4040     }{
4041       % TODO
4042       % tail is not a single group
4043     }
4044   }{
4045     % TODO
4046     % tail is not a single group
4047   }
4048 }
4049
4050 \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
4051   \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
4052     \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
4053       :~\seq_use:Nn #1 {,~}
4054     }
4055   }
4056 }
4057
4058 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
4059   % import module
4060   \stex_import_module_uri:nn { #1 } { #2 }
4061   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
4062   \stex_import_require_module:nnnn
4063     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4064     { \l_stex_import_path_str } { \l_stex_import_name_str }
4065
4066   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
4067   \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
4068
4069   % fields
4070   \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
4071   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4072     \seq_map_inline:cn {c_stex_module_##1_constants}{
4073       \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
4074         ##1 ? #####1
4075       }
4076     }
4077   }
4078
4079   % setup prop
4080   \seq_clear:N \l_tmpa_seq

```

```

4081 \exp_args:Nn \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
4082   name      = \l_stex_current_copymodule_name_str ,
4083   module    = \l_stex_current_module_str ,
4084   from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
4085   includes  = \l_tmpa_seq %,
4086 % fields    = \l_tmpa_seq
4087 }
4088 \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
4089   as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
4090 \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_seq {,
4091 \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,
4092
4093 \stex_if_do_html:T {
4094   \begin{stex_annotate_env} {#4} {
4095     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4096   }
4097   \stex_annotate_invisible:nnn{domain}{\l_stex_import_ns_str ?\l_stex_import_name_str}{}
4098 }
4099 }
4100
4101 \cs_new_protected:Nn \stex_copymodule_end:n {
4102   % apply to every field
4103   \def \l_tmpa_cs ##1 ##2 {#1}
4104
4105   \tl_clear:N \__stex_copymodule_module_tl
4106   \tl_clear:N \__stex_copymodule_exec_tl
4107
4108   %\prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
4109   \seq_clear:N \__stex_copymodule_fields_seq
4110
4111   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4112     \seq_map_inline:cn {c_stex_module_##1_constants}{
4113
4114       \tl_clear:N \__stex_copymodule_curr_symbol_tl % <- wrap in current symbol html
4115       \l_tmpa_cs{##1}{####1}
4116
4117       \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
4118         \str_set_eq:Nc \__stex_copymodule_curr_name_str {l__stex_copymodule_copymodule_##1?####1_name_str}
4119         \stex_if_do_html:T {
4120           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4121             \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?####1_name_str}}
4122           }
4123         }
4124       }{
4125         \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str / ##1 }
4126       }
4127
4128       \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
4129       \prop_put:Nnx \l_tmpa_prop { name } \__stex_copymodule_curr_name_str
4130       \prop_put:Nnx \l_tmpa_prop { module } \l_stex_current_module_str
4131
4132       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
4133         \stex_if_do_html:T {
4134           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {

```

```

4135         $\stex_annotate_invisible:nnn{definiens}{\exp_after:wN \exp_not:N\csname l__st
4136     }
4137 }
4138 \prop_put:Nnn \l_tmpa_prop { defined } { true }
4139 }
4140
4141 \stex_add_constant_to_current_module:n \__stex_copymodule_curr_name_str
4142 \tl_put_right:Nx \__stex_copymodule_module_tl {
4143     \seq_clear:c {l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_n
4144     \prop_set_from_keyval:cn {
4145         l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_name_str _prop
4146     }{
4147         \prop_to_keyval:N \l_tmpa_prop
4148     }
4149 }
4150
4151 \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
4152     \stex_if_do_html:T {
4153         \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4154             \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1
4155         }
4156     }
4157     \tl_put_right:Nx \__stex_copymodule_module_tl {
4158         \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
4159             \stex_invoke_symbol:n {
4160                 \l_stex_current_module_str ? \__stex_copymodule_curr_name_str
4161             }
4162         }
4163     }
4164 }
4165
4166 \seq_put_right:Nx \__stex_copymodule_fields_seq {\l_stex_current_module_str ? \__stex_
4167
4168 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4169     \stex_copy_notations:nn {\l_stex_current_module_str ? \__stex_copymodule_curr_name_s
4170 }
4171
4172 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4173     \stex_if_do_html:TF{
4174         \stex_annotate_invisible:nnn{assignment} {##1?####1} { \exp_after:wN \exp_not:n \e
4175     }{
4176         \exp_after:wN \exp_not:n \exp_after:wN {\__stex_copymodule_curr_symbol_tl}
4177     }
4178 }
4179 }
4180 }
4181
4182
4183 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \__stex_copymodule_fields_seq
4184 \tl_put_left:Nx \__stex_copymodule_module_tl {
4185     \prop_set_from_keyval:cn {
4186         l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _pro
4187     }{
4188         \prop_to_keyval:N \l_stex_current_copymodule_prop

```

```

4189     }
4190   }
4191
4192   \seq_gput_right:cx{c_stex_module_\l_stex_current_module_str _copymodules}{
4193     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4194   }
4195
4196   \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4197   \stex_debug:nn{copymodule}{result:\meaning \__stex_copymodule_module_tl}
4198   \stex_debug:nn{copymodule}{output:\meaning \__stex_copymodule_exec_tl}
4199
4200   \__stex_copymodule_exec_tl
4201   \stex_if_do_html:T {
4202     \end{stex_annotate_env}
4203   }
4204 }
4205
4206 \NewDocumentEnvironment {copymodule} { 0{} m m}{
4207   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ copymodule }
4208   \stex_deactivate_macro:Nn \symdecl {module~environments}
4209   \stex_deactivate_macro:Nn \symdef {module~environments}
4210   \stex_deactivate_macro:Nn \notation {module~environments}
4211   \stex_reactivate_macro:N \assign
4212   \stex_reactivate_macro:N \renamedekl
4213   \stex_reactivate_macro:N \donotcopy
4214   \stex_smsmode_do:
4215 }{
4216   \stex_copymodule_end:n {}
4217 }
4218
4219 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
4220   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ interpretmodule }
4221   \stex_deactivate_macro:Nn \symdecl {module~environments}
4222   \stex_deactivate_macro:Nn \symdef {module~environments}
4223   \stex_deactivate_macro:Nn \notation {module~environments}
4224   \stex_reactivate_macro:N \assign
4225   \stex_reactivate_macro:N \renamedekl
4226   \stex_reactivate_macro:N \donotcopy
4227   \stex_smsmode_do:
4228 }{
4229   \stex_copymodule_end:n {
4230     \tl_if_exist:cF {
4231       l__stex_copymodule_copymodule_##1?##2_def_tl
4232     }{
4233       \str_if_eq:eeF {
4234         \prop_item:cn{
4235           l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
4236         }{ true }{
4237           \msg_error:nxxx{stex}{error/interpretmodule/nodéfiniens}{
4238             ##1?##2
4239           }{\l_stex_current_copymodule_name_str}
4240         }
4241       }
4242     }

```



```

4243 }
4244
4245 \iffalse \begin{stex_annotate_env} \fi
4246 \NewDocumentEnvironment {realization} { 0 } { m } {
4247   \stex_copymodule_start:nnnn { #1 } { #2 } { #2 } { realize }
4248   \stex_deactivate_macro:Nn \symdecl {module~environments}
4249   \stex_deactivate_macro:Nn \symdef {module~environments}
4250   \stex_deactivate_macro:Nn \notation {module~environments}
4251   \stex_reactivate_macro:N \donotcopy
4252   \stex_reactivate_macro:N \assign
4253   \stex_smsmode_do:
4254 } {
4255   \stex_import_module_uri:nn { #1 } { #2 }
4256   \tl_clear:N \__stex_copymodule_exec_tl
4257   \tl_set:Nx \__stex_copymodule_module_tl {
4258     \stex_import_require_module:nnnn
4259     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4260     { \l_stex_import_path_str } { \l_stex_import_name_str }
4261   }
4262
4263   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4264     \seq_map_inline:cn {c_stex_module_##1_constants}{
4265       \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str / #1 }
4266       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?###1_def_tl}{
4267         \stex_if_do_html:T {
4268           \tl_put_right:Nx \__stex_copymodule_exec_tl {
4269             \stex_annotate_invisible:nnn{assignment} {##1?###1} {
4270               $\stex_annotate_invisible:nnn{definien}{}{\exp_after:wN \exp_not:N\csname l__
4271             }
4272           }
4273         }
4274         \tl_put_right:Nx \__stex_copymodule_module_tl {
4275           \prop_put:cnn {l_stex_symdecl_##1?###1_prop}{ defined }{ true }
4276         }
4277       }
4278     }
4279
4280     \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4281
4282     \__stex_copymodule_exec_tl
4283     \stex_if_do_html:T {\end{stex_annotate_env}}
4284   }
4285
4286   \NewDocumentCommand \donotcopy { m } {
4287     \str_clear:N \l_stex_import_name_str
4288     \str_set:Nn \l_tmpa_str { #1 }
4289     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4290     \seq_map_inline:Nn \l_stex_all_modules_seq {
4291       \str_set:Nn \l_tmpb_str { ##1 }
4292       \str_if_eq:eeT { \l_tmpa_str } {
4293         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4294       } {
4295         \seq_map_break:n {
4296           \stex_if_do_html:T {

```

```

4297         \stex_if_smsmode:F {
4298             \stex_annotate_invisible:nnn{donotcopy}{##1}{
4299                 \stex_annotate:nnn{domain}{##1}{}}
4300         }
4301     }
4302 }
4303 \str_set_eq:NN \l_stex_import_name_str \l_tmpb_str
4304 }
4305 }
4306 \seq_map_inline:cn {c_stex_module_###1_copymodules}{
4307     \str_set:Nn \l_tmpb_str { #####1 }
4308     \str_if_eq:eeT { \l_tmpa_str } {
4309         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4310     } {
4311         \seq_map_break:n {\seq_map_break:n {
4312             \stex_if_do_html:T {
4313                 \stex_if_smsmode:F {
4314                     \stex_annotate_invisible:nnn{donotcopy}{#####1}{
4315                         \stex_annotate:nnn{domain}{
4316                             \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4317                         }{}}
4318                 }
4319             }
4320         }
4321         \str_set:Nx \l_stex_import_name_str {
4322             \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4323         }
4324     }}
4325 }
4326 }
4327 }
4328 \str_if_empty:NTF \l_stex_import_name_str {
4329     % TODO throw error
4330 }{
4331     \stex_collect_imports:n {\l_stex_import_name_str }
4332     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4333         \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
4334         \seq_map_inline:cn {c_stex_module_###1_constants}{
4335             \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? #####1 }
4336             \bool_lazy_any:nT {
4337                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_name_str}}
4338                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_macroname_str}}
4339                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_def_tl}}
4340             }{
4341                 % TODO throw error
4342             }
4343         }
4344     }
4345     \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
4346     \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_name_str }
4347     \prop_put:Nno \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
4348 }
4349 \stex_smsmode_do:
4350 }

```

```

4351
4352 \NewDocumentCommand \assign { m m }{
4353   \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
4354   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
4355   \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_def_tl}{#2}
4356   \stex_smsmode_do:
4357 }
4358
4359 \keys_define:nn { stex / renamedekl } {
4360   name          .str_set_x:N = \l_stex_renamedekl_name_str
4361 }
4362 \cs_new_protected:Nn \__stex_copymodule_renamedekl_args:n {
4363   \str_clear:N \l_stex_renamedekl_name_str
4364   \keys_set:nn { stex / renamedekl } { #1 }
4365 }
4366
4367 \NewDocumentCommand \renamedekl { O{} m m }{
4368   \__stex_copymodule_renamedekl_args:n { #1 }
4369   \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
4370   \stex_debug:nn{renamedekl}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
4371   \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_macroname_str}{#3}
4372   \str_if_empty:NTF \l_stex_renamedekl_name_str {
4373     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4374       \l_stex_get_symbol_uri_str
4375     } }
4376   } {
4377     \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_name_str}{\l_stex
4378       \stex_debug:nn{renamedekl}{@~\l_stex_current_module_str ? \l_stex_renamedekl_name_str}
4379       \prop_set_eq:cc {l_stex_symdecl_
4380         \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4381         _prop
4382       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
4383       \seq_set_eq:cc {l_stex_symdecl_
4384         \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4385         _notations
4386       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
4387       \prop_put:cnx {l_stex_symdecl_
4388         \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4389         _prop
4390       }{ name }{ \l_stex_renamedekl_name_str }
4391       \prop_put:cnx {l_stex_symdecl_
4392         \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4393         _prop
4394       }{ module }{ \l_stex_current_module_str }
4395       \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
4396         \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4397       }
4398       \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4399         \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4400       } }
4401     }
4402     \stex_smsmode_do:
4403   }
4404

```

```

4405 \stex_deactivate_macro:Nn \assign {copymodules}
4406 \stex_deactivate_macro:Nn \renamedekl {copymodules}
4407 \stex_deactivate_macro:Nn \donotcopy {copymodules}
4408
4409

```

31.2 The feature environment

structural@feature

```

4410 <@@=stex_features>
4411
4412 \NewDocumentEnvironment{structural_feature_module}{ m m m }{
4413   \stex_if_in_module:F {
4414     \msg_set:nnn{stex}{error/nomodule}{
4415       Structural~Feature~has~to~occur~in~a~module:\\
4416       Feature~#2~of~type~#1\\
4417       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
4418     }
4419     \msg_error:nn{stex}{error/nomodule}
4420   }
4421
4422   \str_set_eq:NN \l_stex_feature_parent_str \l_stex_current_module_str
4423
4424   \stex_module_setup:nn{meta=NONE}{#2 - #1}
4425
4426   \stex_if_do_html:T {
4427     \begin{stex_annotate_env}{ feature:#1 }{\l_stex_feature_parent_str ? #2 - #1}
4428     \stex_annotate_invisible:nnn{header}{\{ #3 }
4429   }
4430 }{
4431   \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
4432   \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
4433   \stex_debug:nn{features}{
4434     Feature: \l_stex_last_feature_str
4435   }
4436   \stex_if_do_html:T {
4437     \end{stex_annotate_env}
4438   }
4439 }

```

31.3 Structure

structure

```

4440 <@@=stex_structures>
4441 \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
4442   \prop_if_exist:cF {c_stex_module_ \l_stex_current_module_str _structures}{
4443     \prop_new:c {c_stex_module_ \l_stex_current_module_str _structures}
4444   }
4445   \prop_gput:cxn{c_stex_module_ \l_stex_current_module_str _structures}
4446   {#1}{#2}
4447 }
4448

```

```

4449 \keys_define:nn { stex / features / structure } {
4450   name          .str_set_x:N = \l__stex_structures_name_str ,
4451 }
4452
4453 \cs_new_protected:Nn \__stex_structures_structure_args:n {
4454   \str_clear:N \l__stex_structures_name_str
4455   \keys_set:nn { stex / features / structure } { #1 }
4456 }
4457
4458 \NewDocumentEnvironment{mathstructure}{m O{}}{
4459   \__stex_structures_structure_args:n { #2 }
4460   \str_if_empty:NT \l__stex_structures_name_str {
4461     \str_set:Nx \l__stex_structures_name_str { #1 }
4462   }
4463   \stex_suppress_html:n {
4464     \exp_args:Nx \stex_symdecl_do:nn {
4465       name = \l__stex_structures_name_str ,
4466       def  = {\STEXsymbol{module-type}}{
4467         \_stex_term_math_oms:nnnn {
4468           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4469             { ns } ?
4470           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4471             { name } / \l__stex_structures_name_str - structure
4472         }{}{0}{}
4473       }}
4474     }{ #1 }
4475   }
4476   \exp_args:Nnnx
4477   \begin{structural_feature_module}{ structure }
4478     { \l__stex_structures_name_str }{}
4479   \stex_smsmode_do:
4480 }{
4481   \end{structural_feature_module}
4482   \stex_reset_up_to_module:n \l_stex_last_feature_str
4483   \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
4484   \seq_clear:N \l_tmpa_seq
4485   \seq_map_inline:Nn \l_stex_collect_imports_seq {
4486     \seq_map_inline:cn{c_stex_module_##1_constants}{
4487       \seq_put_right:Nn \l_tmpa_seq { ##1 ? ####1 }
4488     }
4489   }
4490   \exp_args:Nnno
4491   \prop_gput:cnn {c_stex_module_\l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
4492   \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
4493   \stex_add_structure_to_current_module:nn
4494     \l__stex_structures_name_str
4495     \l_stex_last_feature_str
4496
4497   \stex_execute_in_module:x {
4498     \tl_set:cn { #1 }{
4499       \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures
4500     }
4501   }
4502 }

```

```

4503
4504 \cs_new:Nn \stex_invoke_structure:nn {
4505   \stex_invoke_symbol:n { #1?#2 }
4506 }
4507
4508 \cs_new_protected:Nn \stex_get_structure:n {
4509   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4510     \tl_set:Nn \l_tmpa_tl { #1 }
4511     \__stex_structures_get_from_cs:
4512   }{
4513     \cs_if_exist:cTF { #1 }{
4514       \cs_set_eq:Nc \l_tmpa_cs { #1 }
4515       \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
4516       \str_if_empty:NNTF \l_tmpa_str {
4517         \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs } \stex_invoke_structure:nn {
4518           \__stex_structures_get_from_cs:
4519         }{
4520           \__stex_structures_get_from_string:n { #1 }
4521         }
4522       }{
4523         \__stex_structures_get_from_string:n { #1 }
4524       }
4525     }{
4526       \__stex_structures_get_from_string:n { #1 }
4527     }
4528   }
4529 }
4530
4531 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
4532   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4533     { \tl_tail:N \l_tmpa_tl }
4534   \str_set:Nx \l_tmpa_str {
4535     \exp_after:wN \use_i:nn \l_tmpa_tl
4536   }
4537   \str_set:Nx \l_tmpb_str {
4538     \exp_after:wN \use_ii:nn \l_tmpa_tl
4539   }
4540   \str_set:Nx \l_stex_get_structure_str {
4541     \l_tmpa_str ? \l_tmpb_str
4542   }
4543   \str_set:Nx \l_stex_get_structure_module_str {
4544     \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
4545   }
4546 }
4547
4548 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
4549   \tl_set:Nn \l_tmpa_tl {
4550     \msg_error:nnn{stex}{error/unknownstructure}{#1}
4551   }
4552   \str_set:Nn \l_tmpa_str { #1 }
4553   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4554
4555   \seq_map_inline:Nn \l_stex_all_modules_seq {
4556     \prop_if_exist:cT {c_stex_module_##1_structures} {

```

```

4557 \prop_map_inline:cn {c_stex_module_##1_structures} {
4558   \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
4559     \prop_map_break:n{\seq_map_break:n{
4560       \tl_set:Nn \l_tmpa_tl {
4561         \str_set:Nn \l_stex_get_structure_str {##1?####1}
4562         \str_set:Nn \l_stex_get_structure_module_str {####2}
4563       }
4564     }}
4565   }
4566 }
4567 }
4568 }
4569 \l_tmpa_tl
4570 }

```

\instantiate

```

4571
4572 \keys_define:nn { stex / instantiate } {
4573   name          .str_set_x:N = \l__stex_structures_name_str
4574 }
4575 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
4576   \str_clear:N \l__stex_structures_name_str
4577   \keys_set:nn { stex / instantiate } { #1 }
4578 }
4579
4580 \NewDocumentCommand \instantiate {m O{} m m O{}}{
4581   \begin{group}
4582     \stex_get_structure:n {#3}
4583     \__stex_structures_instantiate_args:n { #2 }
4584     \str_if_empty:NT \l__stex_structures_name_str {
4585       \str_set:Nn \l__stex_structures_name_str { #1 }
4586     }
4587     \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4588     \seq_clear:N \l__stex_structures_fields_seq
4589     \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4590     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4591       \seq_map_inline:cn {c_stex_module_##1_constants}{
4592         \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4593       }
4594     }
4595
4596     \tl_if_empty:nF{#5}{
4597       \seq_set_split:Nnn \l_tmpa_seq , {#5}
4598       \prop_clear:N \l_tmpa_prop
4599       \seq_map_inline:Nn \l_tmpa_seq {
4600         \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4601         \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4602           \msg_error:nnn{stex}{error/keyval}{##1}
4603         }
4604         \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
4605         \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4606         \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
4607         \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}
4608         \exp_args:Nxx \str_if_eq:nnF

```

```

4609         {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4610         {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
4611         \msg_error:nnxxxx{stex}{error/incompatible}
4612         {\l__stex_structures_dom_str}
4613         {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4614         {\l_stex_get_symbol_uri_str}
4615         {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
4616     }
4617     \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str
4618 }
4619 }
4620
4621 \seq_map_inline:Nn \l__stex_structures_fields_seq {
4622     \str_set:Nx \l_tmpa_str {field:\l__stex_structures_name_str . \prop_item:cn {l_stex_sy
4623     \stex_debug:nn{instantiate}{Field~\l_tmpa_str :~##1}
4624
4625     \stex_add_constant_to_current_module:n {\l_tmpa_str}
4626     \stex_execute_in_module:x {
4627         \prop_set_from_keyval:cn { l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _p
4628         name = \l_tmpa_str ,
4629         args = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4630         arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4631         assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4632     }
4633     \seq_clear:c {l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _notations}
4634 }
4635
4636 \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4637     \stex_find_notation:nn{##1}{}
4638     \stex_execute_in_module:x {
4639         \seq_put_right:cn {l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _notation
4640     }
4641
4642     \stex_copy_control_sequence_ii:ccN
4643     {stex_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_notation_
4644     {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4645     \l_tmpa_tl
4646     \exp_args:No \stex_execute_in_module:n \l_tmpa_tl
4647
4648
4649     \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
4650         \tl_set_eq:Nc \l_tmpa_cs {stex_op_notation_##1\c_hash_str \l_stex_notation_variant
4651         \stex_execute_in_module:x {
4652             \tl_set:cn
4653             {stex_op_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_not
4654             { \exp_args:No \exp_not:n \l_tmpa_cs}
4655         }
4656     }
4657
4658 }
4659
4660 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\l_stex_cur
4661 }
4662

```



```

4663 \stex_execute_in_module:x {
4664   \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
4665   domain = \l_stex_get_structure_module_str ,
4666   \prop_to_keyval:N \l_tmpa_prop
4667 }
4668 \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
4669 }
4670 \stex_debug:nn{instantiate}{
4671   Instance~\l_stex_current_module_str?\l__stex_structures_name_str \
4672   \prop_to_keyval:N \l_tmpa_prop
4673 }
4674 \exp_args:Nxx \stex_symdecl_do:nn {
4675   type={\STEXsymbol{module-type}}{
4676     \stex_term_math_oms:nnnn {
4677       \l_stex_get_structure_module_str
4678     }{}{0}{}
4679   }}
4680 }{\l__stex_structures_name_str}
4681 % {
4682   \str_set:Nx \l_stex_get_symbol_uri_str {\l_stex_current_module_str?\l__stex_structures
4683   \tl_set:Nn \l_stex_notation_after_do_tl {\__stex_notation_final:}
4684   \stex_notation_do:nnnnn{}{}{}{}{\comp{#4}}
4685 % }
4686 %\exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}
4687 \endgroup
4688 \stex_smsmode_do:\ignorespacesandpars
4689 }
4690
4691 \cs_new_protected:Nn \stex_symbol_or_var:n {
4692   \cs_if_exist:cTF{#1}{
4693     \cs_set_eq:Nc \l_tmpa_tl { #1 }
4694     \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4695     \str_if_empty:NTF \l_tmpa_str {
4696       \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
4697       \stex_invoke_variable:n {
4698         \bool_set_true:N \l_stex_symbol_or_var_bool
4699         \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
4700         \str_set:Nx \l_stex_get_symbol_uri_str {
4701           \exp_after:wN \use:n \l_tmpa_tl
4702         }
4703       }{
4704         \bool_set_false:N \l_stex_symbol_or_var_bool
4705         \stex_get_symbol:n{#1}
4706       }
4707     }{
4708       \__stex_structures_symbolorvar_from_string:n{ #1 }
4709     }
4710   }{
4711     \__stex_structures_symbolorvar_from_string:n{ #1 }
4712   }
4713 }
4714
4715 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
4716   \prop_if_exist:cTF {l_stex_variable_#1 _prop}{

```

```

4717     \bool_set_true:N \l_stex_symbol_or_var_bool
4718     \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
4719   }{
4720     \bool_set_false:N \l_stex_symbol_or_var_bool
4721     \stex_get_symbol:n{#1}
4722   }
4723 }
4724
4725 \keys_define:nn { stex / varinstantiate } {
4726   name      .str_set_x:N = \l__stex_structures_name_str,
4727   bind      .choices:nn =
4728     {forall,exists}
4729     {\str_set:Nx \l__stex_structures_bind_str {\l_keys_choice_tl}}
4730 }
4731 }
4732 \cs_new_protected:Nn \__stex_structures_varinstantiate_args:n {
4733   \str_clear:N \l__stex_structures_name_str
4734   \str_clear:N \l__stex_structures_bind_str
4735   \keys_set:nn { stex / varinstantiate } { #1 }
4736 }
4737
4738 \NewDocumentCommand \varinstantiate {m O{} m m O{}}{
4739   \begingroup
4740     \stex_get_structure:n {#3}
4741     \__stex_structures_varinstantiate_args:n { #2 }
4742     \str_if_empty:NT \l__stex_structures_name_str {
4743       \str_set:Nn \l__stex_structures_name_str { #1 }
4744     }
4745     \stex_if_do_html:TF{
4746       \stex_annotate:nnn{varinstance}{\l__stex_structures_name_str}
4747     }{\use:n}
4748     {
4749       \stex_if_do_html:T{
4750         \stex_annotate_invisible:nnn{domain}{\l_stex_get_structure_module_str}{}}
4751       }
4752       \seq_clear:N \l__stex_structures_fields_seq
4753       \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4754       \seq_map_inline:Nn \l_stex_collect_imports_seq {
4755         \seq_map_inline:cn {c_stex_module_##1_constants}{
4756           \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4757         }
4758       }
4759       \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4760       \prop_clear:N \l_tmpa_prop
4761       \tl_if_empty:nF {#5} {
4762         \seq_set_split:Nnn \l_tmpa_seq , {#5}
4763         \seq_map_inline:Nn \l_tmpa_seq {
4764           \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4765           \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4766             \msg_error:nnn{stex}{error/keyval}{##1}
4767           }
4768           \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_stru
4769           \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4770           \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol

```

```

4771 \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
4772 \stex_if_do_html:T{
4773   \stex_annotate:nnn{assign}{\l__stex_structures_dom_str,\l_stex_get_symbol_uri_str}
4774 }
4775 \bool_if:NTF \l_stex_symbol_or_var_bool {
4776   \exp_args:Nxx \str_if_eq:nnF
4777     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4778     {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}{
4779     \msg_error:nnxxxx{stex}{error/incompatible}
4780     {\l__stex_structures_dom_str}
4781     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4782     {\l_stex_get_symbol_uri_str}
4783     {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}
4784   }
4785   \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n}
4786 }{
4787   \exp_args:Nxx \str_if_eq:nnF
4788     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4789     {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
4790     \msg_error:nnxxxx{stex}{error/incompatible}
4791     {\l__stex_structures_dom_str}
4792     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4793     {\l_stex_get_symbol_uri_str}
4794     {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
4795   }
4796   \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n}
4797 }
4798 }
4799 }
4800 \tl_gclear:N \g__stex_structures_aftergroup_tl
4801 \seq_map_inline:Nn \l__stex_structures_fields_seq {
4802   \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdecl_\l__stex_structures_fields_seq}{args}}
4803   \stex_debug:nn{varinstantiate}{Field~\l_tmpa_str :~##1}
4804   \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4805     \stex_find_notation:nn{##1}{
4806       \cs_gset_eq:cc{g__stex_structures_tmpa_\l_tmpa_str _cs}
4807         {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4808       \stex_debug:nn{varinstantiate}{Notation:~\cs_meaning:c{g__stex_structures_tmpa_\l_tmpa_str _cs}}
4809       \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
4810         \cs_gset_eq:cc {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
4811           {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4812         \stex_debug:nn{varinstantiate}{Operator~Notation:~\cs_meaning:c{g__stex_structures_tmpa_op_\l_tmpa_str _cs}}
4813       }
4814     }
4815   }
4816   \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4817     \prop_set_from_keyval:cn { l_stex_variable_ \l_tmpa_str _prop}{
4818       name = \l_tmpa_str ,
4819       args = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4820       arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4821       assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4822     }
4823     \cs_set_eq:cc {stex_var_notation_\l_tmpa_str _cs}
4824     {g__stex_structures_tmpa_\l_tmpa_str _cs}

```

```

4825         \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str_cs}
4826         {g__stex_structures_tmpa_op_\l_tmpa_str_cs}
4827     }
4828     \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\stex_inv
4829 }
4830 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4831     \prop_set_from_keyval:cn {l_stex_varinstance_\l__stex_structures_name_str_prop }{
4832         domain = \l_stex_get_structure_module_str ,
4833         \prop_to_keyval:N \l_tmpa_prop
4834     }
4835     \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
4836     \tl_set:cn {l_stex_varinstance_\l__stex_structures_name_str_op_tl}{
4837         \exp_args:Nnx \exp_not:N \use:nn {
4838             \str_set:Nn \exp_not:N \l_stex_current_symbol_str {var://\l__stex_structures_nam
4839             \_stex_term_omv:nn {var://\l__stex_structures_name_str}{
4840                 \exp_not:n{
4841                     \_varcomp{#4}
4842                 }
4843             }
4844             }{
4845                 \exp_not:n{\_stex_reset:N \l_stex_current_symbol_str}
4846             }
4847         }
4848     }
4849 }
4850 \stex_debug:nn{varinstantiate}{\expandafter\detokenize\expandafter{\g__stex_structures_a
4851 \aftergroup\g__stex_structures_aftergroup_tl
4852 \endgroup
4853 \stex_smsmode_do:\ignorespacesandpars
4854 }
4855
4856 \cs_new_protected:Nn \stex_invoke_instance:n {
4857     \peek_charcode_remove:NTF ! {
4858         \stex_invoke_symbol:n{#1}
4859     }{
4860         \_stex_invoke_instance:nn {#1}
4861     }
4862 }
4863
4864
4865 \cs_new_protected:Nn \stex_invoke_varinstance:n {
4866     \peek_charcode_remove:NTF ! {
4867         \exp_args:Nnx \use:nn {
4868             \def\comp{\_varcomp}
4869             \use:c{l_stex_varinstance_#1_op_tl}
4870         }{
4871             \_stex_reset:N \comp
4872         }
4873     }{
4874         \_stex_invoke_varinstance:nn {#1}
4875     }
4876 }
4877
4878 \cs_new_protected:Nn \stex_invoke_instance:nn {

```

```

4879 \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
4880   \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
4881 }{
4882   \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
4883   \msg_error:nnxxx{stex}{error/unknownfield}{#2}{#1}{
4884     \prop_to_keyval:N \l_tmpa_prop
4885   }
4886 }
4887 }
4888
4889 \cs_new_protected:Nn \stex_invoke_varinstance:nn {
4890   \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
4891     \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
4892     \l_tmpa_tl
4893   }{
4894     \msg_error:nnnnn{stex}{error/unknownfield}{#2}{#1}{}
4895   }
4896 }

```

(End definition for `\instantiate`. This function is documented on page 32.)

`\stex_invoke_structure:nnn`

```

4897 % #1: URI of the instance
4898 % #2: URI of the instantiated module
4899 \cs_new_protected:Nn \stex_invoke_structure:nnn {
4900   \tl_if_empty:nTF{ #3 }{
4901     \prop_set_eq:Nc \l__stex_structures_structure_prop {
4902       c_stex_feature_ #2 _prop
4903     }
4904     \tl_clear:N \l_tmpa_tl
4905     \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
4906     \seq_map_inline:Nn \l_tmpa_seq {
4907       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
4908       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
4909       \cs_if_exist:cT {
4910         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
4911       }{
4912         \tl_if_empty:NF \l_tmpa_tl {
4913           \tl_put_right:Nn \l_tmpa_tl {,}
4914         }
4915         \tl_put_right:Nx \l_tmpa_tl {
4916           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
4917         }
4918       }
4919     }
4920     \exp_args:No \mathstrut \l_tmpa_tl
4921   }{
4922     \stex_invoke_symbol:n{#1/#3}
4923   }
4924 }

```

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

```

4925 </package>

```

Chapter 32

STEX -Statements Implementation

```
4926 <*package>
4927
4928 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4929
4930 <@@=stex_statements>
    Warnings and error messages
4931
\titleemph
4932 \def\titleemph#1{\textbf{#1}}
    (End definition for \titleemph. This function is documented on page ??.)
```

32.1 Definitions

definiendum

```
4933 \keys_define:nn {stex / definiendum }{
4934   pre      .tl_set:N      = \l__stex_statements_definiendum_pre_tl,
4935   post     .tl_set:N      = \l__stex_statements_definiendum_post_tl,
4936   root     .str_set_x:N    = \l__stex_statements_definiendum_root_str,
4937   gfa      .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
4938 }
4939 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
4940   \str_clear:N \l__stex_statements_definiendum_root_str
4941   \tl_clear:N \l__stex_statements_definiendum_post_tl
4942   \str_clear:N \l__stex_statements_definiendum_gfa_str
4943   \keys_set:nn { stex / definiendum }{ #1 }
4944 }
4945 \NewDocumentCommand \definiendum { O{} m m } {
4946   \__stex_statements_definiendum_args:n { #1 }
4947   \stex_get_symbol:n { #2 }
4948   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4949   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
4950     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

```

4951     \tl_set:Nn \l_tmpa_tl { #3 }
4952   } {
4953     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
4954     \tl_set:Nn \l_tmpa_tl {
4955       \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
4956     }
4957   }
4958 } {
4959   \tl_set:Nn \l_tmpa_tl { #3 }
4960 }
4961
4962 % TODO root
4963 \stex_html_backend:TF {
4964   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
4965 } {
4966   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
4967 }
4968 }
4969 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

(End definition for definiendum. This function is documented on page 41.)

definame

```

4970
4971 \NewDocumentCommand \definame { 0{ } m } {
4972   \__stex_statements_definiendum_args:n { #1 }
4973   % TODO: root
4974   \stex_get_symbol:n { #2 }
4975   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4976   \str_set:Nx \l_tmpa_str {
4977     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4978   }
4979   \str_replace_all:Nnn \l_tmpa_str {-} {~}
4980   \stex_html_backend:TF {
4981     \stex_if_do_html:T {
4982       \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4983         \l_tmpa_str\l__stex_statements_definiendum_post_tl
4984       }
4985     }
4986   } {
4987     \exp_args:Nnx \defemph@uri {
4988       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4989     } { \l_stex_get_symbol_uri_str }
4990   }
4991 }
4992 \stex_deactivate_macro:Nn \definame {definition~environments}
4993
4994 \NewDocumentCommand \Definame { 0{ } m } {
4995   \__stex_statements_definiendum_args:n { #1 }
4996   \stex_get_symbol:n { #2 }
4997   \str_set:Nx \l_tmpa_str {
4998     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4999   }
5000   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}

```

```

5001 \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5002 \stex_html_backend:TF {
5003   \stex_if_do_html:T {
5004     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5005       \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5006     }
5007   }
5008 } {
5009   \exp_args:Nnx \defemph@uri {
5010     \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5011   } { \l_stex_get_symbol_uri_str }
5012 }
5013 }
5014 \stex_deactivate_macro:Nn \Definame {definition~environments}
5015
5016 \NewDocumentCommand \premise { m }{
5017   \stex_annotate:nnn{ premise }{}{ #1 }
5018 }
5019 \NewDocumentCommand \conclusion { m }{
5020   \stex_annotate:nnn{ conclusion }{}{ #1 }
5021 }
5022 \NewDocumentCommand \definiens { 0{} m }{
5023   \str_clear:N \l_stex_get_symbol_uri_str
5024   \tl_if_empty:nF {#1} {
5025     \stex_get_symbol:n { #1 }
5026   }
5027   \str_if_empty:NT \l_stex_get_symbol_uri_str {
5028     \int_compare:nNnTF {\clist_count:N \l__stex_statements_sdefinition_for_clist} = 1 {
5029       \str_set:Nx \l_stex_get_symbol_uri_str {\clist_item:Nn \l__stex_statements_sdefinition
5030     }{
5031       % TODO throw error
5032     }
5033   }
5034   \str_if_eq:eeT {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{module}}
5035   {\l_stex_current_module_str}{
5036     \str_if_eq:eeF {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defin
5037   }{true}{
5038     \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5039     \exp_args:Nx \stex_add_to_current_module:n {
5040       \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5041     }
5042   }
5043 }
5044 \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
5045 }
5046
5047 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
5048 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
5049 \stex_deactivate_macro:Nn \definiens {definition~environments}
5050

```

(End definition for `definame`. This function is documented on page 41.)

`sdefinition`


```

5051
5052 \keys_define:nn {stex / sdefinition }{
5053   type      .str_set_x:N = \sdefinitiontype,
5054   id        .str_set_x:N = \sdefinitionid,
5055   name      .str_set_x:N = \sdefinitionname,
5056   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5057   title     .tl_set:N     = \sdefinitiontitle
5058 }
5059 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
5060   \str_clear:N \sdefinitiontype
5061   \str_clear:N \sdefinitionid
5062   \str_clear:N \sdefinitionname
5063   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5064   \tl_clear:N \sdefinitiontitle
5065   \keys_set:nn { stex / sdefinition }{ #1 }
5066 }
5067
5068 \NewDocumentEnvironment{sdefinition}{0{}}{
5069   \__stex_statements_sdefinition_args:n{ #1 }
5070   \stex_reactivate_macro:N \definiendum
5071   \stex_reactivate_macro:N \definame
5072   \stex_reactivate_macro:N \Definame
5073   \stex_reactivate_macro:N \premise
5074   \stex_reactivate_macro:N \definiens
5075   \stex_if_smsmode:F{
5076     \seq_clear:N \l_tmpa_seq
5077     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5078       \tl_if_empty:NF{ ##1 }{
5079         \stex_get_symbol:n { ##1 }
5080         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5081           \l_stex_get_symbol_uri_str
5082         }
5083       }
5084     }
5085     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpa_seq
5086     \exp_args:Nnnx
5087     \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {},}}
5088     \str_if_empty:NF \sdefinitiontype {
5089       \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5090     }
5091     \str_if_empty:NF \sdefinitionname {
5092       \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5093     }
5094     \clist_set:Nn \l_tmpa_clist \sdefinitiontype
5095     \tl_clear:N \l_tmpa_tl
5096     \clist_map_inline:Nn \l_tmpa_clist {
5097       \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
5098         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
5099       }
5100     }
5101     \tl_if_empty:NTF \l_tmpa_tl {
5102       \__stex_statements_sdefinition_start:
5103     }{
5104       \l_tmpa_tl

```

```

5105     }
5106   }
5107   \stex_ref_new_doc_target:n \sdefinitionid
5108   \stex_smsmode_do:
5109 }{
5110   \stex_suppress_html:n {
5111     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5112   }
5113   \stex_if_smsmode:F {
5114     \clist_set:No \l_tmpa_clist \sdefinitiontype
5115     \tl_clear:N \l_tmpa_tl
5116     \clist_map_inline:Nn \l_tmpa_clist {
5117       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
5118         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
5119       }
5120     }
5121     \tl_if_empty:NTF \l_tmpa_tl {
5122       \__stex_statements_sdefinition_end:
5123     }{
5124       \l_tmpa_tl
5125     }
5126     \end{stex_annotate_env}
5127   }
5128 }

```

\stexpatchdefinition

```

5129 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
5130   \stex_par:\noindent\titleemph{Definition\tl_if_empty:NF \sdefinitiontitle {
5131     ~(\sdefinitiontitle)
5132   }~}
5133 }
5134 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\stex_par:\medskip}
5135
5136 \newcommand\stexpatchdefinition[3] [] {
5137   \str_set:Nx \l_tmpa_str{ #1 }
5138   \str_if_empty:NTF \l_tmpa_str {
5139     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
5140     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
5141   }{
5142     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
5143     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
5144   }
5145 }

```

(End definition for \stexpatchdefinition. This function is documented on page 47.)

\inlinedef inline:

```

5146 \keys_define:nn {stex / inlinedef }{
5147   type      .str_set_x:N = \sdefinitiontype,
5148   id        .str_set_x:N = \sdefinitionid,
5149   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5150   name      .str_set_x:N = \sdefinitionname
5151 }
5152 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {

```

```

5153 \str_clear:N \sdefinitiontype
5154 \str_clear:N \sdefinitionid
5155 \str_clear:N \sdefinitionname
5156 \clist_clear:N \l__stex_statements_sdefinition_for_clist
5157 \keys_set:nn { stex / inlinedef }{ #1 }
5158 }
5159 \NewDocumentCommand \inlinedef { 0{} m } {
5160 \begingroup
5161 \__stex_statements_inlinedef_args:n{ #1 }
5162 \stex_reactivate_macro:N \definiendum
5163 \stex_reactivate_macro:N \definame
5164 \stex_reactivate_macro:N \Definame
5165 \stex_reactivate_macro:N \premise
5166 \stex_reactivate_macro:N \definiens
5167 \stex_ref_new_doc_target:n \sdefinitionid
5168 \stex_if_smsmode:TF{\stex_suppress_html:n {
5169 \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5170 }}{
5171 \seq_clear:N \l_tmpa_seq
5172 \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5173 \tl_if_empty:nF{ ##1 }{
5174 \stex_get_symbol:n { ##1 }
5175 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5176 \l_stex_get_symbol_uri_str
5177 }
5178 }
5179 }
5180 \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpa_seq
5181 \exp_args:Nnx
5182 \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
5183 \str_if_empty:NF \sdefinitiontype {
5184 \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5185 }
5186 #2
5187 \str_if_empty:NF \sdefinitionname {
5188 \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sdefinitionname}}
5189 \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5190 }
5191 }
5192 }
5193 \endgroup
5194 \stex_smsmode_do:
5195 }

```

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

32.2 Assertions

sassertion

```

5196
5197 \keys_define:nn {stex / sassertion }{
5198 type .str_set_x:N = \sassertiontype,
5199 id .str_set_x:N = \sassertionid,

```

```

5200 title .tl_set:N = \sassertiontitle ,
5201 for .clist_set:N = \l__stex_statements_sassertion_for_clist ,
5202 name .str_set_x:N = \sassertionname
5203 }
5204 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
5205 \str_clear:N \sassertiontype
5206 \str_clear:N \sassertionid
5207 \str_clear:N \sassertionname
5208 \clist_clear:N \l__stex_statements_sassertion_for_clist
5209 \tl_clear:N \sassertiontitle
5210 \keys_set:nn { stex / sassertion }{ #1 }
5211 }
5212
5213 %\tl_new:N \g__stex_statements_aftergroup_tl
5214
5215 \NewDocumentEnvironment{sassertion}{0{}}{
5216 \__stex_statements_sassertion_args:n{ #1 }
5217 \stex_reactivate_macro:N \premise
5218 \stex_reactivate_macro:N \conclusion
5219 \stex_if_smsmode:F {
5220 \seq_clear:N \l_tmpa_seq
5221 \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5222 \tl_if_empty:nF{ ##1 }{
5223 \stex_get_symbol:n { ##1 }
5224 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5225 \l_stex_get_symbol_uri_str
5226 }
5227 }
5228 }
5229 \exp_args:Nnnx
5230 \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}
5231 \str_if_empty:NF \sassertiontype {
5232 \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
5233 }
5234 \str_if_empty:NF \sassertionname {
5235 \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
5236 }
5237 \clist_set:Nn \l_tmpa_clist \sassertiontype
5238 \tl_clear:N \l_tmpa_tl
5239 \clist_map_inline:Nn \l_tmpa_clist {
5240 \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
5241 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
5242 }
5243 }
5244 \tl_if_empty:NTF \l_tmpa_tl {
5245 \__stex_statements_sassertion_start:
5246 }{
5247 \l_tmpa_tl
5248 }
5249 }
5250 \str_if_empty:NTF \sassertionid {
5251 \str_if_empty:NF \sassertionname {
5252 \stex_ref_new_doc_target:n {}
5253 }

```

```

5254 } {
5255   \stex_ref_new_doc_target:n \sassertionid
5256 }
5257 \stex_smsmode_do:
5258 ){
5259   \str_if_empty:NF \sassertionname {
5260     \stex_suppress_html:n{\stex_symdecl_do:nn{ }\sassertionname}}
5261     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5262   }
5263   \stex_if_smsmode:F {
5264     \clist_set:Nn \l_tmpa_clist \sassertiontype
5265     \tl_clear:N \l_tmpa_tl
5266     \clist_map_inline:Nn \l_tmpa_clist {
5267       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
5268         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
5269       }
5270     }
5271     \tl_if_empty:NTF \l_tmpa_tl {
5272       __stex_statements_sassertion_end:
5273     }{
5274       \l_tmpa_tl
5275     }
5276     \end{stex_annotate_env}
5277   }
5278 }

```

\stexpatchassertion

```

5279
5280 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
5281   \stex_par:\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {
5282     (\sassertiontitle)
5283   }~}
5284 }
5285 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\stex_par:\medskip}
5286
5287 \newcommand\stexpatchassertion[3] [] {
5288   \str_set:Nx \l_tmpa_str{ #1 }
5289   \str_if_empty:NTF \l_tmpa_str {
5290     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
5291     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
5292   }{
5293     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2 }
5294     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
5295   }
5296 }

```

(End definition for \stexpatchassertion. This function is documented on page 47.)

\inlineass inline:

```

5297 \keys_define:nn {stex / inlineass }{
5298   type      .str_set_x:N = \sassertiontype,
5299   id        .str_set_x:N = \sassertionid,
5300   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
5301   name      .str_set_x:N = \sassertionname

```

```

5302 }
5303 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
5304   \str_clear:N \sassertiontype
5305   \str_clear:N \sassertionid
5306   \str_clear:N \sassertionname
5307   \clist_clear:N \l__stex_statements_sassertion_for_clist
5308   \keys_set:nn { stex / inlineass }{ #1 }
5309 }
5310 \NewDocumentCommand \inlineass { 0{} m } {
5311   \begingroup
5312   \stex_reactivate_macro:N \premise
5313   \stex_reactivate_macro:N \conclusion
5314   \__stex_statements_inlineass_args:n{ #1 }
5315   \str_if_empty:NTF \sassertionid {
5316     \str_if_empty:NF \sassertionname {
5317       \stex_ref_new_doc_target:n {}
5318     }
5319   } {
5320     \stex_ref_new_doc_target:n \sassertionid
5321   }
5322
5323   \stex_if_smsmode:TF{
5324     \str_if_empty:NF \sassertionname {
5325       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5326       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5327     }
5328   }{
5329     \seq_clear:N \l_tmpa_seq
5330     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5331       \tl_if_empty:nF{ ##1 }{
5332         \stex_get_symbol:n { ##1 }
5333         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5334           \l_stex_get_symbol_uri_str
5335         }
5336       }
5337     }
5338     \exp_args:Nnx
5339     \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{
5340       \str_if_empty:NF \sassertiontype {
5341         \stex_annotate_invisible:nnn{typestrings}{\sassertiontype}{}
5342       }
5343       #2
5344       \str_if_empty:NF \sassertionname {
5345         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5346         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5347         \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
5348       }
5349     }
5350   }
5351   \endgroup
5352   \stex_smsmode_do:
5353 }

```

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

32.3 Examples

sexample

```

5354 \keys_define:nn {stex / sexample }{
5355   type      .str_set_x:N = \exampletype,
5356   id        .str_set_x:N = \sexampleid,
5357   title     .tl_set:N     = \sexampletitle,
5358   name      .str_set_x:N = \sexamplename ,
5359   for       .clist_set:N = \l__stex_statements_sexample_for_clist,
5360 }
5361 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
5362   \str_clear:N \sexampletype
5363   \str_clear:N \sexampleid
5364   \str_clear:N \sexamplename
5365   \tl_clear:N \sexampletitle
5366   \clist_clear:N \l__stex_statements_sexample_for_clist
5367   \keys_set:nn { stex / sexample }{ #1 }
5368 }
5369
5370
5371 \NewDocumentEnvironment{sexample}{0{}}{
5372   \__stex_statements_sexample_args:n{ #1 }
5373   \stex_reactivate_macro:N \premise
5374   \stex_reactivate_macro:N \conclusion
5375   \stex_if_smsmode:F {
5376     \seq_clear:N \l_tmpa_seq
5377     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5378       \tl_if_empty:NF{ ##1 }{
5379         \stex_get_symbol:n { ##1 }
5380         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5381           \l_stex_get_symbol_uri_str
5382         }
5383       }
5384     }
5385     \exp_args:Nnnx
5386     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
5387     \str_if_empty:NF \sexampletype {
5388       \stex_annotate_invisible:nnn{typestrings}{\sexampletype}{}
5389     }
5390     \str_if_empty:NF \sexamplename {
5391       \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}
5392     }
5393     \clist_set:Nn \l_tmpa_clist \sexampletype
5394     \tl_clear:N \l_tmpa_tl
5395     \clist_map_inline:Nn \l_tmpa_clist {
5396       \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
5397         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
5398       }
5399     }
5400     \tl_if_empty:NTF \l_tmpa_tl {
5401       \__stex_statements_sexample_start:
5402     }{
5403       \l_tmpa_tl
5404     }

```

```

5405 }
5406 \str_if_empty:NF \sexampleid {
5407   \stex_ref_new_doc_target:n \sexampleid
5408 }
5409 \stex_smsmode_do:
5410 ){
5411   \str_if_empty:NF \sexamplename {
5412     \stex_suppress_html:n{\stex_symdecl_do:nn}{\sexamplename}}
5413   }
5414   \stex_if_smsmode:F {
5415     \clist_set:Nn \l_tmpa_clist \sexamplotype
5416     \tl_clear:N \l_tmpa_tl
5417     \clist_map_inline:Nn \l_tmpa_clist {
5418       \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
5419         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
5420       }
5421     }
5422     \tl_if_empty:NTF \l_tmpa_tl {
5423       \__stex_statements_sexample_end:
5424     }{
5425       \l_tmpa_tl
5426     }
5427     \end{stex_annotate_env}
5428   }
5429 }

```

\stexpatchexample

```

5430
5431 \cs_new_protected:Nn \__stex_statements_sexample_start: {
5432   \stex_par:\noindent\titllemph{Example~\tl_if_empty:NF \sexampltitle {
5433     (\sexampltitle)
5434   }~}
5435 }
5436 \cs_new_protected:Nn \__stex_statements_sexample_end: {\stex_par:\medskip}
5437
5438 \newcommand\stexpatchexample[3]{} {
5439   \str_set:Nx \l_tmpa_str{ #1 }
5440   \str_if_empty:NTF \l_tmpa_str {
5441     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
5442     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
5443   }{
5444     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
5445     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
5446   }
5447 }

```

(End definition for \stexpatchexample. This function is documented on page 47.)

\inlineex inline:

```

5448 \keys_define:nn {stex / inlineex }{
5449   type      .str_set_x:N = \sexamplotype,
5450   id        .str_set_x:N = \sexampleid,
5451   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
5452   name      .str_set_x:N = \sexamplename

```



```

5453 }
5454 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
5455   \str_clear:N \sexamplotype
5456   \str_clear:N \sexampleid
5457   \str_clear:N \sexamplename
5458   \clist_clear:N \l__stex_statements_sexample_for_clist
5459   \keys_set:nn { stex / inlineex }{ #1 }
5460 }
5461 \NewDocumentCommand \inlineex { 0{ } m } {
5462   \beginngroup
5463   \stex_reactivate_macro:N \premise
5464   \stex_reactivate_macro:N \conclusion
5465   \__stex_statements_inlineex_args:n{ #1 }
5466   \str_if_empty:NF \sexampleid {
5467     \stex_ref_new_doc_target:n \sexampleid
5468   }
5469   \stex_if_smsmode:TF{
5470     \str_if_empty:NF \sexamplename {
5471       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5472     }
5473   }{
5474     \seq_clear:N \l_tmpa_seq
5475     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5476       \tl_if_empty:nF{ ##1 }{
5477         \stex_get_symbol:n { ##1 }
5478         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5479           \l_stex_get_symbol_uri_str
5480         }
5481       }
5482     }
5483     \exp_args:Nnx
5484     \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpa_seq {,}}{
5485       \str_if_empty:NF \sexamplotype {
5486         \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{ }
5487       }
5488       #2
5489       \str_if_empty:NF \sexamplename {
5490         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5491         \stex_annotate_invisible:nnn{statementname}{\sexamplename}{ }
5492       }
5493     }
5494   }
5495   \endgroup
5496   \stex_smsmode_do:
5497 }

```

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

32.4 Logical Paragraphs

sparagraph

```

5498 \keys_define:nn { stex / sparagraph } {
5499   id          .str_set_x:N    = \sparagraphid ,

```

```

5500 title .tl_set:N = \l_stex_sparagraph_title_tl ,
5501 type .str_set_x:N = \sparagraphtype ,
5502 for .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5503 from .tl_set:N = \sparagraphfrom ,
5504 to .tl_set:N = \sparagraphto ,
5505 start .tl_set:N = \l_stex_sparagraph_start_tl ,
5506 name .str_set:N = \sparagraphname ,
5507 imports .tl_set:N = \l__stex_statements_sparagraph_imports_tl
5508 }
5509
5510 \cs_new_protected:Nn \stex_sparagraph_args:n {
5511 \tl_clear:N \l_stex_sparagraph_title_tl
5512 \tl_clear:N \sparagraphfrom
5513 \tl_clear:N \sparagraphto
5514 \tl_clear:N \l_stex_sparagraph_start_tl
5515 \tl_clear:N \l__stex_statements_sparagraph_imports_tl
5516 \str_clear:N \sparagraphid
5517 \str_clear:N \sparagraphtype
5518 \clist_clear:N \l__stex_statements_sparagraph_for_clist
5519 \str_clear:N \sparagraphname
5520 \keys_set:nn { stex / sparagraph }{ #1 }
5521 }
5522 \newif\if@in@omtext\@in@omtextfalse
5523
5524 \NewDocumentEnvironment {sparagraph} { 0{} } {
5525 \stex_sparagraph_args:n { #1 }
5526 \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5527 \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
5528 }{
5529 \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
5530 }
5531 \@in@omtexttrue
5532 \stex_if_smsmode:F {
5533 \seq_clear:N \l_tmpa_seq
5534 \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5535 \tl_if_empty:NF{ ##1 }{
5536 \stex_get_symbol:n { ##1 }
5537 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5538 \l_stex_get_symbol_uri_str
5539 }
5540 }
5541 }
5542 \exp_args:Nnnx
5543 \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
5544 \str_if_empty:NF \sparagraphtype {
5545 \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{ }
5546 }
5547 \str_if_empty:NF \sparagraphfrom {
5548 \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{ }
5549 }
5550 \str_if_empty:NF \sparagraphto {
5551 \stex_annotate_invisible:nnn{to}{\sparagraphto}{ }
5552 }
5553 \str_if_empty:NF \sparagraphname {

```

```

5554     \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{ }
5555 }
5556 \clist_set:No \l_tmpa_clist \sparagraphtype
5557 \tl_clear:N \l_tmpa_tl
5558 \clist_map_inline:Nn \sparagraphtype {
5559     \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
5560         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
5561     }
5562 }
5563 \stex_csl_to_imports:No \usemodule \l__stex_statements_sparagraph_imports_tl
5564 \tl_if_empty:NTF \l_tmpa_tl {
5565     \__stex_statements_sparagraph_start:
5566 }{
5567     \l_tmpa_tl
5568 }
5569 }
5570 \clist_set:No \l_tmpa_clist \sparagraphtype
5571 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5572 {
5573     \stex_reactivate_macro:N \definiendum
5574     \stex_reactivate_macro:N \definame
5575     \stex_reactivate_macro:N \Definame
5576     \stex_reactivate_macro:N \premise
5577     \stex_reactivate_macro:N \definiens
5578 }
5579 \str_if_empty:NTF \sparagraphid {
5580     \str_if_empty:NTF \sparagraphname {
5581         \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5582             \stex_ref_new_doc_target:n {}
5583         }
5584     } {
5585         \stex_ref_new_doc_target:n {}
5586     }
5587 } {
5588     \stex_ref_new_doc_target:n \sparagraphid
5589 }
5590 \exp_args:NNx
5591 \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5592     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5593         \tl_if_empty:nF{ ##1 }{
5594             \stex_get_symbol:n { ##1 }
5595             \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5596         }
5597     }
5598 }
5599 \stex_smsmode_do:
5600 \ignorespacesandpars
5601 }{
5602     \str_if_empty:NF \sparagraphname {
5603         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5604         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5605     }
5606     \stex_if_smsmode:F {
5607         \clist_set:No \l_tmpa_clist \sparagraphtype

```

```

5608 \tl_clear:N \l_tmpa_tl
5609 \clist_map_inline:Nn \l_tmpa_clist {
5610   \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
5611     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
5612   }
5613 }
5614 \tl_if_empty:NTF \l_tmpa_tl {
5615   \__stex_statements_sparagraph_end:
5616 }{
5617   \l_tmpa_tl
5618 }
5619 \end{stex_annotate_env}
5620 }
5621 }

```

\stexpatchparagraph

```

5622
5623 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
5624   \stex_par:\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5625     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
5626       \titleemph{\l_stex_sparagraph_title_tl}:~
5627     }
5628   }{
5629     \titleemph{\l_stex_sparagraph_start_tl}~
5630   }
5631 }
5632 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\stex_par:\medskip}
5633
5634 \newcommand\stexpatchparagraph[3] [] {
5635   \str_set:Nx \l_tmpa_str{ #1 }
5636   \str_if_empty:NTF \l_tmpa_str {
5637     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
5638     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
5639   }{
5640     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
5641     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
5642   }
5643 }
5644
5645 \keys_define:nn { stex / inlinepara} {
5646   id      .str_set_x:N = \sparagraphid ,
5647   type    .str_set_x:N = \sparagraphtype ,
5648   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5649   from    .tl_set:N    = \sparagraphfrom ,
5650   to      .tl_set:N    = \sparagraphto ,
5651   name    .str_set:N   = \sparagraphname
5652 }
5653 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
5654   \tl_clear:N \sparagraphfrom
5655   \tl_clear:N \sparagraphto
5656   \str_clear:N \sparagraphid
5657   \str_clear:N \sparagraphtype
5658   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5659   \str_clear:N \sparagraphname

```

```

5660 \keys_set:nn { stex / inlinepara }{ #1 }
5661 }
5662 \NewDocumentCommand \inlinepara { 0{} m } {
5663   \beginngroup
5664     \__stex_statements_inlinepara_args:n{ #1 }
5665     \clist_set:No \l_tmpa_clist \sparagraphtype
5666     \str_if_empty:NTF \sparagraphid {
5667       \str_if_empty:NTF \sparagraphname {
5668         \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5669           \stex_ref_new_doc_target:n {}
5670         }
5671       } {
5672         \stex_ref_new_doc_target:n {}
5673       }
5674     } {
5675       \stex_ref_new_doc_target:n \sparagraphid
5676     }
5677     \stex_if_smsmode:TF{
5678       \str_if_empty:NF \sparagraphname {
5679         \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
5680       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5681     }
5682   }{
5683     \seq_clear:N \l_tmpa_seq
5684     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5685       \tl_if_empty:nF{ ##1 }{
5686         \stex_get_symbol:n { ##1 }
5687         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5688           \l_stex_get_symbol_uri_str
5689         }
5690       }
5691     }
5692     \exp_args:NNx
5693     \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}{
5694       \str_if_empty:NF \sparagraphtype {
5695         \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
5696       }
5697       \str_if_empty:NF \sparagraphfrom {
5698         \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5699       }
5700       \str_if_empty:NF \sparagraphto {
5701         \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5702       }
5703       \str_if_empty:NF \sparagraphname {
5704         \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
5705       \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
5706       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5707     }
5708     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5709       \clist_map_inline:Nn \l_tmpa_seq {
5710         \stex_ref_new_sym_target:n {##1}
5711       }
5712     }
5713     #2

```

```

5714     }
5715   }
5716   \endgroup
5717   \stex_smsmode_do:
5718 }
5719

```

(End definition for \stexpatchparagraph. This function is documented on page 47.)

```

5720 \endpackage

```

Chapter 33

The Implementation

```
5721 <*package>
5722 <@@=stex_sproof>
5723
5724 %%%%%%%%%% sproof.dtx %%%%%%%%%%
5725
```

33.1 Proofs

We first define some keys for the proof environment.

```
5726 \keys_define:nn { stex / spf } {
5727   id          .str_set_x:N = \spfid,
5728   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,
5729   from        .tl_set:N    = \l__stex_sproof_spf_from_tl ,
5730   proofend    .tl_set:N    = \l__stex_sproof_spf_proofend_tl,
5731   type        .str_set_x:N = \spftype,
5732   title       .tl_set:N    = \spftitle,
5733   continues   .tl_set:N    = \l__stex_sproof_spf_continues_tl,
5734   functions   .tl_set:N    = \l__stex_sproof_spf_functions_tl,
5735   method      .tl_set:N    = \l__stex_sproof_spf_method_tl
5736 }
5737 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
5738   \str_clear:N \spfid
5739   \tl_clear:N \l__stex_sproof_spf_for_tl
5740   \tl_clear:N \l__stex_sproof_spf_from_tl
5741   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
5742   \str_clear:N \spftype
5743   \tl_clear:N \spftitle
5744   \tl_clear:N \l__stex_sproof_spf_continues_tl
5745   \tl_clear:N \l__stex_sproof_spf_functions_tl
5746   \tl_clear:N \l__stex_sproof_spf_method_tl
5747   \bool_set_false:N \l__stex_sproof_inc_counter_bool
5748   \keys_set:nn { stex / spf }{ #1 }
5749 }
```

```
\c__stex_sproof_flow_str We define this macro, so that we can test whether the display key has the value flow
5750 \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.

```

5751 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
5752 \cs_new_protected:Npn \sproofnumber {
5753   \int_set:Nn \l_tmpa_int {1}
5754   \bool_while_do:nn {
5755     \int_compare_p:nNn {
5756       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5757     } > 0
5758   }{
5759     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
5760     \int_incr:N \l_tmpa_int
5761   }
5762 }
5763 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
5764   \int_set:Nn \l_tmpa_int {1}
5765   \bool_while_do:nn {
5766     \int_compare_p:nNn {
5767       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5768     } > 0
5769   }{
5770     \int_incr:N \l_tmpa_int
5771   }
5772   \int_compare:nNnF \l_tmpa_int = 1 {
5773     \int_decr:N \l_tmpa_int
5774   }
5775   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
5776     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1
5777   }
5778 }
5779
5780 \cs_new_protected:Npn \__stex_sproof_add_counter: {
5781   \int_set:Nn \l_tmpa_int {1}
5782   \bool_while_do:nn {
5783     \int_compare_p:nNn {
5784       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5785     } > 0
5786   }{
5787     \int_incr:N \l_tmpa_int
5788   }
5789   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 1 }
5790 }
5791
5792 \cs_new_protected:Npn \__stex_sproof_remove_counter: {
5793   \int_set:Nn \l_tmpa_int {1}
5794   \bool_while_do:nn {

```



```

5795 \int_compare_p:nNn {
5796 \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5797 } > 0
5798 ){
5799 \int_incr:N \l_tmpa_int
5800 }
5801 \int_decr:N \l_tmpa_int
5802 \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 0 }
5803 }

```

\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

```

5804 \def\sproof@box{
5805 \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
5806 }
5807 \def\sproofend{
5808 \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
5809 \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
5810 }
5811 }

```

(End definition for \sproofend. This function is documented on page 46.)

spf@*@kw

```

5812 \def\spf@proofsketch@kw{Proof~Sketch}
5813 \def\spf@proof@kw{Proof}
5814 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

5815 \AddToHook{begindocument}{
5816 \ltx@ifpackageloaded{babel}{
5817 \makeatletter
5818 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5819 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5820 \input{sproof-ngerman.ldf}
5821 }
5822 \clist_if_in:NnT \l_tmpa_clist {finnish}{
5823 \input{sproof-finnish.ldf}
5824 }
5825 \clist_if_in:NnT \l_tmpa_clist {french}{
5826 \input{sproof-french.ldf}
5827 }
5828 \clist_if_in:NnT \l_tmpa_clist {russian}{
5829 \input{sproof-russian.ldf}
5830 }
5831 \makeatother
5832 }{}
5833 }

```

spfsketch

```

5834 \newcommand\spsketch[2][]{
5835 \begin{group}
5836 \let \premise \stex_proof_premise:

```

```

5837 \__stex_sproof_spf_args:n{#1}
5838 \stex_if_smsmode:TF {
5839   \str_if_empty:NF \spfid {
5840     \stex_ref_new_doc_target:n \spfid
5841   }
5842 }{
5843   \seq_clear:N \l_tmpa_seq
5844   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5845     \tl_if_empty:nF{ ##1 }{
5846       \stex_get_symbol:n { ##1 }
5847       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5848         \l_stex_get_symbol_uri_str
5849       }
5850     }
5851   }
5852   \exp_args:Nnx
5853   \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
5854     \str_if_empty:NF \spftype {
5855       \stex_annotate_invisible:nnn{type}{\spftype}{
5856     }
5857     \clist_set:No \l_tmpa_clist \spftype
5858     \tl_set:Nn \l_tmpa_tl {
5859       \titleemph{
5860         \tl_if_empty:NTF \spftitle {
5861           \spf@proofsketch@kw
5862         }{
5863           \spftitle
5864         }
5865       }::~
5866     }
5867     \clist_map_inline:Nn \l_tmpa_clist {
5868       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5869         \tl_clear:N \l_tmpa_tl
5870       }
5871     }
5872     \str_if_empty:NF \spfid {
5873       \stex_ref_new_doc_target:n \spfid
5874     }
5875     \l_tmpa_tl #2 \sproofend
5876   }
5877 }
5878 \endgroup
5879 \stex_smsmode_do:
5880 }
5881

```

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

spfeq This is very similar to `spfsketch`, but uses a computation array¹⁴¹⁵

```

5882 \newenvironment{spfeq}[2][ ]{
5883   \__stex_sproof_spf_args:n{#1}

```

¹⁴EDNOTE: This should really be more like a tabular with an ensuremath in it. or invoke text on the last column

¹⁵EDNOTE: document above

```

5884 \let \premise \stex_proof_premise:
5885 \stex_if_smsmode:TF {
5886   \str_if_empty:NF \spfid {
5887     \stex_ref_new_doc_target:n \spfid
5888   }
5889 }{
5890   \seq_clear:N \l_tmpa_seq
5891   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5892     \tl_if_empty:nF{ ##1 }{
5893       \stex_get_symbol:n { ##1 }
5894       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5895         \l_stex_get_symbol_uri_str
5896       }
5897     }
5898   }
5899   \exp_args:Nnnx
5900   \begin{stex_annotate_env}{spfeq}{\seq_use:Nn \l_tmpa_seq {,}}
5901   \str_if_empty:NF \spftype {
5902     \stex_annotate_invisible:nnn{type}{\spftype}{}
5903   }
5904
5905   \clist_set:No \l_tmpa_clist \spftype
5906   \tl_clear:N \l_tmpa_tl
5907   \clist_map_inline:Nn \l_tmpa_clist {
5908     \tl_if_exist:cT {__stex_sproof_spfeq_##1_start:}{
5909       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_start:}}
5910     }
5911     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5912       \tl_set:Nn \l_tmpa_tl {\use:n{}}
5913     }
5914   }
5915   \tl_if_empty:NTF \l_tmpa_tl {
5916     \__stex_sproof_spfeq_start:
5917   }{
5918     \l_tmpa_tl
5919   }{~#2}
5920   \str_if_empty:NF \spfid {
5921     \stex_ref_new_doc_target:n \spfid
5922   }
5923   \begin{displaymath}\begin{array}{rcll}
5924   }
5925   \stex_smsmode_do:
5926 }{
5927   \stex_if_smsmode:F {
5928     \end{array}\end{displaymath}
5929     \clist_set:No \l_tmpa_clist \spftype
5930     \tl_clear:N \l_tmpa_tl
5931     \clist_map_inline:Nn \l_tmpa_clist {
5932       \tl_if_exist:cT {__stex_sproof_spfeq_##1_end:}{
5933         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_end:}}
5934       }
5935     }
5936     \tl_if_empty:NTF \l_tmpa_tl {
5937       \__stex_sproof_spfeq_end:

```

```

5938   }{
5939     \l_tmpa_tl
5940   }
5941   \end{stex_annotate_env}
5942 }
5943 }
5944
5945 \cs_new_protected:Nn \__stex_sproof_spfeq_start: {
5946   \titleemph{
5947     \tl_if_empty:NTF \spftitle {
5948       \spf@proof@kw
5949     }{
5950       \spftitle
5951     }
5952   }:
5953 }
5954 \cs_new_protected:Nn \__stex_sproof_spfeq_end: {\sproofend}
5955
5956 \newcommand\stexpatchspfeq[3] [] {
5957   \str_set:Nx \l_tmpa_str{ #1 }
5958   \str_if_empty:NTF \l_tmpa_str {
5959     \tl_set:Nn \__stex_sproof_spfeq_start: { #2 }
5960     \tl_set:Nn \__stex_sproof_spfeq_end: { #3 }
5961   }{
5962     \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_start:\endcsname{ #2 }
5963     \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_end:\endcsname{ #3 }
5964   }
5965 }
5966

```

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

sproof 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.

```

5967 \newenvironment{sproof}[2] []{
5968   \let \premise \stex_proof_premise:
5969   \intarray_gzero:N \l__stex_sproof_counter_intarray
5970   \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
5971   \__stex_sproof_spf_args:n{#1}
5972   \stex_if_smsmode:TF {
5973     \str_if_empty:NF \spfid {
5974       \stex_ref_new_doc_target:n \spfid
5975     }
5976   }{
5977     \seq_clear:N \l_tmpa_seq
5978     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5979       \tl_if_empty:nF{ ##1 }{
5980         \stex_get_symbol:n { ##1 }
5981         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5982           \l_stex_get_symbol_uri_str
5983         }
5984       }
5985     }

```

```

5986 \exp_args:Nnnx
5987 \begin{stex_annotate_env}{sproof}{\seq_use:Nn \l_tmpa_seq {,}}
5988 \str_if_empty:NF \spftype {
5989   \stex_annotate_invisible:nnn{type}{\spftype}{}
5990 }
5991
5992 \clist_set:No \l_tmpa_clist \spftype
5993 \tl_clear:N \l_tmpa_tl
5994 \clist_map_inline:Nn \l_tmpa_clist {
5995   \tl_if_exist:cT {__stex_sproof_sproof_##1_start:}{
5996     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_start:}}
5997   }
5998   \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5999     \tl_set:Nn \l_tmpa_tl {\use:n{}}
6000   }
6001 }
6002 \tl_if_empty:NTF \l_tmpa_tl {
6003   \__stex_sproof_sproof_start:
6004 }{
6005   \l_tmpa_tl
6006 }{~#2}
6007 \str_if_empty:NF \spfid {
6008   \stex_ref_new_doc_target:n \spfid
6009 }
6010 \begin{description}
6011 }
6012 \stex_smsmode_do:
6013 ){
6014   \stex_if_smsmode:F{
6015     \end{description}
6016     \clist_set:No \l_tmpa_clist \spftype
6017     \tl_clear:N \l_tmpa_tl
6018     \clist_map_inline:Nn \l_tmpa_clist {
6019       \tl_if_exist:cT {__stex_sproof_sproof_##1_end:}{
6020         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_end:}}
6021       }
6022     }
6023     \tl_if_empty:NTF \l_tmpa_tl {
6024       \__stex_sproof_sproof_end:
6025     }{
6026       \l_tmpa_tl
6027     }
6028     \end{stex_annotate_env}
6029   }
6030 }
6031
6032 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
6033   \par\noindent\titleemph{
6034     \tl_if_empty:NTF \spftype {
6035       \spf@proof@kw
6036     }{
6037       \spftype
6038     }
6039   }::

```

```

6040 }
6041 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
6042
6043 \newcommand\stexpatchproof[3] [] {
6044   \str_set:Nx \l_tmpa_str{ #1 }
6045   \str_if_empty:NTF \l_tmpa_str {
6046     \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
6047     \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
6048   }{
6049     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
6050     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
6051   }
6052 }

```

\spfidea

```

6053 \newcommand\spfidea[2] []{
6054   \__stex_sproof_spf_args:n{#1}
6055   \titleemph{
6056     \tl_if_empty:NTF \spftype {Proof-Idea}{
6057       \spftype
6058     } :
6059   }~#2
6060   \sproofend
6061 }

```

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

The next two environments (proof steps) and comments, are mostly semantical, they take KeyVal arguments that specify their semantic role. In draft mode, they read these values and show them. If the surrounding proof had `display=flow`, then no new `\item` is generated, otherwise it is. In any case, the proof step number (at the current level) is incremented.

spfstep

```

6062 \newenvironment{spfstep}[1] []{
6063   \__stex_sproof_spf_args:n{#1}
6064   \stex_if_smsmode:TF {
6065     \str_if_empty:NF \spfid {
6066       \stex_ref_new_doc_target:n \spfid
6067     }
6068   }{
6069     \@in@omtexttrue
6070     \seq_clear:N \l_tmpa_seq
6071     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6072       \tl_if_empty:nF{ ##1 }{
6073         \stex_get_symbol:n { ##1 }
6074         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6075           \l_stex_get_symbol_uri_str
6076         }
6077       }
6078     }
6079     \exp_args:Nnnx
6080     \begin{stex_annotate_env}{spfstep}{\seq_use:Nn \l_tmpa_seq {},}
6081     \str_if_empty:NF \spftype {
6082       \stex_annotate_invisible:nnn{type}{\spftype}{ }

```

```

6083   }
6084   \clist_set:Nn \l_tmpa_clist \spftype
6085   \tl_set:Nn \l_tmpa_tl {
6086     \item[\sproofnumber]
6087     \bool_set_true:N \l__stex_sproof_inc_counter_bool
6088   }
6089   \clist_map_inline:Nn \l_tmpa_clist {
6090     \exp_args:Nn \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6091       \tl_clear:N \l_tmpa_tl
6092     }
6093   }
6094   \l_tmpa_tl
6095   \tl_if_empty:NF \spftitle {
6096     {(\titleemph{\spftitle})\enspace}
6097   }
6098   \str_if_empty:NF \spfid {
6099     \stex_ref_new_doc_target:n \spfid
6100   }
6101 }
6102 \stex_smsmode_do:
6103 \ignorespacesandpars
6104 }{
6105   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6106     \__stex_sproof_inc_counter:
6107   }
6108   \stex_if_smsmode:F {
6109     \end{stex_annotate_env}
6110   }
6111 }

```

spfcomment

```

6112 \newenvironment{spfcomment}[1][]{
6113   \__stex_sproof_spf_args:n{#1}
6114   \clist_set:Nn \l_tmpa_clist \spftype
6115   \tl_set:Nn \l_tmpa_tl {
6116     \item[\sproofnumber]
6117     \bool_set_true:N \l__stex_sproof_inc_counter_bool
6118   }
6119   \clist_map_inline:Nn \l_tmpa_clist {
6120     \exp_args:Nn \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6121       \tl_clear:N \l_tmpa_tl
6122     }
6123   }
6124   \l_tmpa_tl
6125 }{
6126   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6127     \__stex_sproof_inc_counter:
6128   }
6129 }

```

The next two environments also take a `KeyVal` argument, but also a regular one, which contains a start text. Both environments start a new numbered proof level.

subproof In the `subproof` environment, a new (lower-level) `proproof` environment is started.

```

6130 \newenvironment{subproof}[2][]{
6131   \_stex_sproof_spf_args:n{#1}
6132   \stex_if_smsmode:TF{
6133     \str_if_empty:NF \spfid {
6134       \stex_ref_new_doc_target:n \spfid
6135     }
6136   }{
6137     \seq_clear:N \l_tmpa_seq
6138     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6139       \tl_if_empty:nF{ ##1 }{
6140         \stex_get_symbol:n { ##1 }
6141         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6142           \l_stex_get_symbol_uri_str
6143         }
6144       }
6145     }
6146     \exp_args:Nnnx
6147     \begin{stex_annotate_env}{subproof}{\seq_use:Nn \l_tmpa_seq {,}}
6148     \str_if_empty:NF \spftype {
6149       \stex_annotate_invisible:nnn{type}{\spftype}{ }
6150     }
6151
6152     \clist_set:No \l_tmpa_clist \spftype
6153     \tl_set:Nn \l_tmpa_tl {
6154       \item[\sproofnumber]
6155       \bool_set_true:N \l__stex_sproof_inc_counter_bool
6156     }
6157     \clist_map_inline:Nn \l_tmpa_clist {
6158       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6159         \tl_clear:N \l_tmpa_tl
6160       }
6161     }
6162     \l_tmpa_tl
6163     \tl_if_empty:NF \spftitle {
6164       {(\titleemph{\spftitle})\enspace}
6165     }
6166     {~#2}
6167     \str_if_empty:NF \spfid {
6168       \stex_ref_new_doc_target:n \spfid
6169     }
6170   }
6171   \_stex_sproof_add_counter:
6172   \stex_smsmode_do:
6173 }{
6174   \_stex_sproof_remove_counter:
6175   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6176     \_stex_sproof_inc_counter:
6177   }
6178   \stex_if_smsmode:F{
6179     \end{stex_annotate_env}
6180   }
6181 }

```

spfcases In the **pfcases** environment, the start text is displayed as the first comment of the proof.


```

6182 \newenvironment{spfcases}[2] [] {
6183   \tl_if_empty:nTF{#1}{
6184     \begin{subproof}[method=by-cases]{#2}
6185   }{
6186     \begin{subproof}[#1,method=by-cases]{#2}
6187   }
6188 }{
6189   \end{subproof}
6190 }

```

spfcase In the pfcase environment, the start text is displayed specification of the case after the `\item`

```

6191 \newenvironment{spfcase}[2] [] {
6192   \__stex_sproof_spf_args:n{#1}
6193   \stex_if_smsmode:TF {
6194     \str_if_empty:NF \spfid {
6195       \stex_ref_new_doc_target:n \spfid
6196     }
6197   }{
6198     \seq_clear:N \l_tmpa_seq
6199     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6200       \tl_if_empty:nF{ ##1 }{
6201         \stex_get_symbol:n { ##1 }
6202         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6203           \l_stex_get_symbol_uri_str
6204         }
6205       }
6206     }
6207     \exp_args:Nnnx
6208     \begin{stex_annotate_env}{spfcase}{\seq_use:Nn \l_tmpa_seq {,}}
6209     \str_if_empty:NF \spftype {
6210       \stex_annotate_invisible:nnn{type}{\spftype}{ }
6211     }
6212     \clist_set:No \l_tmpa_clist \spftype
6213     \tl_set:Nn \l_tmpa_tl {
6214       \item[\sproofnumber]
6215       \bool_set_true:N \l__stex_sproof_inc_counter_bool
6216     }
6217     \clist_map_inline:Nn \l_tmpa_clist {
6218       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6219         \tl_clear:N \l_tmpa_tl
6220       }
6221     }
6222     \l_tmpa_tl
6223     \tl_if_empty:nF{#2}{
6224       \titleemph{#2}:~
6225     }
6226   }
6227   \__stex_sproof_add_counter:
6228   \stex_smsmode_do:
6229 }{
6230   \__stex_sproof_remove_counter:
6231   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6232     \__stex_sproof_inc_counter:

```

```

6233 }
6234 \stex_if_smsmode:F{
6235   \clist_set:No \l_tmpa_clist \spftype
6236   \tl_set:Nn \l_tmpa_tl{\sproofend}
6237   \clist_map_inline:Nn \l_tmpa_clist {
6238     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6239       \tl_clear:N \l_tmpa_tl
6240     }
6241   }
6242   \l_tmpa_tl
6243   \end{stex_annotate_env}
6244 }
6245 }

```

spfcase similar to **spfcase**, takes a third argument.

```

6246 \newcommand\spfcasesketch[3] [] {
6247   \begin{spfcase}[#1]{#2}#3\end{spfcase}
6248 }

```

33.2 Justifications

We define the actions that are undertaken, when the keys for justifications are encountered. Here this is very simple, we just define an internal macro with the value, so that we can use it later.

```

6249 \keys_define:nn { stex / just }{
6250   id      .str_set_x:N = \l__stex_sproof_just_id_str,
6251   method  .tl_set:N   = \l__stex_sproof_just_method_tl,
6252   premises .tl_set:N   = \l__stex_sproof_just_premises_tl,
6253   args    .tl_set:N   = \l__stex_sproof_just_args_tl
6254 }

```

The next three environments and macros are purely semantic, so we ignore the keyval arguments for now and only display the content.¹⁶

\spfjust

```

6255 \newcommand\spfjust[1] [] {}

```

(End definition for **\spfjust**. This function is documented on page 45.)

\premise

```

6256 \newcommand\stex_proof_premise:[2] [] {#2}

```

(End definition for **\premise**. This function is documented on page 45.)

\justarg the **\justarg** macro is purely semantic, so we ignore the keyval arguments for now and only display the content.

```

6257 \newcommand\justarg[2] [] {#2}
6258 \</package>

```

(End definition for **\justarg**. This function is documented on page 45.)

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

¹⁶EdNOTE: need to do something about the premise in draft mode.

Chapter 34

STEX -Others Implementation

```
6259 <*package>
6260
6261 %%%%%%%%%% others.dtx %%%%%%%%%%
6262
6263 <@@=stex_others>
        Warnings and error messages
6264 % None

\MSC Math subject classifier

6265 \NewDocumentCommand \MSC {m} {
6266 % TODO
6267 }

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

6268 \@ifpackageloaded{tikzinput}{
6269 \RequirePackage{stex-tikzinput}
6270 }{}
6271
6272 \bool_if:NT \c_stex_persist_mode_bool {
6273 \input{\jobname.sms}
6274 \prop_if_exist:NT\c_stex_mathhub_main_manifest_prop{
6275 \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
6276 \l_tmpa_str
6277 \prop_set_eq:cN { c_stex_mathhub\_l_tmpa_str_manifest_prop }
6278 \c_stex_mathhub_main_manifest_prop
6279 \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
6280 }
6281 }
6282 </package>
```

Chapter 35

STEX -Metatheory Implementation

```
6283 <*package>
6284 <@@=stex_modules>
6285
6286 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
6287
6288 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX/meta}
6289 \begingroup
6290 \stex_module_setup:nn{
6291   ns=\c_stex_metatheory_ns_str,
6292   meta=NONE
6293 }{Metatheory}
6294 \stex_reactivate_macro:N \symdecl
6295 \stex_reactivate_macro:N \notation
6296 \stex_reactivate_macro:N \symdef
6297 \ExplSyntaxOff
6298 \csname stex_suppress_html:n\endcsname{
6299   % is-a (a:A, a \in A, a is an A, etc.)
6300   \symdecl{isa}[args=ai]
6301   \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
6302   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
6303   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
6304
6305   % bind (\forall, \Pi, \lambda etc.)
6306   \symdecl{bind}[args=Bi]
6307   \notation{bind}[forall]{\comp\forall #1.;#2}{##1 \comp, ##2}
6308   \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6309   \notation{bind}[deffun]{\comp( #1 \comp{ }\;\to\; ) #2}{##1 \comp, ##2}
6310
6311   % implicit bind
6312   \symdef{implicitbind}[args=Bi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6313
6314   % dummy variable
6315   \symdecl{dummyvar}
6316   \notation{dummyvar}[underscore]{\comp\_}
6317   \notation{dummyvar}[dot]{\comp\cdot}
```

```

6318 \notation{dummyvar}[dash]{\comp{\rm --}}
6319
6320 %fromto (function space, Hom-set, implication etc.)
6321 \symdecl{fromto}[args=ai]
6322 \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
6323 \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
6324
6325 % mapto (lambda etc.)
6326 \symdecl{mapto}[args=Bi]
6327 %\notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
6328 %\notation{mapto}[lambda]{\comp\lambda #1 \comp.; #2}{#1 \comp, #2}
6329 %\notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.; #2}{#1 \comp, #2}
6330
6331 % function/operator application
6332 \symdecl{apply}[args=ia]
6333 \notation{apply}[prec=0;0x\infpres,parens]{#1 \comp( #2 \comp)}{##1 \comp, ##2}
6334 \notation{apply}[prec=0;0x\infpres,lambda]{#1 \; #2 }{##1 \; ; ##2}
6335
6336 % collection of propositions/booleans/truth values
6337 \symdecl{prop}[name=proposition]
6338 \notation{prop}[prop]{\comp{\rm prop}}
6339 \notation{prop}[BOOL]{\comp{\rm BOOL}}
6340
6341 \symdecl{judgmentholds}[args=1]
6342 \notation{judgmentholds}[vdash,op=\vdash]{\comp\vdash\; ; #1}
6343
6344 % sequences
6345 \symdecl{seqtype}[args=1]
6346 \notation{seqtype}[kleene]{#1^{\comp\ast}}
6347
6348 \symdecl{seqexpr}[args=a]
6349 \notation{seqexpr}[angle,prec=nobrackets]{\comp\langle #1\comp\rangle}{##1\comp,##2}
6350
6351 \symdef{seqmap}[args=abi,setlike]{\comp{\#3 \comp| #2\comp\in \dobrackets{#1} \comp\}}{##1\comp,##2}
6352 \symdef{seqprepend}[args=ia]{#1 \comp{:} #2}{##1 \comp, ##2}
6353 \symdef{seqappend}[args=ai]{#1 \comp{:} #2}{##1 \comp, ##2}
6354 \symdef{seqfoldleft}[args=iabbi]{ \comp{foldl}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\comp}
6355 \symdef{seqfoldright}[args=iabbi,op=foldr]{ \comp{foldr}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\comp}
6356 \symdef{seqhead}[args=a]{\comp{head}\dobrackets{#1}}{##1 \comp, ##2}
6357 \symdef{seqtail}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6358 \symdef{seqlast}[args=a]{\comp{last}\dobrackets{#1}}{##1 \comp, ##2}
6359 \symdef{seqinit}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6360
6361 \symdef{sequence-index}[args=2,li,prec=nobrackets]{\comp{#1}_{#2}}
6362 \notation{sequence-index}[ui,prec=nobrackets]{\comp{#1}^{#2}}
6363
6364 \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{\,\ellipses}}{##1\comp,##2}
6365 \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{\,\ellipses,}#2}{##1\comp,##2}
6366 \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{\,\ellipses,}#2\comp{\,\ellipses,}#3}
6367
6368 % letin ("let", local definitions, variable substitution)
6369 \symdecl{letin}[args=bii]
6370 \notation{letin}[let]{\comp{\rm let}}{#1\comp{=}#2\; \comp{\rm in}}{#3}
6371 \notation{letin}[subst]{#3 \comp[ #1 \comp/ #2 \comp]}

```

```

6372 \notation{letin}[frac]{#3 \comp[ \frac{#2}{#1} \comp]}
6373
6374 % structures
6375 \symdecl*{module-type}[args=1]
6376 \notation{module-type}{\comp{\mathtt{MOD}}} #1}
6377 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
6378 \notation{mathstruct}[angle,prec=nobrackets]{\comp\langle #1 \comp\rangle}{##1 \comp, ##2}
6379
6380 % objects
6381 \symdecl{object}
6382 \notation{object}{\comp{\mathtt{OBJECT}}}
6383
6384 }
6385
6386 % The following are abbreviations in the sTeX corpus that are left over from earlier
6387 % developments. They will eventually be phased out.
6388
6389 \ExplSyntaxOn
6390 \stex_add_to_current_module:n{
6391   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
6392   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
6393   \def\livar{\csname sequence-index\endcsname[li]}
6394   \def\uivar{\csname sequence-index\endcsname[ui]}
6395   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
6396   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
6397 }
6398 \__stex_modules_end_module:
6399 \endgroup
6400 \</package>

```

Chapter 36

Tikzinput Implementation

```
6401 <@@=tikzinput>
6402 <*package>
6403
6404 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
6405
6406 \ProvidesExplPackage{tikzinput}{2022/02/26}{3.0.1}{tikzinput package}
6407 \RequirePackage{l3keys2e}
6408
6409 \keys_define:nn { tikzinput } {
6410   image .bool_set:N = \c_tikzinput_image_bool,
6411   image .default:n = false ,
6412   unknown .code:n = {}
6413 }
6414
6415 \ProcessKeysOptions { tikzinput }
6416
6417 \bool_if:NTF \c_tikzinput_image_bool {
6418   \RequirePackage{graphicx}
6419
6420   \providecommand\usetikzlibrary[]{}
6421   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
6422 }{
6423   \RequirePackage{tikz}
6424   \RequirePackage{standalone}
6425
6426   \newcommand \tikzinput [2] [] {
6427     \setkeys{Gin}{#1}
6428     \ifx \Gin@ewidth \Gin@exclamation
6429       \ifx \Gin@eheight \Gin@exclamation
6430         \input { #2 }
6431       \else
6432         \resizebox{!}{ \Gin@eheight }{
6433           \input { #2 }
6434         }
6435       \fi
6436     \else
6437       \ifx \Gin@eheight \Gin@exclamation
6438         \resizebox{ \Gin@ewidth }{!}{
```

```

6439         \input { #2 }
6440     }
6441     \else
6442         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
6443             \input { #2 }
6444         }
6445     \fi
6446 \fi
6447 }
6448 }
6449
6450 \newcommand \ctikzinput [2] [] {
6451     \begin{center}
6452         \tikzinput [#1] {#2}
6453     \end{center}
6454 }
6455
6456 \@ifpackageloaded{stex}{
6457     \RequirePackage{stex-tikzinput}
6458 }{}
6459
6460 \</package>
6461 \<stex>
6462 \ProvidesExplPackage{stex-tikzinput}{2022/02/26}{3.0.1}{stex-tikzinput}
6463 \RequirePackage{stex}
6464 \RequirePackage{tikzinput}
6465
6466 \newcommand\mhtikzinput[2] []{%
6467     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
6468     \stex_in_repository:nn\Gin@mhrepos{
6469         \tikzinput[#1]{\mhp\path{##1}{#2}}
6470     }
6471 }
6472 \newcommand\cmhtikzinput[2] []{\begin{center}\mhtikzinput[#1]{#2}\end{center}}
6473
6474 \cs_new_protected:Nn \__tikzinput_usetikzlibrary:nn {
6475     \pgfkeys@spdef\pgf@temp{#1}
6476     \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
6477     \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgf@temp
6478     \expandafter\edef\csname tikz@library@#1@atcode\endcsname{\the\catcode'\@}
6479     \expandafter\edef\csname tikz@library@#1@barcode\endcsname{\the\catcode'\|}
6480     \expandafter\edef\csname tikz@library@#1@dollarcode\endcsname{\the\catcode'\$}
6481     \catcode'\@=11
6482     \catcode'\|=12
6483     \catcode'\$=3
6484     \pgfutil@InputIfFileExists{#2}{-}{-}
6485     \catcode'\@=\csname tikz@library@#1@atcode\endcsname
6486     \catcode'\|=\csname tikz@library@#1@barcode\endcsname
6487     \catcode'\$=\csname tikz@library@#1@dollarcode\endcsname
6488 }
6489
6490
6491 \newcommand\libusetikzlibrary[1]{

```



```

6492 \prop_if_exist:NF \l_stex_current_repository_prop {
6493   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6494 }
6495 \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
6496   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6497 }
6498 \seq_clear:N \l__tikzinput_libinput_files_seq
6499 \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
6500 \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
6501
6502 \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
6503   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / tikzlibrary
6504   \IfFileExists{ \l_tmpa_str }{
6505     \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6506   }{}
6507   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
6508   \seq_put_right:No \l_tmpa_seq \l_tmpa_str
6509 }
6510
6511 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / tikzlibrary #1 .code.t
6512 \IfFileExists{ \l_tmpa_str }{
6513   \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6514 }{}
6515
6516 \seq_if_empty:NTF \l__tikzinput_libinput_files_seq {
6517   \msg_error:nxxx{stex}{error/nofile}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .code.t
6518 }{
6519   \int_compare:nNnTF {\seq_count:N \l__tikzinput_libinput_files_seq} = 1 {
6520     \seq_map_inline:Nn \l__tikzinput_libinput_files_seq {
6521       \__tikzinput_usetikzlibrary:nn{#1}{ ##1 }
6522     }
6523   }{
6524     \msg_error:nxxx{stex}{error/twofiles}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .co
6525   }
6526 }
6527 }
6528 </stex>

```

LocalWords: bibfolder jobname.dtx tikzinput.dtx usetikzlibrary Gin@ewidth Gin@eheight
 LocalWords: resizebox ctikzinput mhtikzinput Gin@mhrepos mhpath

Chapter 37

document-structure.sty Implementation

```
6529 <*package>
6530 <@@=document_structure>
6531 \ProvidesExplPackage{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure}
6532 \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).

```
6533
6534 \keys_define:nn{ document-structure }{
6535   class      .str_set_x:N = \c_document_structure_class_str,
6536   topsect    .str_set_x:N = \c_document_structure_topsect_str,,
6537   unknown    .code:n      = {
6538     \PassOptionsToClass{\CurrentOption}{stex}
6539     \PassOptionsToClass{\CurrentOption}{tikzinput}
6540   }
6541   % showignores .bool_set:N = \c_document_structure_showignores_bool,
6542 }
6543 \ProcessKeysOptions{ document-structure }
6544 \str_if_empty:NT \c_document_structure_class_str {
6545   \str_set:Nn \c_document_structure_class_str {article}
6546 }
6547 \str_if_empty:NT \c_document_structure_topsect_str {
6548   \str_set:Nn \c_document_structure_topsect_str {section}
6549 }
```

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

```
6550 \RequirePackage{xspace}
6551 \RequirePackage{comment}
6552 \RequirePackage{stex}
6553 \AddToHook{begindocument}{
```

```

6554 \ltx@ifpackageloaded{babel}{
6555     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6556     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6557         \makeatletter\input{document-structure-ngerman.ldf}\makeatother
6558     }
6559 }{}
6560 }

```

`\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.

```

6561 \int_new:N \l_document_structure_section_level_int
6562 \str_case:NnF \c_document_structure_topsect_str {
6563     {part}}{
6564         \int_set:Nn \l_document_structure_section_level_int {0}
6565     }
6566     {chapter}{
6567         \int_set:Nn \l_document_structure_section_level_int {1}
6568     }
6569 }{
6570     \str_case:NnF \c_document_structure_class_str {
6571         {book}{
6572             \int_set:Nn \l_document_structure_section_level_int {0}
6573         }
6574         {report}{
6575             \int_set:Nn \l_document_structure_section_level_int {0}
6576         }
6577     }{
6578         \int_set:Nn \l_document_structure_section_level_int {2}
6579     }
6580 }

```

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.¹⁷

EdN:17

```

6581 \def\current@section@level{document}%
6582 \newcommand\currentsectionlevel{\lowercase\expandafter\current@section@level\xspace}%
6583 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%

```

(End definition for `\currentsectionlevel`. This function is documented on page 52.)

`\skipfragment`

```

6584 \cs_new_protected:Npn \skipfragment {

```

¹⁷EdNOTE: MK: we may have to experiment with the more powerful uppercasing macro from `mfirstuc.sty` once we internationalize.

```

6585 \ifcase\l_document_structure_section_level_int
6586 \or\stepcounter{part}
6587 \or\stepcounter{chapter}
6588 \or\stepcounter{section}
6589 \or\stepcounter{subsection}
6590 \or\stepcounter{subsubsection}
6591 \or\stepcounter{paragraph}
6592 \or\stepcounter{subparagraph}
6593 \fi
6594 }

```

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

blindfragment

```

6595 \newcommand\at@begin@blindsfragment[1]{
6596 \newenvironment{blindfragment}
6597 {
6598 \int_incr:N\l_document_structure_section_level_int
6599 \at@begin@blindsfragment\l_document_structure_section_level_int
6600 }{}

```

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

```

6601 \newcommand\sfragment@nonum[2]{
6602 \ifx\hyper@anchor\@undefined\else\phantomsection\fi
6603 \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
6604 }

```

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

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

```

6605 \newcommand\sfragment@num[2]{
6606 \tl_if_empty:NTF \l__document_structure_sfragment_short_tl {
6607 \@nameuse{#1}{#2}
6608 }{
6609 \cs_if_exist:NTF\rdfmata@sectioning{
6610 \@nameuse{rdfmata@#1@old}[\l__document_structure_sfragment_short_tl]{#2}
6611 }{
6612 \@nameuse{#1}[\l__document_structure_sfragment_short_tl]{#2}
6613 }
6614 }
6615 %\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\sfragment@id
6616 }

```

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

sfragment

```

6617 \keys_define:nn { document-structure / sfragment }{
6618 id .str_set_x:N = \l__document_structure_sfragment_id_str,
6619 date .str_set_x:N = \l__document_structure_sfragment_date_str,

```

```

6620 creators      .clist_set:N = \l__document_structure_sfragment_creators_clist,
6621 contributors  .clist_set:N = \l__document_structure_sfragment_contributors_clist,
6622 srccite       .tl_set:N     = \l__document_structure_sfragment_srccite_tl,
6623 type         .tl_set:N     = \l__document_structure_sfragment_type_tl,
6624 short        .tl_set:N     = \l__document_structure_sfragment_short_tl,
6625 display      .tl_set:N     = \l__document_structure_sfragment_display_tl,
6626 intro        .tl_set:N     = \l__document_structure_sfragment_intro_tl,
6627 imports      .tl_set:N     = \l__document_structure_sfragment_imports_tl,
6628 loadmodules  .bool_set:N   = \l__document_structure_sfragment_loadmodules_bool
6629 }
6630 \cs_new_protected:Nn \l__document_structure_sfragment_args:n {
6631   \str_clear:N \l__document_structure_sfragment_id_str
6632   \str_clear:N \l__document_structure_sfragment_date_str
6633   \clist_clear:N \l__document_structure_sfragment_creators_clist
6634   \clist_clear:N \l__document_structure_sfragment_contributors_clist
6635   \tl_clear:N \l__document_structure_sfragment_srccite_tl
6636   \tl_clear:N \l__document_structure_sfragment_type_tl
6637   \tl_clear:N \l__document_structure_sfragment_short_tl
6638   \tl_clear:N \l__document_structure_sfragment_display_tl
6639   \tl_clear:N \l__document_structure_sfragment_imports_tl
6640   \tl_clear:N \l__document_structure_sfragment_intro_tl
6641   \bool_set_false:N \l__document_structure_sfragment_loadmodules_bool
6642   \keys_set:nn { document-structure / sfragment } { #1 }
6643 }

```

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.

`\at@begin@sfragment`

```

6644 \newif@if@mainmatter\@mainmattertrue
6645 \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.

```

6646 \keys_define:nn { document-structure / sectioning }{
6647   name      .str_set_x:N = \l__document_structure_sect_name_str ,
6648   ref       .str_set_x:N = \l__document_structure_sect_ref_str  ,
6649   clear     .bool_set:N  = \l__document_structure_sect_clear_bool,
6650   clear     .default:n   = {true} ,
6651   num       .bool_set:N  = \l__document_structure_sect_num_bool  ,
6652   num       .default:n   = {true}
6653 }
6654 \cs_new_protected:Nn \l__document_structure_sect_args:n {
6655   \str_clear:N \l__document_structure_sect_name_str
6656   \str_clear:N \l__document_structure_sect_ref_str
6657   \bool_set_false:N \l__document_structure_sect_clear_bool
6658   \bool_set_false:N \l__document_structure_sect_num_bool
6659   \keys_set:nn { document-structure / sectioning } { #1 }
6660 }
6661 \newcommand\omdoc@sectioning[3][]{
6662   \l__document_structure_sect_args:n {#1 }
6663   \let\omdoc@sect@name\l__document_structure_sect_name_str
6664   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
6665   \if@mainmatter% numbering not overridden by frontmatter, etc.
6666     \bool_if:NTF \l__document_structure_sect_num_bool {

```

```

6667     \sfragment@num{#2}{#3}
6668   }{
6669     \sfragment@nonum{#2}{#3}
6670   }
6671   \def\current@section@level{\omdoc@sect@name}
6672 \else
6673   \sfragment@nonum{#2}{#3}
6674 \fi
6675 }% 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.

```

6676 \newcommand\sfragment@redefine@addtocontents[1]{%
6677   %\edef\__document_structureimport{#1}%
6678   %\@for\@I:=\__document_structureimport\do{%
6679     %\edef\@path{\csname module@\@I @path\endcsname}%
6680     %\@ifundefined{tf@toc}\relax%
6681     %    {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
6682   %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
6683   %\def\addcontentsline##1##2##3{%
6684     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6685   %\else% hyperref.sty not loaded
6686   %\def\addcontentsline##1##2##3{%
6687     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6688   %\fi
6689   }% hyperref.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.

```

6690 \newenvironment{sfragment}[2][ ]% keys, title
6691 {
6692   \__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.

```

6693   \stex_csl_to_imports:No \usemodule \l__document_structure_sfragment_imports_tl
6694
6695   \bool_if:NT \l__document_structure_sfragment_loadmodules_bool {
6696     \sfragment@redefine@addtocontents{
6697       %\@ifundefined{module@id}\used@modules%
6698       %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}
6699     }
6700   }

```

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

```

6701
6702   \stex_document_title:n { #2 }
6703
6704   \int_incr:N\l__document_structure_section_level_int
6705   \ifcase\l__document_structure_section_level_int
6706     \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
6707     \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}

```

```

6708 \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
6709 \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
6710 \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
6711 \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
6712 \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
6713 \fi
6714 \at@begin@sfragment[#1]\l_document_structure_section_level_int{#2}
6715 \str_if_empty:NF \l__document_structure_sfragment_id_str {
6716   \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
6717 }
6718 }% for customization
6719 {}

```

and finally, we localize the sections

```

6720 \newcommand\omdoc@part@kw{Part}
6721 \newcommand\omdoc@chapter@kw{Chapter}
6722 \newcommand\omdoc@section@kw{Section}
6723 \newcommand\omdoc@subsection@kw{Subsection}
6724 \newcommand\omdoc@subsubsection@kw{Subsubsection}
6725 \newcommand\omdoc@paragraph@kw{paragraph}
6726 \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`

```

6727 \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).

```

6728 \cs_if_exist:NTF\frontmatter{
6729   \let\__document_structure_orig_frontmatter\frontmatter
6730   \let\frontmatter\relax
6731 }{
6732   \tl_set:Nn\__document_structure_orig_frontmatter{
6733     \clearpage
6734     \@mainmatterfalse
6735     \pagenumbering{roman}
6736   }
6737 }
6738 \cs_if_exist:NTF\backmatter{
6739   \let\__document_structure_orig_backmatter\backmatter
6740   \let\backmatter\relax
6741 }{
6742   \tl_set:Nn\__document_structure_orig_backmatter{
6743     \clearpage
6744     \@mainmatterfalse

```

```

6745     \pagenumbering{roman}
6746   }
6747 }

```

Using these, we can now define the `frontmatter` and `backmatter` environments

frontmatter we use the `\orig@frontmatter` macro defined above and `\mainmatter` if it exists, otherwise we define it.

```

6748 \newenvironment{frontmatter}{
6749   \_document_structure_orig_frontmatter
6750 }{
6751   \cs_if_exist:NTF\mainmatter{
6752     \mainmatter
6753   }{
6754     \clearpage
6755     \@mainmattertrue
6756     \pagenumbering{arabic}
6757   }
6758 }

```

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

```

6759 \newenvironment{backmatter}{
6760   \_document_structure_orig_backmatter
6761 }{
6762   \cs_if_exist:NTF\mainmatter{
6763     \mainmatter
6764   }{
6765     \clearpage
6766     \@mainmattertrue
6767     \pagenumbering{arabic}
6768   }
6769 }

```

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

```

6770 \@mainmattertrue\pagenumbering{arabic}

```

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

```

6771 \def \c__document_structure_document_str{document}
6772 \newcommand\afterprematurestop{}
6773 \def\prematurestop@endsfragment{
6774   \unless\ifx\@currenvir\c__document_structure_document_str
6775     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
6776       \expandafter\prematurestop@endsfragment
6777     }
6778   }
6779 \providecommand\prematurestop{
6780   \message{Stopping~sTeX~processing~prematurely}
6781   \prematurestop@endsfragment
6782   \afterprematurestop
6783   \end{document}
6784 }

```

(End definition for `\prematurestop`. This function is documented on page 52.)

37.4 Global Variables

\setSGvar set a global variable

```
6785 \RequirePackage{etoolbox}
6786 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}
```

(End definition for \setSGvar. This function is documented on page 52.)

\useSGvar use a global variable

```
6787 \newrobustcmd\useSGvar[1]{%
6788   \@ifundefined{sTeX@Gvar@#1}
6789   {\PackageError{document-structure}
6790    {The sTeX Global variable #1 is undefined}
6791    {set it with \protect\setSGvar}}
6792   \@nameuse{sTeX@Gvar@#1}}
```

(End definition for \useSGvar. This function is documented on page 52.)

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

```
6793 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
6794   \@ifundefined{sTeX@Gvar@#1}
6795   {\PackageError{document-structure}
6796    {The sTeX Global variable #1 is undefined}
6797    {set it with \protect\setSGvar}}
6798   {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}
```

(End definition for \ifSGvar. This function is documented on page 52.)

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.

```
6799 \*cls)
6800 \@@=notesslides}
6801 \ProvidesExplClass{notesslides}{2022/02/28}{3.1.0}{notesslides Class}
6802 \RequirePackage{13keys2e}
6803
6804 \keys_define:nn{notesslides / cls}{
6805   class .str_set_x:N = \c__notesslides_class_str,
6806   notes .bool_set:N = \c__notesslides_notes_bool ,
6807   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
6808   docopt .str_set_x:N = \c__notesslides_docopt_str,
6809   unknown .code:n = {
6810     \PassOptionsToPackage{\CurrentOption}{document-structure}
6811     \PassOptionsToClass{\CurrentOption}{beamer}
6812     \PassOptionsToPackage{\CurrentOption}{notesslides}
6813     \PassOptionsToPackage{\CurrentOption}{stex}
6814   }
6815 }
6816 \ProcessKeysOptions{ notesslides / cls }
6817
6818 \str_if_empty:NF \c__notesslides_class_str {
6819   \PassOptionsToPackage{class=\c__notesslides_class_str}{document-structure}
6820 }
6821
6822 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{book}{
6823   \PassOptionsToPackage{defaultttopsect=part}{notesslides}
6824 }
6825 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{report}{
6826   \PassOptionsToPackage{defaultttopsect=part}{notesslides}
6827 }
6828
6829 \RequirePackage{stex}
```

```

6830 \stex_html_backend:T {
6831   \bool_set_true:N\c__notesslides_notes_bool
6832 }
6833
6834 \bool_if:NTF \c__notesslides_notes_bool {
6835   \PassOptionsToPackage{notes=true}{notesslides}
6836 }{
6837   \PassOptionsToPackage{notes=false}{notesslides}
6838 }
6839 \</cls>

```

now we do the same for the notesslides package.

```

6840 \*package>
6841 \ProvidesExplPackage{notesslides}{2022/02/28}{3.1.0}{notesslides Package}
6842 \RequirePackage{13keys2e}
6843
6844 \keys_define:nn{notesslides / pkg}{
6845   topsect          .str_set_x:N = \c__notesslides_topsect_str,
6846   defaulttopsect   .str_set_x:N = \c__notesslides_defaulttopsec_str,
6847   notes            .bool_set:N = \c__notesslides_notes_bool ,
6848   slides           .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
6849   sectocframes     .bool_set:N = \c__notesslides_sectocframes_bool ,
6850   frameimages      .bool_set:N = \c__notesslides_frameimages_bool ,
6851   fiboxed          .bool_set:N = \c__notesslides_fiboxed_bool ,
6852   nopproblems      .bool_set:N = \c__notesslides_nopproblems_bool,
6853   unknown          .code:n      = {
6854     \PassOptionsToClass{\CurrentOption}{stex}
6855     \PassOptionsToClass{\CurrentOption}{tikzinput}
6856   }
6857 }
6858 \ProcessKeysOptions{ notesslides / pkg }
6859
6860 \RequirePackage{stex}
6861 \stex_html_backend:T {
6862   \bool_set_true:N\c__notesslides_notes_bool
6863 }
6864
6865 \newif\ifnotes
6866 \bool_if:NTF \c__notesslides_notes_bool {
6867   \notesttrue
6868 }{
6869   \notesfalse
6870 }
6871

```

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

```

6872 \str_if_empty:NTF \c__notesslides_topsect_str {
6873   \str_set_eq:NN \__notesslidestopsect \c__notesslides_defaulttopsec_str
6874 }{
6875   \str_set_eq:NN \__notesslidestopsect \c__notesslides_topsect_str
6876 }
6877 \PassOptionsToPackage{topsect=\__notesslidestopsect}{document-structure}
6878 \</package>

```

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

```

6879 <*cls>
6880 \bool_if:NTF \c__notesslides_notes_bool {
6881   \str_if_empty:NT \c__notesslides_class_str {
6882     \str_set:Nn \c__notesslides_class_str {article}
6883   }
6884   \exp_after:wN\LoadClass\exp_after:wN[\c__notesslides_docostr]
6885     {\c__notesslides_class_str}
6886 }{
6887   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
6888   \newcounter{Item}
6889   \newcounter{paragraph}
6890   \newcounter{subparagraph}
6891   \newcounter{Hfootnote}
6892 }
6893 \RequirePackage{document-structure}

```

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

```

6894 \RequirePackage{notesslides}
6895 </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.

```

6896 <*package>
6897 \bool_if:NT \c__notesslides_notes_bool {
6898   \RequirePackage{a4wide}
6899   \RequirePackage{marginnote}
6900   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
6901   \RequirePackage{mdframed}
6902   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
6903   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
6904 }
6905 \RequirePackage{stex-tikzinput}
6906 \RequirePackage{etoolbox}
6907 \RequirePackage{amssymb}
6908 \RequirePackage{amsmath}
6909 \RequirePackage{comment}
6910 \RequirePackage{textcomp}
6911 \RequirePackage{url}
6912 \RequirePackage{graphicx}
6913 \RequirePackage{pgf}

```

38.2 Notes and Slides

For the lecture notes cases, we also provide the `\usetheme` macro that would otherwise come from the `beamer` class. While the latter loads `beamertheme<theme>.sty`, the

notes version loads `beamernotestheme(theme).sty`.¹⁸

```

6914 \bool_if:NT \c__notesslides_notes_bool {
6915   \renewcommand\usetheme[2] [] {\usepackage[#1]{beamernotestheme#2}}
6916 }
6917
6918
6919 \NewDocumentCommand \libusetheme {O{} m} {
6920   \bool_if:NTF \c__notesslides_notes_bool {
6921     \libusepackage[#1]{beamernotestheme#2}
6922   }{
6923     \libusepackage[#1]{beamertheme#2}
6924   }
6925 }

```

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

```

6926 \newcounter{slide}
6927 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
6928 \newlength{\slideheight}\setlength{\slideheight}{9cm}

```

note 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.

```

6929 \bool_if:NTF \c__notesslides_notes_bool {
6930   \renewenvironment{note}{\ignorespaces}{}
6931 }{
6932   \excludcomment{note}
6933 }

```

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.

```

6934 \bool_if:NT \c__notesslides_notes_bool {
6935   \newlength{\slideframewidth}
6936   \setlength{\slideframewidth}{1.5pt}

```

frame We first define the keys.

```

6937 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
6938   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
6939     \bool_set_true:N #1
6940   }{
6941     \bool_set_false:N #1
6942   }
6943 }
6944 \keys_define:nn{notesslides / frame}{
6945   label .str_set_x:N = \l__notesslides_frame_label_str,
6946   allowframebreaks .code:n = {
6947     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
6948   },
6949   allowdisplaybreaks .code:n = {

```

¹⁸EDNOTE: MK: This is not ideal, but I am not sure that I want to be able to provide the full theme functionality there.

```

6950     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_allowdisplaybreaks\_bool { #1 }
6951 },
6952 fragile .code:n = {
6953     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_fragile\_bool { #1 }
6954 },
6955 shrink .code:n = {
6956     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_shrink\_bool { #1 }
6957 },
6958 squeeze .code:n = {
6959     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_squeeze\_bool { #1 }
6960 },
6961 t .code:n = {
6962     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_t\_bool { #1 }
6963 },
6964 unknown .code:n = {}
6965 }
6966 \cs\_new\_protected:Nn \_notesslides\_frame\_args:n {
6967     \str\_clear:N \l\_notesslides\_frame\_label\_str
6968     \bool\_set\_true:N \l\_notesslides\_frame\_allowframebreaks\_bool
6969     \bool\_set\_true:N \l\_notesslides\_frame\_allowdisplaybreaks\_bool
6970     \bool\_set\_true:N \l\_notesslides\_frame\_fragile\_bool
6971     \bool\_set\_true:N \l\_notesslides\_frame\_shrink\_bool
6972     \bool\_set\_true:N \l\_notesslides\_frame\_squeeze\_bool
6973     \bool\_set\_true:N \l\_notesslides\_frame\_t\_bool
6974     \keys\_set:nn { notesslides / frame }{ #1 }
6975 }

```

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

```

6976 \renewenvironment{frame}[1][]{
6977     \_notesslides\_frame\_args:n{#1}
6978     \sffamily
6979     \stepcounter{slide}
6980     \def\@currentlabel{\theslide}
6981     \str\_if\_empty:NF \l\_notesslides\_frame\_label\_str {
6982         \label{\l\_notesslides\_frame\_label\_str}
6983     }

```

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

```

6984 \def\itemize@level{outer}
6985 \def\itemize@outer{outer}
6986 \def\itemize@inner{inner}
6987 \renewcommand\newpage{\addtocounter{framenum}{1}}
6988 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
6989 \renewenvironment{itemize}{
6990     \ifx\itemize@level\itemize@outer
6991         \def\itemize@label{\$ \rhd \$}
6992     \fi
6993     \ifx\itemize@level\itemize@inner
6994         \def\itemize@label{\$ \scriptstyle \rhd \$}
6995     \fi
6996     \begin{list}
6997     {\itemize@label}
6998     {\setlength{\labelsep}{.3em}
6999     \setlength{\labelwidth}{.5em}
7000     \setlength{\leftmargin}{1.5em}

```

```

7001     }
7002     \edef\itemize@level{\itemize@inner}
7003   }{
7004     \end{list}
7005   }

```

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

```

7006     \stex_html_backend:TF {
7007       \begin{stex_annotate_env}{frame}{}\vbox\bgroup
7008         \mdf@patchamsthm
7009     }{
7010       \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwid
7011     }
7012   }{
7013     \stex_html_backend:TF {
7014       \miko@slidelabel\egroup\end{stex_annotate_env}
7015     }{\medskip\miko@slidelabel\end{mdframed}}
7016   }

```

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

`\frametitle`

```

7017   \renewcommand{\frametitle}[1]{
7018     \stex_document_title:n { #1 }
7019     {\Large\bf\sf\color{blue}{#1}}\medskip
7020   }
7021 }

```

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

EdN:19

`\pause`

```

19
7022 \bool_if:NT \c__notesslides_notes_bool {
7023   \newcommand\pause{}
7024 }

```

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

`nparagraph`

```

7025 \bool_if:NTF \c__notesslides_notes_bool {
7026   \newenvironment{nparagraph}[1] []{\begin{sparagraph}[#1]}\end{sparagraph}}
7027 }{
7028   \excludacomment{nparagraph}
7029 }

```

`nfragment`

```

7030 \bool_if:NTF \c__notesslides_notes_bool {
7031   \newenvironment{nfragment}[2] []{\begin{sfragment}[#1]{#2}}\end{sfragment}}
7032 }{
7033   \excludacomment{nfragment}
7034 }

```

¹⁹EdNOTE: MK: fake it in notes mode for now

ndefinition

```
7035 \bool_if:NTF \c__notesslides_notes_bool {
7036   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}\end{sdefinition}}
7037 }{
7038   \excludecomment{ndefinition}
7039 }
```

nassertion

```
7040 \bool_if:NTF \c__notesslides_notes_bool {
7041   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}\end{sassertion}}
7042 }{
7043   \excludecomment{nassertion}
7044 }
```

nsproof

```
7045 \bool_if:NTF \c__notesslides_notes_bool {
7046   \newenvironment{nproof}[2] [] {\begin{sproof}[#1]{#2}}\end{sproof}}
7047 }{
7048   \excludecomment{nproof}
7049 }
```

nexample

```
7050 \bool_if:NTF \c__notesslides_notes_bool {
7051   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}\end{sexample}}
7052 }{
7053   \excludecomment{nexample}
7054 }
```

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

```
7055 \def\inputref@preskip{\smallskip}
7056 \def\inputref@postskip{\medskip}
```

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

\inputref*

```
7057 \let\orig@inputref\inputref
7058 \def\inputref{\@ifstar\ninputref\orig@inputref}
7059 \newcommand\ninputref[2] [] {
7060   \bool_if:NT \c__notesslides_notes_bool {
7061     \orig@inputref[#1]{#2}
7062   }
7063 }
```

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

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.

\setslidelo The default logo is the \TeX logo. Customization can be done by `\setslidelo{<logo name>}`.

```

7064 \newlength{\slideloheight}
7065
7066 \bool_if:NTF \c_notesslides_notes_bool {
7067   \setlength{\slideloheight}{.4cm}
7068 }{
7069   \setlength{\slideloheight}{1cm}
7070 }
7071 \newsavebox{\slidelo}
7072 \sbox{\slidelo}{\TeX}
7073 \newrobustcmd{\setslidelo}[1]{
7074   \sbox{\slidelo}{\includegraphics[height=\slideloheight]{#1}}
7075 }

```

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

\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.

```

7076 \def\source{Michael Kohlhase}% customize locally
7077 \newrobustcmd{\setsource}[1]{\def\source{#1}}

```

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

\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.

```

7078 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
7079 \newsavebox{\cclogo}
7080 \sbox{\cclogo}{\includegraphics[height=\slideloheight]{stex-cc_somerights}}
7081 \newif\ifcchref\cchreffalse
7082 \AtBeginDocument{
7083   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
7084 }
7085 \def\licensing{
7086   \ifcchref
7087     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
7088   \else
7089     {\usebox{\cclogo}}
7090   \fi
7091 }
7092 \newrobustcmd{\setlicensing}[2][]{
7093   \def\@url{#1}
7094   \sbox{\cclogo}{\includegraphics[height=\slideloheight]{#2}}
7095   \ifx\@url\@empty
7096     \def\licensing{{\usebox{\cclogo}}}
7097   \else
7098     \def\licensing{
7099       \ifcchref
7100         \href{#1}{\usebox{\cclogo}}
7101       \else
7102         {\usebox{\cclogo}}
7103       \fi

```

```

7104     }
7105     \fi
7106 }

```

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

EdN:20

`\slidelabel` Now, we set up the slide label for the article mode.²⁰

```

7107 \newrobustcmd\miko@slidelabel{
7108   \vbox to \slidelogoheight{
7109     \vss\hbox to \slidewidth
7110     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
7111   }
7112 }

```

(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.

```

7113 \def\Gin@mhrepos{}
7114 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7115 \define@key{Gin}{label}{\def\currentlabel{\arabic{slide}}\label{#1}}
7116 \newrobustcmd\frameimage[2][]{
7117   \stepcounter{slide}
7118   \bool_if:NT \c__notesslides_frameimages_bool {
7119     \def\Gin@ewidth{}\setkeys{Gin}{#1}
7120     \bool_if:NF \c__notesslides_notes_bool { \vfill }
7121     \begin{center}
7122       \bool_if:NTF \c__notesslides_fiboxed_bool {
7123         \fbox{
7124           \ifx\Gin@ewidth\@empty
7125             \ifx\Gin@mhrepos\@empty
7126               \mhgraphics[width=\slidewidth,#1]{#2}
7127             \else
7128               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7129             \fi
7130           \else% Gin@ewidth empty
7131             \ifx\Gin@mhrepos\@empty
7132               \mhgraphics[#1]{#2}
7133             \else
7134               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7135             \fi
7136           \fi% Gin@ewidth empty
7137         }
7138       }{
7139         \ifx\Gin@ewidth\@empty
7140           \ifx\Gin@mhrepos\@empty
7141             \mhgraphics[width=\slidewidth,#1]{#2}
7142           \else
7143             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7144           \fi

```

²⁰EdNOTE: see that we can use the themes for the slides some day. This is all fake.

```

7145         \ifx\Gin@mhrepos\@empty
7146         \mhgraphics[#1]{#2}
7147     \else
7148         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7149     \fi
7150     \fi% Gin@ewidth empty
7151 }
7152 \end{center}
7153 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
7154 \bool_if:NF \c__notesslides_notes_bool { \vfill }
7155 }
7156 } % ifmks@sty@frameimages

```

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

38.5 Colors and Highlighting

We first specify sans serif fonts as the default.

```

7157 \sffamily

```

Now, we set up an infrastructure for highlighting phrases in slides. Note that we use content-oriented macros for highlighting rather than directly using color markup. The first thing to do is to adapt the green so that it is dark enough for most beamers

```

7158 \AddToHook{begindocument}{
7159     \definecolor{green}{rgb}{0,.5,0}
7160     \definecolor{purple}{cmyk}{.3,1,0,.17}
7161 }

```

We customize the `\defemph`, `\symrefemph`, `\compemph`, and `\titleemph` macros with colors. Furthermore we customize the `__omtextlec` macro for the appearance of line end comments in `\lec`.

```

7162 % \def\STpresent#1{\textcolor{blue}{#1}}
7163 \def\defemph#1{\textcolor{magenta}{#1}}
7164 \def\symrefemph#1{\textcolor{cyan}{#1}}
7165 \def\compemph#1{\textcolor{blue}{#1}}
7166 \def\titleemph#1{\textcolor{blue}{#1}}
7167 \def\__omtext_lec#1{\textcolor{green}{#1}}

```

I like to use the dangerous bend symbol for warnings, so we provide it here.

`\textwarning` as the macro can be used quite often we put it into a box register, so that it is only loaded once.

```

7168 \pgfdeclareimage[width=.8em]{miko@small@dbend}{stex-dangerous-bend}
7169 \def\smalltextwarning{
7170     \pgfuseimage{miko@small@dbend}
7171     \xspace
7172 }
7173 \pgfdeclareimage[width=1.2em]{miko@dbend}{stex-dangerous-bend}
7174 \newrobustcmd\textwarning{
7175     \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
7176     \xspace
7177 }
7178 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{stex-dangerous-bend}

```

```

7179 \newrobustcmd\bigtextwarning{
7180   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
7181   \xspace
7182 }
(End definition for \textwarning. This function is documented on page 55.)
7183 \newrobustcmd\putgraphicsat[3]{
7184   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
7185 }
7186 \newrobustcmd\putat[2]{
7187   \begin{picture}(0,0)\put(#1){#2}\end{picture}
7188 }

```

38.6 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`.

```

7189 \stex_html_backend:F {
7190   \bool_if:NT \c__notesslides_sectocframes_bool {
7191     \str_if_eq:VnTF \__notesslidesstopsect{part}{
7192       \newcounter{chapter}\counterwithin*{section}{chapter}
7193     }{
7194       \str_if_eq:VnT \__notesslidesstopsect{chapter}{
7195         \newcounter{chapter}\counterwithin*{section}{chapter}
7196       }
7197     }
7198   }
7199 }

```

`\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
7200 \def\part@prefix{}
7201 \@ifpackageloaded{document-structure}{}{
7202   \str_case:VnF \__notesslidesstopsect {
7203     {part}{
7204       \int_set:Nn \l_document_structure_section_level_int {0}
7205       \def\thesection{\arabic{chapter}.\arabic{section}}
7206       \def\part@prefix{\arabic{chapter}.}
7207     }
7208     {chapter}{
7209       \int_set:Nn \l_document_structure_section_level_int {1}
7210       \def\thesection{\arabic{chapter}.\arabic{section}}
7211       \def\part@prefix{\arabic{chapter}.}
7212     }
7213   }{
7214     \int_set:Nn \l_document_structure_section_level_int {2}
7215     \def\part@prefix{}
7216   }
7217 }
7218
7219 \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 choses the L^AT_EX sectioning macros according to `\section@level`.

`sfragment`

```

7220 \renewenvironment{sfragment}[2][]{
7221   \_document_structure_sfragment_args:n { #1 }
7222   \int_incr:N \l_document_structure_section_level_int
7223   \bool_if:NT \c__notesslides_sectocframes_bool {
7224     \stepcounter{slide}
7225     \begin{frame}[noframenumbering]
7226     \vfill\Large\centering
7227     \red{
7228       \ifcase\l_document_structure_section_level_int\or
7229         \stepcounter{part}
7230         \def\_notesslideslabel{\omdoc@part@kw}~\Roman{part}}
7231         \def\currentsectionlevel{\omdoc@part@kw}
7232       \or
7233         \stepcounter{chapter}
7234         \def\_notesslideslabel{\omdoc@chapter@kw}~\arabic{chapter}}
7235         \def\currentsectionlevel{\omdoc@chapter@kw}
7236       \or
7237         \stepcounter{section}
7238         \def\_notesslideslabel{\part@prefix\arabic{section}}
7239         \def\currentsectionlevel{\omdoc@section@kw}
7240       \or
7241         \stepcounter{subsection}
7242         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
7243         \def\currentsectionlevel{\omdoc@subsection@kw}
7244       \or
7245         \stepcounter{subsubsection}
7246         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
7247         \def\currentsectionlevel{\omdoc@subsubsection@kw}
7248       \or
7249         \stepcounter{paragraph}
7250         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
7251         \def\currentsectionlevel{\omdoc@paragraph@kw}
7252       \else
7253         \def\_notesslideslabel{}
7254         \def\currentsectionlevel{\omdoc@paragraph@kw}
7255       \fi% end ifcase
7256       \_notesslideslabel%\sref@label@id\_notesslideslabel
7257       \quad #2%
7258     }%
7259     \vfill%
7260     \end{frame}%
7261   }
7262   \str_if_empty:NF \l_document_structure_sfragment_id_str {
7263     \stex_ref_new_doc_target:n\l_document_structure_sfragment_id_str
7264   }
7265   }{}
7266 }

```

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

```

7267 \def\inserttheorembodyfont{\normalfont}
7268 %\bool_if:NF \c__notesslides_notes_bool {
7269 % \defbeamertemplate{theorem begin}{miko}
7270 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
7271 % \ifx\inserttheoremaddition\empty\else\ (\inserttheoremaddition)\fi%
7272 % \inserttheorempunctuation\inserttheorembodyfont\xspace}
7273 % \defbeamertemplate{theorem end}{miko}{}
```

and we set it as the default one.

```

7274 % \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.

```

7275 % \expandafter\def\csname Parent2\endcsname{}
7276 %}
7277
7278 \AddToHook{begindocument}{ % this does not work for some reason
7279 \setbeamertemplate{theorems}[ams style]
7280 }
7281 \bool_if:NT \c__notesslides_notes_bool {
7282 \renewenvironment{columns}[1][]{%
7283 \par\noindent%
7284 \begin{minipage}%
7285 \slidewidth\centering\leavevmode%
7286 }{%
7287 \end{minipage}\par\noindent%
7288 }%
7289 \newsavebox\columnbox%
7290 \renewenvironment<>{column}[2][]{%
7291 \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
7292 }{%
7293 \end{minipage}\end{lrbox}\usebox\columnbox%
7294 }%
7295 }
7296 \bool_if:NTF \c__notesslides_noproblems_bool {
7297 \newenvironment{problems}{}{}
7298 }{
7299 \excludecomment{problems}
7300 }
```

38.7 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.

```

7301 \gdef\printexcursions{}
7302 \newcommand\excursionref[2]{% label, text
7303 \bool_if:NT \c__notesslides_notes_bool {
7304 \begin{sparagraph}[title=Excursion]
7305 #2 \sref[fallback=the appendix]{#1}.
7306 \end{sparagraph}}
```

```

7307 }
7308 }
7309 \newcommand\activate@excursion[2][]{
7310   \gappto\printexcursions{\inputref{#1}{#2}}
7311 }
7312 \newcommand\excursion[4][]{% repos, label, path, text
7313   \bool_if:NT \c__notesslides_notes_bool {
7314     \activate@excursion{#1}{#3}\excursionref{#2}{#4}
7315   }
7316 }

```

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

\excursiongroup

```

7317 \keys_define:nn{notesslides / excursiongroup }{
7318   id          .str_set_x:N = \l__notesslides_excursion_id_str,
7319   intro       .tl_set:N   = \l__notesslides_excursion_intro_tl,
7320   mhrepos     .str_set_x:N = \l__notesslides_excursion_mhrepos_str
7321 }
7322 \cs_new_protected:Nn \__notesslides_excursion_args:n {
7323   \tl_clear:N \l__notesslides_excursion_intro_tl
7324   \str_clear:N \l__notesslides_excursion_id_str
7325   \str_clear:N \l__notesslides_excursion_mhrepos_str
7326   \keys_set:nn {notesslides / excursiongroup }{ #1 }
7327 }
7328 \newcommand\excursiongroup[1][]{
7329   \__notesslides_excursion_args:n{ #1 }
7330   \ifdefempty\printexcursions{}% only if there are excursions
7331   {\begin{note}
7332     \begin{sfragment}[#1]{Excursions}%
7333     \ifdefempty\l__notesslides_excursion_intro_tl{\{
7334       \inputref[\l__notesslides_excursion_mhrepos_str]{
7335         \l__notesslides_excursion_intro_tl
7336       }
7337     }
7338     \printexcursions%
7339     \end{sfragment}
7340   }\end{note}}
7341 }
7342 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{\fi
7343 \</package>

```

(End definition for \excursiongroup. This function is documented on page 56.)

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.

```
7344 <*package>
7345 <@@=problems>
7346 \ProvidesExplPackage{problem}{2022/02/26}{3.0.1}{Semantic Markup for Problems}
7347 \RequirePackage{l3keys2e,stex}
7348
7349 \keys_define:nn { problem / pkg }{
7350   notes      .default:n    = { true },
7351   notes      .bool_set:N   = \c__problems_notes_bool,
7352   gnotes     .default:n    = { true },
7353   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
7354   hints      .default:n    = { true },
7355   hints      .bool_set:N   = \c__problems_hints_bool,
7356   solutions  .default:n    = { true },
7357   solutions  .bool_set:N   = \c__problems_solutions_bool,
7358   pts        .default:n    = { true },
7359   pts        .bool_set:N   = \c__problems_pts_bool,
7360   min        .default:n    = { true },
7361   min        .bool_set:N   = \c__problems_min_bool,
7362   boxed      .default:n    = { true },
7363   boxed      .bool_set:N   = \c__problems_boxed_bool,
7364   unknown    .code:n       = {}
7365 }
7366 \newif\ifsolutions
7367
7368 \ProcessKeysOptions{ problem / pkg }
7369 \bool_if:NTF \c__problems_solutions_bool {
7370   \solutionstrue
7371 }{
7372   \solutionsfalse
7373 }
```

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


```
7374 \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.

```
7375 \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.

```
7376 \def\prob@problem@kw{Problem}
7377 \def\prob@solution@kw{Solution}
7378 \def\prob@hint@kw{Hint}
7379 \def\prob@note@kw{Note}
7380 \def\prob@gnote@kw{Grading}
7381 \def\prob@pt@kw{pt}
7382 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
7383 \AddToHook{begindocument}{
7384   \ltx@ifpackageloaded{babel}{
7385     \makeatletter
7386     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7387     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7388       \input{problem-ngerman.ldf}
7389     }
7390     \clist_if_in:NnT \l_tmpa_clist {finnish}{
7391       \input{problem-finnish.ldf}
7392     }
7393     \clist_if_in:NnT \l_tmpa_clist {french}{
7394       \input{problem-french.ldf}
7395     }
7396     \clist_if_in:NnT \l_tmpa_clist {russian}{
7397       \input{problem-russian.ldf}
7398     }
7399     \makeatother
7400   }{}
7401 }
```

39.2 Problems and Solutions

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

```
7402 \keys_define:nn{ problem / problem }{
7403   id      .str_set_x:N = \l__problems_prob_id_str,
7404   pts     .tl_set:N    = \l__problems_prob_pts_tl,
7405   min     .tl_set:N    = \l__problems_prob_min_tl,
7406   title   .tl_set:N    = \l__problems_prob_title_tl,
7407   type    .tl_set:N    = \l__problems_prob_type_tl,
7408   imports .tl_set:N    = \l__problems_prob_imports_tl,
7409   name    .str_set_x:N = \l__problems_prob_name_str,
7410   refnum  .int_set:N   = \l__problems_prob_refnum_int
```

```

7411 }
7412 \cs_new_protected:Nn \__problems_prob_args:n {
7413   \str_clear:N \l__problems_prob_id_str
7414   \str_clear:N \l__problems_prob_name_str
7415   \tl_clear:N \l__problems_prob_pts_tl
7416   \tl_clear:N \l__problems_prob_min_tl
7417   \tl_clear:N \l__problems_prob_title_tl
7418   \tl_clear:N \l__problems_prob_type_tl
7419   \tl_clear:N \l__problems_prob_imports_tl
7420   \int_zero_new:N \l__problems_prob_refnum_int
7421   \keys_set:nn { problem / problem }{ #1 }
7422   \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
7423     \let\l__problems_prob_refnum_int\undefined
7424   }
7425 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

7426 \newcounter{problem}[section]
7427 \newcommand\numberproblemsin[1]{\@addtoreset{problem}{#1}}

```

(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.

```

7428 \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

```

7429 \newcommand\prob@number{
7430   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
7431     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
7432   }{
7433     \int_if_exist:NTF \l__problems_prob_refnum_int {
7434       \prob@label{\int_use:N \l__problems_prob_refnum_int }
7435     }{
7436       \prob@label\theproblem
7437     }
7438   }
7439 }

```

(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.

```

7440 \newcommand\prob@title[3]{%
7441   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
7442     #2 \l__problems_inclprob_title_tl #3
7443   }{
7444     \tl_if_exist:NTF \l__problems_prob_title_tl {
7445       #2 \l__problems_prob_title_tl #3
7446     }{
7447       #1

```

```

7448     }
7449   }
7450 }

```

(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.

```

7451 \def\prob@heading{
7452   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
7453   %\sref@label@id{\prob@problem@kw~\prob@number}{~}
7454 }

```

(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`

```

7455 \newenvironment{sproblem}[1][{}]{
7456   \__problems_prob_args:n{#1}%\sref@target%
7457   \@in@omtexttrue% we are in a statement (for inline definitions)
7458   \stepcounter{problem}\record@problem
7459   \def\current@section@level{\prob@problem@kw}
7460
7461   \str_if_empty:NT \l__problems_prob_name_str {
7462     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
7463     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
7464     \seq_get_left:NN \l_tmpa_seq \l__problems_prob_name_str
7465   }
7466
7467   \stex_if_do_html:T{
7468     \tl_if_empty:NF \l__problems_prob_title_tl {
7469       \exp_args:No \stex_document_title:n \l__problems_prob_title_tl
7470     }
7471   }
7472
7473   \exp_args:Nno\stex_module_setup:nn{type=problem}\l__problems_prob_name_str
7474
7475   \stex_reactivate_macro:N \STEXexport
7476   \stex_reactivate_macro:N \importmodule
7477   \stex_reactivate_macro:N \symdecl
7478   \stex_reactivate_macro:N \notation
7479   \stex_reactivate_macro:N \symdef
7480
7481   \stex_if_do_html:T{
7482     \begin{stex_annotate_env} {problem} {
7483       \l_stex_module_ns_str ? \l_stex_module_name_str
7484     }
7485
7486     \stex_annotate_invisible:nnn{header}{} {
7487       \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}

```

```

7488     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
7489     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
7490         \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
7491     }
7492 }
7493 }
7494
7495 \stex_csl_to_imports:No \importmodule \l__problems_prob_imports_tl
7496
7497
7498 \tl_if_exist:NTF \l__problems_inclprob_type_tl {
7499     \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
7500 }{
7501     \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
7502 }
7503 \str_if_exist:NTF \l__problems_inclprob_id_str {
7504     \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
7505 }{
7506     \str_set_eq:NN \sproblemid \l__problems_prob_id_str
7507 }
7508
7509
7510 \stex_if_smsmode:F {
7511     \clist_set:No \l_tmpa_clist \sproblemtype
7512     \tl_clear:N \l_tmpa_tl
7513     \clist_map_inline:Nn \l_tmpa_clist {
7514         \tl_if_exist:cT {__problems_sproblem_##1_start:}{
7515             \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_start:}}
7516         }
7517     }
7518     \tl_if_empty:NTF \l_tmpa_tl {
7519         \__problems_sproblem_start:
7520     }{
7521         \l_tmpa_tl
7522     }
7523 }
7524 \stex_ref_new_doc_target:n \sproblemid
7525 \stex_smsmode_do:
7526 }{
7527     \__stex_modules_end_module:
7528     \stex_if_smsmode:F{
7529         \clist_set:No \l_tmpa_clist \sproblemtype
7530         \tl_clear:N \l_tmpa_tl
7531         \clist_map_inline:Nn \l_tmpa_clist {
7532             \tl_if_exist:cT {__problems_sproblem_##1_end:}{
7533                 \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_end:}}
7534             }
7535         }
7536         \tl_if_empty:NTF \l_tmpa_tl {
7537             \__problems_sproblem_end:
7538         }{
7539             \l_tmpa_tl
7540         }
7541     }

```

```

7542 \stex_if_do_html:T{
7543   \end{stex_annotate_env}
7544 }
7545
7546 \smallskip
7547 }
7548
7549 \seq_put_right:Nx\g_stex_smsmode_allowedenvs_seq{\tl_to_str:n{sproblem}}
7550
7551
7552
7553 \cs_new_protected:Nn \__problems_sproblem_start: {
7554   \par\noindent\textbf{\prob@heading\show@pts\show@min\\\ignorespacesandpars
7555 }
7556 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
7557
7558 \newcommand\stexpatchproblem[3][] {
7559   \str_set:Nx \l_tmpa_str{ #1 }
7560   \str_if_empty:NTF \l_tmpa_str {
7561     \tl_set:Nn \__problems_sproblem_start: { #2 }
7562     \tl_set:Nn \__problems_sproblem_end: { #3 }
7563   }{
7564     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
7565     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
7566   }
7567 }
7568
7569
7570 \bool_if:NT \c__problems_boxed_bool {
7571   \surroundwithmdframed{problem}
7572 }

```

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

```

7573 \def\record@problem{
7574   \protected@write\@auxout{}
7575   {
7576     \string\@problem{\prob@number}
7577     {
7578       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7579         \l__problems_inclprob_pts_tl
7580       }{
7581         \l__problems_prob_pts_tl
7582       }
7583     }%
7584     {
7585       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7586         \l__problems_inclprob_min_tl
7587       }{
7588         \l__problems_prob_min_tl
7589       }
7590     }
7591   }
7592 }

```

(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).

```
7593 \def\@problem#1#2#3{}
```

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

`solution` 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.

```
7594 \keys_define:nn { problem / solution }{
7595   id          .str_set_x:N = \l__problems_solution_id_str ,
7596   for         .tl_set:N   = \l__problems_solution_for_tl ,
7597   height      .dim_set:N  = \l__problems_solution_height_dim ,
7598   creators    .clist_set:N = \l__problems_solution_creators_clist ,
7599   contributors .clist_set:N = \l__problems_solution_contributors_clist ,
7600   srccite     .tl_set:N    = \l__problems_solution_srccite_tl
7601 }
7602 \cs_new_protected:Nn \__problems_solution_args:n {
7603   \str_clear:N \l__problems_solution_id_str
7604   \tl_clear:N \l__problems_solution_for_tl
7605   \tl_clear:N \l__problems_solution_srccite_tl
7606   \clist_clear:N \l__problems_solution_creators_clist
7607   \clist_clear:N \l__problems_solution_contributors_clist
7608   \dim_zero:N \l__problems_solution_height_dim
7609   \keys_set:nn { problem / solution }{ #1 }
7610 }
```

the next step is to define a helper macro that does what is needed to start a solution.

```
7611 \newcommand\@startsolution[1][]{
7612   \__problems_solution_args:n { #1 }
7613   \@in@omtexttrue% we are in a statement.
7614   \bool_if:NF \c__problems_boxed_bool { \hrule }
7615   \smallskip\noindent
7616   {\textbf{\prob@solution@kw :}\enspace}
7617   \begin{small}
7618   \def\current@section@level{\prob@solution@kw}
7619   \ignorespacesandpars
7620 }
```

`\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.

```
7621 \box_new:N \l__problems_solution_box
7622 \newenvironment{solution}[1][]{
7623   \stex_html_backend:TF{
7624     \stex_if_do_html:T{
7625       \begin{stex_annotate_env}{solution}{}}
7626   }
7627   }{
7628     \setbox\l__problems_solution_box\vbox\bgroup
7629     \par\smallskip\hrule\smallskip
7630     \noindent\textbf{Solution:}~
7631   }
7632 }{
7633   \stex_html_backend:TF{
```

```

7634 \stex_if_do_html:T{
7635 \end{stex_annotate_env}
7636 }
7637 }{
7638 \smallskip\hrule
7639 \egroup
7640 \bool_if:NT \c__problems_solutions_bool {
7641 \box\l__problems_solution_box
7642 }
7643 }
7644 }
7645
7646 \newcommand\startsolutions{
7647 \bool_set_true:N \c__problems_solutions_bool
7648 % \specialcomment{solution}{\@startsolution}{
7649 % \bool_if:NF \c__problems_boxed_bool {
7650 % \hrule\medskip
7651 % }
7652 % \end{small}}%
7653 % }
7654 % \bool_if:NT \c__problems_boxed_bool {
7655 % \surroundwithmdframed{solution}
7656 % }
7657 }

```

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

\stopsolutions

```

7658 \newcommand\stopsolutions{\bool_set_false:N \c__problems_solutions_bool}%\excludecomment{sol

```

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

so it only remains to start/stop solutions depending on what option was specified.

```

7659 \ifsolutions
7660 \startsolutions
7661 \else
7662 \stopsolutions
7663 \fi

```

exnote

```

7664 \bool_if:NTF \c__problems_notes_bool {
7665 \newenvironment{exnote}[1][{}]{
7666 \par\smallskip\hrule\smallskip
7667 \noindent\textbf{\prob@note@kw :~ }\small
7668 }{
7669 \smallskip\hrule
7670 }
7671 }{
7672 \excludecomment{exnote}
7673 }

```

hint

```

7674 \bool_if:NTF \c__problems_notes_bool {
7675 \newenvironment{hint}[1][{}]{
7676 \par\smallskip\hrule\smallskip

```

```

7677 \noindent\textbf{\prob@hint@kw :~ }\small
7678 }{
7679 \smallskip\hrule
7680 }
7681 \newenvironment{exhint}[1][]{
7682 \par\smallskip\hrule\smallskip
7683 \noindent\textbf{\prob@hint@kw :~ }\small
7684 }{
7685 \smallskip\hrule
7686 }
7687 }{
7688 \excludecomment{hint}
7689 \excludecomment{exhint}
7690 }

```

gnote

```

7691 \bool_if:NTF \c__problems_notes_bool {
7692 \newenvironment{gnote}[1][]{
7693 \par\smallskip\hrule\smallskip
7694 \noindent\textbf{\prob@gnote@kw :~ }\small
7695 }{
7696 \smallskip\hrule
7697 }
7698 }{
7699 \excludecomment{gnote}
7700 }

```

39.3 Multiple Choice Blocks

EdN:21

mcb 21

```

7701 \newenvironment{mcb}{
7702 \begin{enumerate}
7703 }{
7704 \end{enumerate}
7705 }

```

we define the keys for the mcb macro

```

7706 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
7707 \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
7708 \bool_set_true:N #1
7709 }{
7710 \bool_set_false:N #1
7711 }
7712 }
7713 \keys_define:nn { problem / mcb }{
7714 id .str_set:x:N = \l__problems_mcc_id_str ,
7715 feedback .tl_set:N = \l__problems_mcc_feedback_tl ,
7716 T .default:n = { false } ,
7717 T .bool_set:N = \l__problems_mcc_t_bool ,
7718 F .default:n = { false } ,
7719 F .bool_set:N = \l__problems_mcc_f_bool ,

```

²¹EdNOTE: MK: maybe import something better here from a dedicated MC package


```

7720 Ttext      .tl_set:N      = \l__problems_mcc_Ttext_str ,
7721 Ftext      .tl_set:N      = \l__problems_mcc_Ftext_str
7722 }
7723 \cs_new_protected:Nn \l__problems_mcc_args:n {
7724   \str_clear:N \l__problems_mcc_id_str
7725   \tl_clear:N \l__problems_mcc_feedback_tl
7726   \bool_set_false:N \l__problems_mcc_t_bool
7727   \bool_set_false:N \l__problems_mcc_f_bool
7728   \tl_clear:N \l__problems_mcc_Ttext_tl
7729   \tl_clear:N \l__problems_mcc_Ftext_tl
7730   \str_clear:N \l__problems_mcc_id_str
7731   \keys_set:nn { problem / mcc }{ #1 }
7732 }

```

\mcc

```

7733 \def\mccTrueText{\textbf{(true)}~}
7734 \def\mccFalseText{\textbf{(false)}~}
7735 \newcommand\mcc[2][]{}
7736   \l__problems_mcc_args:n{ #1 }
7737   \item[{$\Box$}] #2
7738   \ifsolutions
7739     \\\
7740     \bool_if:NT \l__problems_mcc_t_bool {
7741       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccTrueText\l__problems_mcc_Ttext_tl
7742     }
7743     \bool_if:NT \l__problems_mcc_f_bool {
7744       \tl_if_empty:NTF\l__problems_mcc_Ftext_tl\mccFalseText\l__problems_mcc_Ftext_tl
7745     }
7746     \tl_if_empty:NF \l__problems_mcc_feedback_tl {
7747       \emph{(\l__problems_mcc_feedback_tl)}
7748     }
7749   \fi
7750 } %solutions

```

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

39.4 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.

```

7751
7752 \keys_define:nn{ problem / inclproblem }{
7753   id      .str_set_x:N = \l__problems_inclprob_id_str,
7754   pts     .tl_set:N    = \l__problems_inclprob_pts_tl,
7755   min     .tl_set:N    = \l__problems_inclprob_min_tl,
7756   title   .tl_set:N    = \l__problems_inclprob_title_tl,
7757   refnum  .int_set:N    = \l__problems_inclprob_refnum_int,
7758   type    .tl_set:N    = \l__problems_inclprob_type_tl,
7759   mhrepos .str_set_x:N = \l__problems_inclprob_mhrepos_str
7760 }
7761 \cs_new_protected:Nn \__problems_inclprob_args:n {
7762   \str_clear:N \l__problems_prob_id_str

```

```

7763 \tl_clear:N \l__problems_inclprob_pts_tl
7764 \tl_clear:N \l__problems_inclprob_min_tl
7765 \tl_clear:N \l__problems_inclprob_title_tl
7766 \tl_clear:N \l__problems_inclprob_type_tl
7767 \int_zero_new:N \l__problems_inclprob_refnum_int
7768 \str_clear:N \l__problems_inclprob_mhrepos_str
7769 \keys_set:nn { problem / inclproblem }{ #1 }
7770 \tl_if_empty:NT \l__problems_inclprob_pts_tl {
7771   \let\l__problems_inclprob_pts_tl\undefined
7772 }
7773 \tl_if_empty:NT \l__problems_inclprob_min_tl {
7774   \let\l__problems_inclprob_min_tl\undefined
7775 }
7776 \tl_if_empty:NT \l__problems_inclprob_title_tl {
7777   \let\l__problems_inclprob_title_tl\undefined
7778 }
7779 \tl_if_empty:NT \l__problems_inclprob_type_tl {
7780   \let\l__problems_inclprob_type_tl\undefined
7781 }
7782 \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
7783   \let\l__problems_inclprob_refnum_int\undefined
7784 }
7785 }
7786
7787 \cs_new_protected:Nn \__problems_inclprob_clear: {
7788   \let\l__problems_inclprob_id_str\undefined
7789   \let\l__problems_inclprob_pts_tl\undefined
7790   \let\l__problems_inclprob_min_tl\undefined
7791   \let\l__problems_inclprob_title_tl\undefined
7792   \let\l__problems_inclprob_type_tl\undefined
7793   \let\l__problems_inclprob_refnum_int\undefined
7794   \let\l__problems_inclprob_mhrepos_str\undefined
7795 }
7796 \__problems_inclprob_clear:
7797
7798 \newcommand\includeproblem[2][ ]{
7799   \__problems_inclprob_args:n{ #1 }
7800   \exp_args:No \stex_in_repository:nn\l__problems_inclprob_mhrepos_str{
7801     \stex_html_backend:TF {
7802       \str_clear:N \l_tmpa_str
7803       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
7804         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
7805       }
7806       \stex_annotate_invisible:nnn{includeproblem}{
7807         \l_tmpa_str / #2
7808       }{}
7809     }{
7810       \begingroup
7811         \inputreftrue
7812         \tl_if_empty:nTF{ ##1 }{
7813           \input{#2}
7814         }{
7815           \input{ \c_stex_mathhub_str / ##1 / source / #2 }
7816         }

```

```

7817     \endgroup
7818   }
7819 }
7820 \__problems_inclprob_clear:
7821 }

```

(End definition for `\includeproblem`. This function is documented on page 59.)

39.5 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.

```

7822 \AddToHook{enddocument}{
7823   \bool_if:NT \c__problems_pts_bool {
7824     \message{Total:~\arabic{pts}~points}
7825   }
7826   \bool_if:NT \c__problems_min_bool {
7827     \message{Total:~\arabic{min}~minutes}
7828   }
7829 }

```

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

```

7830 \def\pts#1{
7831   \bool_if:NT \c__problems_pts_bool {
7832     \marginpar{#1~\prob@pt@kw}
7833   }
7834 }
7835 \def\min#1{
7836   \bool_if:NT \c__problems_min_bool {
7837     \marginpar{#1~\prob@min@kw}
7838   }
7839 }

```

`\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.

```

7840 \newcounter{pts}
7841 \def\show@pts{
7842   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7843     \bool_if:NT \c__problems_pts_bool {
7844       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
7845       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
7846     }
7847   }{
7848     \tl_if_exist:NT \l__problems_prob_pts_tl {
7849       \bool_if:NT \c__problems_pts_bool {
7850         \tl_if_empty:NTF \l__problems_prob_pts_tl{
7851           \tl_set:Nn \l__problems_prob_pts_tl {0}
7852         }
7853         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
7854         \addtocounter{pts}{\l__problems_prob_pts_tl}
7855       }
7856     }

```

```

7857 }
7858 }

```

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

\show@min

```

7859 \newcounter{min}
7860 \def\show@min{
7861   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7862     \bool_if:NT \c__problems_min_bool {
7863       \marginpar{\l__problems_inclprob_pts_tl\ min}
7864       \addtocounter{min}{\l__problems_inclprob_min_tl}
7865     }
7866   }{
7867     \tl_if_exist:NT \l__problems_prob_min_tl {
7868       \bool_if:NT \c__problems_min_bool {
7869         \tl_if_empty:NT\l__problems_prob_min_tl{
7870           \tl_set:Nn \l__problems_prob_min_tl {0}
7871         }
7872         \marginpar{\l__problems_prob_min_tl\ min}
7873         \addtocounter{min}{\l__problems_prob_min_tl}
7874       }
7875     }
7876   }
7877 }
7878 \</package>

```

(End definition for \show@min. 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.

```
7879 \*package>
7880 \ProvidesExplPackage{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
7881 \RequirePackage{13keys2e}
7882
7883 \newif\iftest\testfalse
7884 \DeclareOption{test}{\testtrue}
7885 \newif\ifmultiple\multiplefalse
7886 \DeclareOption{multiple}{\multipletrue}
7887 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
7888 \ProcessOptions
```

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

```
7889 \RequirePackage{keyval}[1997/11/10]
7890 \RequirePackage{problem}
```

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

```
7891 \newcommand\hwexam@assignment@kw{Assignment}
7892 \newcommand\hwexam@given@kw{Given}
7893 \newcommand\hwexam@due@kw{Due}
7894 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
7895 blank~for~extra~space}
7896 \def\hwexam@minutes@kw{minutes}
7897 \newcommand\correction@probs@kw{prob.}
7898 \newcommand\correction@pts@kw{total}
7899 \newcommand\correction@reached@kw{reached}
7900 \newcommand\correction@sum@kw{Sum}
7901 \newcommand\correction@grade@kw{grade}
7902 \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

```

7903 \AddToHook{begindocument}{
7904 \ltx@ifpackageloaded{babel}{
7905 \makeatletter
7906 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7907 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7908 \input{hwexam-ngerman.ldf}
7909 }
7910 \clist_if_in:NnT \l_tmpa_clist {finnish}{
7911 \input{hwexam-finnish.ldf}
7912 }
7913 \clist_if_in:NnT \l_tmpa_clist {french}{
7914 \input{hwexam-french.ldf}
7915 }
7916 \clist_if_in:NnT \l_tmpa_clist {russian}{
7917 \input{hwexam-russian.ldf}
7918 }
7919 \makeatother
7920 }{}
7921 }
7922

```

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.

```

7923 \newcounter{assignment}
7924 %\numberproblemsin{assignment}

We will prepare the keyval support for the assignment environment.

7925 \keys_define:nn { hwexam / assignment } {
7926 id .str_set:N = \l_@@_assign_id_str,
7927 number .int_set:N = \l_@@_assign_number_int,
7928 title .tl_set:N = \l_@@_assign_title_tl,
7929 type .tl_set:N = \l_@@_assign_type_tl,
7930 given .tl_set:N = \l_@@_assign_given_tl,
7931 due .tl_set:N = \l_@@_assign_due_tl,
7932 loadmodules .code:n = {
7933 \bool_set_true:N \l_@@_assign_loadmodules_bool
7934 }
7935 }
7936 \cs_new_protected:Nn \_@@_assignment_args:n {
7937 \str_clear:N \l_@@_assign_id_str
7938 \int_set:Nn \l_@@_assign_number_int {-1}
7939 \tl_clear:N \l_@@_assign_title_tl
7940 \tl_clear:N \l_@@_assign_type_tl
7941 \tl_clear:N \l_@@_assign_given_tl
7942 \tl_clear:N \l_@@_assign_due_tl
7943 \bool_set_false:N \l_@@_assign_loadmodules_bool
7944 \keys_set:nn { hwexam / assignment }{ #1 }
7945 }

```

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.

```

7946 \newcommand\given@due[2]{
7947 \bool_lazy_all:nF {
7948 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
7949 { \tl_if_empty_p:V \l_@@_assign_given_tl }
7950 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
7951 { \tl_if_empty_p:V \l_@@_assign_due_tl }
7952 }{ #1 }
7953
7954 \tl_if_empty:NTF \l_@@_inclasssign_given_tl {
7955 \tl_if_empty:NF \l_@@_assign_given_tl {
7956 \hwexam@given@kw\xspace\l_@@_assign_given_tl
7957 }
7958 }{
7959 \hwexam@given@kw\xspace\l_@@_inclasssign_given_tl
7960 }
7961
7962 \bool_lazy_or:nnF {
7963 \bool_lazy_and_p:nn {
7964 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
7965 }{
7966 \tl_if_empty_p:V \l_@@_assign_due_tl
7967 }
7968 }{
7969 \bool_lazy_and_p:nn {
7970 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
7971 }{
7972 \tl_if_empty_p:V \l_@@_assign_due_tl
7973 }
7974 }{ ,~ }
7975
7976 \tl_if_empty:NTF \l_@@_inclasssign_due_tl {
7977 \tl_if_empty:NF \l_@@_assign_due_tl {
7978 \hwexam@due@kw\xspace \l_@@_assign_due_tl
7979 }
7980 }{
7981 \hwexam@due@kw\xspace \l_@@_inclasssign_due_tl
7982 }
7983
7984 \bool_lazy_all:nF {
7985 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
7986 { \tl_if_empty_p:V \l_@@_assign_given_tl }
7987 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
7988 { \tl_if_empty_p:V \l_@@_assign_due_tl }
7989 }{ #2 }
7990 }

```

`\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.

```

7991 \newcommand\assignment@title[3]{
7992 \tl_if_empty:NTF \l_@@_inclasssign_title_tl {
7993 \tl_if_empty:NTF \l_@@_assign_title_tl {
7994 #1
7995 }{
7996 #2\l_@@_assign_title_tl#3
7997 }
7998 }{
7999 #2\l_@@_inclasssign_title_tl#3
8000 }
8001 }

```

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

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

```

8002 \newcommand\assignment@number{
8003 \int_compare:nNnTF \l_@@_inclasssign_number_int = {-1} {
8004 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8005 \arabic{assignment}
8006 } {
8007 \int_use:N \l_@@_assign_number_int
8008 }
8009 }{
8010 \int_use:N \l_@@_inclasssign_number_int
8011 }
8012 }

```

(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 For the assignment environment we delegate the work to the @assignment environment that depends on whether multiple option is given.

```

8013 \newenvironment{assignment}[1][]{
8014 \_@@_assignment_args:n { #1 }
8015 %\sref@target
8016 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8017 \global\stepcounter{assignment}
8018 }{
8019 \global\setcounter{assignment}{\int_use:N\l_@@_assign_number_int}
8020 }
8021 \setcounter{problem}{0}
8022 \renewcommand\prob@label[1]{\assignment@number.##1}
8023 \def\current@section@level{\document@hwexamtype}
8024 %\sref@label{id}{\document@hwexamtype \thesection}
8025 \begin{@assignment}
8026 }{
8027 \end{@assignment}
8028 }

```


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

```

8029 \def\ass@title{
8030 {\protect\document@hwexamtype}\arabic{assignment}
8031 \assignment@title{}\;\;(\;)\;\} -- \given@due{}\;}
8032 }
8033 \ifmultiple
8034 \newenvironment{@assignment}{
8035 \bool_if:NTF \l_@@_assign_loadmodules_bool {
8036 \begin{sfragment}[loadmodules]{\ass@title}
8037 }{
8038 \begin{sfragment}{\ass@title}
8039 }
8040 }{
8041 \end{sfragment}
8042 }

```

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

```

8043 \else
8044 \newenvironment{@assignment}{
8045 \begin{center}\bf
8046 \Large@title\strut\
8047 \document@hwexamtype}\arabic{assignment}\assignment@title{}\;\;(\;)\;\}
8048 \large\given@due{--\;\;}\;\;--}
8049 \end{center}
8050 }{}
8051 \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.

```

8052 \keys_define:nn { hwexam / inclassignment } {
8053 %id .str_set_x:N = \l_@@_assign_id_str,
8054 number .int_set:N = \l_@@_inclassign_number_int,
8055 title .tl_set:N = \l_@@_inclassign_title_tl,
8056 type .tl_set:N = \l_@@_inclassign_type_tl,
8057 given .tl_set:N = \l_@@_inclassign_given_tl,
8058 due .tl_set:N = \l_@@_inclassign_due_tl,
8059 mhrepos .str_set_x:N = \l_@@_inclassign_mhrepos_str
8060 }
8061 \cs_new_protected:Nn \l_@@_inclassignment_args:n {
8062 \int_set:Nn \l_@@_inclassign_number_int {-1}
8063 \tl_clear:N \l_@@_inclassign_title_tl
8064 \tl_clear:N \l_@@_inclassign_type_tl
8065 \tl_clear:N \l_@@_inclassign_given_tl
8066 \tl_clear:N \l_@@_inclassign_due_tl
8067 \str_clear:N \l_@@_inclassign_mhrepos_str
8068 \keys_set:nn { hwexam / inclassignment }{ #1 }
8069 }
8070 \l_@@_inclassignment_args:n {}
8071
8072 \newcommand\inputassignment[2][{}]{

```

```

8073 \_@@_inclassignment_args:n { #1 }
8074 \str_if_empty:NTF \l_@@_inclassign_mhrepos_str {
8075   \input{#2}
8076 }{
8077   \stex_in_repository:nn{\l_@@_inclassign_mhrepos_str}{
8078     \input{\mhpath{\l_@@_inclassign_mhrepos_str}{#2}}
8079   }
8080 }
8081 \_@@_inclassignment_args:n {}
8082 }
8083 \newcommand\includeassignment[2][]{
8084   \newpage
8085   \inputassignment[#1]{#2}
8086 }

```

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

40.4 Typesetting Exams

\quizheading

```

8087 \ExplSyntaxOff
8088 \newcommand\quizheading[1]{%
8089   \def\@tas{#1}%
8090   \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
8091   \ifx\@tas\@empty\else%
8092     \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
8093   \fi%
8094 }
8095 \ExplSyntaxOn

```

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

\testheading

```

8096
8097 \def\hwexamheader{\input{hwexam-default.header}}
8098
8099 \def\hwexamminutes{
8100   \tl_if_empty:NTF \testheading@duration {
8101     {\testheading@min}~\hwexam@minutes@kw
8102   }{
8103     \testheading@duration
8104   }
8105 }
8106
8107 \keys_define:nn { hwexam / testheading } {
8108   min .tl_set:N = \testheading@min,
8109   duration .tl_set:N = \testheading@duration,
8110   reqpts .tl_set:N = \testheading@reqpts,
8111   tools .tl_set:N = \testheading@tools
8112 }
8113 \cs_new_protected:Nn \_@@_testheading_args:n {
8114   \tl_clear:N \testheading@min
8115   \tl_clear:N \testheading@duration

```

```

8116 \tl_clear:N \testheading@reqpts
8117 \tl_clear:N \testheading@tools
8118 \keys_set:nn { hwexam / testheading }{ #1 }
8119 }
8120 \newenvironment{testheading}[1][]{
8121 \_@@_testheading_args:n{ #1 }
8122 \newcount\check@time\check@time=\testheading@min
8123 \advance\check@time by -\theassignment@totalmin
8124 \newif\if@bonuspoints
8125 \tl_if_empty:NTF \testheading@reqpts {
8126 \@bonuspointsfalse
8127 }{
8128 \newcount\bonus@pts
8129 \bonus@pts=\theassignment@totalpts
8130 \advance\bonus@pts by -\testheading@reqpts
8131 \edef\bonus@pts{\the\bonus@pts}
8132 \@bonuspointstrue
8133 }
8134 \edef\check@time{\the\check@time}
8135
8136 \makeatletter\hwexamheader\makeatother
8137 }{
8138 \newpage
8139 }

```

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

\testspace

```

8140 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

8141 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

8142 \newcommand\testemptypage[1][]{\iftest\begin{center}\hwexam@testemptypage@kw\end{center}\vfi}

```

(End definition for \testemptypage. 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.

```

8143 <@@=problems>
8144 \renewcommand\@problem[3]{
8145 \stepcounter{assignment@probs}
8146 \def\__problemspts{#2}
8147 \ifx\__problemspts\@empty\else
8148 \addtocounter{assignment@totalpts}{#2}
8149 \fi
8150 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
8151 \xdef\correction@probs{\correction@probs & #1}%
8152 \xdef\correction@pts{\correction@pts & #2}
8153 \xdef\correction@reached{\correction@reached &}

```

```

8154 }
8155 <@@=hwexam>

```

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

`\correction@table` This macro generates the correction table

```

8156 \newcounter{assignment@probs}
8157 \newcounter{assignment@totalpts}
8158 \newcounter{assignment@totalmin}
8159 \def\correction@probs{\correction@probs@kw}
8160 \def\correction@pts{\correction@pts@kw}
8161 \def\correction@reached{\correction@reached@kw}
8162 \stepcounter{assignment@probs}
8163 \newcommand\correction@table{
8164 \resizebox{\textwidth}{!}{%
8165 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
8166 &\multicolumn{\theassignment@probs}{c|}||%|
8167 {\footnotesize\correction@forgrading@kw} &\\ \hline
8168 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
8169 \correction@pts & \theassignment@totalpts & \\ \hline
8170 \correction@reached & & \[.7cm]\hline
8171 \end{tabular}}
8172 </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

- [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).

²²EDNOTE: 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).